# AI501/DM879 Introduction to Artificial Intelligence

List of exercises

***N.B.:*** *Many exercises, including the accompanying figures, are reproduced or adapted from the list of exercises at* `https://aimacode.github.io/aima-exercises/`*, included in the first three editions of the reference book. They are reproduced here for convenience.*

## 1 Intelligent agents

1. For each of the following assertions, say whether it is true or false and support your answer with examples or counterexamples where appropriate.
   (a) An agent that senses only partial information about the state cannot be perfectly rational.
   (b) There exist task environments in which no pure reflex agent can behave rationally.
   (c) There exists a task environment in which every agent is rational.
   (d) The input to an agent program is the same as the input to the agent function.
   (e) Every agent function is implementable by some program/machine combination.
   (f) Suppose an agent selects its action uniformly at random from the set of possible actions. There exists a deterministic task environment in which this agent is rational.
   (g) It is possible for a given agent to be perfectly rational in two distinct task environments.
   (h) Every agent is rational in an unobservable environment.
   (i) A perfectly rational poker-playing agent never loses.
2. For each of the following activities, give a PEAS description of the task environment and characterize it.
   (a) Playing soccer.
   (b) Exploring the subsurface oceans of Titan.
   (c) Shopping for used AI books on the Internet.
   (d) Playing a tennis match.
   (e) Practicing tennis against a wall.
   (f) Performing a high jump.
   (g) Knitting a sweater.
   (h) Bidding on an item at an auction.
3. Let us examine the rationality of various vacuum-cleaner agent functions.
   (a) Show that the simple vacuum-cleaner agent function described in Figure 2.3 is indeed rational.
   (b) Describe a rational agent function for the case in which each movement costs one point. Does the corresponding agent program require internal state?
   (c) Discuss possible agent designs for the cases in which clean squares can become dirty and the geography of the environment is unknown. Does it make sense for the agent to learn from its experience in these cases? If so, what should it learn? If not, why not?

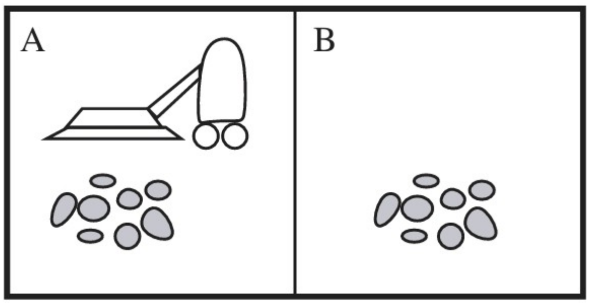


**Figure 2.2** A vacuum-cleaner world with just two locations.

| Percept sequence | Action |
|---|---|
| $[A, Clean]$ | $Right$ |
| $[A, Dirty]$ | $Suck$ |
| $[B, Clean]$ | $Left$ |
| $[B, Dirty]$ | $Suck$ |
| $[A, Clean], [A, Clean]$ | $Right$ |
| $[A, Clean], [A, Dirty]$ | $Suck$ |
| $\vdots$ | $\vdots$ |
| $[A, Clean], [A, Clean], [A, Clean]$ | $Right$ |
| $[A, Clean], [A, Clean], [A, Dirty]$ | $Suck$ |
| $\vdots$ | $\vdots$ |

**Figure 2.3** Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

4. Define in your own words the following terms: agent, agent function, agent program, rationality, autonomy, reflex agent, model-based agent, goal-based agent, utility-based agent, learning agent.

5. Consider a modified version of the vacuum environment in which the agent is penalized one point for each movement.
   (a) Can a simple reflex agent be perfectly rational for this environment? Explain.
   (b) What about a reflex agent with state? Design such an agent.
   (c) How do your answers to a and b change if the agent's percepts give it the clean/dirty status of every square in the environment?

6. Consider a modified version of the vacuum environment in which the geography of the environment – its extent, boundaries, and obstacles – is unknown, as is the initial dirt configuration. In this variant, the agent can go Up and Down as well as Left and Right.
   (a) Can a simple reflex agent be perfectly rational for this environment? Explain.
   (b) Can a simple reflex agent with a randomized agent function outperform a simple reflex agent? Design such an agent and measure its performance on several environments.
   (c) Can you design an environment in which your randomized agent will perform poorly? Show your results.

(d) Can a reflex agent with state outperform a simple reflex agent? Design such an agent and measure its performance on several environments. Can you design a rational agent of this type?

7. Repeat the previous exercise for the case in which the location sensor is replaced with a "bump" sensor that detects the agent's attempts to move into an obstacle or to cross the boundaries of the environment. Suppose the bump sensor stops working; how should the agent behave?

8. The vacuum environments in the preceding exercises have all been deterministic. Discuss possible agent programs for each of the following stochastic versions:

   (a) Murphy's law: twenty-five percent of the time, the Suck action fails to clean the floor if it is dirty and deposits dirt onto the floor if the floor is clean. How is your agent program affected if the dirt sensor gives the wrong answer 10% of the time?

   (b) Small children: At each time step, each clean square has a 10% chance of becoming dirty. Can you come up with a rational agent design for this case?

9. This exercise explores the differences between agent functions and agent programs.

   (a) Can there be more than one agent program that implements a given agent function? Give an example, or show why one is not possible.

   (b) Are there agent functions that cannot be implemented by any agent program?

   (c) Given a fixed machine architecture, does each agent program implement exactly one agent function?

   (d) Given an architecture with $n$ bits of storage, how many different possible agent programs are there?

   (e) Suppose we keep the agent program fixed but speed up the machine by a factor of two. Does that change the agent function?

## 2 Basic search

1. Give a complete problem formulation for each of the following problems. Choose a formulation that is precise enough to be implemented.

   (a) There are six glass boxes in a row, each with a lock. Each of the first five boxes holds a key unlocking the next box in line; the last box holds a banana. You have the key to the first box, and you want the banana.

   (b) You start with the sequence ABABAECCEC, or in general any sequence made from A, B, C, and E. You can transform this sequence using the following equalities: AC = E, AB = BC, BB = E, and E$x$ = $x$ for any $x$. For example, ABBC can be transformed into AEC, and then AC, and then E. Your goal is to produce the sequence E.

   (c) There is an $n \times n$ grid of squares, each square initially being either unpainted floor or a bottomless pit. You start standing on an unpainted floor square, and can either paint the square under you or move onto an adjacent unpainted floor square. You want the whole floor painted.

   (d) A container ship is in port, loaded high with containers. There 13 rows of containers, each 13 containers wide and 5 containers tall. You control a crane that can move to any location above the ship, pick up the container under it, and move it onto the dock. You want the ship unloaded.

   (e) Using only four colors, you have to color a planar map in such a way that no two adjacent regions have the same color.

   (f) A 3-foot-tall monkey is in a room where some bananas are suspended from the 8-foot ceiling. He would like to get the bananas. The room contains two stackable, movable, climbable 3-foot-high crates.

(g) You have a program that outputs the message "illegal input record" when fed a certain file of input records. You know that processing of each record is independent of the other records. You want to discover what record is illegal.

(h) You have three jugs, measuring 12 liters, 8 liters, and 3 liters, and a water faucet. You can fill the jugs up or empty them out from one to another or onto the ground. You need to measure out exactly one liter.

2. A robot has the task of navigating out of a maze. The robot starts in the center of the maze facing north. It can turn to face north, east, south, or west. The robot can also move forward a certain distance, although it will stop before hitting a wall.

   (a) Formulate this problem. How large is the state space?

   (b) In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now?

   (c) From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot's orientation now?

   (d) In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made.

3. Consider a $9 \times 9$ grid of squares, each of which can be colored red or blue. The grid is initially colored all blue, but you can change the color of any square any number of times. Imagining the grid divided into nine $3 \times 3$ sub-squares, you want each sub-square to be all one color but neighboring sub-squares to be different colors.

   (a) Formulate this problem in the straightforward way. Compute the size of the state space.

   (b) You need color a square only once. Reformulate, and compute the size of the state space. Would breadth-first graph search perform faster on this problem than on the one in (a)? How about iterative deepening tree search?

   (c) Given the goal, we need consider only colorings where each sub-square is uniformly colored. Reformulate the problem and compute the size of the state space.

   (d) How many solutions does this problem have?

   (e) Parts (b) and (c) successively abstracted the original problem (a). Can you give a translation from solutions in problem (c) into solutions in problem (b), and from solutions in problem (b) into solutions for problem (a)?

4. Explain why problem formulation must follow goal formulation.

5. Consider the problem of finding the shortest path between two points on a plane that has convex polygonal obstacles as shown in Figure 3.31. This is an idealization of the problem that a robot has to solve to navigate in a crowded environment.

   (a) Suppose the state space consists of all positions $(x, y)$ in the plane. How many states are there? How many paths are there to the goal?

   (b) Explain briefly why the shortest path from one polygon vertex to any other in the scene must consist of straight-line segments joining some of the vertices of the polygons. Define a good state space now. How large is this state space?

   (c) Define the necessary functions to implement the search problem, including an function that takes a vertex as input and returns a set of vectors, each of which maps the current vertex to one of the vertices that can be reached in a straight line. (Do not forget the neighbors on the same polygon.) Use the straight-line distance for the heuristic function.

   (d) Apply one or more of the algorithms in this chapter to solve a range of problems in the domain, and comment on their performance.

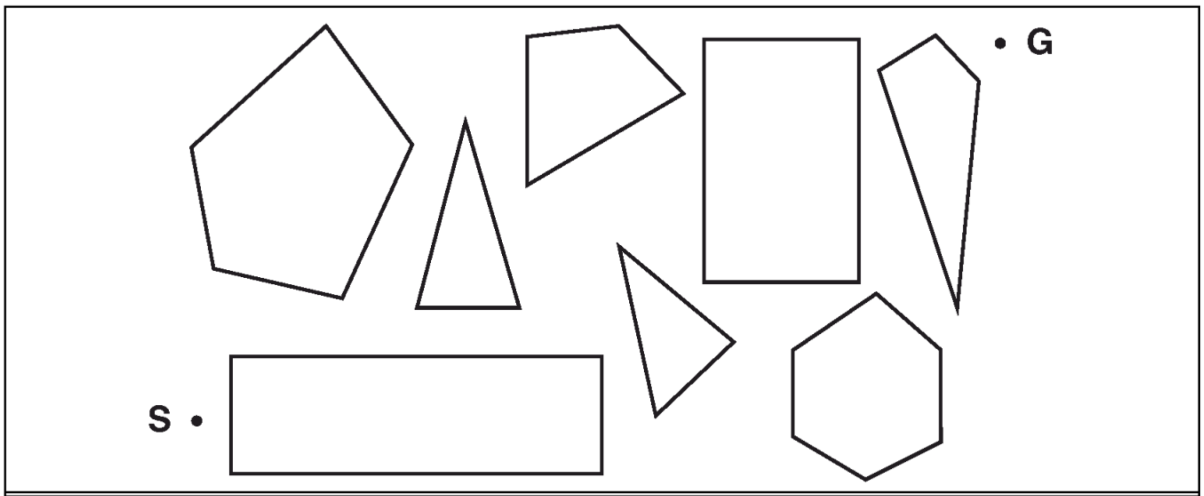


**Figure 3.31** A scene with polygonal obstacles. $S$ and $G$ are the start and goal states.

6. Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. [1]
   (a) Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution. Draw a diagram of the complete state space.
   (b) Solve the problem optimally using an appropriate search algorithm. Is it a good idea to check for repeated states?
   (c) Why do you think people have a hard time solving this puzzle, given that the state space is so simple?

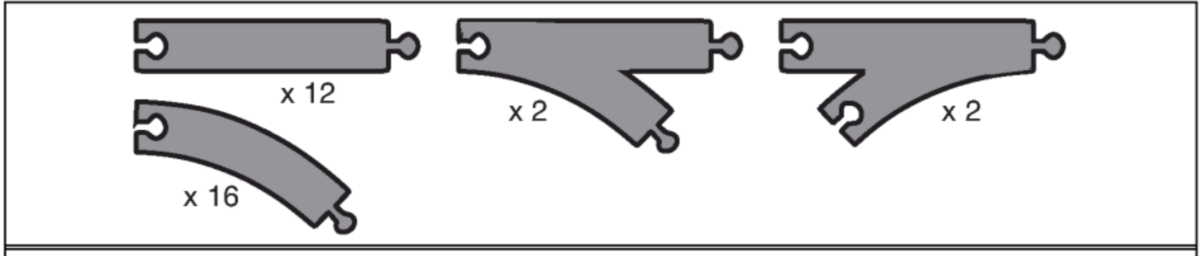


**Figure 3.32** The track pieces in a wooden railway set; each is labeled with the number of copies in the set. Note that curved pieces and "fork" pieces ("switches" or "points") can be flipped over so they can curve in either direction. Each curve subtends 45 degrees.

7. A basic wooden railway set contains the pieces shown in Figure 3.32. The task is to connect these pieces into a railway that has no overlapping tracks and no loose ends where a train could run off onto the floor.
   (a) Suppose that the pieces fit together exactly with no slack. Give a precise formulation of the task as a search problem.
   (b) Identify a suitable uninformed search algorithm for this task and explain your choice.
   (c) Explain why removing any one of the "fork" pieces makes the problem unsolvable.

[1] This problem is famous in AI because it was the subject of the first paper that approached problem formulation from an analytical viewpoint (Amarel, 1968).

(d) Give an upper bound on the total size of the state space defined by your formulation. (Hint: think about the maximum branching factor for the construction process and the maximum depth, ignoring the problem of overlapping pieces and loose ends. Begin by pretending that every piece is unique.)

8. Consider a state space where the start state is number 1 and each state $k$ has two successors: numbers $2k$ and $2k + 1$.

   (a) Draw the portion of the state space for states 1 to 15.

   (b) Suppose the goal state is 11. List the order in which nodes will be visited for breadth-first search, depth-limited search with limit 3, and iterative deepening search.

   (c) How well would bidirectional search work on this problem? What is the branching factor in each direction of the bidirectional search?

   (d) Does the answer to (c) suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?

   (e) Call the action going from $k$ to $2k$ Left, and the action going to $2k + 1$ Right. Can you find an algorithm that outputs the solution to this problem without any search at all?

9. An action such as Go(Sibiu) really consists of a long sequence of finer-grained actions: turn on the car, release the brake, accelerate forward, etc. Having composite actions of this kind reduces the number of steps in a solution sequence, thereby reducing the search time. Suppose we take this to the logical extreme, by making super-composite actions out of every possible sequence of Go actions. Then every problem instance is solved by a single super-composite action, such as Go(Sibiu)–Go(Rimnicu Vilcea)–Go(Pitesti)–Go(Bucharest). Explain how search would work in this formulation. Is this a practical approach for speeding up problem solving?

10. Define in your own words the following terms: state, state space, search tree, search node, goal, action, transition model, and branching factor.

11. What is the difference between a world state, a state description, and a search node? Why is this distinction useful?

12. Does a finite state space always lead to a finite search tree? How about a finite state space that is a tree? Can you be more precise about what types of state spaces always lead to finite search trees?

13. Consider the problem of find a path of links from one web URL to another. What is an appropriate search strategy? Is bidirectional search a good idea? Could a search engine be used to implement a predecessor function?

14. Describe a state space in which iterative deepening search performs much worse than depth-first search (for example, $O(n^2)$ vs $O(n)$).

## 3 Informed search

1. Suppose two friends live in different cities. On every turn, the two friends can simultaneously move to a neighboring city on the map. The amount of time needed to move from city $i$ to neighbor $j$ is equal to the road distance $d(i, j)$ between the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.

   (a) Write a detailed formulation for this search problem. (You will find it helpful to define some formal notation here.)

   (b) Let $D(i, j)$ be the straight-line distance between cities $i$ and $j$. Which of the following heuristic functions are admissible?

      i. $D(i, j)$

    ii. $2 \cdot D(i, j)$
    iii. $D(i, j)/2$

(c) Are there completely connected maps for which no solution exists?

(d) Are there maps in which all solutions require one friend to visit the same city twice?

2. Consider a regular infinite 2D grid. The start state is at the origin, (0,0), and the goal state is at $(m, n)$.

   (a) What is the branching factor $b$ in this state space?

   (b) How many distinct states are there at depth $k$ (for $k > 0$)?

   (c) What is the maximum number of nodes expanded by breadth-first tree search?

   (d) What is the maximum number of nodes expanded by breadth-first graph search?

   (e) Is $h = |u - m| + |v - n|$ an admissible heuristic for a state at $(u, v)$? Explain.

   (f) How many nodes are expanded by $A^*$ graph search using $h$?

   (g) Does $h$ remain admissible if some links are removed?

   (h) Does $h$ remain admissible if some links are added between nonadjacent states?

3. Consider the problem of moving $k$ knights from $k$ starting squares $s_1, \dots, s_k$ to $k$ goal squares $g_1, \dots, g_k$, on an unbounded chessboard, subject to the rule that no two knights can land on the same square at the same time. Each action consists of moving up to $k$ knights simultaneously. We would like to complete the maneuver in the smallest number of actions.

   (a) What is the maximum branching factor in this state space, expressed as a function of $k$?

   (b) Suppose $h_i$ is an admissible heuristic for the problem of moving knight $i$ to goal $g_i$ by itself. Which of the following heuristics are admissible for the $k$-knight problem? Of those, which is the best?

       i. $\sum_{i=1}^{n} h_i$
       ii. $\max h_1, \dots, h_n$.
       iii. $\min h_1, \dots, h_n$.

   (c) Repeat (b) for the case where you are allowed to move only one knight at a time.

4. A set of $n$ vehicles occupy squares $(1, 1)$ through $(n, 1)$ (i.e., the bottom row) of an $n \times n$ grid. The vehicles must be moved to the top row but in reverse order; so the vehicle $i$ that starts in $(i, 1)$ must end up in $(n - i + 1, n)$. On each time step, every one of the $n$ vehicles can move one square up, down, left, or right, or stay put.

   (a) Calculate the size of the state space as a function of $n$.

   (b) Calculate the branching factor as a function of $n$.

   (c) Suppose that vehicle $i$ is at $(x_i, y_i)$; write a nontrivial admissible heuristic $h_i$ for the number of moves it will require to get to its goal location $(n - i + 1, n)$, assuming no other vehicles are on the grid.

   (d) Which of the following heuristics are admissible for the problem of moving all $n$ vehicles to their destinations? Explain.

       i. $\sum_{i=1}^{n} h_i$
       ii. $\max h_1, \dots, h_n$.
       iii. $\min h_1, \dots, h_n$.

5. Can you draw some conclusions comparing the answers to the previous two exercises?

6. Trace the operation of $A^*$ search applied to the problem of getting to Bucharest from Lugoj using the straight-line distance heuristic. That is, show the sequence of nodes that the algorithm will consider and the $f$, $g$ and $h$ score for each node.

7. We saw that the straight-line distance heuristic leads greedy best-first search astray on the problem of going from Iasi to Fagaras. However, the heuristic is perfect on the opposite problem: going from Fagaras to Iasi. Are there problems for which the heuristic is misleading in both directions?

8. Prove that if a heuristic is consistent, it must be admissible. Construct an admissible heuristic that is not consistent.

9. The heuristic path algorithm (Pohl, 1977) is a best-first search in which the evaluation function is $f(n) = (2 - w)g(n) + wh(n)$. For what values of $w$ is this complete? For what values is it optimal, assuming that $h$ is admissible? What kind of search does this perform for $w = 0, w = 1, andw = 2$?

10. Devise a state space in which $A^*$ using returns a suboptimal solution with an $h$ function that is admissible but inconsistent.

11. Sometimes there is no good evaluation function for a problem but there is a good comparison method: a way to tell whether one node is better than another without assigning numerical values to either. Show that this is enough to do a best-first search. Is there an analog of $A^*$ for this setting?

12. Which of the following are true and which are false? Explain your answers.
    (a) Depth-first search always expands at least as many nodes as $A^*$ search with an admissible heuristic.
    (b) $h(n) = 0$ is an admissible heuristic for the 8-puzzle.
    (c) $A^*$ is of no use in robotics because percepts, states, and actions are continuous.
    (d) Breadth-first search is complete even if zero step costs are allowed.
    (e) Assume that a rook can move on a chessboard any number of squares in a straight line, vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the problem of moving the rook from square A to square B in the smallest number of moves

13. Prove each of the following statements, or give a counterexample:
    (a) Breadth-first search is a special case of uniform-cost search.
    (b) Depth-first search is a special case of best-first tree search.
    (c) Uniform-cost search is a special case of $A^*$ search.

## 4 Search in complex environments

1. Give the name of the algorithm that results from each of the following special cases:
    (a) Local beam search with $k = 1$.
    (b) Local beam search with one initial state and no limit on the number of states retained.
    (c) Simulated annealing with $T = 0$ at all times (and omitting the termination test).
    (d) Simulated annealing with $T = \infty$ at all times.
    (e) Genetic algorithm with population size $N = 1$.

2. In this exercise, we examine hill climbing in the context of robot navigation, using the environment in exercise 2.5 as an example.
    (a) Repeat exercise 2.5 using hill climbing. Does your agent ever get stuck in a local minimum? Is it possible for it to get stuck with convex obstacles?
    (b) Construct a nonconvex polygonal environment in which the agent gets stuck.
    (c) Modify the hill-climbing algorithm so that, instead of doing a depth-1 search to decide where to go next, it does a depth-$k$ search. It should find the best $k$-step path and do one step along it, and then repeat the process.

(d) Is there some $k$ for which the new algorithm is guaranteed to escape from local minima?

(e) Explain how LRTA enables the agent to escape from local minima in this case.

3. The And-Or-Graph-Search algorithm checks for repeated states only on the path from the root to the current state. Suppose that, in addition, the algorithm were to store every visited state and check against that list. Determine the information that should be stored and how the algorithm should use that information when a repeated state is found.

   *Hint:* You will need to distinguish at least between states for which a successful subplan was constructed previously and states for which no subplan could be found.

4. Explain precisely how to modify the And-Or-Graph-Search algorithm to generate a cyclic plan if no acyclic plan exists. You will need to deal with three issues: labeling the plan steps so that a cyclic plan can point back to an earlier part of the plan, modifying Or-Search so that it continues to look for acyclic plans after finding a cyclic plan, and augmenting the plan representation to indicate whether a plan is cyclic. Show how your algorithm works on (a) the slippery vacuum world, and (b) the slippery, erratic vacuum world.

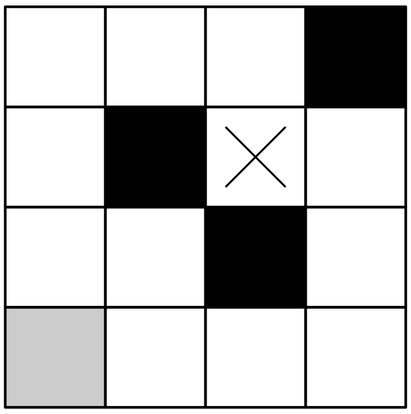


Figure 1: Yet another simple maze.

5. Consider the maze in Figure 1 and the problem of moving from the bottom left corner (shaded) to the treasure, marked with X. Draw the full And-Or search tree for this problem, assuming that the possible actions are tried in the order *Up*, *Right*, *Left*, *Down*, and use it to write a conditional plan that solves the problem.

6. Consider the sensorless version of the erratic vacuum world shown in Figure 4.14.

   (a) Draw the belief-state space reachable from the initial belief state 1,2,3,4,5,6,7,8, and explain why the problem is unsolvable.

   (b) Draw the belief-state space reachable from the initial belief state 1,3,5,7, and explain why the problem is unsolvable.

7. In Section 4.4.1 we introduced belief states to solve sensorless search problems. A sequence of actions solves a sensorless problem if it maps every physical state in the initial belief state $b$ to a goal state. Suppose the agent knows $h^*(s)$, the true optimal cost of solving the physical state $s$ in the fully observable problem, for every state $s$ in b. Find an admissible heuristic $h(b)$ for the sensorless problem in terms of these costs, and prove its admissibilty. Comment on the accuracy of this heuristic on the sensorless vacuum problem of Figure 4.14. How well does A* perform?

8. This exercise explores subset–superset relations between belief states in sensorless or partially observable environments.

   (a) Prove that if an action sequence is a solution for a belief state $b$, it is also a solution for any subset of $b$. Can anything be said about supersets of $b$?

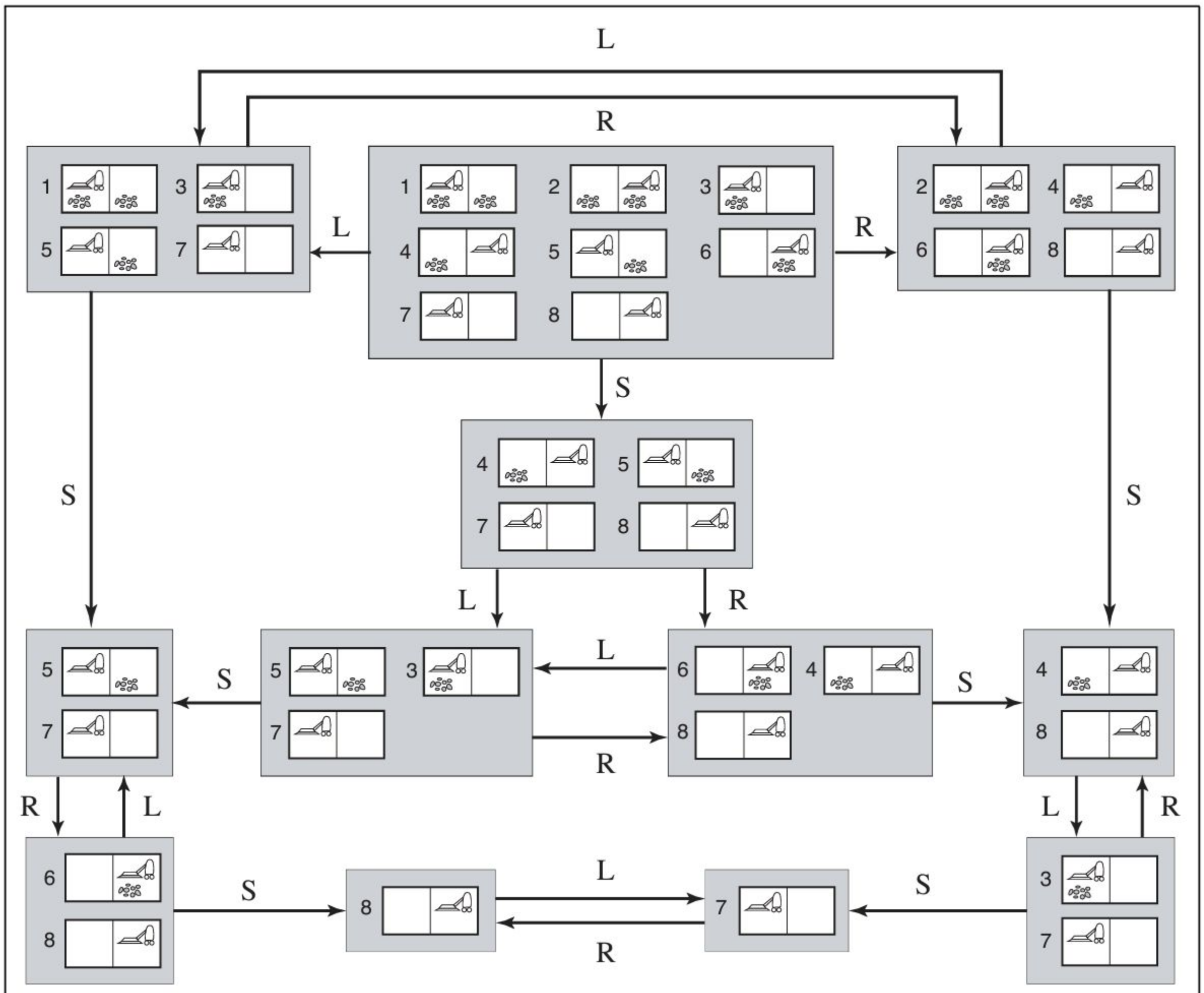


**Figure 4.14** The reachable portion of the belief-state space for the deterministic, sensorless vacuum world. Each shaded box corresponds to a single belief state. At any given point, the agent is in a particular belief state but does not know which physical state it is in. The initial belief state (complete ignorance) is the top center box. Actions are represented by labeled links. Self-loops are omitted for clarity.

(b) Explain in detail how to modify graph search for sensorless problems to take advantage of your answers in (a).

(c) Explain in detail how to modify and–or search for partially observable problems, beyond the modifications you describe in (b).

9. Consider the following problem: there are two coins in a table, both covered (so it is unknown which side is facing up). The only possible action at each stage is: choose one coin, toss it, check the result, and cover it again. The goal is to have both coins facing the same way (heads or tails).

   Draw the complete And-Or search tree for this problem and use it to make a contingency plan that will solve it.

10. Suppose that an agent is in a $3 \times 3$ maze environment like the one shown in Figure 4.19. The agent knows that its initial location is (1,1), that the goal is at (3,3), and that the actions *Up*, *Down*, *Left* and *Right* have their usual effects unless blocked by a wall. The agent does not know where the internal walls are. In any given state, the agent perceives the set of legal actions; it can also tell whether the state is one it has visited before.

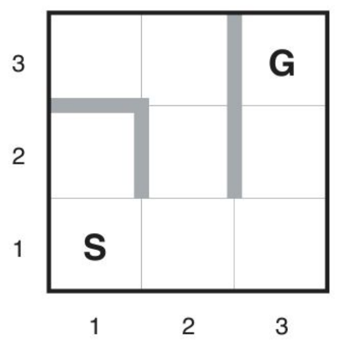


**Figure 4.19** A simple maze problem. The agent starts at $S$ and must reach $G$ but knows nothing of the environment.

(a) Explain how this online search problem can be viewed as an offline search in belief-state space, where the initial belief state includes all possible environment configurations. How large is the initial belief state? How large is the space of belief states?

(b) How many distinct percepts are possible in the initial state?

(c) Describe the first few branches of a contingency plan for this problem. How large (roughly) is the complete plan?

Notice that this contingency plan is a solution for every possible environment fitting the given description. Therefore, interleaving of search and execution is not strictly necessary even in unknown environments.

11. Like depth-first search, online depth-first search is incomplete for reversible state spaces with infinite paths. For example, suppose that states are points on the infinite two-dimensional grid and actions are unit vectors $(1, 0), (0, 1), (-1, 0), (0, -1)$, tried in that order. Show that online depth-first search starting at $(0, 0)$ will not reach $(1, -1)$. Suppose the agent can observe, in addition to its current state, all successor states and the actions that would lead to them. Write an algorithm that is complete even for bidirected state spaces with infinite paths. What states does it visit in reaching $(1, -1)$?

## 5 Constraint satisfaction problems

1. Give precise formulations for each of the following as constraint satisfaction problems.

   (a) Rectilinear floor-planning: find non-overlapping places in a large rectangle for a number of smaller rectangles.

   (b) Class scheduling: There is a fixed number of professors and classrooms, a list of classes to be offered, and a list of possible time slots for classes. Each professor has a set of classes that he or she can teach.

   (c) Hamiltonian tour: given a network of cities connected by roads, choose an order to visit all cities in a country without repeating any.

2. Consider the problem of placing $k$ knights on an $n \times n$ chessboard such that no two knights are attacking each other, where $k$ is given and $k \leq n^2$.

   (a) Formulate this problem as a constraint satisfaction problem. In your formulation, what are the variables?

   (b) What are the possible values of each variable?

   (c) What sets of variables are constrained, and how?

(d) Now consider the problem of putting as many knights as possible on the board without any attacks. Explain how to solve this with local search by defining appropriate ACTIONS and RESULT functions and a sensible objective function.

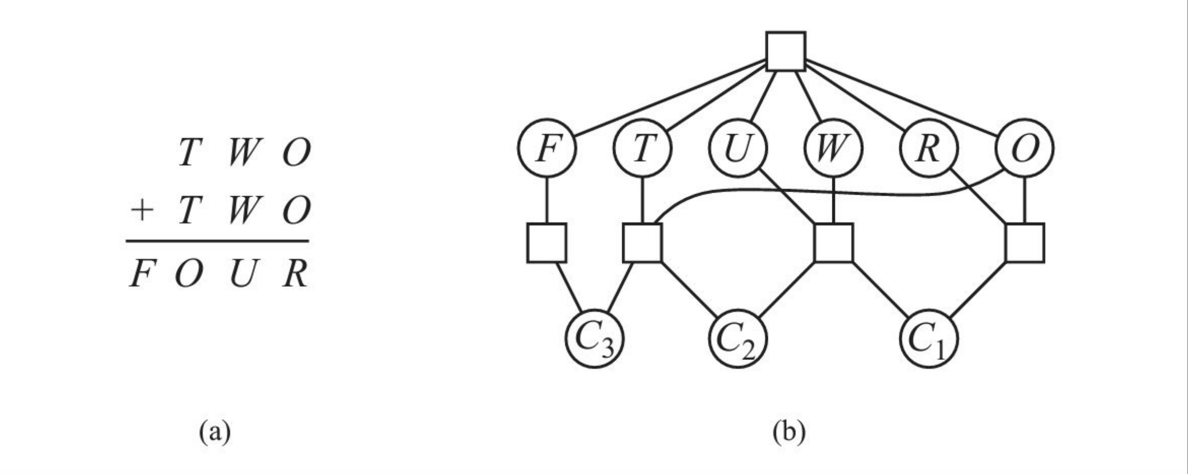


**Figure 6.2** (a) A cryptarithmetic problem. Each letter stands for a distinct digit; the aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct, with the added restriction that no leading zeroes are allowed. (b) The constraint hypergraph for the cryptarithmetic problem, showing the *Alldiff* constraint (square box at the top) as well as the column addition constraints (four square boxes in the middle). The variables $C_1$, $C_2$, and $C_3$ represent the carry digits for the three columns.

3. Solve the cryptarithmetic problem in Figure 6.2 by hand, using the strategy of backtracking with forward checking and the MRV (Most restricted variable) and least-constraining-value heuristics.

4. Consider the problem of *constructing* (not solving) crossword puzzles fitting words into a rectangular grid. The grid, which is given as part of the problem, specifies which squares are blank and which are shaded. Assume that a list of words (i.e., a dictionary) is provided and that the task is to fill in the blank squares by using any subset of the list. Formulate this problem precisely in two ways:

   (a) As a general search problem. Choose an appropriate search algorithm and specify a heuristic function. Is it better to fill in blanks one letter at a time or one word at a time?

   (b) As a constraint satisfaction problem. Should the variables be words or letters?

   Which formulation do you think will be better? Why?

5. Show how a single ternary constraint such as "$A + B = C$" can be turned into three binary constraints by using an auxiliary variable. You may assume finite domains.

   *Hint:* Consider a new variable that takes on values that are pairs of other values, and consider constraints such as "$X$ is the first element of the pair $Y$".) Next, show how constraints with more than three variables can be treated similarly. Finally, show how unary constraints can be eliminated by altering the domains of variables. This completes the demonstration that any CSP can be transformed into a CSP with only binary constraints.

6. Consider the following logic puzzle: In five houses, each with a different color, live five persons of different nationalities, each of whom prefers a different brand of candy, a different drink, and a different pet. Given the following facts, the questions to answer are "Where does the zebra live, and in which house do they drink water?"

   - The Englishman lives in the red house.

- The Spaniard owns the dog.
- The Norwegian lives in the first house on the left.
- The green house is immediately to the right of the ivory house.
- The man who eats Hershey bars lives in the house next to the man with the fox.
- Kit Kats are eaten in the yellow house.
- The Norwegian lives next to the blue house.
- The Smarties eater owns snails.
- The Snickers eater drinks orange juice.
- The Ukrainian drinks tea.
- The Japanese eats Milky Ways.
- Kit Kats are eaten in a house next to the house where the horse is kept.
- Coffee is drunk in the green house.
- Milk is drunk in the middle house.

Discuss different representations of this problem as a CSP. Why would one prefer one representation over another?

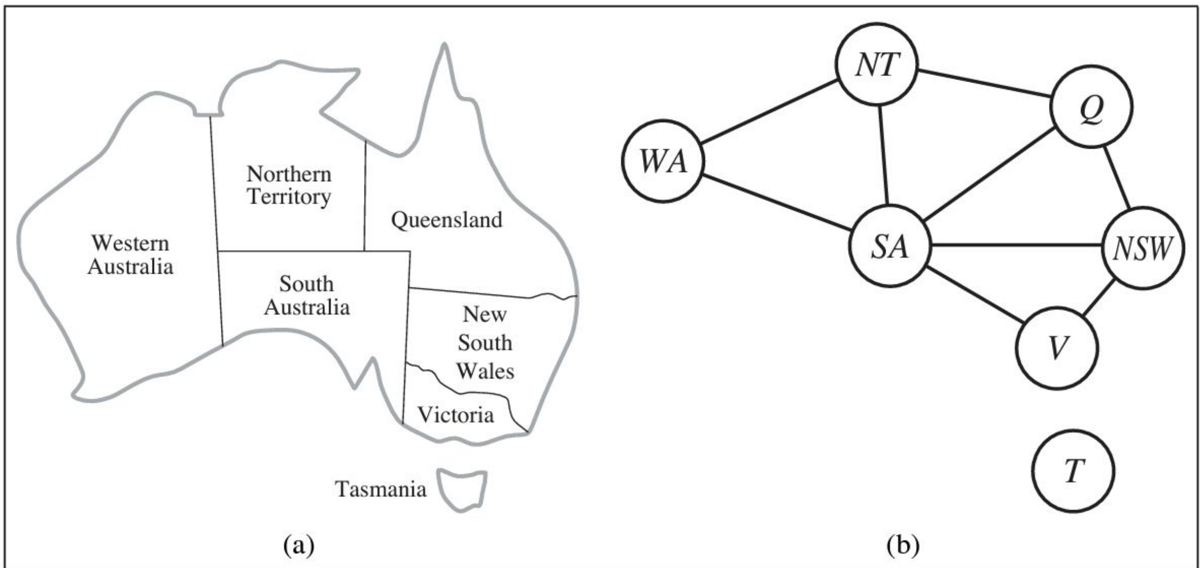


**Figure 6.1** (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

7. How many solutions are there for the map-coloring problem in Figure 6.1? How many solutions if four colors are allowed? Two colors?

8. Define in your own words the terms constraint, backtracking search, arc consistency, backjumping, min-conflicts, and cycle cutset.

9. Use the AC-3 algorithm to show that arc consistency can detect the inconsistency of the partial assignment $\{WA = \textit{green}, V = \textit{red}\}$ for the problem shown in Figure 6.1.

10. Consider the graph with 8 nodes $A_1$, $A_2$, $A_3$, $A_4$, $H$, $T$, $F_1$, $F_2$. Each node $A_i$ is connected to $A_{i+1}$ and to $H$; $H$ is connected to $T$; and $T$ is connected to each $F_i$. Find a 3-coloring of this graph by hand using backtracking with conflict-directed backjumping, the variable order $A_1$, $H$, $A_4$, $F_1$, $A_2$, $F_2$, $A_3$, $T$, and the value order *red*, *green*, *blue*.

11. Explain why, in a CSP search, it is a good heuristic to choose the variable that is most constrained but the value that is least constraining.

12. Consider the problem of tiling a surface (completely and exactly covering it) with $n$ dominoes ($2 \times 1$ rectangles). The surface is an arbitrary edge-connected (i.e., adjacent along an edge, not just a corner) collection of $2n 1 \times 1$ squares (e.g., a checkerboard, a checkerboard with some squares missing, a $10 \times 1$ row of squares, etc.).

    (a) Formulate this problem precisely as a CSP where the dominoes are the variables.

    (b) Formulate this problem precisely as a CSP where the squares are the variables, keeping the state space as small as possible.

    *Hint:* does it matter which particular domino goes on a given pair of squares?

    (c) Construct a surface consisting of 6 squares such that your CSP formulation from part (b) has a tree-structured constraint graph.

    (d) Describe exactly the set of solvable instances that have a *tree-structured* constraint graph.

13. AC-3 puts back on the queue every arc $(X_k, X_i)$ whenever any value is deleted from the domain of $X_i$, even if each value of $X_k$ is consistent with several remaining values of $X_i$. Suppose that, for every arc $(X_k, X_i)$, we keep track of the number of remaining values of $X_i$ that are consistent with each value of $X_k$. Explain how to update these numbers efficiently and hence show that arc consistency can be enforced in total time $O(n^2 d^2)$.

## 6 Adversarial search and games

1. Suppose you have an oracle, OM($s$), that correctly predicts the opponent's move in any state $s$. Using this, formulate the definition of a game as a (single-agent) search problem. Describe an algorithm for finding the optimal move.

2. Describe and implement state descriptions, move generators, terminal tests, utility functions, and evaluation functions for one or more of the following stochastic games: Monopoly, Scrabble, bridge play with a given contract, or Texas hold'em poker.

3. Discuss how well the standard approach to game playing would apply to games such as tennis, pool, and croquet, which take place in a continuous physical state space.

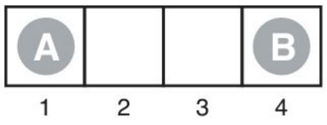


**Figure 5.17** The starting position of a simple game. Player $A$ moves first. The two players take turns moving, and each player must move his token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then a player may jump over the opponent to the next open space if any. (For example, if $A$ is on 3 and $B$ is on 2, then $A$ may move back to 1.) The game ends when one player reaches the opposite end of the board. If player $A$ reaches space 4 first, then the value of the game to $A$ is $+1$; if player $B$ reaches space 1 first, then the value of the game to $A$ is $-1$.

4. Consider the two-player game described in Figure 5.17.

    (a) Draw the complete game tree, using the following conventions:

    - Write each state as $(s_A, s_B)$, where $s_A$ and $s_B$ denote the token locations.
    - Put each terminal state in a square box and write its game value in a circle.

- Put loop states (states that already appear on the path to the root) in double square boxes. Since their value is unclear, annotate each with a "?" in a circle.

(b) Now mark each node with its backed-up minimax value (also in a circle). Explain how you handled the "?" values and why.

(c) Explain why the standard minimax algorithm would fail on this game tree and briefly sketch how you might fix it, drawing on your answer to (b). Does your modified algorithm give optimal decisions for all games with loops?

(d) This 4-square game can be generalized to $n$ squares for any $n > 2$. Prove that $A$ wins if $n$ is even and loses if $n$ is odd.

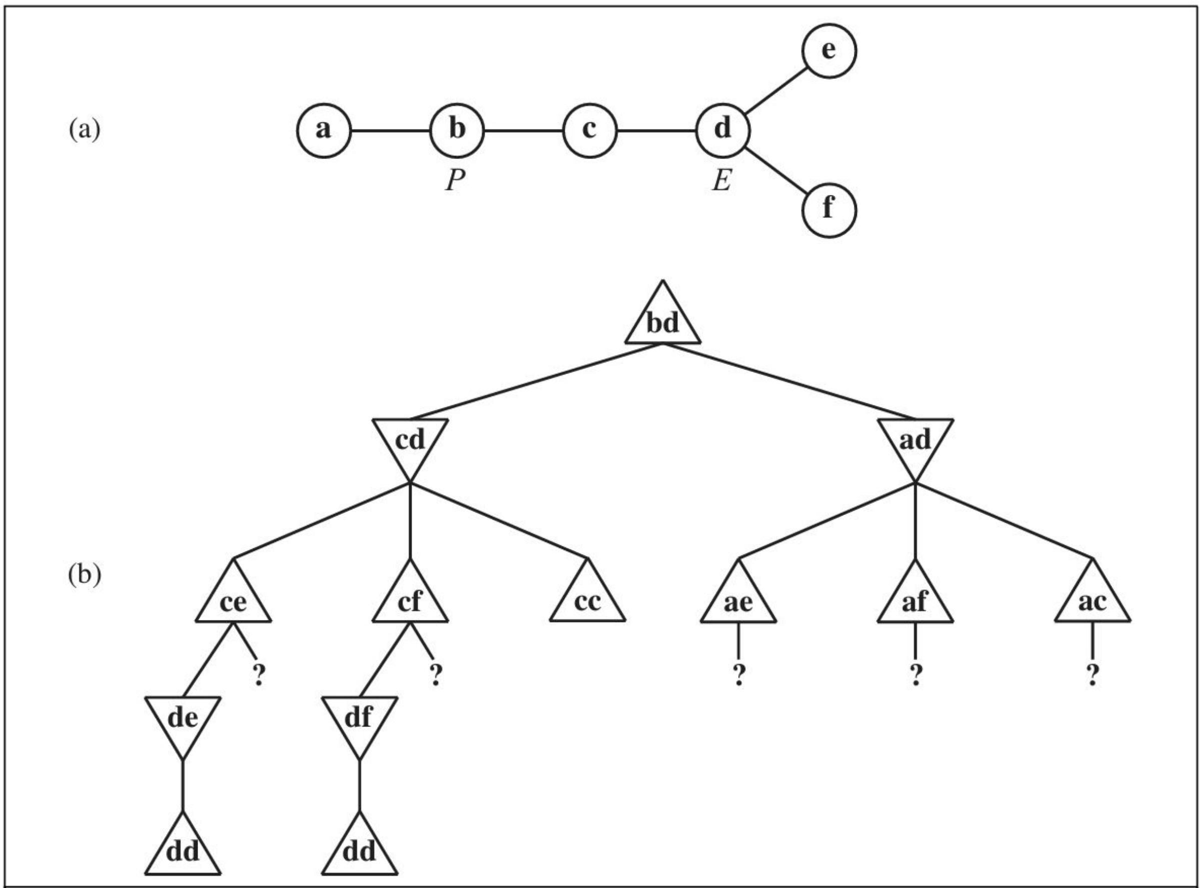


**Figure 5.16** (a) A map where the cost of every edge is 1. Initially the pursuer $P$ is at node **b** and the evader $E$ is at node **d**. (b) A partial game tree for this map. Each node is labeled with the $P, E$ positions. $P$ moves first. Branches marked "?" have yet to be explored.

5. Consider the following game, played on the map in Figure 5.16 (a). There are two players, a pursuer (P) and an evader (E); the pursuer wants to catch the evader, and the evader wants to avoid being caught. We assume now that the players take turns moving. The game ends only when the players are on the same node; the terminal payoff to the pursuer is minus the total time taken. (The evader "wins" by never losing.) An example is shown in Figure 5-16 (b).

   (a) Copy the game tree and mark the values of the terminal nodes.

   (b) Next to each internal node, write the strongest fact you can infer about its value (a number, one or more inequalities such as "$\geq 14$", or a "?").

   (c) Beneath each question mark, write the name of the node reached by that branch.

(d) Explain how a bound on the value of the nodes in (c) can be derived from consideration of shortest-path lengths on the map, and derive such bounds for these nodes. Remember the cost to get to each leaf as well as the cost to solve it.

(e) Now suppose that the tree as given, with the leaf bounds from (d), is evaluated from left to right. Circle those "?" nodes that would not need to be expanded further, given the bounds from part (d), and cross out those that need not be considered at all.

(f) Can you prove anything in general about who wins the game on a map that is a tree?

6. Prove the following assertion: For every game tree, the utility obtained by max using minimax decisions against a suboptimal min will never be lower than the utility obtained playing against an optimal min. Can you come up with a game tree in which max can do still better using a suboptimal strategy against a suboptimal min?

7. This problem exercises the basic concepts of game playing, using tic-tac-toe as an example. We define $X_n$ as the number of rows, columns, or diagonals with exactly $n$ $X$s and no $O$s. Similarly, $O_n$ is the number of rows, columns, or diagonals with just $n$ $O$s. The utility function assigns $+1$ to any position with $X_3 = 1$ and $-1$ to any position with $O_3 = 1$. All other terminal positions have utility 0. For nonterminal positions, we use a linear evaluation function defined as $\text{Eval}(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$.

   (a) Approximately how many possible games of tic-tac-toe are there?

   (b) Show the whole game tree starting from an empty board down to depth 2 (i.e., one $X$ and one $O$ on the board), taking symmetry into account.

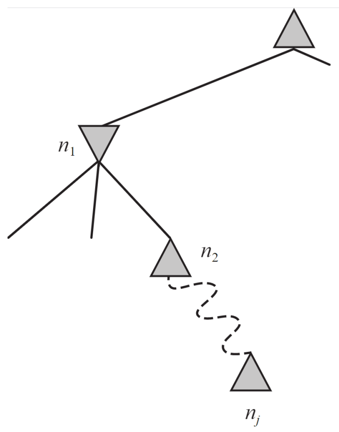
   (c) Mark on your tree the evaluations of all the positions at depth 2.

   (d) Using the minimax algorithm, mark on your tree the backed-up values for the positions at depths 1 and 0, and use those values to choose the best starting move.

   (e) Circle the nodes at depth 2 that would not be evaluated if alpha–beta pruning were applied, assuming the nodes are generated in the optimal order for alpha–beta pruning.

8. Describe how the minimax and alpha–beta algorithms change for two-player, non-zero-sum games in which each player has a distinct utility function and both utility functions are known to both players. If there are no constraints on the two terminal utilities, is it possible for any node to be pruned by alpha–beta? What if the player's utility functions on any state differ by at most a constant $k$, making the game almost cooperative? What if the player's utility functions on any state sum to a number between constants $-k$ and $k$, making the game almost zero-sum?

9. Develop a formal proof of correctness for alpha–beta pruning. To do this, consider the situation shown in the figure below.



The question is whether to prune node $n_j$, which is a max-node and a descendant of node $n_1$. The basic idea is to prune it if and only if the minimax value of $n_1$ can be shown to be independent of the value of $n_j$.

(a) Node $n_1$ takes on the minimum value among its children: $n_1 = \min(n_2, n_{2,1}, \ldots, n_{2,b_2})$. Find a similar expression for $n_2$ and hence an expression for $n_1$ in terms of $n_j$.

(b) Let $l_i$ be the minimum (or maximum) value of the nodes to the left of node $n_i$ at depth $i$, whose minimax value is already known. Similarly, let $r_i$ be the minimum (or maximum) value of the unexplored nodes to the right of $n_i$ at depth $i$. Rewrite your expression for $n_1$ in terms of the $l_i$ and $r_i$ values.

(c) Now reformulate the expression to show that in order to affect $n_1$, $n_j$ must not exceed a certain bound derived from the $l_i$ values.

(d) Repeat the process for the case where $n_j$ is a min-node.

10. Prove that the alpha–beta algorithm takes time $O(b^{m/2})$ with optimal move ordering, where $m$ is the maximum depth of the game tree.

11. Prove that with a positive linear transformation of leaf values (i.e., transforming a value $x$ to $ax+b$ where $a > 0$), the choice of move remains unchanged in a game tree.

# 7 Logical Agents

1. Suppose the agent in the Wumpus world has explored three squares, having perceived nothing in [1,1], a breeze in [2,1], and a stench in [1,2], and is now concerned with the contents of [1,3], [2,2], and [3,1]. Each of these can contain a pit, and at most one can contain a wumpus. Following the example in the book, construct the set of possible worlds. (You should find 32 of them.) Mark the worlds in which the KB is true and those in which each of the following sentences is true:

$$\begin{aligned}
\alpha_1 &= \text{“There is no pit in [1,3].”} \\
\alpha_2 &= \text{“There is no pit in [2,2].”} \\
\alpha_3 &= \text{“There is no pit in [3,1].”}
\end{aligned}$$

Use this information to show that $KB \models \alpha_1$, $KB \models \alpha_2$ and $KB \not\models \alpha_3$.

2. Consider a vocabulary with only four propositions, $A$, $B$, $C$ and $D$. How many models are there for the following sentences?

   (a) $B \lor C$

   (b) $\neg A \lor \neg B \lor \neg C \lor \neg D$

   (c) $(A \to B) \land A \land \neg B \land C \land D$.

3. Which of the following are correct?

   (a) $(A \lor B) \land \neg(A \to B)$ is satisfiable.

   (b) $(A \leftrightarrow B) \land (\neg A \lor B)$ is satisfiable.

   (c) $A \land B \models A \leftrightarrow B$

   (d) $A \leftrightarrow B \models A \lor B$

   (e) $A \leftrightarrow B \models \neg A \lor B$

   (f) $(A \land B) \to C \models (A \to C) \lor (B \to C)$

   (g) $(A \lor B) \land (\neg C \lor \neg D \lor E) \models A \lor B$

   (h) $(A \lor B) \land (\neg C \lor \neg D \lor E) \models (A \lor B) \land (\neg D \lor E)$

   (i) $\mathit{False} \models \mathit{True}$

   (j) $\mathit{True} \models \mathit{False}$

   (k) $(C \lor (\neg A \land \neg B))$ is equivalent to $((A \to C) \land (B \to C))$

   (l) $(A \leftrightarrow B) \leftrightarrow C$ has the same number of models as $A \leftrightarrow B$ for any fixed set of proposition symbols that include $A$, $B$ and $C$.

4. Prove each of the following assertions:
   (a) $\alpha$ is valid if and only if $\mathit{True} \models \alpha$.
   (b) For any $\alpha$, $\mathit{False} \models \alpha$.
   (c) $\alpha \models \beta$ if and only if $\alpha \rightarrow \beta$ is valid.
   (d) $\alpha \equiv \beta$ if and only if $\alpha \leftrightarrow \beta$ is valid.
   (e) $\alpha \models \beta$ if and only if $\alpha \wedge \neg\beta$ is unsatisfiable.

5. Prove, or find a counterexample to, each of the following assertions:
   (a) If $\alpha \models \gamma$ or $\beta \models \gamma$ (or both) then $\alpha \wedge \beta \models \gamma$.
   (b) If $\alpha \wedge \beta \models \gamma$ then $\alpha \models \gamma$ or $\beta \models \gamma$ (or both).
   (c) If $\alpha \models \beta \vee \gamma$ then $\alpha \models \beta$ or $\alpha \models \gamma$ (or both).

6. We have defined four binary logical connectives.
   (a) Are there any others that might be useful?
   (b) How many binary connectives can there be?
   (c) Why are some of them not very useful?

7. Consider the following information.
   - If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal.
   - If the unicorn is immortal or a mammal, then it is horned.
   - The unicorn is magical if it is horned.

   (a) Can you prove that the unicorn is mythical? How about magical? Horned?
   (b) Write the facts in propositional logic and argue formally for your previous answer.
   (c) Check your answers using tableaux.
   (d) Translate the formulas you wrote into CNF and apply resolution to confirm your results.

8. According to some political pundits, a person who is radical ($R$) is electable ($E$) if they are conservative ($C$), but otherwise is not electable.
   (a) Which of the following are correct representations of this assertion?
      i. $(R \wedge E) \leftrightarrow C$
      ii. $R \rightarrow (E \leftrightarrow C)$
      iii. $R \rightarrow ((C \rightarrow E) \vee \neg E)$
   (b) Which of the sentences in (a) can be expressed in Horn form?

9. Decide whether each of the following sentences is valid, unsatisfiable, or neither. Check your conclusions using tableaux.
   (a) $\mathit{Smoke} \rightarrow \mathit{Smoke}$
   (b) $\mathit{Smoke} \rightarrow \mathit{Fire}$
   (c) $(\mathit{Smoke} \rightarrow \mathit{Fire}) \rightarrow (\neg\mathit{Smoke} \rightarrow \neg\mathit{Fire})$
   (d) $\mathit{Smoke} \vee \mathit{Fire} \vee \neg\mathit{Fire}$
   (e) $((\mathit{Smoke} \wedge \mathit{Heat}) \rightarrow \mathit{Fire}) \leftrightarrow ((\mathit{Smoke} \rightarrow \mathit{Fire}) \vee (\mathit{Heat} \rightarrow \mathit{Fire}))$
   (f) $(\mathit{Smoke} \rightarrow \mathit{Fire}) \rightarrow ((\mathit{Smoke} \wedge \mathit{Heat}) \rightarrow \mathit{Fire})$
   (g) $\mathit{Big} \vee \mathit{Dumb} \vee (\mathit{Big} \rightarrow \mathit{Dumb})$

10. Consider the following sentence:

$$((\mathit{Food} \rightarrow \mathit{Party}) \vee (\mathit{Drinks} \rightarrow \mathit{Party})) \rightarrow ((\mathit{Food} \wedge \mathit{Drinks}) \rightarrow \mathit{Party})\,.$$

(a) Determine, using the semantics, whether this sentence is valid, satisfiable (but not valid), or unsatisfiable.

(b) Check your answer using tableaux.

(c) Convert the left- and righthandsides of the main implication into CNF, showing each step, and explain how the results confirm your answer to (a).

(d) Prove your answer to (a) using resolution.

11. This exercise looks into the relationship between clauses and implication sentences.

(a) Show that the clause $(\neg P_1 \vee \ldots \vee \neg P_n \vee Q)$ is logically equivalent to the implication sentence $(P_1 \wedge \ldots \wedge P_n) \rightarrow Q$.

(b) Show that every clause (regardless of the number of positive literals) can be written in the form $(P_1 \wedge \ldots \wedge P_n) \rightarrow (Q_1 \vee \ldots \vee Q_n)$, where $P$s and $Q$s are proposition symbols. A knowledge base consisting of such sentences is in implicative normal form or Kowalski form.

(c) Write down the full resolution rule for sentences in implicative normal form.

12. Explain why every nonempty propositional clause, by itself, is satisfiable. Prove rigorously that every set of five 3-SAT clauses is satisfiable, provided that each clause mentions exactly three distinct variables. What is the smallest set of such clauses that is unsatisfiable? Construct such a set.

13. A propositional 2-CNF expression is a conjunction of clauses, each containing exactly 2 literals, e.g.,

$$(A \vee B) \wedge (\neg A \vee C) \wedge (\neg B \vee D) \wedge (\neg C \vee G) \wedge (\neg D \vee G)$$

(a) Prove that the above sentence entails $G$ using:

i. resolution;

ii. forward chaining;

iii. backward chaining.

(b) Two clauses are semantically distinct if they are not logically equivalent. How many semantically distinct 2-CNF clauses can be constructed from $n$ proposition symbols?

(c) Using your answer to (b), prove that propositional resolution always terminates in time polynomial in $n$ given a 2-CNF sentence containing no more than $n$ distinct symbols.

(d) Explain why your argument in (c) does not apply to 3-CNF.

14. Is a randomly generated 4-CNF sentence with $n$ symbols and $m$ clauses more or less likely to be satisfiable than a randomly generated 3-CNF sentence with $n$ symbols and $m$ clauses? Explain.

15. Prove each of the following assertions:

(a) Every pair of propositional clauses either has no resolvents, or all their resolvents are logically equivalent.

(b) There is no clause that, when resolved with itself, yields (after factoring) the clause $\neg P \vee \neg Q$.

(c) If a propositional clause $C$ can be resolved with a copy of itself, it must be logically equivalent to *True*.

16. Convert the following set of sentences to clausal form.

- $A \leftrightarrow (B \vee E)$.
- $E \rightarrow D$.
- $C \wedge F \rightarrow \neg B$

- $E \to B$
- $B \to F$
- $B \to C$

(a) Give a trace of the execution of DPLL on the conjunction of these clauses.

(b) Use resolution to prove that these clauses entail $\neg A \wedge \neg B$.

17. This question considers representing satisfiability (SAT) problems as CSPs.

    (a) Draw the constraint graph corresponding to the SAT problem
    $$(\neg X_1 \vee X_2) \wedge (\neg X_2 \vee X_3) \wedge \ldots \wedge (\neg X_{n-1} \vee X_n)$$
    for the particular case $n = 5$.

    (b) How many solutions are there for this general SAT problem as a function of $n$?

    (c) Suppose we apply Backtracking to find all solutions to a SAT CSP of the type given in (a). (To find all solutions to a CSP, we simply modify the basic algorithm so it continues searching after each solution is found.) Assume that variables are ordered $X_1, \ldots, X_n$ and *false* is ordered before *true*. How much time will the algorithm take to terminate as a function of $n$?

    (d) We know that SAT problems in Horn form can be solved in linear time by forward chaining (unit propagation). We also know that every tree-structured binary CSP with discrete, finite domains can be solved in time linear in the number of variables. Are these two facts connected? Discuss.

18. Minesweeper, the well-known computer game, is closely related to the wumpus world. A minesweeper world is a rectangular grid of $N$ squares with $M$ invisible mines scattered among them. Any square may be probed by the agent; instant death follows if a mine is probed. Minesweeper indicates the presence of mines by revealing, in each probed square, the number of mines that are directly or diagonally adjacent. The goal is to probe every unmined square.

    (a) Let $X_{i,j}$ be true iff square $[i, j]$ contains a mine. Write down the assertion that exactly two mines are adjacent to $[1, 1]$ as a sentence involving some logical combination of $X_{i,j}$ propositions.

    (b) Generalize your assertion from (a) by explaining how to construct a CNF sentence asserting that $k$ of $n$ neighbors contain mines.

    (c) Explain precisely how an agent can use DPLL to prove that a given square does (or does not) contain a mine, ignoring the global constraint that there are exactly $M$ mines in all.

    (d) Suppose that the global constraint is constructed from your method from part (b). How does the number of clauses depend on $M$ and $N$? Suggest a way to modify DPLL so that the global constraint does not need to be represented explicitly.

    (e) Are any conclusions derived by the method in part (c) invalidated when the global constraint is taken into account?

    (f) Give examples of configurations of probe values that induce long-range dependencies such that the contents of a given unprobed square would give information about the contents of a far-distant square. (Hint: consider an $N \times 1$ board.)

19. Write a successor-state axiom for the Locked predicate, which applies to doors, assuming the only actions available are Lock and Unlock.

20. Suppose an agent inhabits a world with two states, $S$ and $\neg S$, and can do exactly one of two actions, $a$ and $b$. Action $a$ does nothing and action $b$ flips from one state to the other. Let $S^t$ be the proposition that the agent is in state $S$ at time $t$, and let $a^t$ be the proposition that the agent does action $a$ at time $t$ (similarly for $b^t$).

    (a) Write a successor-state axiom for $S^{t+1}$.

    (b) Convert the sentence in (a) into CNF.

    (c) Show a resolution refutation proof that if the agent is in $\neg S$ at time $t$ and does $a$, it will still be in $\neg S$ at time $t + 1$.

## 8 First Order Logic

1. A logical knowledge base represents the world using a set of sentences with no explicit structure. An *analogical* representation, on the other hand, has a physical structure that corresponds directly to the structure of the thing represented.

   For example, a road map of a country is an analogical representation of facts about the country -— it represents facts with a map language. The two-dimensional structure of the map corresponds to the two-dimensional surface of the area.

   (a) Give five examples of *symbols* in the map language.

   (b) An *explicit* sentence is a sentence that the creator of the representation actually writes down. An *implicit* sentence is a sentence that results from explicit sentences because of properties of the analogical representation. Give three examples each of *implicit* and *explicit* sentences in the map language.

   (c) Give three examples of facts about the physical structure of your country that cannot be represented in the map language.

   (d) Give two examples of facts that are much easier to express in the map language than in first-order logic.

   (e) Give two other examples of useful analogical representations. What are the advantages and disadvantages of each of these languages?

2. In each of the following we give an English sentence and a number of candidate logical expressions written in a suggestive vocabulary. For each of the logical expressions, state whether it (1) correctly expresses the English sentence; (2) is syntactically invalid and therefore meaningless; or (3) is syntactically valid but does not express the meaning of the English sentence.

   (a) Paris and Marseilles are both in France.

      i. $In(Paris \land Marseilles, France)$
      ii. $In(Paris, France) \land In(Marseilles, France)$
      iii. $In(Paris, France) \lor In(Marseilles, France)$

   (b) There is a country that borders both Iraq and Pakistan.

      i. $\exists c. Country(c) \land Borders(c, Iraq) \land Borders(c, Pakistan)$
      ii. $\exists c. Country(c) \to (Borders(c, Iraq) \land Borders(c, Pakistan))$
      iii. $(\exists c. Country(c)) \to (Borders(c, Iraq) \land Borders(c, Pakistan))$
      iv. $\exists c. Borders(Country(c), Iraq \land Pakistan)$

   (c) All countries that border Ecuador are in South America.

      i. $\forall c. Country(c) \land Borders(c, Ecuador) \to In(c, SouthAmerica)$
      ii. $\forall c. Country(c) \to (Borders(c, Ecuador) \to In(c, SouthAmerica))$
      iii. $\forall c. (Country(c) \to Borders(c, Ecuador)) \to In(c, SouthAmerica)$
      iv. $\forall c. Country(c) \land Borders(c, Ecuador) \land In(c, SouthAmerica)$

   (d) No region in South America borders any region in Europe.

      i. $\neg(\exists c, d. In(c, SouthAmerica) \land In(d, Europe) \land Borders(c, d))$
      ii. $\forall c, d. (In(c, SouthAmerica) \land In(d, Europe)) \to \neg Borders(c, d))$
      iii. $(\neg \forall c. In(c, SouthAmerica)) \to (\exists d. In(d, Europe) \land \neg Borders(c, d))$
      iv. $\forall c. In(c, SouthAmerica) \to (\forall d. In(d, Europe) \to \neg Borders(c, d))$

   (e) No two adjacent countries have the same map color.

      i. $\forall x, y. \neg Country(x) \lor \neg Country(y) \lor \neg Borders(x, y) \lor \neg(MapColor(x) = MapColor(y))$
      ii. $\forall x, y. (Country(x) \land Country(y) \land Borders(x, y) \land \neg(x = y)) \to \neg(MapColor(x) = MapColor(y))$
      iii. $\forall x, y. Country(x) \land Country(y) \land Borders(x, y) \land \neg(MapColor(x) = MapColor(y))$
      iv. $\forall x, y. (Country(x) \land Country(y) \land Borders(x, y)) \to MapColor(x \neq y)$

(f) Every cat loves its mother or father.

i. $\forall x. Cat(x) \rightarrow Loves(x, Mother(x) \vee Father(x))$

ii. $\forall x. \neg Cat(x) \vee Loves(x, Mother(x)) \vee Loves(x, Father(x))$

iii. $\forall x. Cat(x) \wedge (Loves(x, Mother(x)) \vee Loves(x, Father(x)))$

(g) Every dog who loves one of its brothers is happy.

i. $\forall x. Dog(x) \wedge (\exists y. Brother(y, x) \wedge Loves(x, y)) \rightarrow Happy(x)$

ii. $\forall x, y. Dog(x) \wedge Brother(y, x) \wedge Loves(x, y) \rightarrow Happy(x)$

iii. $\forall x. Dog(x) \wedge (\forall y. Brother(y, x) \leftrightarrow Loves(x, y)) \rightarrow Happy(x)$

(h) No dog bites a child of its owner.

i. $\forall x. Dog(x) \rightarrow \neg Bites(x, Child(Owner(x)))$

ii. $\neg \exists x, y. Dog(x) \wedge Child(y, Owner(x)) \wedge Bites(x, y)$

iii. $\forall x. Dog(x) \rightarrow (\forall y. Child(y, Owner(x)) \rightarrow \neg Bites(x, y))$

iv. $\neg \exists x. Dog(x) \rightarrow (\exists y. Child(y, Owner(x)) \wedge Bites(x, y))$

(i) Everyone's zip code within a state has the same first digit.

i. $\forall x, s, z_1. (State(s) \wedge LivesIn(x, s) \wedge Zip(x) = z_1) \rightarrow (\forall y, z_2. LivesIn(y, s) \wedge Zip(y) = z_2 \rightarrow Digit(1, z_1) = Digit(1, z_2))$

ii. $\forall x, s. (State(s) \wedge LivesIn(x, s) \wedge \exists z_1. Zip(x) = z_1) \rightarrow (\forall y, z_2. LivesIn(y, s) \wedge Zip(y) = z_2 \wedge Digit(1, z_1) = Digit(1, z_2))$

iii. $\forall x, y, s. State(s) \wedge LivesIn(x, s) \wedge LivesIn(y, s) \rightarrow Digit(1, Zip(x) = Zip(y))$

iv. $\forall x, y, s. State(s) \wedge LivesIn(x, s) \wedge LivesIn(y, s) \rightarrow Digit(1, Zip(x)) = Digit(1, Zip(y))$

3. (a) Translate into good, *natural* English (no $x$s or $y$s!):

$$\forall x, y, l. SpeaksLanguage(x, l) \wedge SpeaksLanguage(y, l) \rightarrow Understands(x, y) \wedge Understands(y, x)$$

(b) Explain why the previous sentence is entailed by the sentence:

$$\forall x, y, l. SpeaksLanguage(x, l) \wedge SpeaksLanguage(y, l) \rightarrow Understands(x, y)$$

4. Consider a vocabulary with the following symbols.

- $Occupation(p, o)$: predicate denoting that person $p$ has occupation $o$.
- $Customer(p1, p2)$: predicate denoting that person $p1$ is a customer of person $p2$.
- $Boss(p1, p2)$: predicate denoting that person $p1$ is a boss of person $p2$.
- *Doctor*, *Surgeon*, *Lawyer*, *Actor*: constants denoting occupations.
- *Emily*, *Joe*: constants denoting people.

Use these symbols to write the following assertions in first-order logic:

(a) Emily is either a surgeon or a lawyer.

(b) Joe is an actor, but he also holds another job.

(c) All surgeons are doctors.

(d) Joe does not have a lawyer (i.e., is not a customer of any lawyer).

(e) Emily has a boss who is a lawyer.

(f) There exists a lawyer all of whose customers are doctors.

(g) Every surgeon has a lawyer.

5. Consider a first-order logical knowledge base that describes worlds containing people, songs, albums (e.g., "Meet the Beatles") and disks (i.e., particular physical instances of CDs). The vocabulary contains the following symbols:

- $CopyOf(d, a)$: predicate denoting that disk $d$ is a copy of album $a$.

- *Owns*$(p, d)$: predicate denoting that person $p$ owns disk $d$.
- *Sings*$(p, s, a)$: predicate denoting that album $a$ includes a recording of song $s$ sung by person $p$.
- *Wrote*$(p, s)$: predicate denoting that person $p$ wrote song $s$.
- constants *McCartney*, *Gershwin*, *BillieHoliday*, *Joe*, *EleanorRigby*, *TheManILove*, and *Revolver*.

Express the following statements in first-order logic:

(a) Gershwin wrote "The Man I Love".
(b) Gershwin did not write "Eleanor Rigby".
(c) Either Gershwin or McCartney wrote "The Man I Love".
(d) Joe has written at least one song.
(e) Joe owns a copy of *Revolver*.
(f) Every song that McCartney sings on *Revolver* was written by McCartney.
(g) Gershwin did not write any of the songs on *Revolver*.
(h) Every song that Gershwin wrote has been recorded on some album. (Possibly different songs are recorded on different albums.)
(i) There is a single album that contains every song that Joe has written.
(j) Joe owns a copy of an album that has Billie Holiday singing "The Man I Love".
(k) Joe owns a copy of every album that has a song sung by McCartney. (Of course, each different album is instantiated in a different physical CD.)
(l) Joe owns a copy of every album on which all the songs are sung by Billie Holiday.

6. Write in first-order logic the assertion that every key and at least one of every pair of socks will eventually be lost forever, using only the following vocabulary: *Key*$(x)$, denoting that $x$ is a key; *Sock*$(x)$, denoting that $x$ is a sock; *Pair*$(x, y)$, denoting that $x$ and $y$ are a pair; *Now*, denoting the current time; *Before*$(t_1, t_2)$, denoting that time $t_1$ comes before time $t_2$; and *Lost*$(x, t)$, denoting that object $x$ is lost at time $t$.

7. Assuming predicates *Parent* and *Female* and constants *Joan* and *Kevin*, with the obvious meanings, express each of the following sentences in first-order logic. (You may use the abbreviation $\exists!$ to mean "there exists exactly one".)

(a) Joan has a daughter (possibly more than one, and possibly sons as well).
(b) Joan has exactly one daughter (but may have sons as well).
(c) Joan has exactly one child, a daughter.
(d) Joan and Kevin have exactly one child together.
(e) Joan has at least one child with Kevin, and no children with anyone else.

8. Arithmetic assertions can be written in first-order logic with the predicate symbol $<$, the function symbols $+$ and $\times$, and the constant symbols 0 and 1. Additional predicates can also be defined with biconditionals.

(a) Represent the property "$x$ is an even number".
(b) Represent the property "$x$ is prime".
(c) Represent Goldbach's conjecture – the conjecture (unproven as yet) that every even number is equal to the sum of two primes.

9. Represent the following sentences in first-order logic, using a consistent vocabulary (which you must define).

(a) Some students took French in spring 2009.

(b) Every student who takes French passes it.

(c) Only one student took Greek in spring 2009.

(d) The best score in Greek is always higher than the best score in French.

(e) Every person who buys a policy is smart.

(f) No person buys an expensive policy.

(g) There is an agent who sells policies only to people who are not insured.

(h) There is a barber who shaves all men in town who do not shave themselves.

(i) A person born in the UK, each of whose parents is a UK citizen or a UK resident, is a UK citizen by birth.

(j) A person born outside the UK, one of whose parents is a UK citizen by birth, is a UK citizen by descent.

(k) Politicians can fool some of the people all of the time, and they can fool all of the people some of the time, but they can't fool all of the people all of the time.

(l) All Greeks speak the same language.

10. Translate into first-order logic the following sentences, using a vocabulary that you define.

    (a) Understanding leads to friendship.

    (b) Friendship is transitive.

    Remember to define all predicates, functions, and constants you use.

11. The formula $B_{x,y} \leftrightarrow P_{x+1,y} \vee P_{x,y+1} \vee P_{x-1,y} \vee P_{x,y-1}$ defines the conditions under which a square is breezy. Here we consider two other ways to describe this aspect of the wumpus world.

    (a) We can write *diagnostic rules* leading from observed effects to hidden causes. For finding pits, the obvious diagnostic rules say that if a square is breezy, some adjacent square must contain a pit; and if a square is not breezy, then no adjacent square contains a pit. Write these two rules in first-order logic and show that their conjunction is logically equivalent to the formula above.

    (b) We can write *causal rules* leading from cause to effect. One obvious causal rule is that a pit causes all adjacent squares to be breezy. Write this rule in first-order logic, explain why it is incomplete compared to the formula above, and supply the missing axiom.

12. Write out the axioms required for reasoning about the wumpus's location, using a constant symbol *Wumpus* and a binary predicate *At*(*Wumpus*, *Location*). Remember that there is only one wumpus.

13. Explain what is wrong with the following proposed definition of adjacent squares in the wumpus world.

$$\forall x, y. Adjacent([x, y], [x + 1, y]) \wedge Adjacent([x, y], [x, y + 1])$$

14. Explain what is wrong with the following proposed definition of the set membership predicate.

$$\forall x, s. x \in \{x|s\}$$
$$\forall x, s. x \in s \rightarrow \forall y. x \in \{y|s\}$$

15. Using the set axioms as examples, write axioms for the list domain, including all the constants, functions, and predicates mentioned in chapter 8.

16. Does the fact $\neg$*Spouse*(*George*, *Laura*) follow from the facts *Jim* $\neq$ *George* and *Spouse*(*Jim*, *Laura*)? If so, give a proof; if not, supply additional axioms as needed.

    What happens if we use *Spouse* as a unary function symbol instead of a binary predicate?

17. Consider a vocabulary consisting of the functions *Father* and *Mother* and of the predicates *Male*, *Female*, *Parent*, *Sibling*, *Brother*, *Sister*, *Child*, *Daughter*, *Son*, *Spouse*, *Wife*, *Husband*, *Grandparent*, *Grandchild*, *Ancestor*, *FirstCousin*, *Aunt* and *Uncle*.

(a) Write axioms defining these predicates in terms of each other.

(b) Find out the proper definition of $m$-th cousin $n$ times removed, and write the definition in first-order logic.

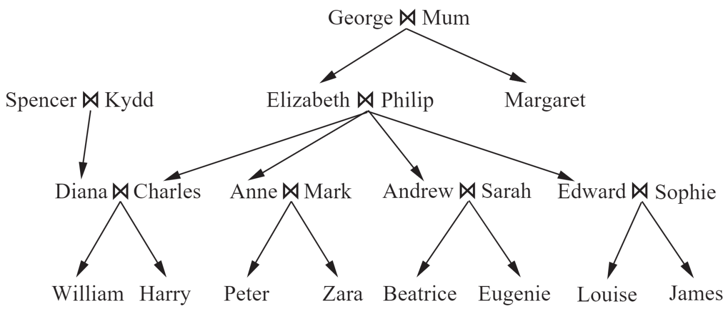


Figure 2: A typical family tree. The symbol $\bowtie$ connects spouses and arrows point to children.

(c) Write down the basic facts depicted in the family tree in Figure 2.

(d) Using a suitable logical reasoning system and the sentences you have written down, derive who are Elizabeth's grandchildren, Diana's brothers-in-law, Zara's great-grandparents, and Eugenie's ancestors.

18. (a) Consider a knowledge base containing just two sentences: $P(a)$ and $P(b)$. Does this knowledge base entail $\forall x.P(x)$? Why/why not?

    (b) Write down a logical sentence such that every world in which it is true contains exactly one object.

    (c) Write down a logical sentence such that every world in which it is true contains exactly two objects.

19. Which of the following are valid (necessarily true) sentences?

    (a) $(\exists x.R(x,x)) \to \forall y.\exists z.R(y,z))$

    (b) $\forall x.P(x) \vee \neg P(x)$

    (c) $\forall x.Smart(x) \vee R(x,x)$

20. Consider a version of the semantics for first-order logic in which models with empty domains are allowed. Give at least two examples of sentences that are valid according to the standard semantics but not according to the new semantics. Discuss which outcome makes more intuitive sense for your examples.

21. In Chapter 5, we used equality to indicate the relation between a variable and its value. For instance, we wrote $WA = red$ to mean that Western Australia is colored red. Representing this in first-order logic, we must write more verbosely $ColorOf(WA) = red$. What incorrect inference could be drawn if we wrote sentences such as $WA = red$ directly as logical assertions?

22. Write a general set of facts and axioms to represent the assertion "Wellington heard about Napoleon's death" that can correctly answer the question "Did Napoleon hear about Wellington's death?".

# 9 Inference in First-Order Logic

1. For each of the following pairs of atomic sentences, decide whether they are unifiable. If this is the case, give the most general unifier.

   (a) $P(A,B,B)$ and $P(x,y,z)$

   (b) $Q(y,G(A,B))$ and $Q(G(x,x),y)$

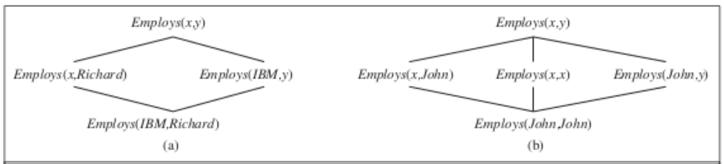


**Figure 9.2** (a) The subsumption lattice whose lowest node is $Employs(IBM, Richard)$. (b) The subsumption lattice for the sentence $Employs(John, John)$.

(c) $Older(Father(y), y)$ and $Older(Father(x), John)$

(d) $Knows(Father(y), y)$ and $Knows(x, x)$

2. Consider the subsumption lattices shown in Figure 9.2.

   (a) Construct the lattice for the sentence $Employs(Mother(John), Father(Richard))$.

   (b) Construct the lattice for the sentence $Employs(IBM, y)$ ("Everyone works for IBM"). Remember to include every kind of query that unifies with the sentence.

   (c) Assume that STORE indexes each sentence under every node in its subsumption lattice. Explain how FETCH should work when some of these sentences contain variables; use as examples the sentences in (a) and (b) and the query $Employs(x, Father(x))$.

3. From $Likes(Jerry, IceCream)$ it seems reasonable to infer $\exists x.Likes(x, IceCream)$. Write down a general inference rule that sanctions this inference. State carefully the conditions that must be satisfied by the variables and terms involved.

4. Suppose a knowledge base contains just the sentence $\exists x.AsHighAs(x, Everest)$. Which of the following are legitimate results of applying Existential Instantiation?

   (a) $AsHighAs(Everest, Everest)$

   (b) $AsHighAs(Kilimanjaro, Everest)$

   (c) $AsHighAs(Kilimanjaro, Everest) \wedge AsHighAs(BenNevis, Everest)$ (after two applications)

5. These questions concern concern issues with substitution and Skolemization.

   (a) Given the premise $\forall x \exists y.P(x, y)$, it is not valid to conclude that $\exists q.P(q, q)$. Give an example of a predicate $P$ where the first is true but the second is false.

   (b) Suppose that an inference engine is incorrectly written with the occurs check omitted, so that it allows a literal like $P(x, F(x))$ to be unified with $P(q, q)$. (As mentioned, most standard implementations of Prolog actually do allow this.) Show that such an inference engine will allow the conclusion $\exists y.P(q, q)$ to be inferred from the premise $\forall x \exists y.P(x, y)$.

   (c) Suppose that a procedure that converts first-order logic to clausal form incorrectly Skolemizes $\forall x \exists y.P(x, y)$ to $P(x, Sk_0)$ – that is, it replaces $y$ by a Skolem constant rather than by a Skolem function of $x$. Show that an inference engine that uses such a procedure will likewise allow $\exists q.P(q, q)$ to be inferred from the premise $\forall x \exists y.P(x, y)$.

   (d) A common error is to suppose that, in unification, one is allowed to substitute a term for a Skolem constant instead of for a variable. For instance, they will say that the formulas $P(Sk_1)$ and $P(A)$ can be unified under the substitution $\{Sk_1/A\}$. Give an example where this leads to an invalid inference.

6. Write down logical representations for the following sentences, suitable for use with Generalized Modus Ponens:

   - Horses, cows, and pigs are mammals.
   - An offspring of a horse is a horse.
   - Bluebeard is a horse.
   - Bluebeard is Charlie's parent.
   - Offspring and parent are inverse relations.
   - Every mammal has a parent.

Now use these sentences to answer a question by using a backward-chaining algorithm.

(a) Draw the proof tree generated by an exhaustive backward-chaining algorithm for the query $\exists h.Horse(h)$, where clauses are matched in the order given.

(b) What do you notice about this domain?

(c) How many solutions for $h$ actually follow from your sentences?

(d) Can you think of a way to find all of them?

7. Suppose you are given the following axioms:

1. $0 \leq 3$
2. $7 \leq 9$
3. $\forall x.x \leq x$
4. $\forall x.x \leq x + 0$
5. $\forall x.x + 0 \leq x$
6. $\forall x, y.x + y \leq y + x.$
7. $\forall w, x, y, z.w \leq y \wedge x \leq z \rightarrow w + x \leq y + z$
8. $\forall x, y, z.x \leq y \wedge y \leq z \rightarrow x \leq z$

(a) Give a backward-chaining proof of the sentence $7 \leq 3 + 9$. (Be sure, of course, to use only the axioms given here, not anything else you may know about arithmetic.) Show only the steps that leads to success, not the irrelevant steps.

(b) Give a forward-chaining proof of the sentence $7 \leq 3 + 9$. Again, show only the steps that lead to success.

8. This question considers Horn KBs, such as the following:

$$\begin{aligned} P(F(x)) &\rightarrow P(x) \\ Q(x) &\rightarrow P(F(x)) \\ &P(A) \\ &Q(B) \end{aligned}$$

Let FC be a breadth-first forward-chaining algorithm that repeatedly adds all consequences of currently satisfied rules; let BC be a depth-first left-to-right backward-chaining algorithm that tries clauses in the order given in the KB. Which of the following are true?

(a) FC will infer the literal $Q(A)$.

(b) FC will infer the literal $P(B)$.

(c) If FC has failed to infer a given literal, then it is not entailed by the KB.

(d) BC will return *true* given the query $P(B)$.

(e) If BC does not return *true* given a query literal, then it is not entailed by the KB.

9. The following Prolog code defines a predicate `P`. (Remember that uppercase terms are variables, not constants, in Prolog.)

```
P(X,[X|Y]).
P(X,[Y|Z]) :- P(X,Z).
```

(a) Show proof trees and solutions for the queries `P(A,[2,1,3])` and `P(2,[1,A,3])`.

(b) What standard list operation does `P` represent?

10. This exercise looks at sorting in Prolog.

(a) Write Prolog clauses that define a predicate `sorted(L)`, which is true if and only if list `L` is sorted in ascending order.

(b) Write a Prolog definition for the predicate `perm(L,M)`, which is true if and only if `L` is a permutation of `M`.

(c) Define `sort(L,M)` (`M` is a sorted version of `L`) using `perm` and `sorted`.

(d) Run `sort` on longer and longer lists until you lose patience. What is the time complexity of your program?

(e) Write a faster sorting algorithm, such as insertion sort or Quicksort, in Prolog.

11. This exercise looks at the recursive application of rewrite rules, using logic programming. A *rewrite rule* (or *demodulator* in OTTER terminology) is an equation with a specified direction. For example, the rewrite rule $x+0 \rightarrow x$ suggests replacing any expression that matches $x+0$ with the expression $x$.

    Rewrite rules are a key component of equational reasoning systems. Use the predicate `rewrite(X,Y)` to represent rewrite rules. (For example, the earlier rewrite rule is written as `rewrite(X+0,X)`.) Some terms are primitive and cannot be further simplified; thus, we write `primitive(0)` to say that `0` is a primitive term.

    (a) Write a definition of a predicate `simplify(X,Y)` that is true when `Y` is a simplified version of `X`, that is, when no further rewrite rules apply to any subexpression of `Y`.

    (b) Write a collection of rules for the simplification of expressions involving arithmetic operators, and apply your simplification algorithm to some sample expressions.

    (c) Write a collection of rewrite rules for symbolic differentiation, and use them along with your simplification rules to differentiate and simplify expressions involving arithmetic expressions, including exponentiation.

12. Suppose a knowledge base contains just the following first-order Horn clauses.

$$\begin{aligned} & \mathit{Ancestor}(\mathit{Mother}(x), x) \\ \mathit{Ancestor}(x,y) \wedge \mathit{Ancestor}(y,z) \rightarrow \; & \mathit{Ancestor}(x,z) \end{aligned}$$

    Consider a forward chaining algorithm that, on the $j$-th iteration, terminates if the KB contains a sentence that unifies with the query, otherwise adds to the KB every atomic sentence that can be inferred from the sentences already in the KB after iteration $j-1$.

    (a) For each of the following queries, say whether the algorithm will (1) give an answer (if so, write down that answer); (2) terminate with no answer; or (3) never terminate.

    i. $\mathit{Ancestor}(\mathit{Mother}(y), \mathit{John})$
    ii. $\mathit{Ancestor}(\mathit{Mother}(\mathit{Mother}(y)), \mathit{John})$
    iii. $\mathit{Ancestor}(\mathit{Mother}(\mathit{Mother}(\mathit{Mother}(y))), \mathit{Mother}(y))$
    iv. $\mathit{Ancestor}(\mathit{Mother}(\mathit{John}), \mathit{Mother}(\mathit{Mother}(\mathit{John})))$

    (b) Can a resolution algorithm prove the sentence $\neg \mathit{Ancestor}(\mathit{John}, \mathit{John})$ from the original knowledge base? Explain how, or why not.

    (c) Suppose we add the assertion that $\neg(\mathit{Mother}(x) = x)$ and augment the resolution algorithm with inference rules for equality. Now what is the answer to (b)?

13. Let $\mathcal{L}$ be the first-order language with a single predicate $S(p,q)$, meaning "$p$ shaves $q$". Assume a domain of people.

    (a) Consider the sentence "There exists a person $P$ who shaves every one who does not shave themselves, and only people that do not shave themselves". Express this in $\mathcal{L}$.

    (b) Convert the sentence you wrote to clausal form.

    (c) Construct a resolution proof to show that the clauses you obtained are inherently inconsistent. (Note: you do not need any additional axioms.)

14. How can resolution be used to show that a sentence is valid? Unsatisfiable?

15. Construct an example of two clauses that can be resolved together in two different ways giving two different outcomes.

16. From "Horses are animals", it follows that "The head of a horse is the head of an animal". Demonstrate that this inference is valid by carrying out the following steps:

    (a) Translate the premise and the conclusion into the language of first-order logic. Use three predicates: $\mathit{HeadOf}(h, x)$ (meaning "$h$ is the head of $x$"), $\mathit{Horse}(x)$, and $\mathit{Animal}(x)$.

    (b) Negate the conclusion, and convert the premise and the negated conclusion into conjunctive normal form.

    (c) Use resolution to show that the conclusion follows from the premise.

17. Here are two sentences in the language of first-order logic:

    (A) $\forall x \exists y . x \geq y$

    (B) $\exists y \forall x . x \geq y$

    (a) Assume that the variables range over all the natural numbers and that predicate $\geq$ means "is greater than or equal to". Under this interpretation, translate (A) and (B) into English.

    (b) Is (A) true under this interpretation?

    (c) Is (B) true under this interpretation?

    (d) Does (A) logically entail (B)?

    (e) Does (B) logically entail (A)?

    (f) Using resolution, try to prove that (A) follows from (B). Do this even if you think that (B) does not logically entail (A); continue until the proof breaks down and you cannot proceed (if it does break down). Show the unifying substitution for each resolution step. If the proof fails, explain exactly where, how, and why it breaks down.

    (g) Now try to prove that (B) follows from (A).

18. We said in this chapter that resolution cannot be used to generate all logical consequences of a set of sentences. Can any algorithm do this?

19. A popular children's riddle is "Brothers and sisters have I none, but that man's father is my father's son". Use the rules of the family domain to show who that man is. You may apply any of the inference methods described in this chapter. Why do you think that this riddle is difficult?

# 10 Default Logic

1. Default Logic can be used to model the Closed World Assumption (everything you cannot prove, you assume to be false).

    (a) You ask your friend Melisa if she has any siblings and she tells you she has an older sister. What default rule is needed to derive that your friend has no brother?

    (b) Generalize this to any system. What rules are needed to enforce the Closed World Assumption?

    (c) Consider a system where there are five predicates with the following semantics:

        i. $\mathit{HasYoungerSister}(x)$: $x$ has a younger sister.
        ii. $\mathit{HasOlderSister}(x)$: $x$ has an older sister.
        iii. $\mathit{HasYoungerBrother}(x)$: $x$ has a younger brother.
        iv. $\mathit{HasOlderBrother}(x)$: $x$ has an older brother.
        v. $\mathit{IsYoungestSibling}(x)$: $x$ is the youngest sibling.

        Add the rules needed to enforce the closed world assumption as specified by your answer to (b). In addition, add a rule for deducting whether someone is the youngest sibling. Use this to infer that *Melisa* is the youngest sibling based on the answer she gave in question (a).

2. Specify the following as default rules, using a suggestive vocabulary.

(a) You do not expect someone to be waiting outside your door.

(b) If your doorbell does not ring, then you do not expect someone to be waiting outside your door.

(c) If your doorbell is not broken and it does not ring, then you do not expect someone to be waiting outside your door.

(d) If your doorbell is not broken and does not ring, or if your doorbell is broken and you don't hear a knock, then you don't expect someone to be waiting outside your door.

(e) If you hear a knock, and you see no-one through the peephole of your door, then you don't expect someone to be waiting outside or you expect someone small to be waiting outside.

3. Consider the following scenario.

   - You are heading for a party on a Sunday evening.
   - You wonder if the party will be good.
   - For the party to be good there have to be food and drinks.
   - You have not been asked to bring food nor have you been asked to bring drinks.
   - If you haven't been asked to bring it, you assume it is already there.

   Model the relevant information as a default theory. Can you answer the question of whether the party will be good?

4. Consider the following default rule.

$$\frac{\mathit{Bird}(x) : \mathit{Flies}(x)}{\mathit{Flies}(x)}$$

(a) Consider the default theory containing the above rule and the formula $\mathit{Bird}(\mathit{Tweety})$. Show directly (using the definition) that $\mathit{Th}(\{\mathit{Bird}(\mathit{Tweety})), \mathit{Flies}(\mathit{Tweety})\})$ is an extension of this theory.

(b) Consider now the default theory containing the above rule together with the set

$$\Gamma = \{\mathit{Penguin}(\mathit{Tweety}), \forall x.\mathit{Penguin}(x) \to \mathit{Bird}(x), \forall x.\mathit{Penguin}(x) \to \neg\mathit{Flies}(x)\}\,.$$

Show directly that this theory admits the extension $\mathit{Th}(\Gamma)$, i.e. that the default rule cannot be applied to infer any new knowledge.

(c) Use the algorithm for computing extensions to confirm the results from the two previous questions.

5. Consider the following information.

   - University professors are usually politically left-wing.
   - Left-wing people are usually not conservative.
   - University professors are usually conservative.
   - No one can be simultaneously left-wing and right-wing.
   - John is a right-wing university professor.
   - Mary is a university professor.

(a) Formalise this knowledge as default rules.

(b) Compute all possible extensions of this theory.

6. Which of the following default rules is a better match for the sentence "All adult birds can fly"? Why?

$$\frac{\mathit{Bird}(x) \wedge \mathit{Adult}(x) : \mathit{Flies}(x)}{\mathit{Flies}(x)} \qquad \text{or} \qquad \frac{\mathit{Bird}(x) : \mathit{Adult}(x), \mathit{Flies}(x)}{\mathit{Flies}(x)}$$

7. The justifications of a default rule need to be consistent with the current theory, but not necessarily with each other. This means that the two defaults

$$\frac{true : P(x), \neg P(x)}{P(x)} \qquad \text{and} \qquad \frac{true : P(x) \wedge \neg P(x)}{P(x)}$$

are not equivalent: the second one can never be applied, while the first one can be applied whenever nothing is known about $P(x)$. Can you give an example of a predicate $P$ for which the first default rule makes sense? Is there a simpler default rule that is equivalent (in the sense that it will yield the same extensions for all knowledge bases)?

## 11 Classical Planning

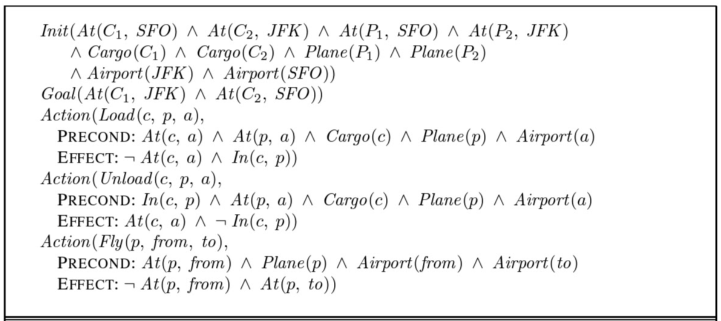
1. The monkey-and-bananas problem is faced by a monkey in a laboratory with some bananas hanging out of reach from the ceiling. A box is available that will enable the monkey to reach the bananas if he climbs on it. Initially, the monkey is at $A$, the bananas at $B$, and the box at $C$. The monkey and box have height *Low*, but if the monkey climbs onto the box he will have height *High*, the same as the bananas. The actions available to the monkey include *Go* from one place to another, *Push* an object from one place to another, *ClimbUp* onto or *ClimbDown* from an object, and *Grasp* or *Ungrasp* an object. The result of a *Grasp* is that the monkey holds the object if the monkey and object are in the same place at the same height.

   (a) Write down the initial state description.

   (b) Write the six action schemas.

   (c) Suppose the monkey wants to fool the scientists, who are off to tea, by grabbing the bananas, but leaving the box in its original place. Write this as a general goal (i.e., not assuming that the box is necessarily at $C$) in the language of situation calculus. Can this goal be solved by a classical planning system?

   (d) Your schema for pushing is probably incorrect, because if the object is too heavy, its position will remain the same when the *Push* schema is applied. Fix your action schema to account for heavy objects.



$Init(At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO) \wedge At(P_2, JFK)$
$\quad \wedge Cargo(C_1) \wedge Cargo(C_2) \wedge Plane(P_1) \wedge Plane(P_2)$
$\quad \wedge Airport(JFK) \wedge Airport(SFO))$
$Goal(At(C_1, JFK) \wedge At(C_2, SFO))$
$Action(Load(c, p, a),$
PRECOND: $At(c, a) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
EFFECT: $\neg At(c, a) \wedge In(c, p))$
$Action(Unload(c, p, a),$
PRECOND: $In(c, p) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
EFFECT: $At(c, a) \wedge \neg In(c, p))$
$Action(Fly(p, from, to),$
PRECOND: $At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
EFFECT: $\neg At(p, from) \wedge At(p, to))$

**Figure 10.1** A PDDL description of an air cargo transportation planning problem.

2. Given the action schemas and initial state from Figure 10.1, what are all the applicable concrete instances of $Fly(p, from, to)$ in the state described by the following formula?

$$At(P_1, JFK) \wedge At(P_2, SFO) \wedge Plane(P_1) \wedge Plane(P_2) \wedge Airport(JFK) \wedge Airport(SFO)$$

3. Consider a robot whose operation is described by the following PDDL operators:

$$Op(\{Go(x,y)\}, \{At(Robot,x)\}, \{\neg At(Robot,x) \wedge At(Robot,y)\})$$
$$Op(\{Pick(o)\}, \{At(Robot,x) \wedge At(o,x)\}, \{\neg At(o,x) \wedge Holding(o)\})$$
$$Op(\{Drop(o)\}, \{At(Robot,x) \wedge Holding(o)\}, \{At(o,x) \wedge \neg Holding(o)\}$$

(a) The operators allow the robot to hold more than one object. Show how to modify them with an *EmptyHand* predicate for a robot that can hold only one object.

(b) Assuming that these are the only actions in the world, write a successor-state axiom for *EmptyHand*.

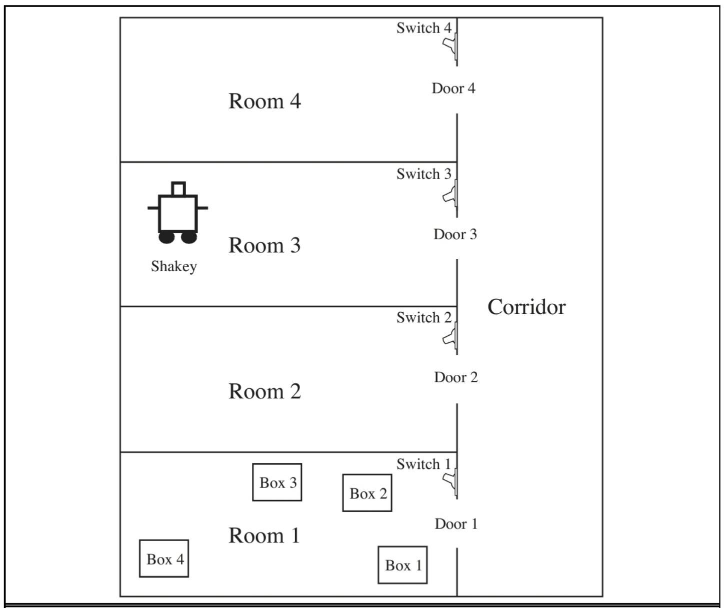


**Figure 10.14** Shakey's world. Shakey can move between landmarks within a room, can pass through the door between rooms, can climb climbable objects and push pushable objects, and can flip light switches.

4. The original STRIPS planner was designed to control Shakey the robot. Figure 10.14 shows a version of Shakey's world consisting of four rooms lined up along a corridor, where each room has a door and a light switch. The actions in Shakey's world include moving from place to place, pushing movable objects (such as boxes), climbing onto and down from rigid objects (such as boxes), and turning light switches on and off. The robot itself could not climb on a box or toggle a switch, but the planner was capable of finding and printing out plans that were beyond the robot's abilities. Shakey's six actions are the following:

   - $Go(x, y, r)$, which requires that Shakey be at $x$ and that $x$ and $y$ be locations in the same room $r$ (expressed by adequate predicates *At* and *In*). By convention a door between two rooms is in both of them.
   - Push a box $b$ from location $x$ to location $y$ within the same room: $Push(b, x, y, r)$. You will need the predicate *Box* and constants for the boxes.
   - Climb onto a box from position $x$: $ClimbUp(x, b)$; climb down from a box to position $x$: $ClimbDown(b, x)$. You will need the predicate *On* and the constant *Floor*.
   - Turn a light switch on or off: $TurnOn(s, b)$; $TurnOff(s, b)$. To turn a light on or off, Shakey must be on top of a box at the light switch's location.

   Write PDDL sentences for Shakey's six actions and the initial state. From these, construct a plan for Shakey to get $Box_2$ into $Room_2$.

5. Figures 10.3 and 10.4 show a blocks-world problem that is known as the Sussman anomaly. The problem was considered anomalous because the noninterleaved planners of the early 1970s could not solve it. Write a definition of the problem and solve it, either by hand or with a planning program.

   A noninterleaved planner is a planner that, when given two subgoals $G_1$ and $G_2$, produces either a plan for $G_1$ concatenated with a plan for $G_2$, or vice versa. Can a noninterleaved planner solve this problem? How, or why not?

$Init(On(A, Table) \wedge On(B, Table) \wedge On(C, A)$
$\quad \wedge Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(B) \wedge Clear(C))$
$Goal(On(A, B) \wedge On(B, C))$
$Action(Move(b, x, y),$
  PRECOND: $On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge Block(y) \wedge$
  $\quad (b \neq x) \wedge (b \neq y) \wedge (x \neq y),$
  EFFECT: $On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y))$
$Action(MoveToTable(b, x),$
  PRECOND: $On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x),$
  EFFECT: $On(b, Table) \wedge Clear(x) \wedge \neg On(b, x))$

**Figure 10.3** A planning problem in the blocks world: building a three-block tower. One solution is the sequence $[MoveToTable(C, A), Move(B, Table, C), Move(A, Table, B)]$.

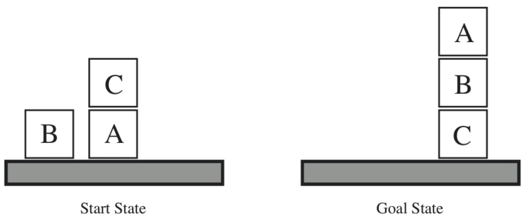


**Figure 10.4** Diagram of the blocks-world problem in Figure 10.3.

6. A finite Turing machine has a finite one-dimensional tape of cells, each cell containing one of a finite number of symbols. One cell has a read and write head above it. There is a finite set of states the machine can be in, one of which is the accept state. At each time step, depending on the symbol on the cell under the head and the machine's current state, there are a set of actions we can choose from. Each action involves writing a symbol to the cell under the head, transitioning the machine to a state, and optionally moving the head left or right. The mapping that determines which actions are allowed is the Turing machine's program. Your goal is to control the machine into the accept state.

   Represent the Turing machine acceptance problem as a planning problem. If you can do this, it demonstrates that determining whether a planning problem has a solution is at least as hard as the Turing acceptance problem, which is PSPACE-hard.

7. Explain why dropping negative effects from every action schema results in a relaxed problem, provided that preconditions and goals contain only positive literals.

8. Describe the differences and similarities between problem solving and planning.

$Init(At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO) \wedge At(P_2, JFK)$
$\quad \wedge Cargo(C_1) \wedge Cargo(C_2) \wedge Plane(P_1) \wedge Plane(P_2)$
$\quad \wedge Airport(JFK) \wedge Airport(SFO))$
$Goal(At(C_1, JFK) \wedge At(C_2, SFO))$
$Action(Load(c, p, a),$
  PRECOND: $At(c, a) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
  EFFECT: $\neg At(c, a) \wedge In(c, p))$
$Action(Unload(c, p, a),$
  PRECOND: $In(c, p) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
  EFFECT: $At(c, a) \wedge \neg In(c, p))$
$Action(Fly(p, from, to),$
  PRECOND: $At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
  EFFECT: $\neg At(p, from) \wedge At(p, to))$

**Figure 10.1** A PDDL description of an air cargo transportation planning problem.

9. You have a number of trucks with which to deliver a set of packages. Each package starts at some location on a grid map, and has a destination somewhere else. Each truck is directly controlled by moving forward and turning. Construct a hierarchy of high-level actions for this problem. What knowledge about the solution does your hierarchy encode?

10. Suppose that a high-level action has exactly one implementation as a sequence of primitive actions.

Give an algorithm for computing its preconditions and effects, given the complete refinement hierarchy and schemas for the primitive actions.

11. Suppose that the optimistic reachable set of a high-level plan is a superset of the goal set; can anything be concluded about whether the plan achieves the goal? What if the pessimistic reachable set doesn't intersect the goal set? Explain.

12. Write an algorithm that takes an initial state (specified by a set of propositional literals) and a sequence of HLAs (each defined by preconditions and angelic specifications of optimistic and pessimistic reachable sets) and computes optimistic and pessimistic descriptions of the reachable set of the sequence.

13. Consider the following argument.

    > In a framework that allows uncertain initial states, nondeterministic effects are just a notational convenience, not a source of additional representational power. For any action schema $a$ with nondeterministic effect $P \lor Q$, we could always replace it with the conditional effects $R : P \land \neg R : Q$, which in turn can be reduced to two regular actions. The proposition $R$ stands for a random proposition that is unknown in the initial state and for which there are no sensing actions.

    Is this argument correct? Consider separately two cases, one in which only one instance of action schema $a$ is in the plan, the other in which more than one instance is.

14. Suppose the *Flip* action always changes the truth value of variable $L$. Show how to define its effects by using an action schema with conditional effects. Show that, despite the use of conditional effects, a 1-CNF belief state representation remains in 1-CNF after a *Flip*.

15. In the blocks world we were forced to introduce two action schemas, *Move* and *MoveToTable*, in order to maintain the *Clear* predicate properly. Show how conditional effects can be used to represent both of these cases with a single action.

16. Conditional effects were illustrated for the *Suck* action in the vacuum world—which square becomes clean depends on which square the robot is in. Can you think of a new set of propositional variables to define states of the vacuum world, such that *Suck* has an unconditional description? Write out the descriptions of *Suck*, *Left*, and *Right*, using your propositions, and demonstrate that they suffice to describe all possible states of the world.

17. The following quotes are from the backs of shampoo bottles. Identify each as an unconditional, conditional, or execution-monitoring plan.

    (a) Lather. Rinse. Repeat.

    (b) Apply shampoo to scalp and let it remain for several minutes. Rinse and repeat if necessary.

    (c) See a doctor if problems persist.

18. A patient arrives at the doctor's office with symptoms that could have been caused either by dehydration or by disease D (but not both). There are two possible actions: *Drink*, which unconditionally cures dehydration, and *Medicate*, which cures disease D but has an undesirable side effect if taken when the patient is dehydrated.

    Write the problem description, and diagram a sensorless plan that solves the problem, enumerating all relevant possible worlds.

19. To the medication problem in the previous exercise, add a *Test* action that has the conditional effect *CultureGrowth* when *Disease* is true and in any case has the perceptual effect *Known*(*CultureGrowth*). Diagram a conditional plan that solves the problem and minimizes the use of the *Medicate* action.