1 INTRODUCTION

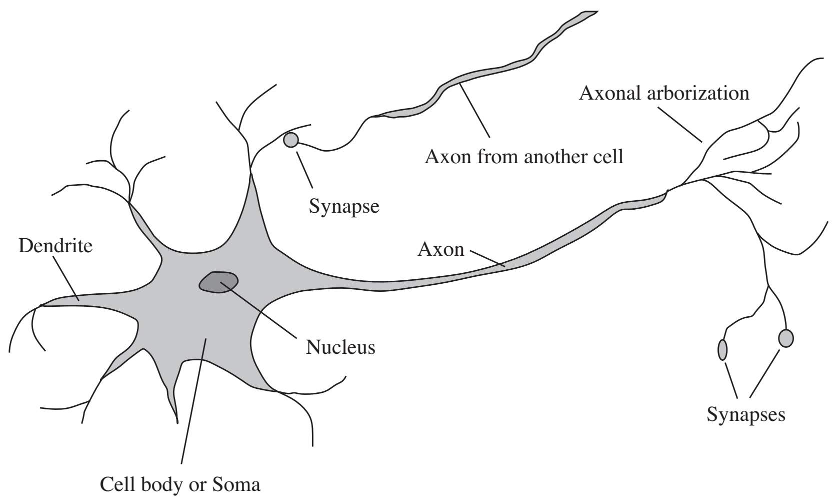


Figure 1.2 FILES: figures/neuron.eps (Tue Nov 3 16:23:13 2009). The parts of a nerve cell or neuron. Each neuron consists of a cell body, or soma, that contains a cell nucleus. Branching out from the cell body are a number of fibers called dendrites and a single long fiber called the axon. The axon stretches out for a long distance, much longer than the scale in this diagram indicates. Typically, an axon is 1 cm long (100 times the diameter of the cell body), but can reach up to 1 meter. A neuron makes connections with 10 to 100,000 other neurons at junctions called synapses. Signals are propagated from neuron to neuron by a complicated electrochemical reaction. The signals control brain activity in the short term and also enable long-term changes in the connectivity of neurons. These mechanisms are thought to form the basis for learning in the brain. Most information processing goes on in the cerebral cortex, the outer layer of the brain. The basic organizational unit appears to be a column of tissue about 0.5 mm in diameter, containing about 20,000 neurons and extending the full depth of the cortex about 4 mm in humans).

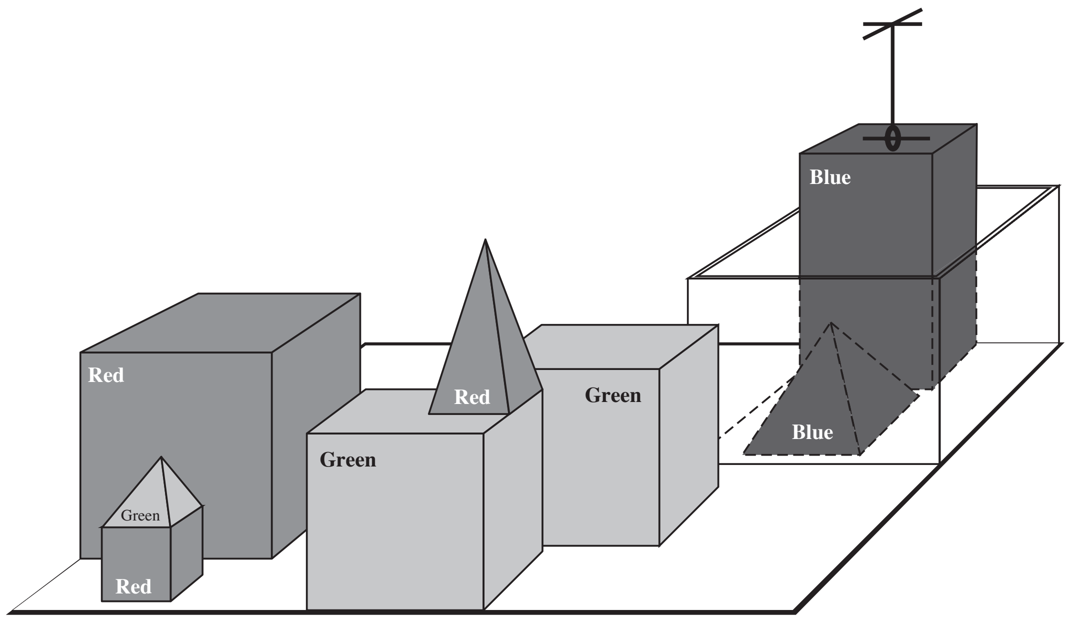


Figure 1.4 FILES: figures/blocks-world.eps (Tue Nov 3 16:22:27 2009). A scene from the blocks world. SHRDLU (?) has just completed the command “Find a block which is taller than the one you are holding and put it in the box.”

2 INTELLIGENT AGENTS

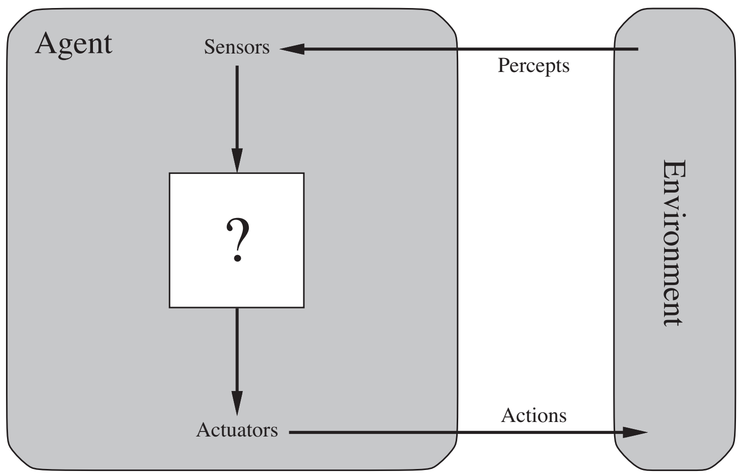


Figure 2.1 **FILES: figures/agent-environment.eps (Tue Nov 3 16:22:19 2009).** Agents interact with environments through sensors and actuators.

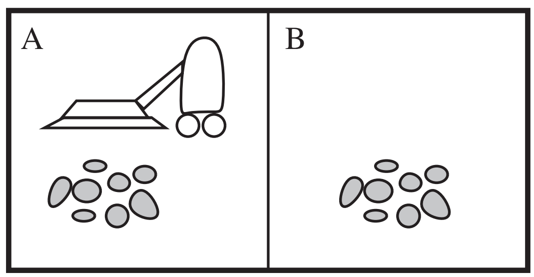


Figure 2.2 **FILES: figures/vacuum2-environment.eps (Tue Nov 3 16:24:01 2009).** A vacuum-cleaner world with just two locations.

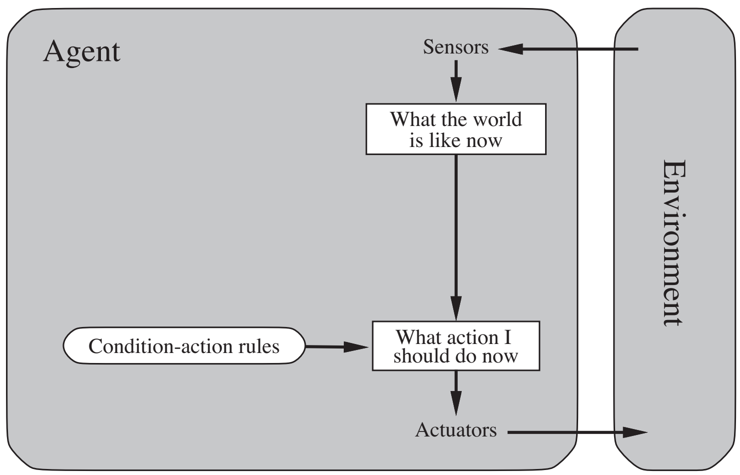


Figure 2.9 FILES: figures/simple-reflex-agent.eps (Tue Nov 3 16:23:44 2009). Schematic diagram of a simple reflex agent.

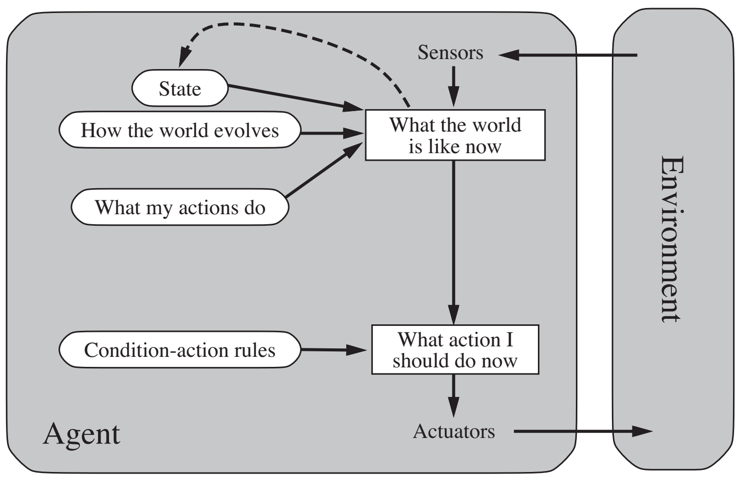


Figure 2.11 **FILES: figures/model-based-reflex-agent.eps (Tue Nov 3 16:23:11 2009).** A model-based reflex agent.

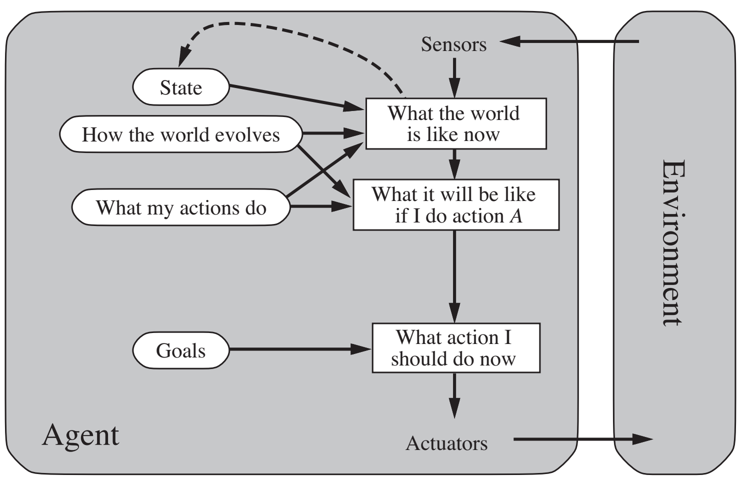


Figure 2.13 FILES: figures/goal-based-agent.eps (Tue Nov 3 16:22:54 2009). A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

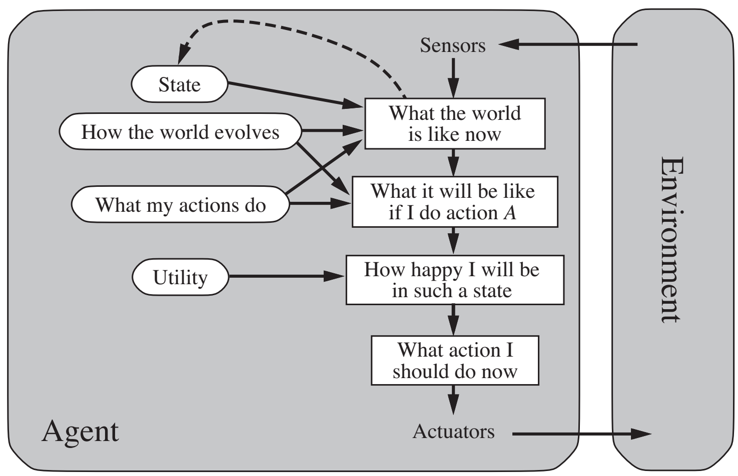


Figure 2.14 FILES: figures/utility-based-agent.eps (Tue Nov 3 16:23:59 2009). A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

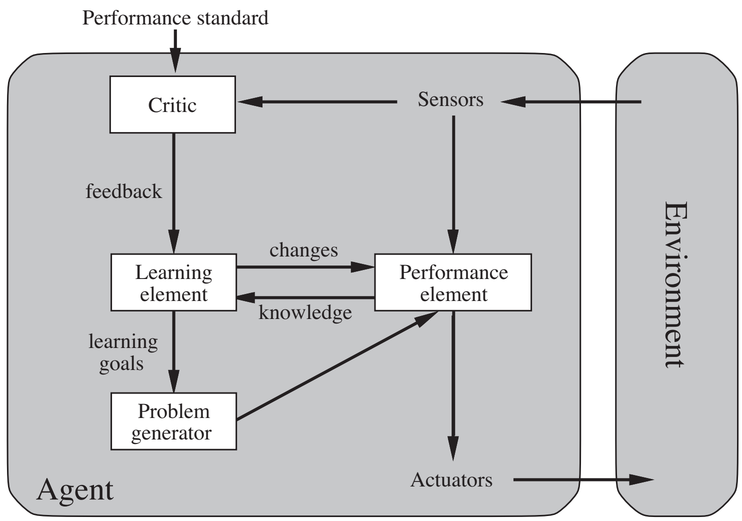


Figure 2.15 FILES: figures/learning-agent.eps (Tue Nov 3 16:23:06 2009). A general learning agent.

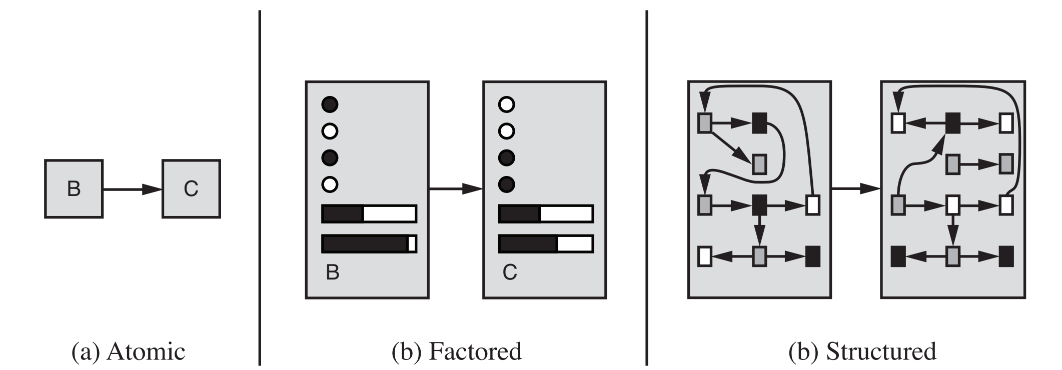


Figure 2.16 **FILES: figures/atomic-factored-structured.eps (Wed Nov 4 14:29:51 2009).** Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed set of symbols. (c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects.

3 SOLVING PROBLEMS BY SEARCHING

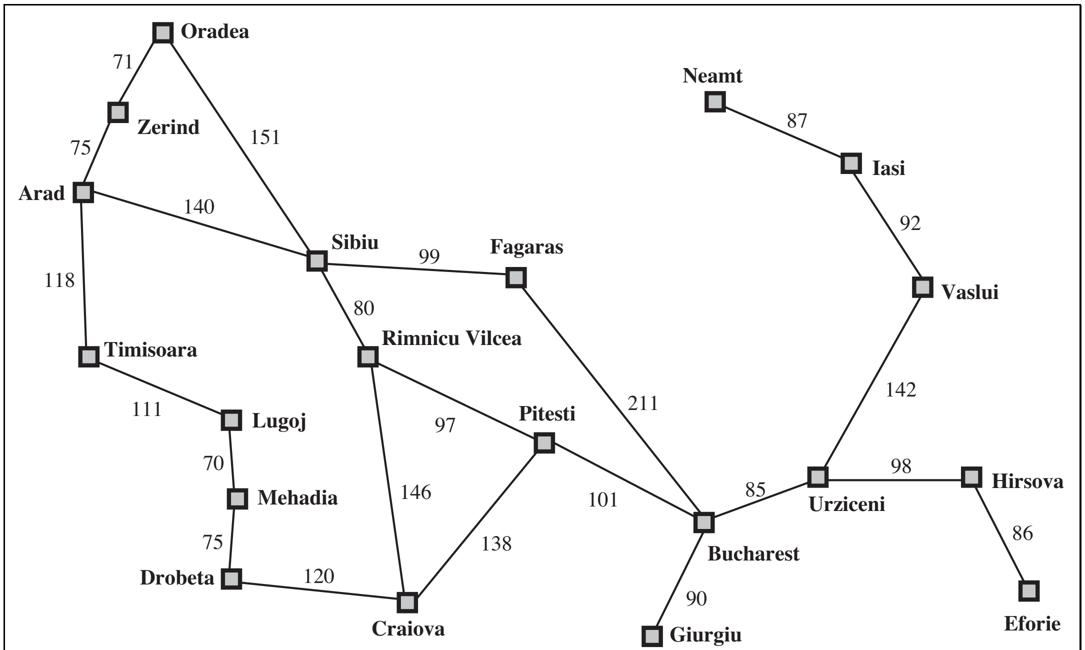


Figure 3.2 **FILES: figures/romania-distances.eps (Tue Nov 3 16:23:37 2009).** A simplified road map of part of Romania.

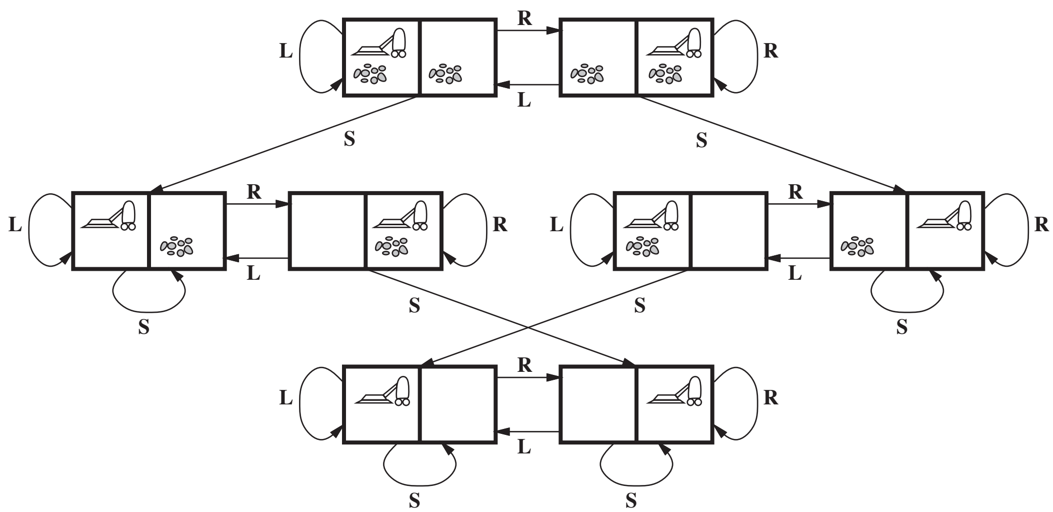


Figure 3.3 **FILES: figures/vacuum2-state-space.eps (Tue Nov 3 16:24:01 2009).** The state space for the vacuum world. Links denote actions: L = *Left*, R = *Right*, S = *Suck*.

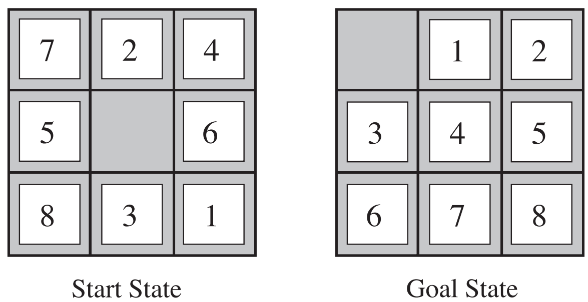


Figure 3.4 **FILES: figures/8puzzle.eps (Tue Nov 3 16:22:11 2009).** A typical instance of the 8-puzzle.

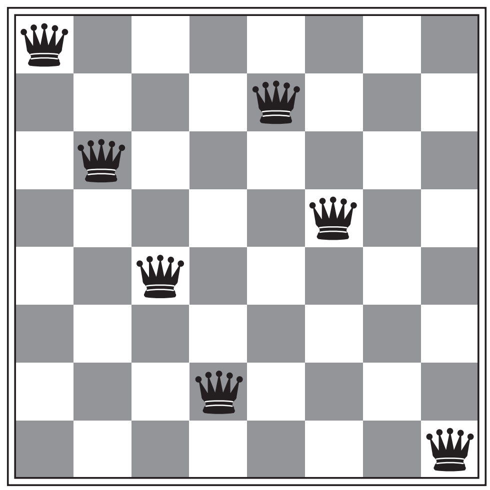


Figure 3.5 **FILES: figures/8queens.eps (Wed Nov 4 16:21:52 2009).** Almost a solution to the 8-queens problem. (Solution is left as an exercise.)

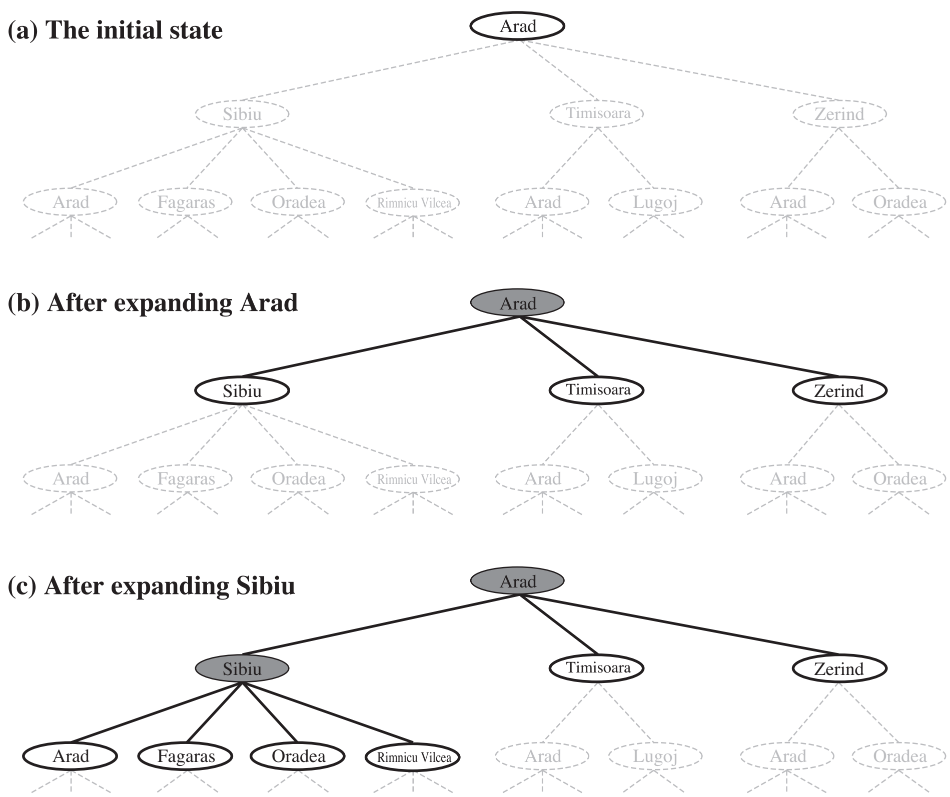


Figure 3.6 FILES: figures/search-map.eps (Tue Nov 3 16:23:38 2009). Partial search trees for finding a route from Arad to Bucharest. Nodes that have been expanded are shaded; nodes that have been generated but not yet expanded are outlined in bold; nodes that have not yet been generated are shown in faint dashed lines.

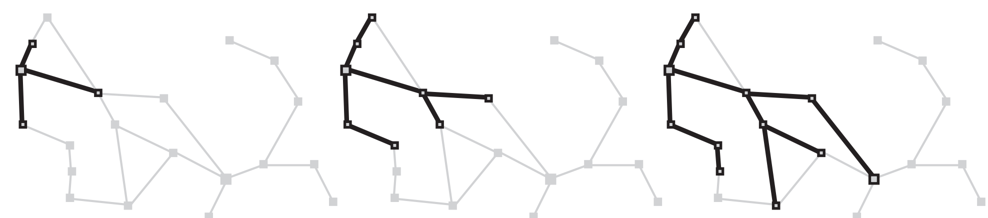


Figure 3.8 FILES: figures/romania-graph-search.eps (Tue Nov 3 13:48:17 2009). A sequence of search trees generated by a graph search on the Romania problem of Figure 3.2. At each stage, we have extended each path by one step. Notice that at the third stage, the northernmost city (Oradea) has become a dead end: both of its successors are already explored via other paths.

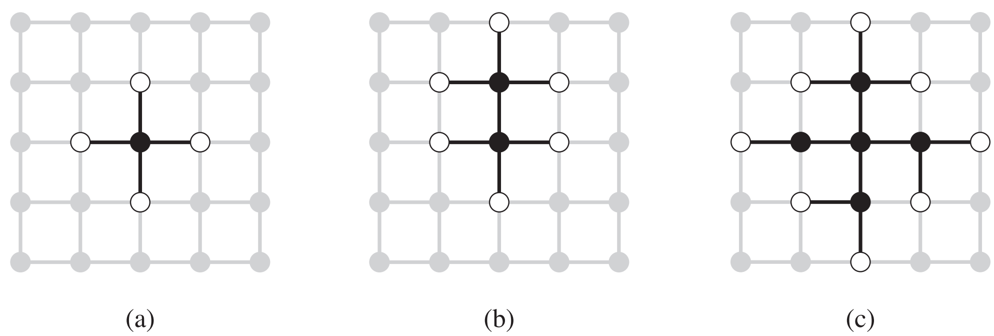


Figure 3.9 **FILES: figures/graph-separation.eps (Tue Nov 3 13:36:17 2009).** The separation property of GRAPH-SEARCH, illustrated on a rectangular-grid problem. The frontier (white nodes) always separates the explored region of the state space (black nodes) from the unexplored region (gray nodes). In (a), just the root has been expanded. In (b), one leaf node has been expanded. In (c), the remaining successors of the root have been expanded in clockwise order.

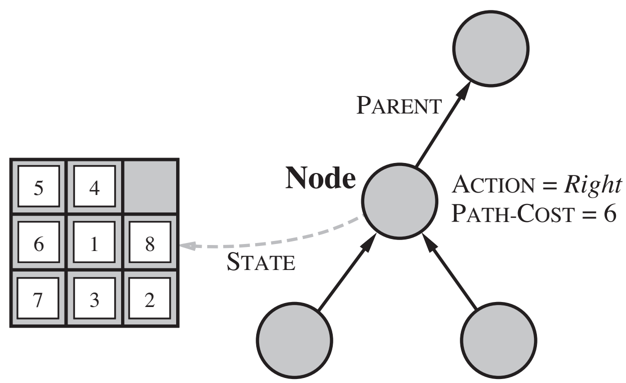


Figure 3.10 FILES: figures/state-vs-node.eps (Tue Nov 3 13:50:06 2009). Nodes are the data structures from which the search tree is constructed. Each has a parent, a state, and various bookkeeping fields. Arrows point from child to parent.

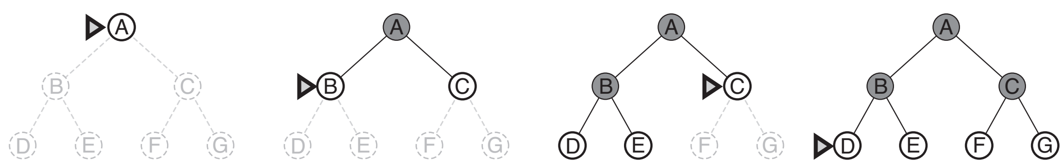


Figure 3.12 **FILES: figures/bfs-progress.eps (Tue Nov 3 16:22:26 2009).** Breadth-first search on a simple binary tree. At each stage, the node to be expanded next is indicated by a marker.

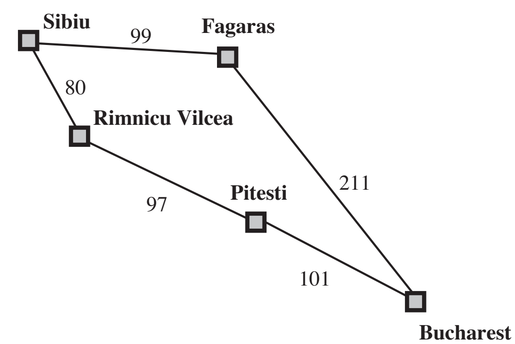


Figure 3.15 FILES: figures/romania-subgraph.eps (Tue Nov 3 13:48:07 2009). Part of the Romania state space, selected to illustrate uniform-cost search.

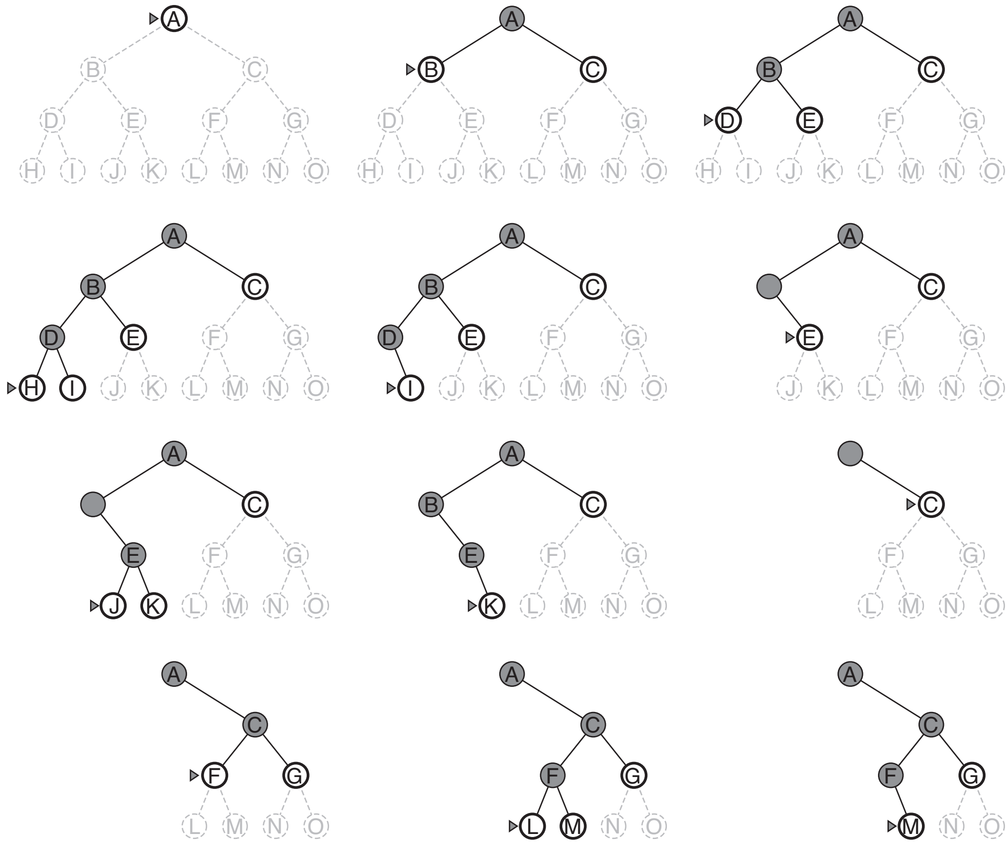


Figure 3.16 FILES: figures/dfs-progress-noblack.eps (Tue Nov 3 13:30:55 2009). Depth-first search on a binary tree. The unexplored region is shown in light gray. Explored nodes with no descendants in the frontier are removed from memory. Nodes at depth 3 have no successors and M is the only goal node.

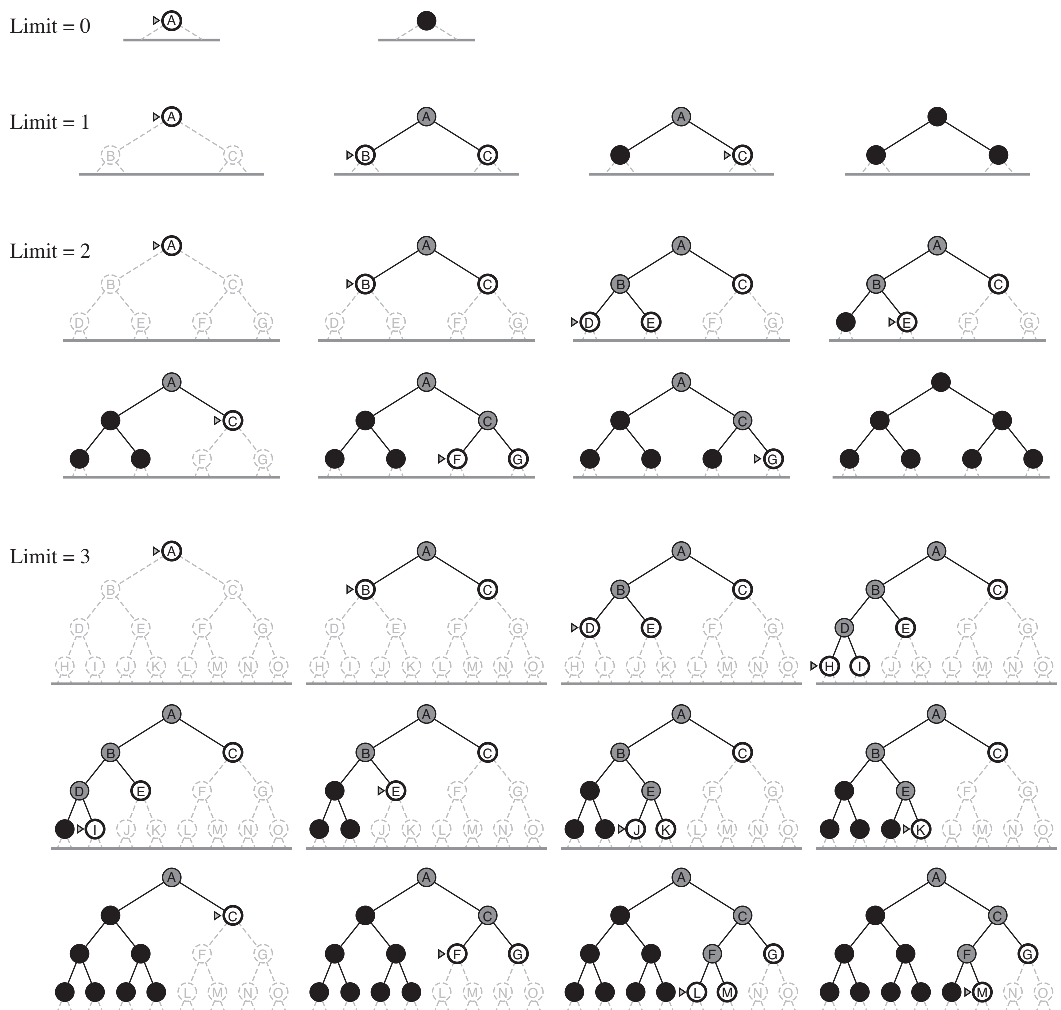


Figure 3.19 FILES: figures/ids-progress.eps (Tue Nov 3 16:23:04 2009). Four iterations of iterative deepening search on a binary tree.

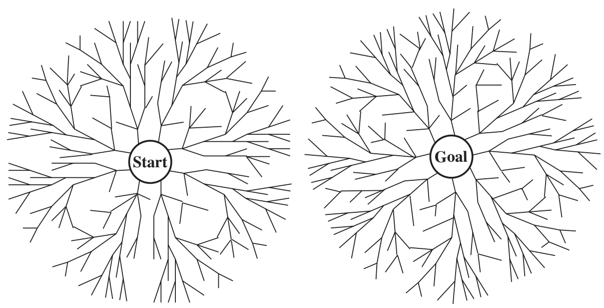


Figure 3.20 **FILES: figures/bidirectional.eps (Tue Nov 3 16:22:27 2009).** A schematic view of a bidirectional search that is about to succeed when a branch from the start node meets a branch from the goal node.

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Figure 3.22 FILES: figures/romania-sld.eps (Tue Nov 3 16:23:37 2009). Values of h_{SLD}—straight-line distances to Bucharest.

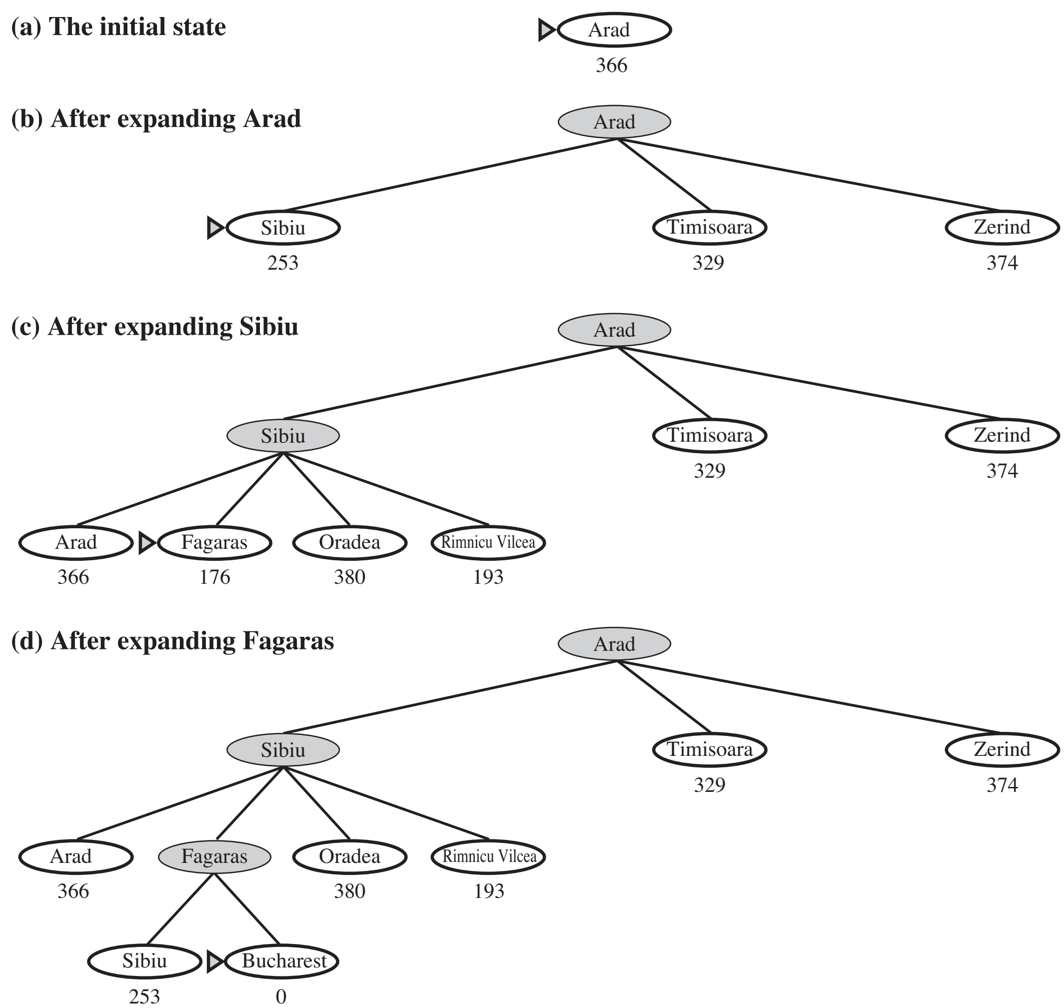


Figure 3.23 **FILES: figures/greedy-progress.eps (Tue Nov 3 16:22:55 2009).** Stages in a greedy best-first tree search for Bucharest with the straight-line distance heuristic h_{SLD}. Nodes are labeled with their h-values.

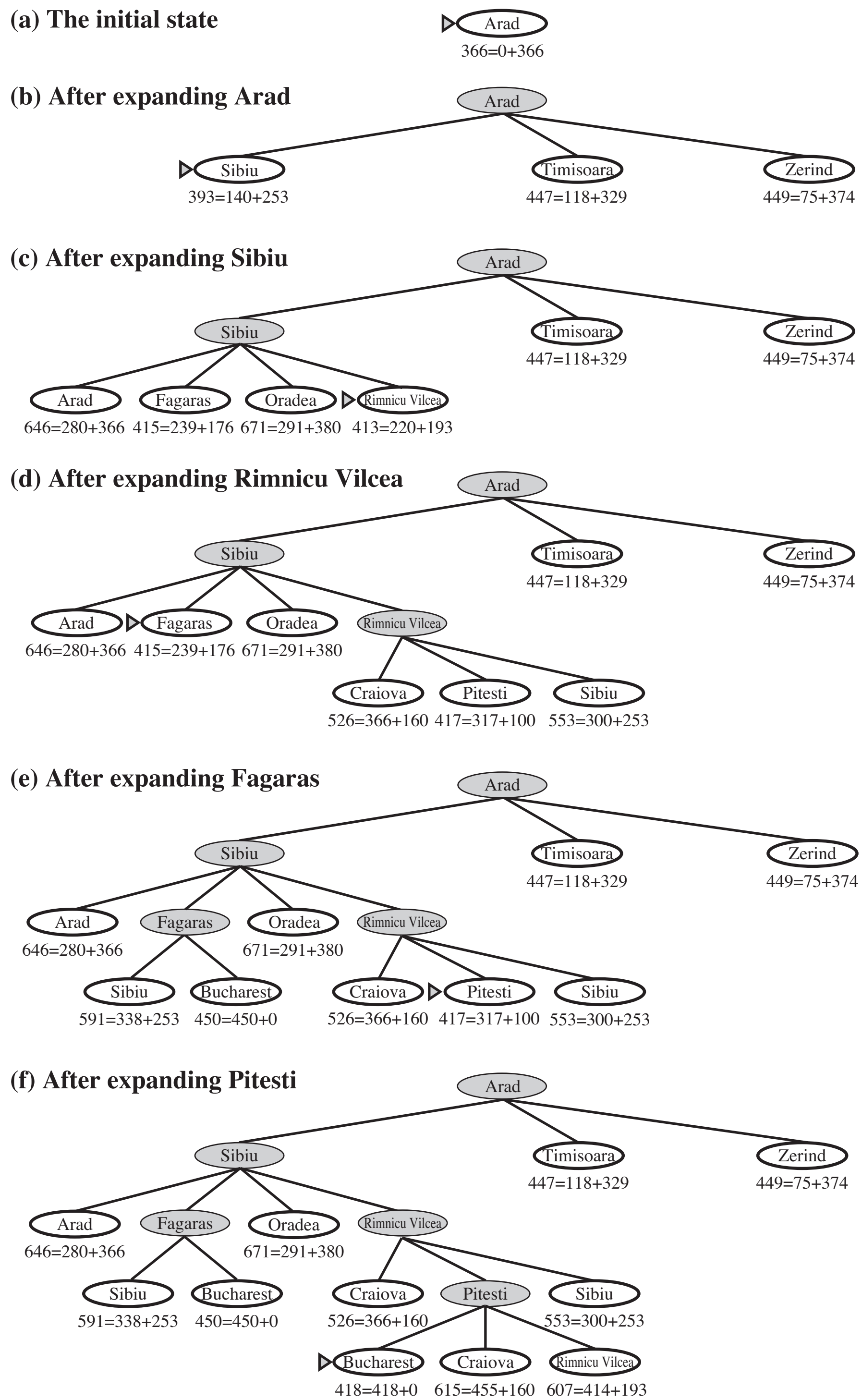


Figure 3.24 **FILES: figures/astar-progress.eps (Tue Nov 3 16:22:24 2009).** Stages in an A^* search for Bucharest. Nodes are labeled with $f = g + h$. The h values are the straight-line distances to Bucharest taken from Figure 3.20.

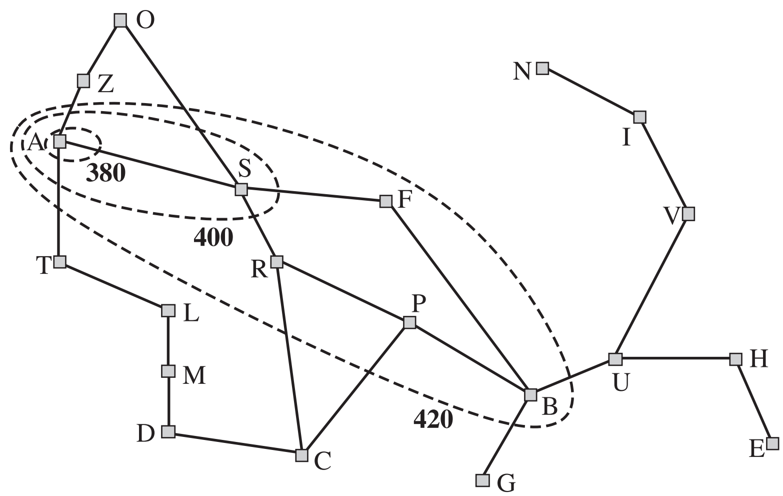


Figure 3.25 FILES: figures/f-circles.eps (Tue Nov 3 16:22:45 2009). Map of Romania showing contours at $f = 380$, $f = 400$, and $f = 420$, with Arad as the start state. Nodes inside a given contour have f-costs less than or equal to the contour value.

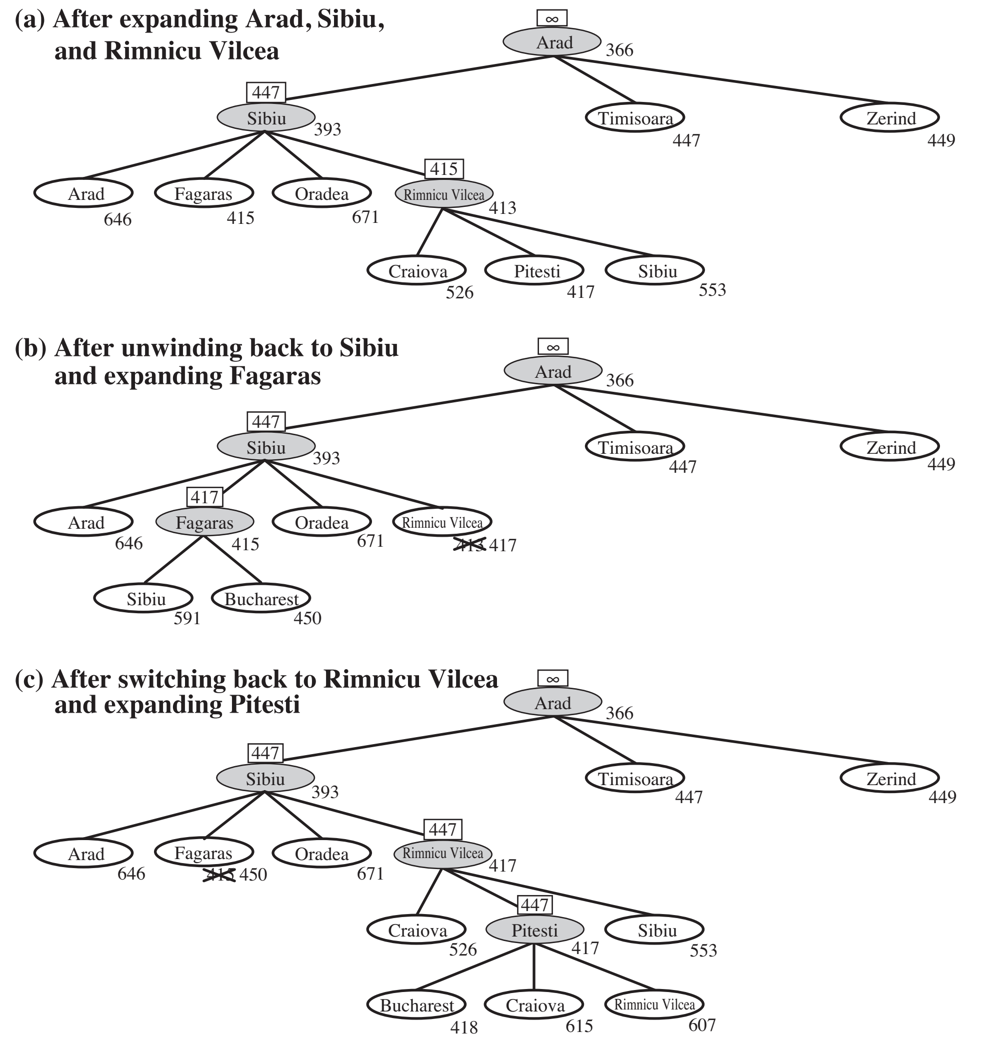


Figure 3.27 FILES: figures/rbfs-progress.eps (Tue Nov 3 16:23:27 2009). Stages in an RBFS search for the shortest route to Bucharest. The f-limit value for each recursive call is shown on top of each current node, and every node is labeled with its f-cost. (a) The path via Rimnicu Vilcea is followed until the current best leaf (Pitesti) has a value that is worse than the best alternative path (Fagaras). (b) The recursion unwinds and the best leaf value of the forgotten subtree (417) is backed up to Rimnicu Vilcea; then Fagaras is expanded, revealing a best leaf value of 450. (c) The recursion unwinds and the best leaf value of the forgotten subtree (450) is backed up to Fagaras; then Rimnicu Vilcea is expanded. This time, because the best alternative path (through Timisoara) costs at least 447, the expansion continues to Bucharest.

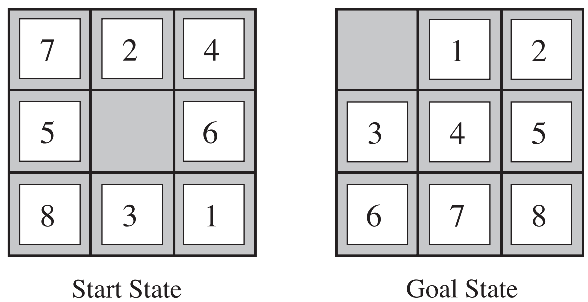


Figure 3.28 FILES: figures/8puzzle.eps (Tue Nov 3 16:22:11 2009). A typical instance of the 8-puzzle. The solution is 26 steps long.

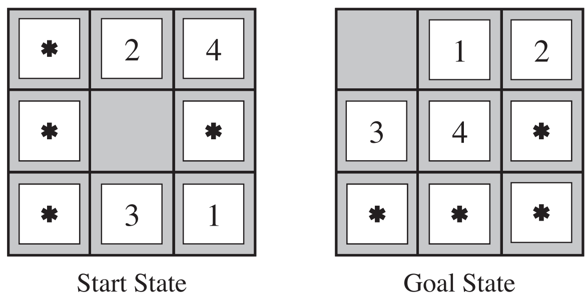


Figure 3.30 **FILES: figures/8puzzle-pattern.eps (Tue Nov 3 16:22:11 2009).** A subproblem of the 8-puzzle instance given in Figure 3.26. The task is to get tiles 1, 2, 3, and 4 into their correct positions, without worrying about what happens to the other tiles.

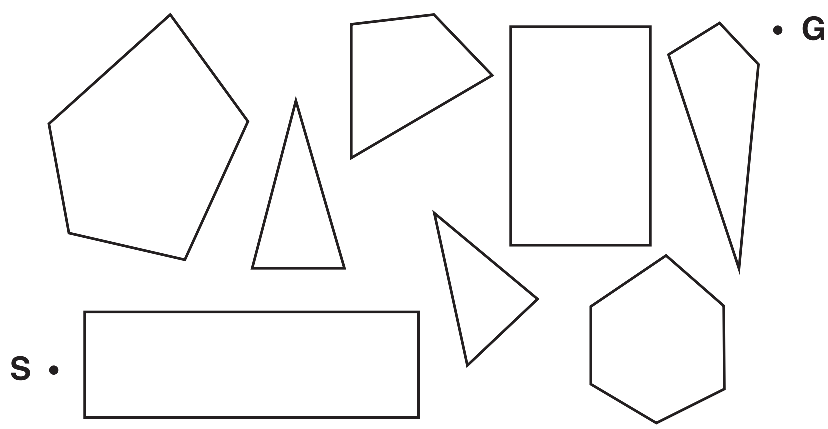


Figure 3.31 **FILES: figures/geometric-scene.eps (Tue Nov 3 16:22:54 2009).** A scene with polygonal obstacles. S and G are the start and goal states.

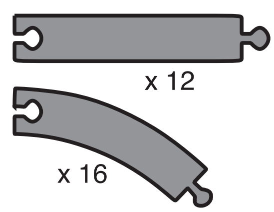


Figure 3.32 FILES: figures/brio.eps (Wed Nov 4 14:35:23 2009). The track pieces in a wooden railway set; each is labeled with the number of copies in the set. Note that curved pieces and “fork” pieces (“switches” or “points”) can be flipped over so they can curve in either direction. Each curve subtends 45 degrees.

4 BEYOND CLASSICAL SEARCH

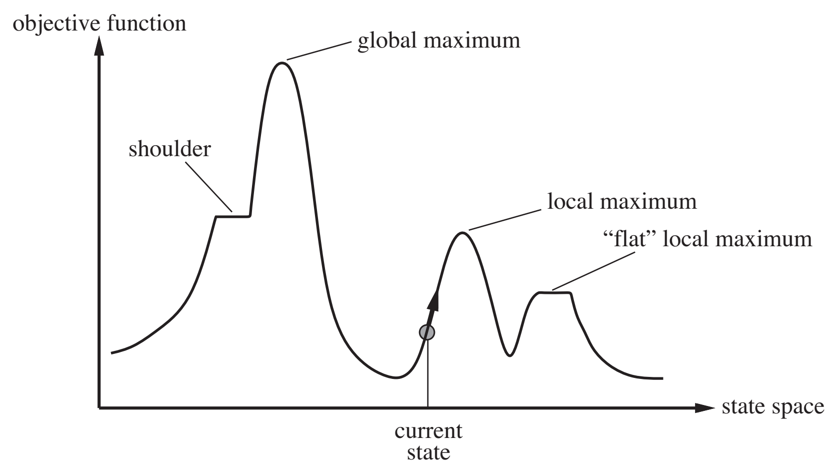
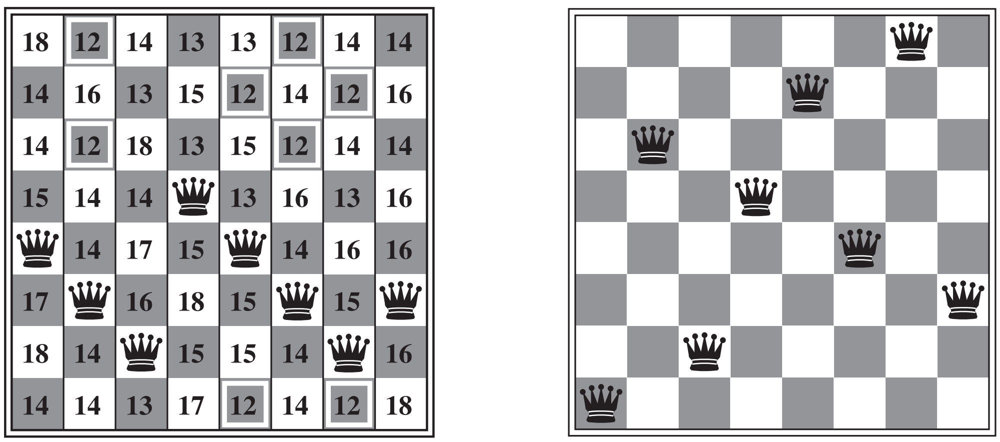


Figure 4.1 **FILES: figures/hill-climbing.eps (Tue Nov 3 16:23:03 2009).** A one-dimensional state-space landscape in which elevation corresponds to the objective function. The aim is to find the global maximum. Hill-climbing search modifies the current state to try to improve it, as shown by the arrow. The various topographic features are defined in the text.



(a) (b)

Figure 4.3 FILES: figures/8queens-successors.eps (Wed Nov 4 16:23:55 2009) figures/8queens-local-minimum.eps (Wed Nov 4 16:14:15 2009). (a) An 8-queens state with heuristic cost estimate $h = 17$, showing the value of h for each possible successor obtained by moving a queen within its column. The best moves are marked. (b) A local minimum in the 8-queens state space; the state has $h = 1$ but every successor has a higher cost.

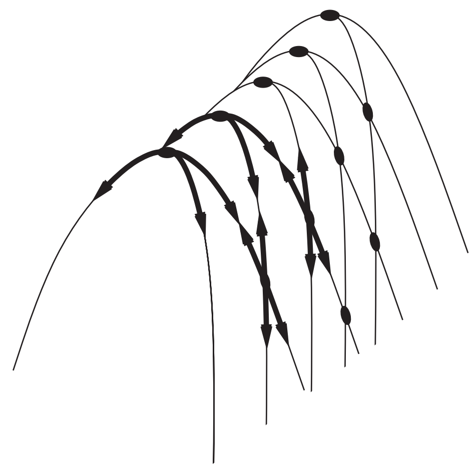


Figure 4.4 FILES: figures/ridge.eps (Tue Nov 3 16:23:29 2009). Illustration of why ridges cause difficulties for hill climbing. The grid of states (dark circles) is superimposed on a ridge rising from left to right, creating a sequence of local maxima that are not directly connected to each other. From each local maximum, all the available actions point downhill.

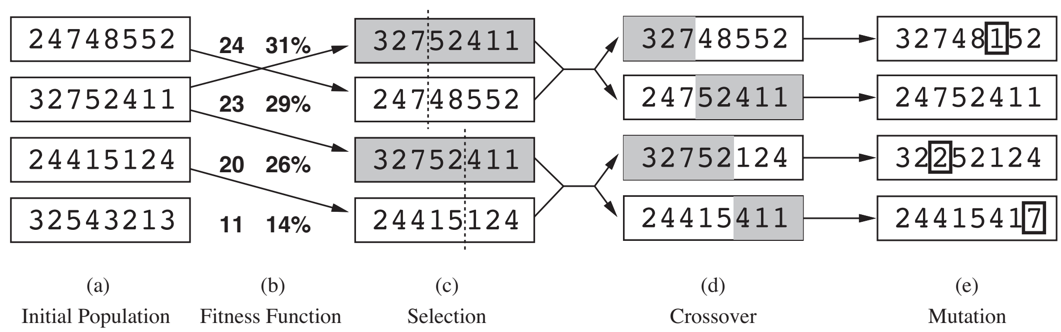


Figure 4.6 FILES: figures/genetic.eps (Tue Nov 3 16:22:53 2009). The genetic algorithm, illustrated for digit strings representing 8-queens states. The initial population in (a) is ranked by the fitness function in (b), resulting in pairs for mating in (c). They produce offspring in (d), which are subject to mutation in (e).

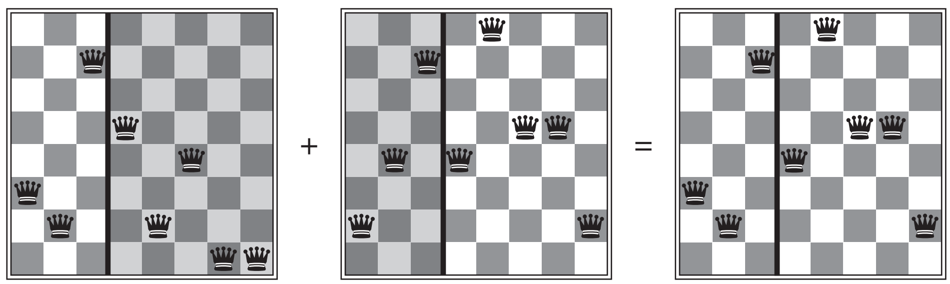


Figure 4.7 FILES: figures/8queens-crossover.eps (Wed Nov 4 16:11:32 2009). The 8-queens states corresponding to the first two parents in Figure 4.6(c) and the first offspring in Figure 4.6(d). The shaded columns are lost in the crossover step and the unshaded columns are retained.

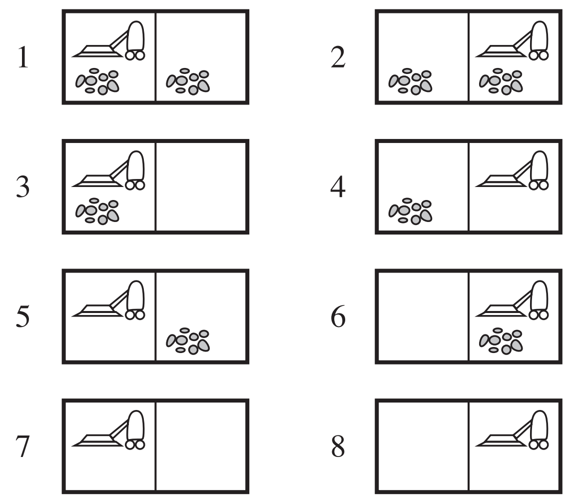


Figure 4.9 **FILES: figures/vacuum2-states.eps (Tue Nov 3 16:24:02 2009).** The eight possible states of the vacuum world; states 7 and 8 are goal states.

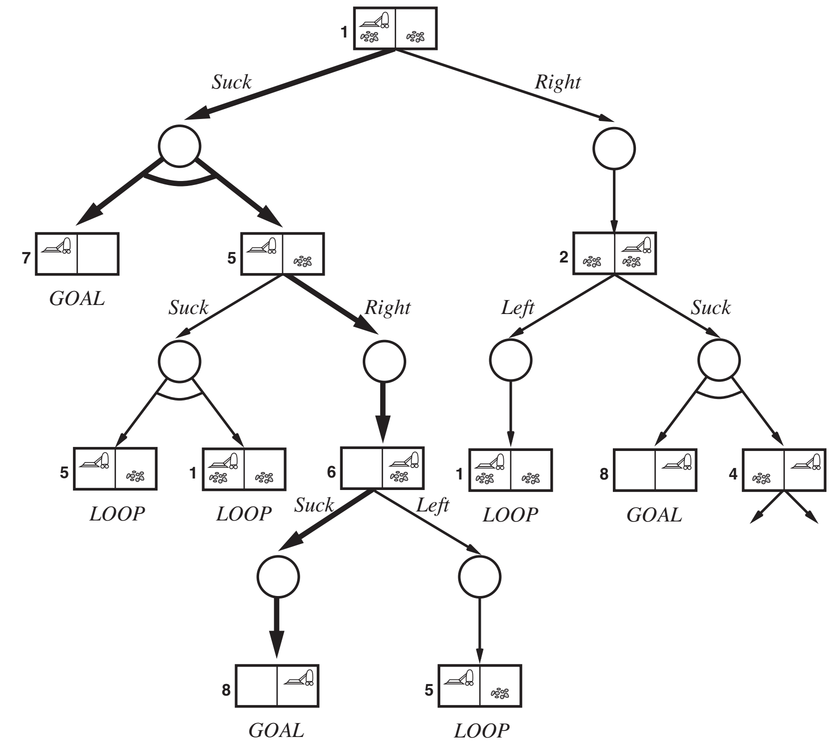


Figure 4.10 **FILES: figures/erratic-vacuum-and-or-plan.eps (Tue Nov 3 13:32:58 2009).** The first two levels of the search tree for the erratic vacuum world. State nodes are OR nodes where some action must be chosen. At the AND nodes, shown as circles, every outcome must be handled, as indicated by the arc linking the outgoing branches. The solution found is shown in bold lines.

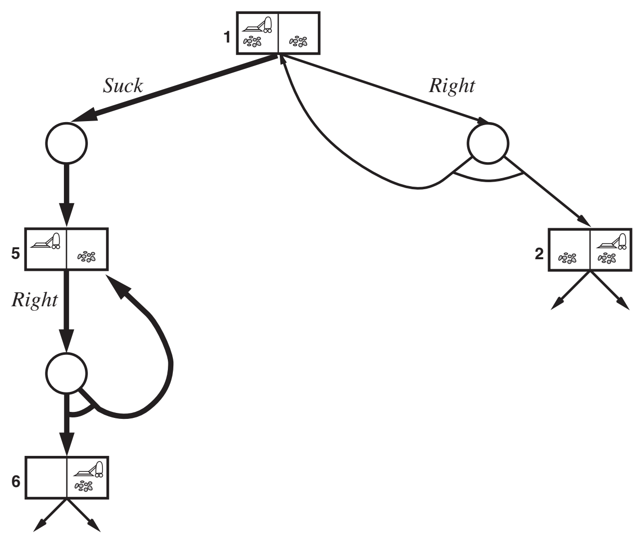


Figure 4.12 **FILES: figures/slippery-vacuum-loop-plan.eps (Tue Nov 3 13:48:56 2009).** Part of the search graph for the slippery vacuum world, where we have shown (some) cycles explicitly. All solutions for this problem are cyclic plans because there is no way to move reliably.

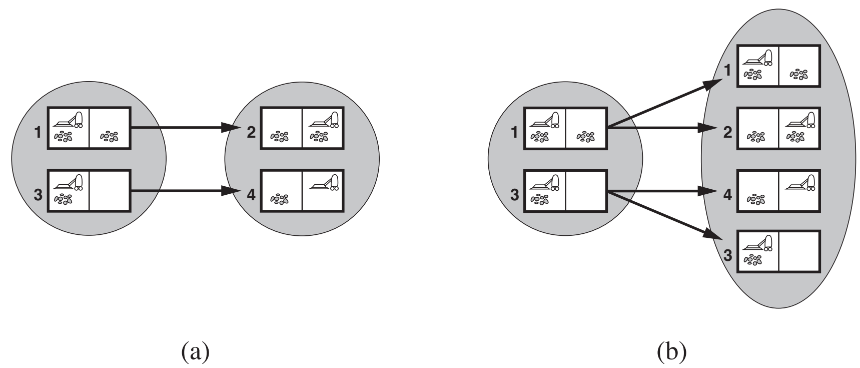


Figure 4.13 FILES: figures/vacuum-prediction.eps (Tue Nov 3 13:51:56 2009). (a) Predicting the next belief state for the sensorless vacuum world with a deterministic action, $Right$. (b) Prediction for the same belief state and action in the slippery version of the sensorless vacuum world.

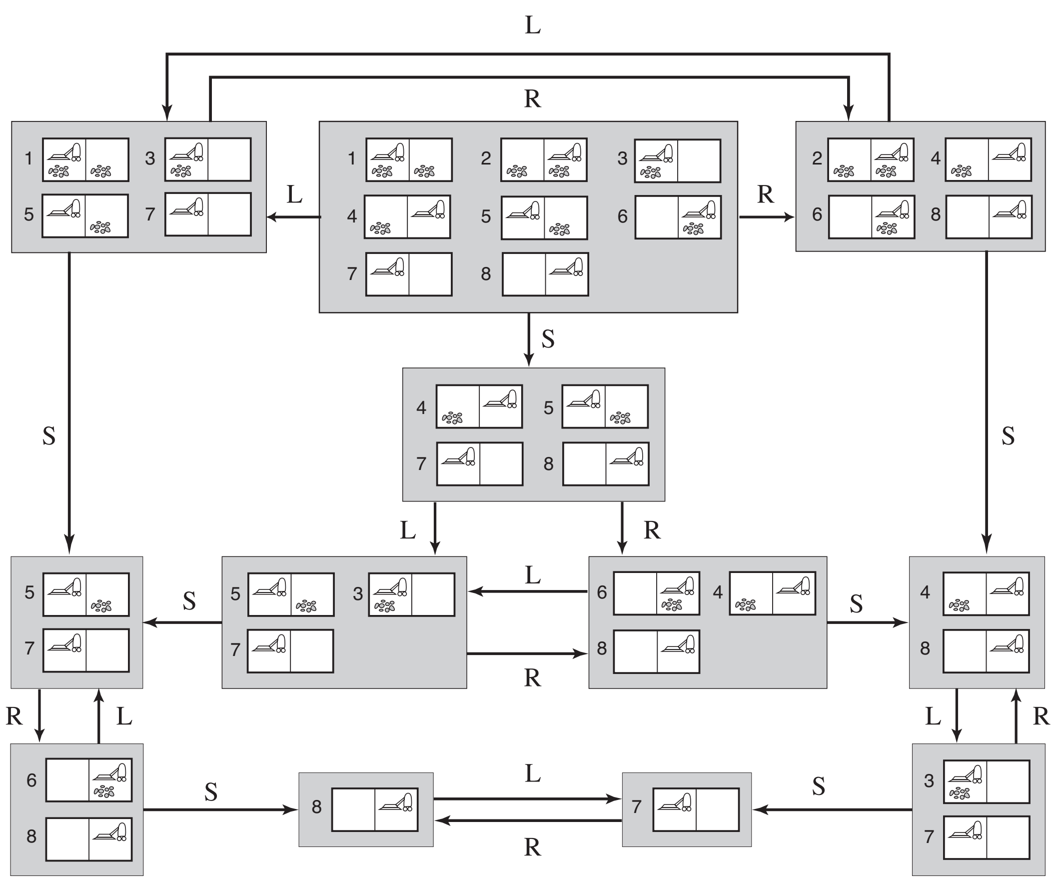


Figure 4.14 FILES: figures/vacuum2-sets.eps (Tue Nov 3 16:24:01 2009). The reachable portion of the belief-state space for the deterministic, sensorless vacuum world. Each shaded box corresponds to a single belief state. At any given point, the agent is in a particular belief state but does not know 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.

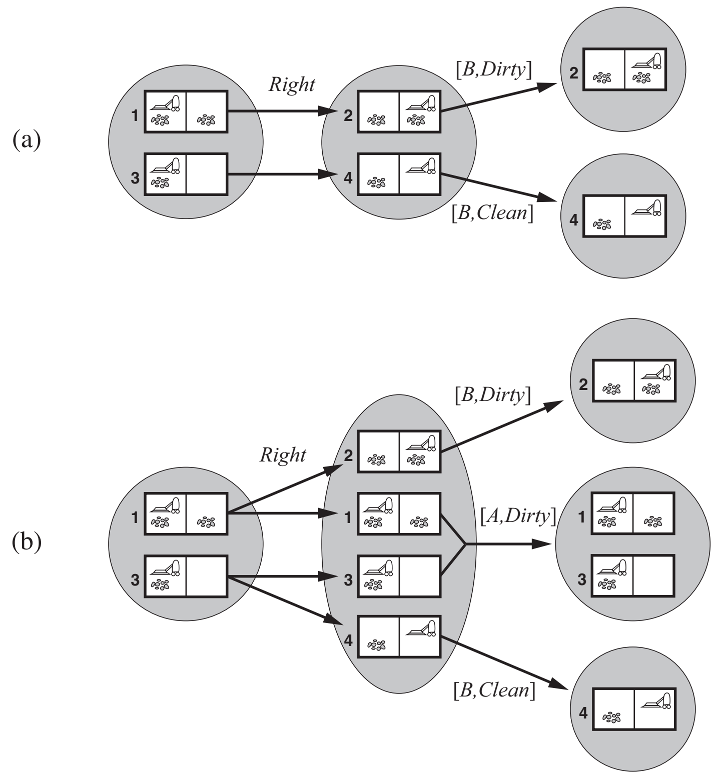


Figure 4.15 FILES: figures/vacuum-prediction-update.eps (Tue Nov 3 13:52:01 2009). Two example of transitions in local-sensing vacuum worlds. (a) In the deterministic world, *Right* is applied in the initial belief state, resulting in a new belief state with two possible physical states; for those states, the possible percepts are $[B, Dirty]$ and $[B, Clean]$, leading to two belief states, each of which is a singleton. (b) In the slippery world, *Right* is applied in the initial belief state, giving a new belief state with four physical states; for those states, the possible percepts are $[A, Dirty]$, $[B, Dirty]$, and $[B, Clean]$, leading to three belief states as shown.

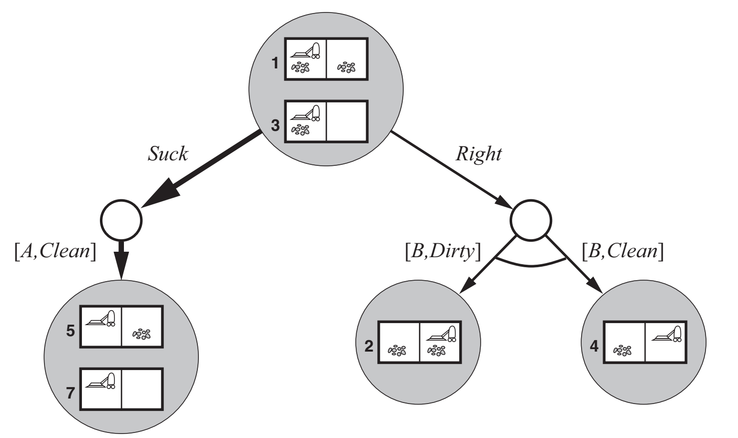


Figure 4.16 FILES: figures/local-sensing-vacuum-and-or.eps (Tue Nov 3 13:42:56 2009). The first level of the AND–OR search tree for a problem in the local-sensing vacuum world; *Suck* is the first step of the solution.

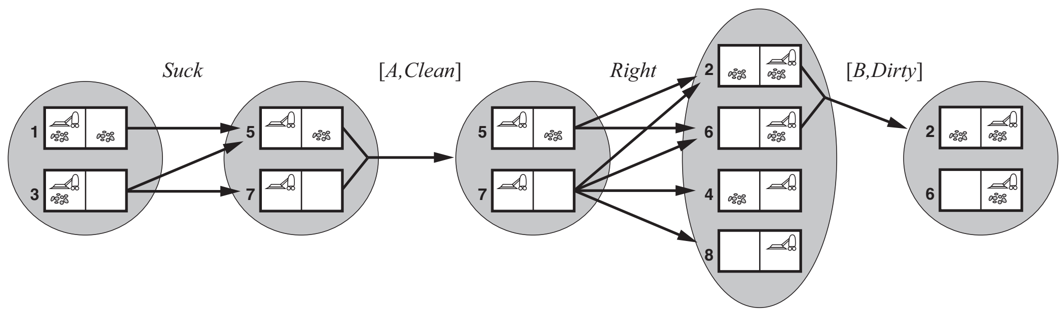


Figure 4.17 FILES: figures/kindergarten-vacuum-filtering.eps (Tue Nov 3 13:41:48 2009). Two prediction–update cycles of belief-state maintenance in the kindergarten vacuum world with local sensing.

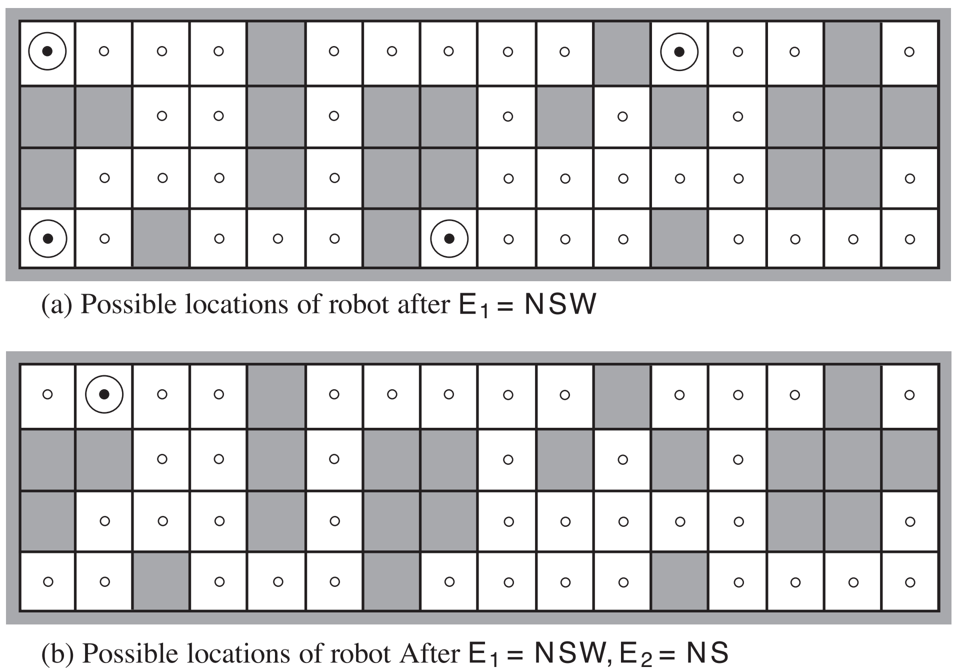


Figure 4.18 FILES: figures/localization-figures-a.eps (Tue Nov 3 16:23:06 2009). Possible positions of the robot, $\odot$, (a) after one observation $E_1 = NSW$ and (b) after a second observation $E_2 = NS$. When sensors are noiseless and the transition model is accurate, there are no other possible locations for the robot consistent with this sequence of two observations.

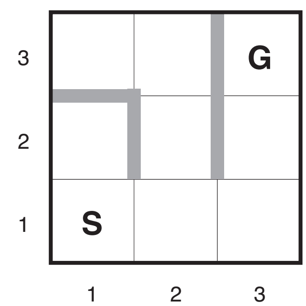


Figure 4.19 FILES: figures/maze-3x3.eps (Tue Nov 3 16:23:08 2009). A simple maze problem. The agent starts at S and must reach G but knows nothing of the environment.

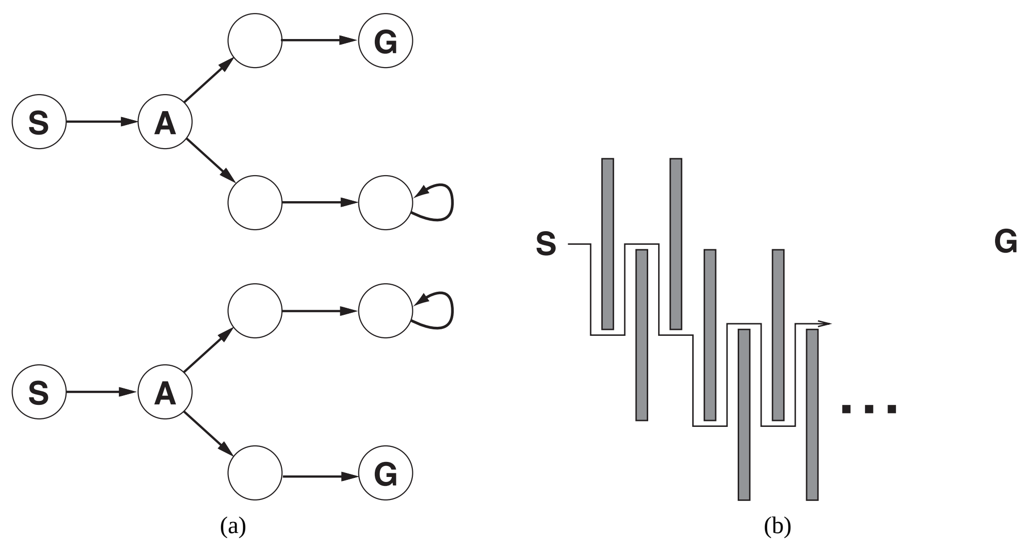


Figure 4.20 FILES: figures/adversary-spaces.eps (Tue Nov 3 16:22:18 2009) figures/adversary-blocks.eps (Sun Oct 25 01:08:26 2009). (a) Two state spaces that might lead an online search agent into a dead end. Any given agent will fail in at least one of these spaces. (b) A two-dimensional environment that can cause an online search agent to follow an arbitrarily inefficient route to the goal. Whichever choice the agent makes, the adversary blocks that route with another long, thin wall, so that the path followed is much longer than the best possible path.

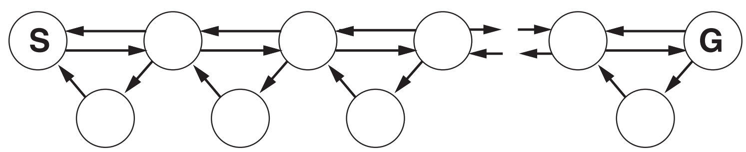


Figure 4.22 **FILES: figures/quicksand.eps (Tue Nov 3 13:45:58 2009).** An environment in which a random walk will take exponentially many steps to find the goal.

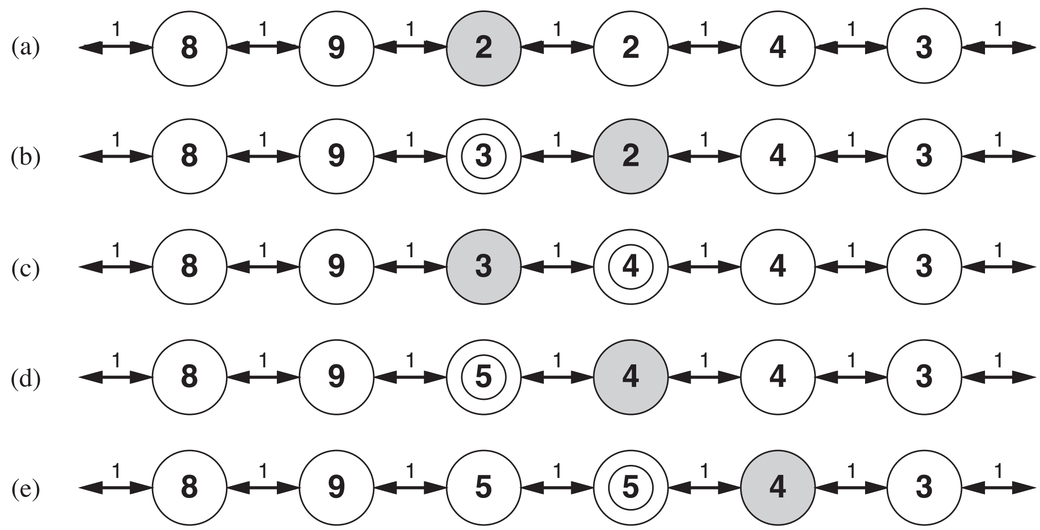


Figure 4.23 FILES: figures/lrta-progress.eps (Tue Nov 3 16:23:08 2009). Five iterations of LRTA* on a one-dimensional state space. Each state is labeled with $H(s)$, the current cost estimate to reach a goal, and each link is labeled with its step cost. The shaded state marks the location of the agent, and the updated cost estimates at each iteration are circled.

5 ADVERSARIAL SEARCH

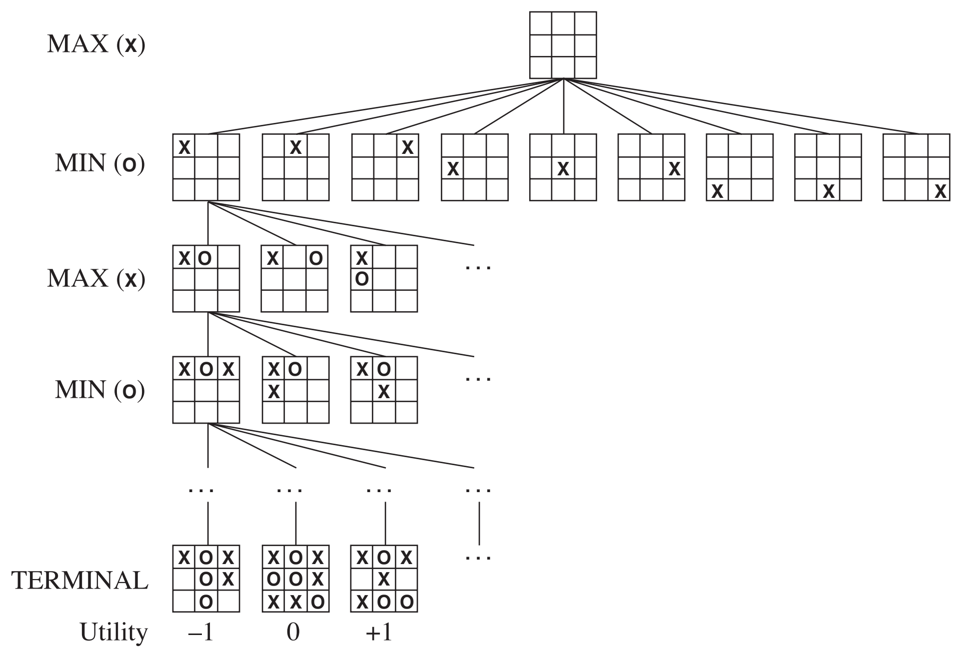


Figure 5.1 **FILES: figures/tictactoe.eps (Tue Nov 3 16:23:55 2009).** A (partial) game tree for the game of tic-tac-toe. The top node is the initial state, and MAX moves first, placing an X in an empty square. We show part of the tree, giving alternating moves by MIN (O) and MAX (X), until we eventually reach terminal states, which can be assigned utilities according to the rules of the game.

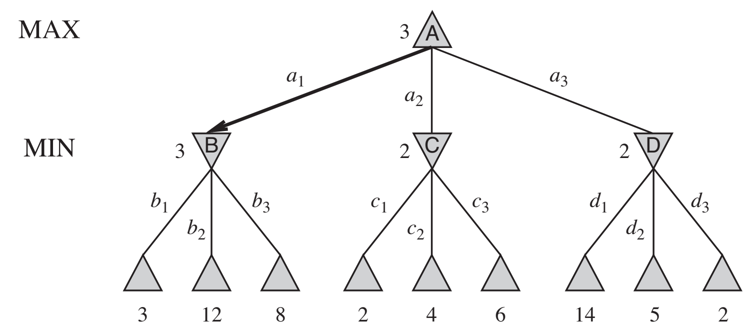


Figure 5.2 FILES: figures/minimax.eps (Tue Nov 3 16:23:11 2009). A two-ply game tree. The △ nodes are "MAX nodes," in which it is MAX's turn to move, and the ▽ nodes are "MIN nodes." The terminal nodes show the utility values for MAX; the other nodes are labeled with their minimax values. MAX's best move at the root is a_1, because it leads to the state with the highest minimax value, and MIN's best reply is b_1, because it leads to the state with the lowest minimax value.

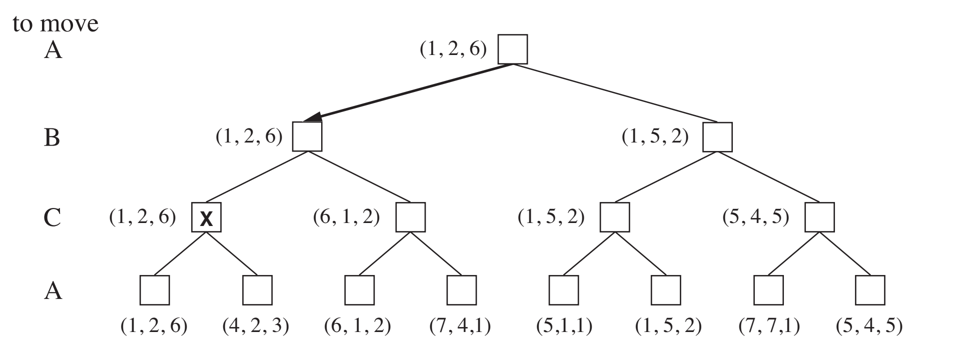


Figure 5.4 **FILES: figures/minimax3.eps (Tue Nov 3 16:23:11 2009).** The first three plies of a game tree with three players (A, B, C). Each node is labeled with values from the viewpoint of each player. The best move is marked at the root.

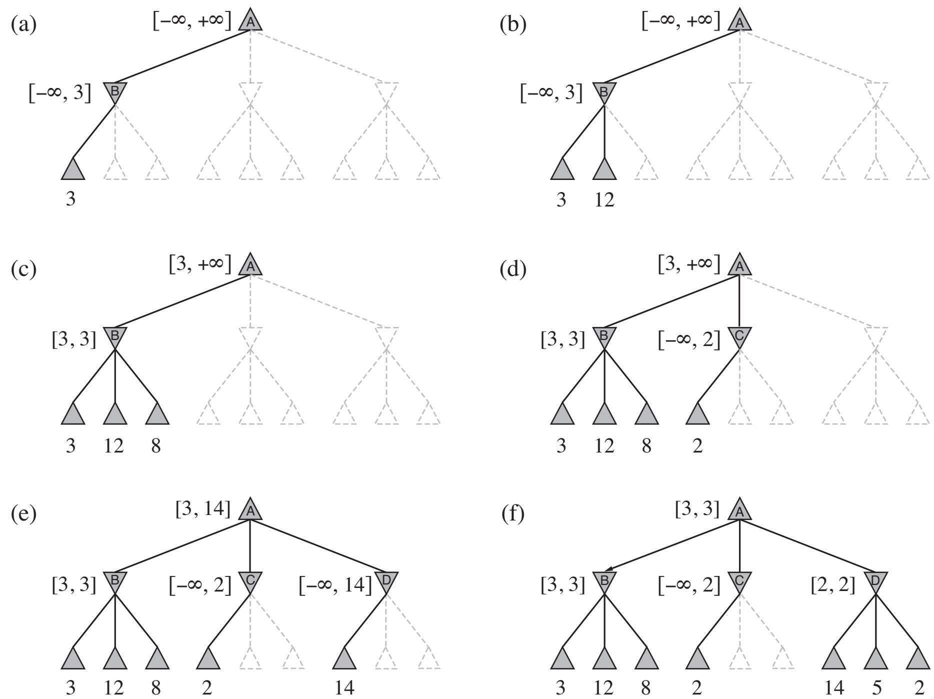


Figure 5.5 FILES: figures/alpha-beta-progress.eps (Tue Nov 3 16:22:20 2009). Stages in the calculation of the optimal decision for the game tree in Figure 5.2. At each point, we show the range of possible values for each node. (a) The first leaf below B has the value 3. Hence, B, which is a MIN node, has a value of *at most* 3. (b) The second leaf below B has a value of 12; MIN would avoid this move, so the value of B is still at most 3. (c) The third leaf below B has a value of 8; we have seen all B's successor states, so the value of B is exactly 3. Now, we can infer that the value of the root is *at least* 3, because MAX has a choice worth 3 at the root. (d) The first leaf below C has the value 2. Hence, C, which is a MIN node, has a value of *at most* 2. But we know that B is worth 3, so MAX would never choose C. Therefore, there is no point in looking at the other successor states of C. This is an example of alpha–beta pruning. (e) The first leaf below D has the value 14, so D is worth *at most* 14. This is still higher than MAX's best alternative (i.e., 3), so we need to keep exploring D's successor states. Notice also that we now have bounds on all of the successors of the root, so the root's value is also at most 14. (f) The second successor of D is worth 5, so again we need to keep exploring. The third successor is worth 2, so now D is worth exactly 2. MAX's decision at the root is to move to B, giving a value of 3.

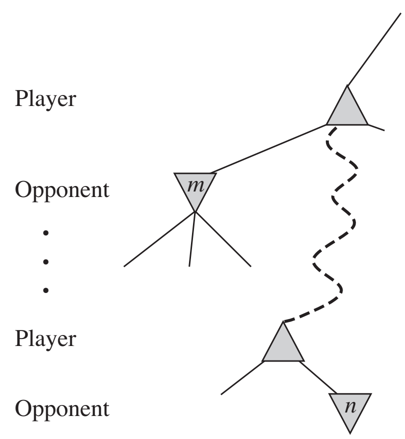


Figure 5.6 FILES: figures/alpha-beta-general.eps (Tue Nov 3 16:22:20 2009). The general case for alpha–beta pruning. If m is better than n for Player, we will never get to n in play.

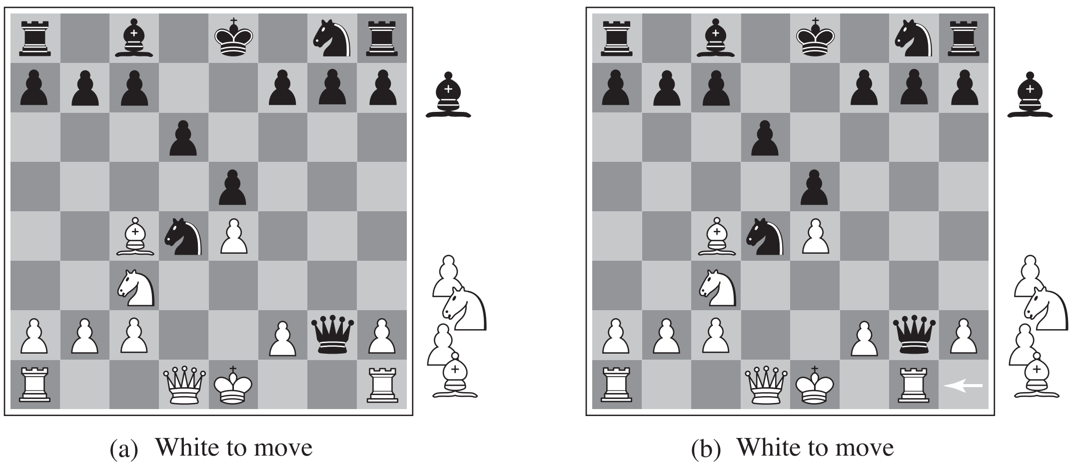


Figure 5.8 FILES: figures/chess-evaluation3.eps (Tue Nov 3 16:22:33 2009). Two chess positions that differ only in the position of the rook at lower right. In (a), Black has an advantage of a knight and two pawns, which should be enough to win the game. In (b), White will capture the queen, giving it an advantage that should be strong enough to win.

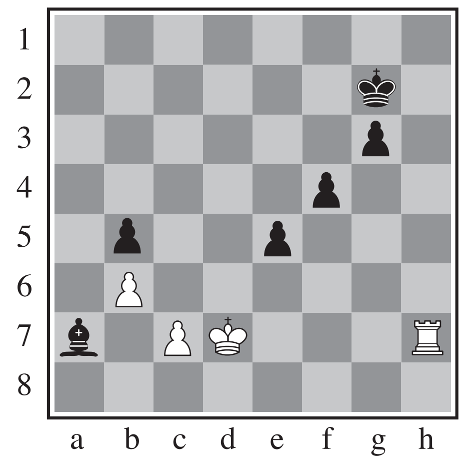


Figure 5.9 **FILES: figures/horizon.eps (Tue Nov 3 16:23:03 2009).** The horizon effect. With Black to move, the black bishop is surely doomed. But Black can forestall that event by checking the white king with its pawns, forcing the king to capture the pawns. This pushes the inevitable loss of the bishop over the horizon, and thus the pawn sacrifices are seen by the search algorithm as good moves rather than bad ones.

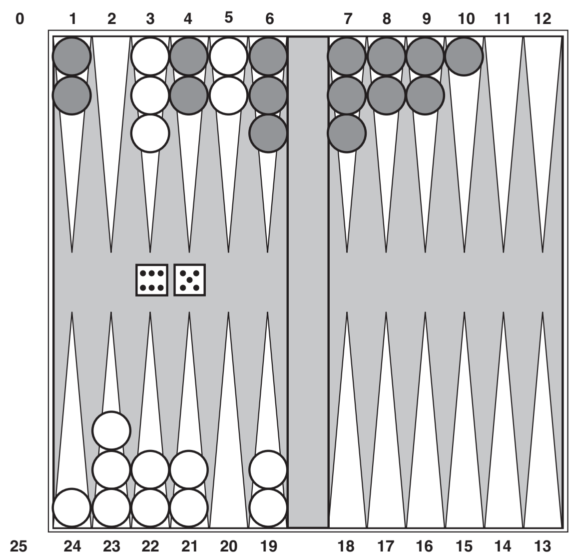


Figure 5.10 FILES: figures/backgammon-position.eps (Tue Nov 3 16:22:26 2009). A typical backgammon position. The goal of the game is to move all one's pieces off the board. White moves clockwise toward 25, and Black moves counterclockwise toward 0. A piece can move to any position unless multiple opponent pieces are there; if there is one opponent, it is captured and must start over. In the position shown, White has rolled 6–5 and must choose among four legal moves: (5–10,5–11), (5–11,19–24), (5–10,10–16), and (5–11,11–16), where the notation (5–11,11–16) means move one piece from position 5 to 11, and then move a piece from 11 to 16.

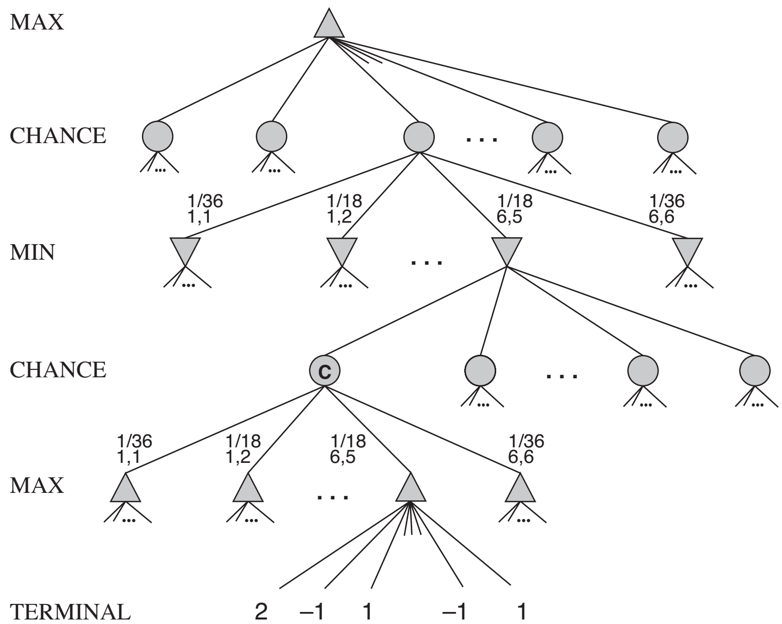


Figure 5.11 FILES: figures/backgammon-tree.eps (Tue Nov 3 16:22:26 2009). Schematic game tree for a backgammon position.

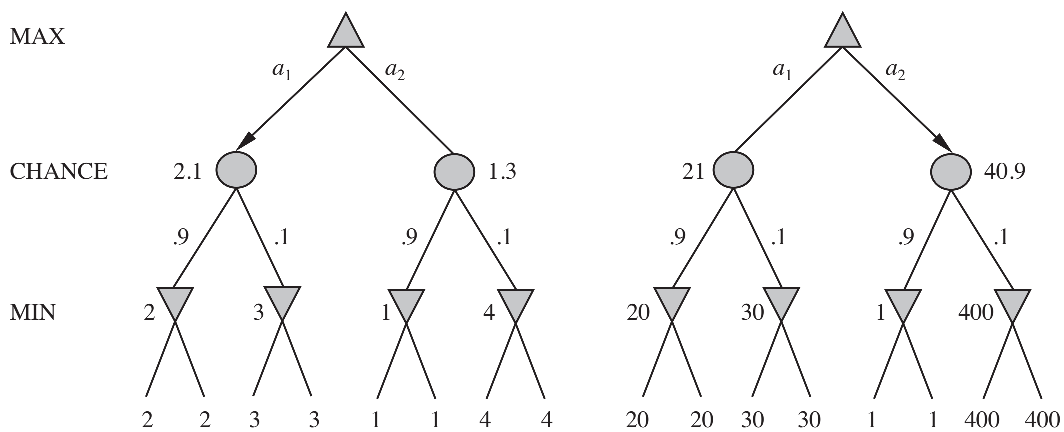


Figure 5.12 FILES: figures/chance-evaluation.eps (Tue Nov 3 16:22:32 2009). An order-preserving transformation on leaf values changes the best move.

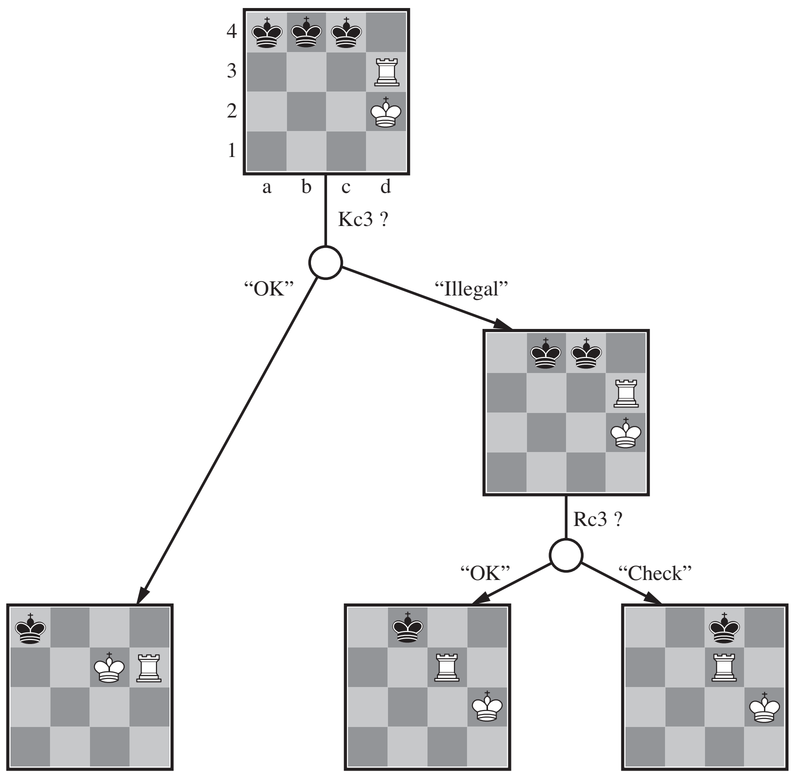


Figure 5.13 **FILES: figures/kriegspiel-krk.eps (Tue Nov 3 13:41:43 2009).** Part of a guaranteed checkmate in the KRK endgame, shown on a reduced board. In the initial belief state, Black’s king is in one of three possible locations. By a combination of probing moves, the strategy narrows this down to one. Completion of the checkmate is left as an exercise.

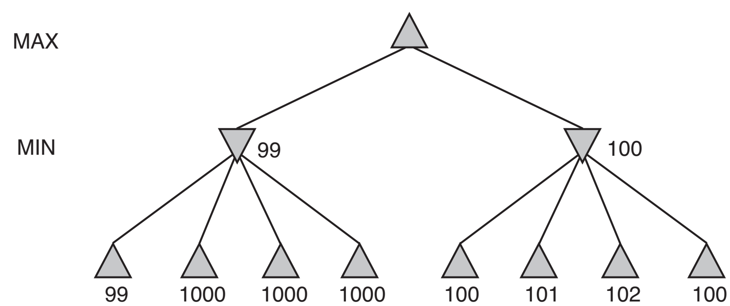


Figure 5.14 FILES: figures/minimax-error.eps (Tue Nov 3 16:23:11 2009). A two-ply game tree for which heuristic minimax may make an error.

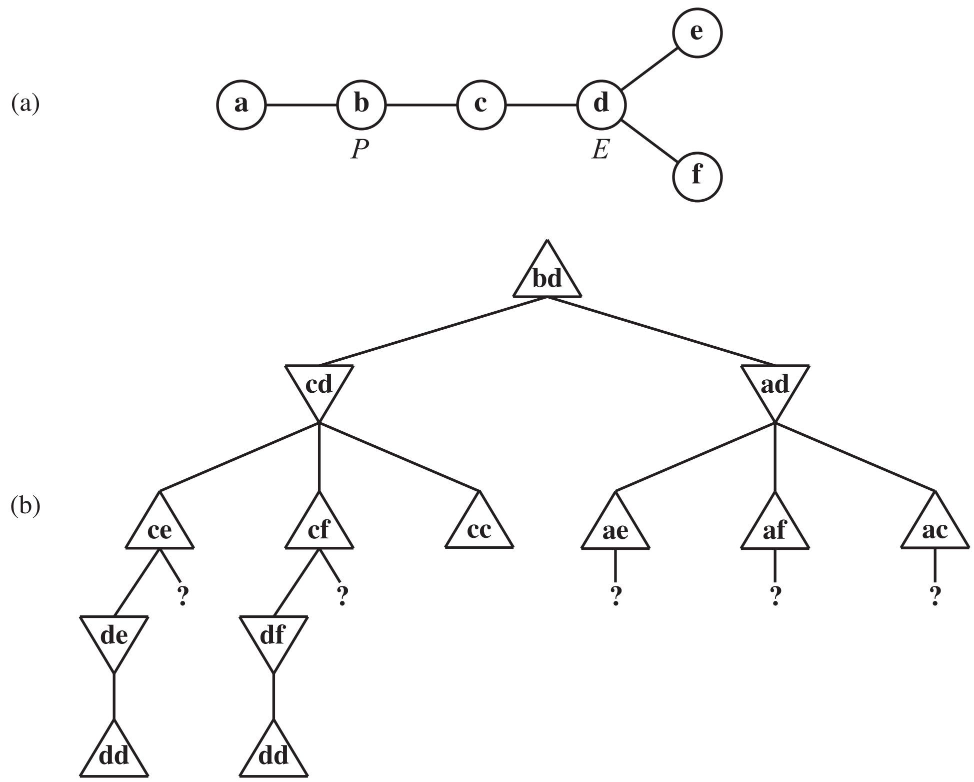


Figure 5.15 FILES: figures/pursuit-evasion-game.eps (Tue Nov 3 13:46:04 2009). (a) A map where the cost of every edge is 1. Initially the pursuer P is at node **b** and the evader E is at node **d**. (b) A partial game tree for this map. Each node is labeled with the P, E positions. P moves first. Branches marked “?” have yet to be explored.

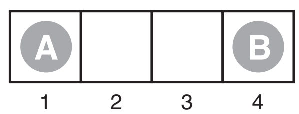


Figure 5.16 FILES: figures/line-game4.eps (Tue Nov 3 16:23:06 2009). The starting position of a simple game. Player A moves first. The two players take turns moving, and each player must move his token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then a player 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.

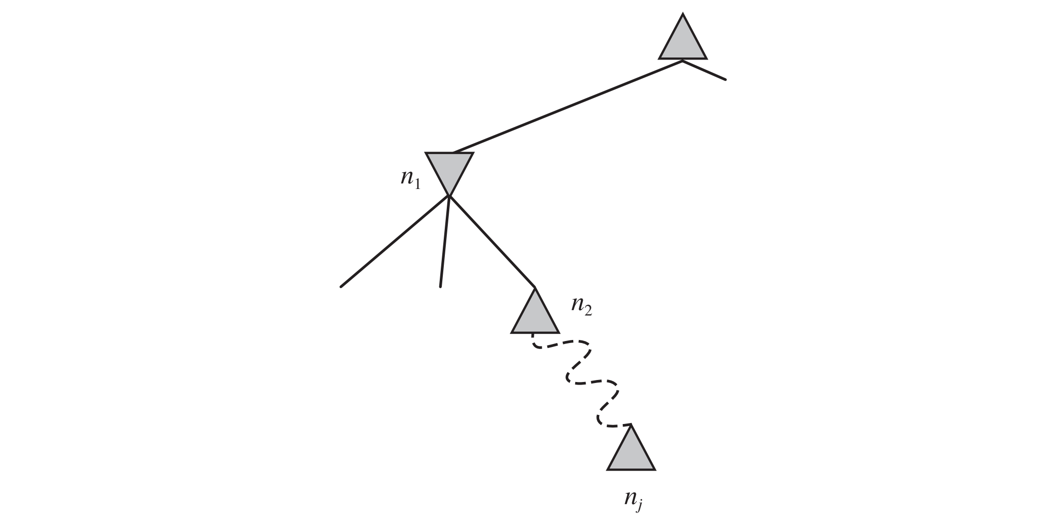


Figure 5.17 **FILES: figures/alpha-beta-proof.eps (Tue Nov 3 16:22:21 2009).** Situation when considering whether to prune node n_j.

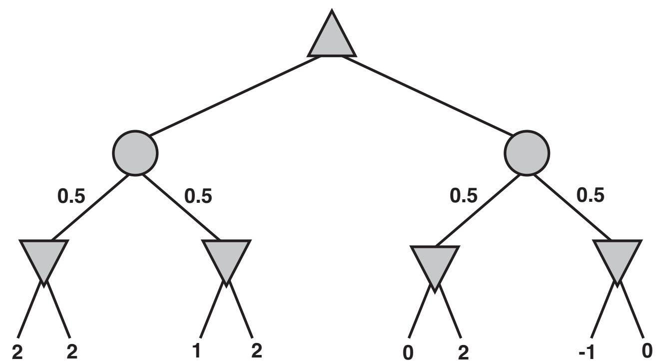


Figure 5.18 **FILES: figures/pruning.eps (Tue Nov 3 16:23:22 2009).** The complete game tree for a trivial game with chance nodes.

(a) (b)

Figure 5.19 FILES: figures/NewellSimonMcCarthy.eps (Tue Nov 3 16:22:16 2009). Pioneers in computer chess: (a) Herbert Simon and Allen Newell, developers of the NSS program (1958); (b) John McCarthy and the Kotok–McCarthy program on an IBM 7090 (1967).

6 CONSTRAINT SATISFACTION PROBLEMS

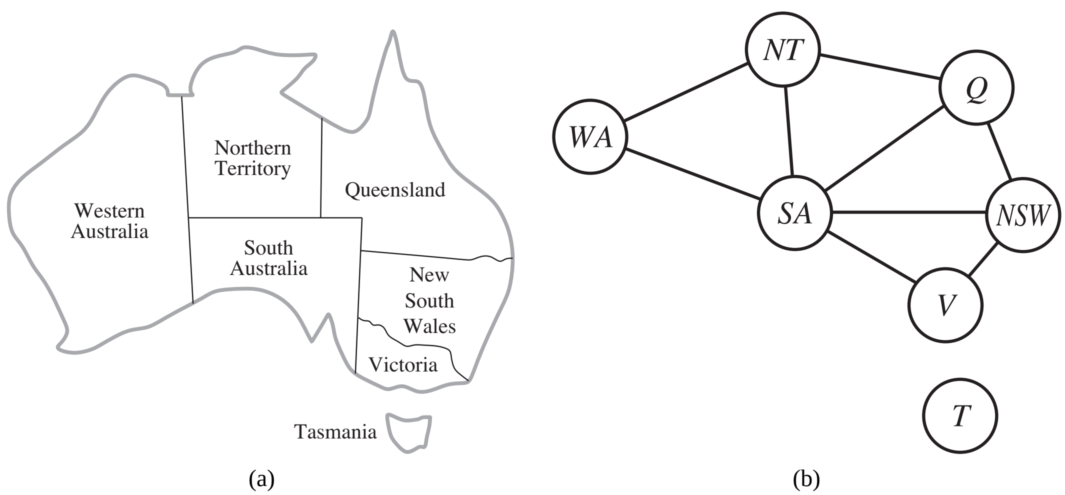


Figure 6.1 FILES: figures/australia.eps (Tue Nov 3 16:22:26 2009) figures/australia-csp.eps (Tue Nov 3 16:22:25 2009). (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

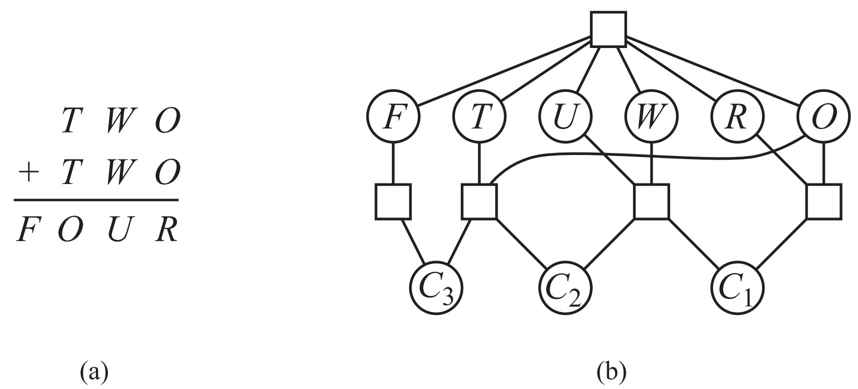


Figure 6.2 FILES: figures/cryptarithmetic.eps (Tue Nov 3 13:31:28 2009). (a) A cryptarithmetic problem. Each letter stands for a distinct digit; the aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct, with the added restriction that no leading zeroes 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.

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

(a)

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

(b)

Figure 6.4 FILES: figures/sudoku.eps (Tue Nov 3 13:49:46 2009). (a) A Sudoku puzzle and (b) its solution.

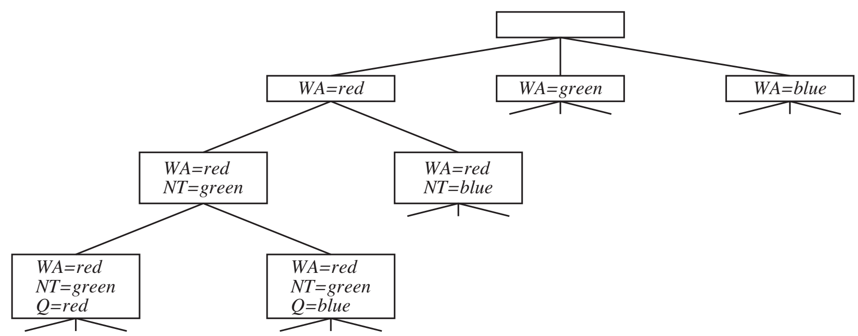


Figure 6.6 FILES: figures/australia-search.eps (Tue Nov 3 16:22:25 2009). Part of the search tree for the map-coloring problem in Figure 6.1.

	WA	*NT*	*Q*	*NSW*	*V*	*SA*	*T*
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After *WA*=*red*	(R)	G B	R G B	R G B	R G B	G B	R G B
After *Q*=*green*	(R)	B	(G)	R B	R G B	B	R G B
After *V*=*blue*	(R)	B	(G)	R	(B)		R G B

Figure 6.7 FILES: figures/australia-fc.eps (Tue Nov 3 16:22:25 2009). The progress of a map-coloring search with forward checking. $WA = red$ is assigned first; then forward checking deletes red from the domains of the neighboring variables NT and SA. After $Q = green$ is assigned, $green$ is deleted from the domains of NT, SA, and NSW. After $V = blue$ is assigned, $blue$ is deleted from the domains of NSW and SA, leaving SA with no legal values.

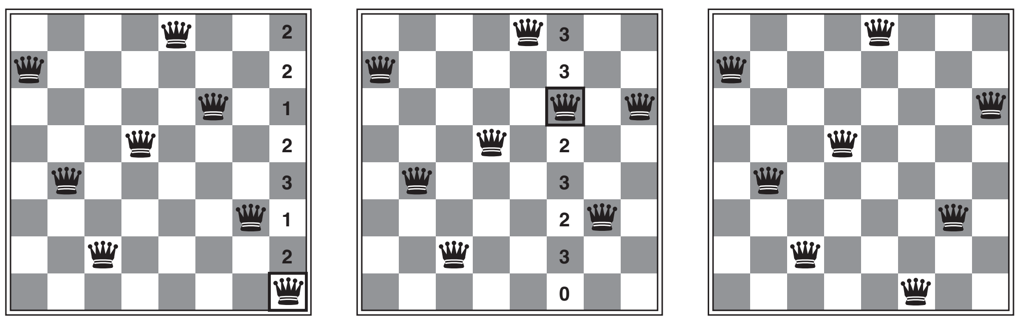


Figure 6.9 FILES: figures/8queens-min-conflicts.eps (Wed Nov 4 16:20:15 2009). A two-step solution using min-conflicts for an 8-queens problem. At each stage, a queen is chosen for reassignment in its column. The number of conflicts (in this case, the number of attacking queens) is shown in each square. The algorithm moves the queen to the min-conflicts square, breaking ties randomly.

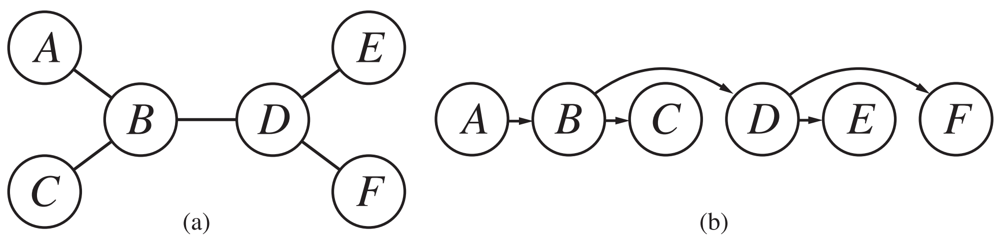


Figure 6.10 **FILES: figures/tree-csp.eps (Tue Nov 3 16:23:58 2009).** (a) The constraint graph of a tree-structured CSP. (b) A linear ordering of the variables consistent with the tree with A as the root. This is known as a **topological sort** of the variables.

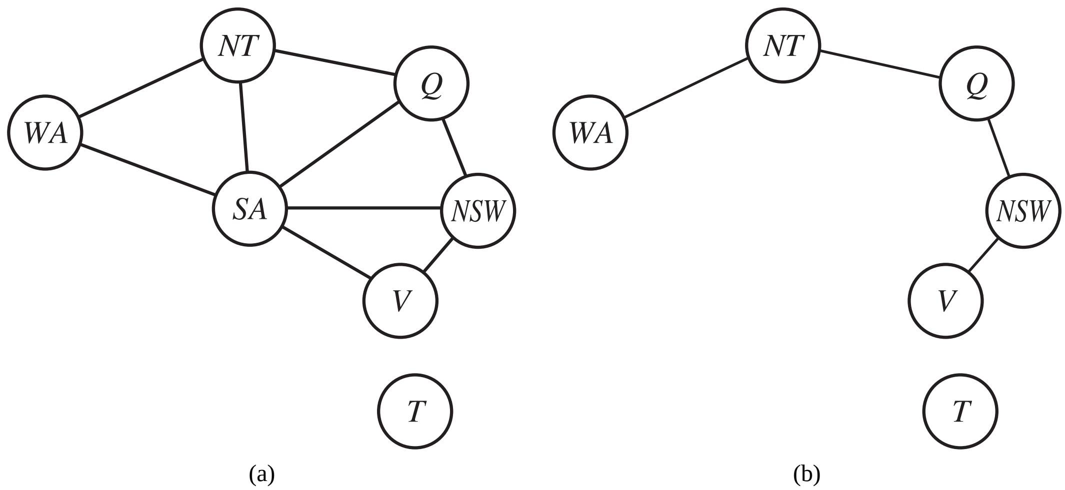


Figure 6.12 FILES: figures/australia-csp.eps (Tue Nov 3 16:22:25 2009) figures/australia-tree.eps (Tue Nov 3 16:22:26 2009). (a) The original constraint graph from Figure 6.1. (b) The constraint graph after the removal of SA.

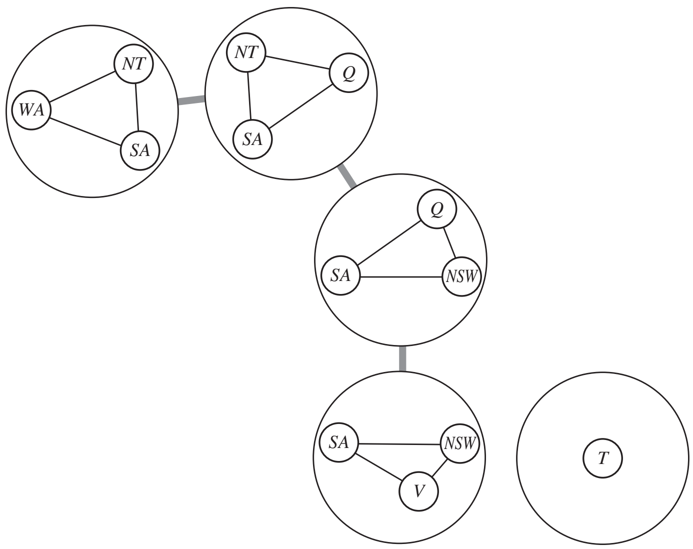


Figure 6.13 FILES: figures/australia-decomposition.eps (Tue Nov 3 16:22:25 2009). A tree decomposition of the constraint graph in Figure 6.12(a).

7 LOGICAL AGENTS

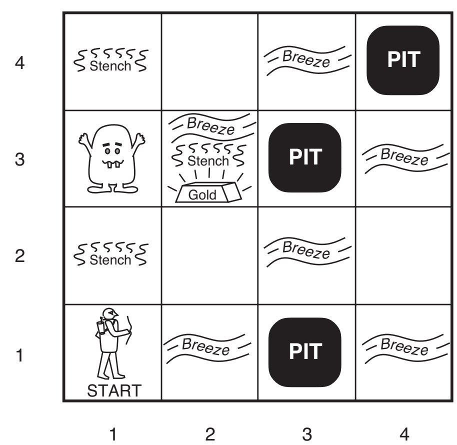


Figure 7.2 FILES: figures/wumpus-world.eps (Tue Nov 3 16:24:13 2009). A typical wumpus world. The agent is in the bottom left corner, facing right.

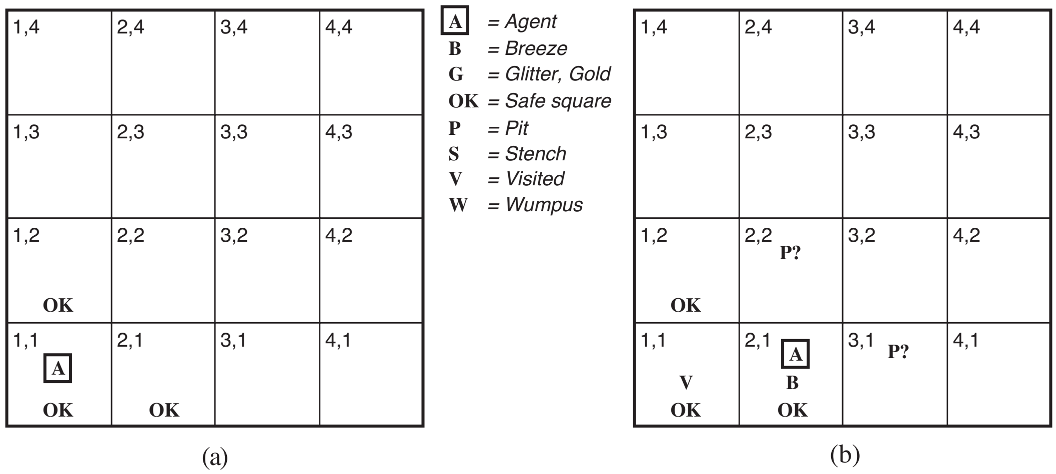


Figure 7.3 FILES: figures/wumpus-seq01.eps (Tue Nov 3 16:24:10 2009). The first step taken by the agent in the wumpus world. (a) The initial situation, after percept $[None, None, None, None, None]$. (b) After one move, with percept $[None, Breeze, None, None, None]$.

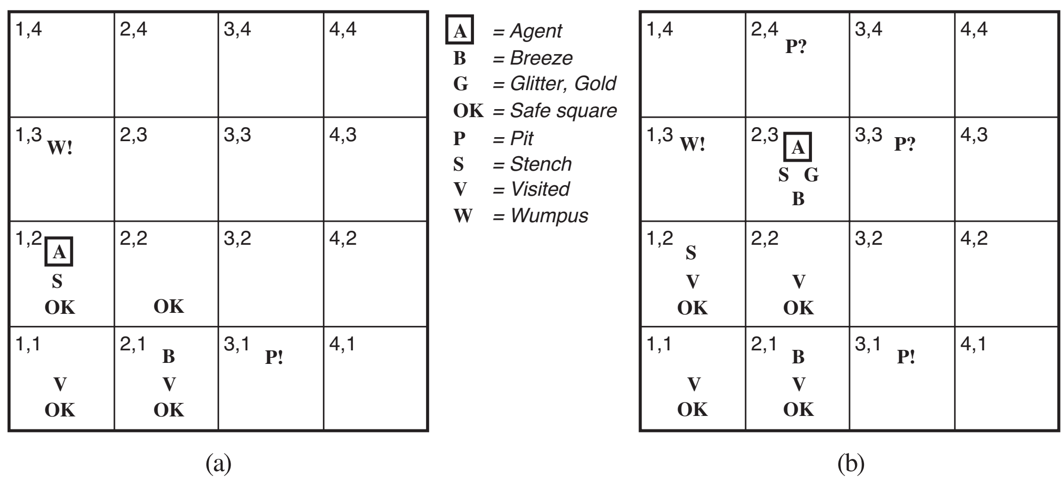
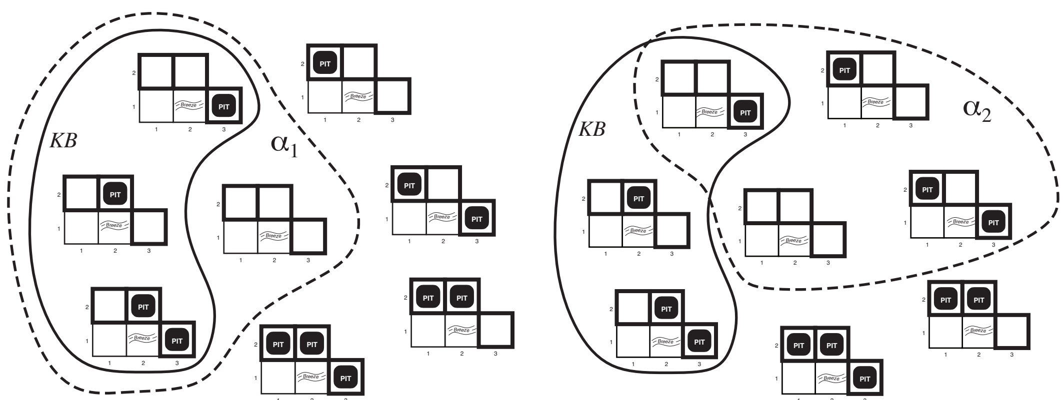


Figure 7.4 FILES: figures/wumpus-seq35.eps (Tue Nov 3 16:24:11 2009). Two later stages in the progress of the agent. (a) After the third move, with percept $[Stench, None, None, None, None]$. (b) After the fifth move, with percept $[Stench, Breeze, Glitter, None, None]$.



(a) (b)

Figure 7.5 FILES: figures/wumpus-entailment.eps (Tue Nov 3 16:24:09 2009) figures/wumpus-nonentailment.eps (Tue Nov 3 16:24:10 2009). Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]. The KB corresponding to the observations of nothing in [1,1] and a breeze in [2,1] is shown by the solid line. (a) Dotted line shows models of α_1 (no pit in [1,2]). (b) Dotted line shows models of α_2 (no pit in [2,2]).

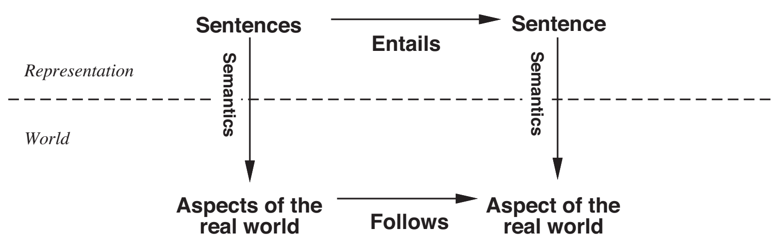


Figure 7.6 FILES: figures/follows+entails.eps (Tue Nov 3 16:22:52 2009). Sentences are physical configurations of the agent, and reasoning is a process of constructing new physical configurations from old ones. Logical reasoning should ensure that the new configurations represent aspects of the world that actually follow from the aspects that the old configurations represent.

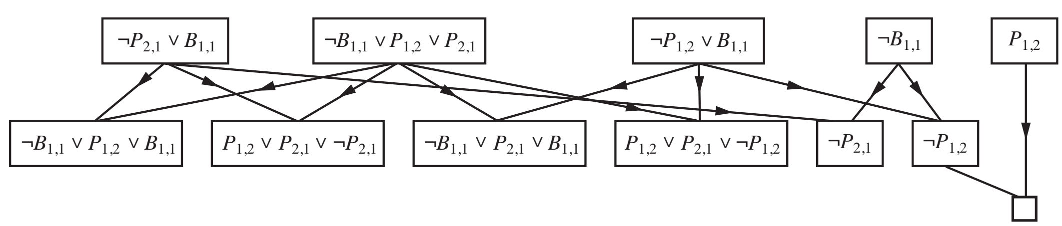
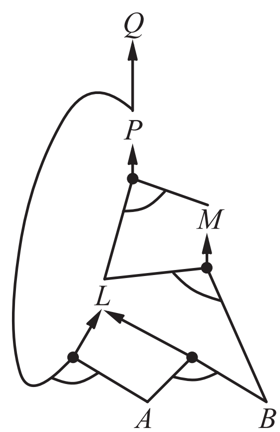


Figure 7.13 FILES: figures/wumpus-resolution.eps (Tue Nov 3 16:24:10 2009). Partial application of PL-RESOLUTION to a simple inference in the wumpus world. $\neg P_{1,2}$ is shown to follow from the first four clauses in the top row.

$$P \Rightarrow Q$$
$$L \land M \Rightarrow P$$
$$B \land L \Rightarrow M$$
$$A \land P \Rightarrow L$$
$$A \land B \Rightarrow L$$
$$A$$
$$B$$

(a)



(b)

Figure 7.16 FILES: figures/pl-horn-example.eps (Tue Nov 3 13:45:07 2009). (a) A set of Horn clauses. (b) The corresponding AND–OR graph.

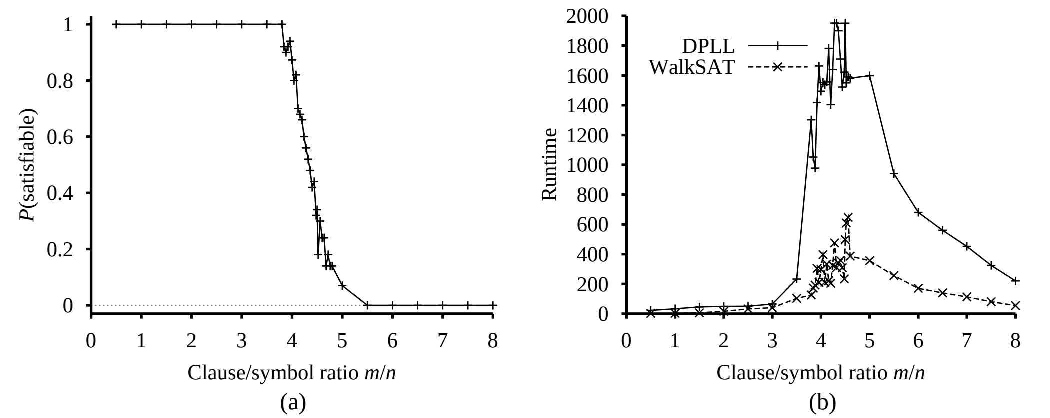


Figure 7.19 **FILES: .** (a) Graph showing the probability that a random 3-CNF sentence with $n = 50$ symbols is satisfiable, as a function of the clause/symbol ratio m/n. (b) Graph of the median run time (measured in number of recursive calls to DPLL, a good proxy) on random 3-CNF sentences. The most difficult problems have a clause/symbol ratio of about 4.3.

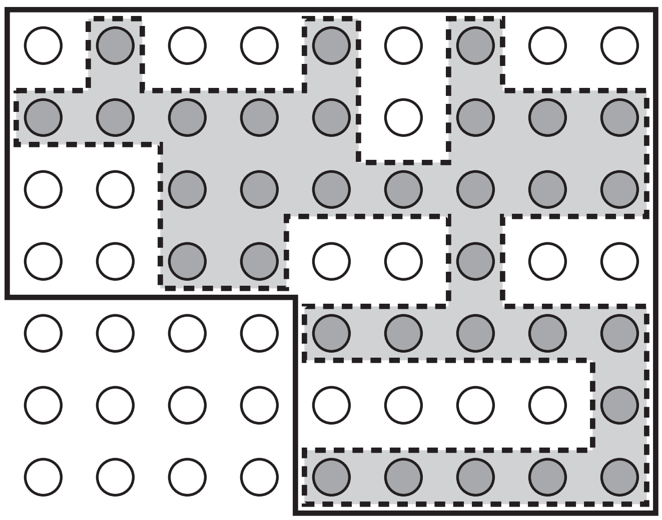


Figure 7.21 FILES: figures/wiggly-belief-state.eps (Tue Nov 3 13:53:12 2009). Depiction of a 1-CNF belief state (bold outline) as a simply representable, conservative approximation to the exact (wiggly) belief state (shaded region with dashed outline). Each possible world is shown as a circle; the shaded ones are consistent with all the percepts.

8 FIRST-ORDER LOGIC

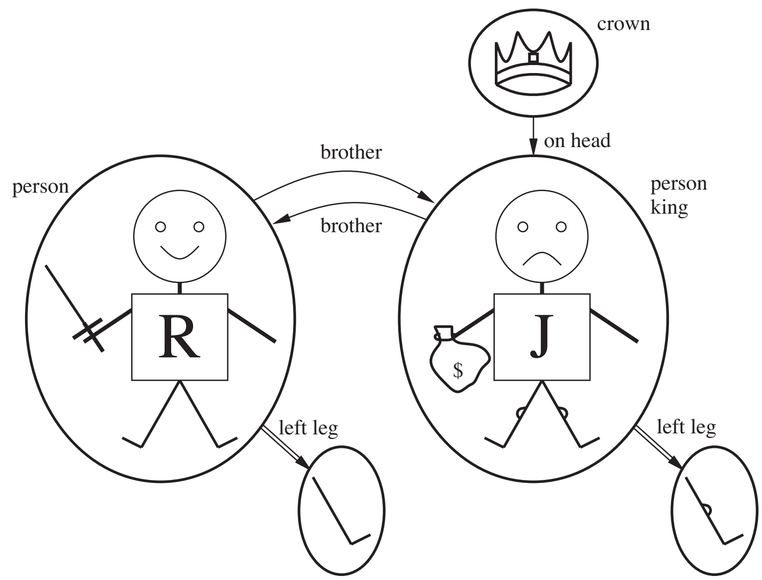


Figure 8.2 FILES: figures/fol-model.eps (Tue Nov 3 16:22:52 2009). A model containing five objects, two binary relations, three unary relations (indicated by labels on the objects), and one unary function, left-leg.

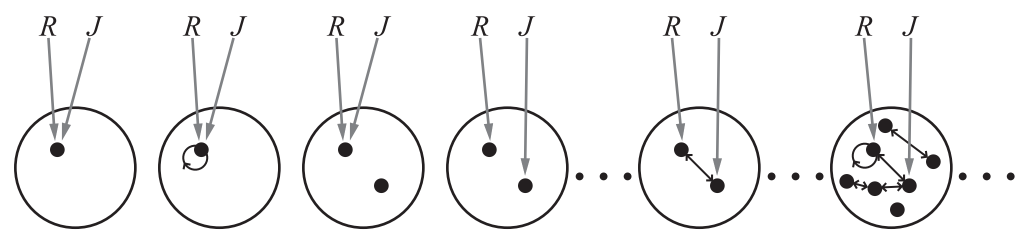


Figure 8.4 FILES: figures/all-models-standard.eps (Tue Nov 3 13:21:28 2009). Some members of the set of all models for a language with two constant symbols, R and J, and one binary relation symbol. The interpretation of each constant symbol is shown by a gray arrow. Within each model, the related objects are connected by arrows.

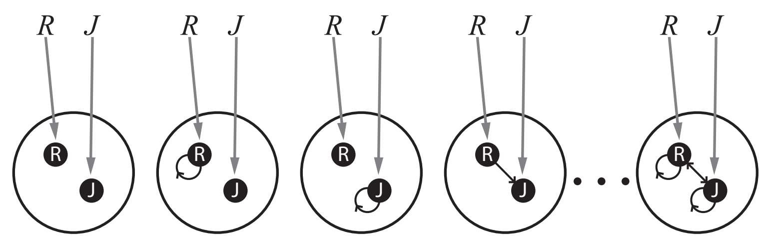


Figure 8.5 **FILES: figures/all-models-database.eps (Tue Nov 3 13:21:39 2009).** Some members of the set of all models for a language with two constant symbols, R and J, and one binary relation symbol, under database semantics. The interpretation of the constant symbols is fixed, and there is a distinct object for each constant symbol.

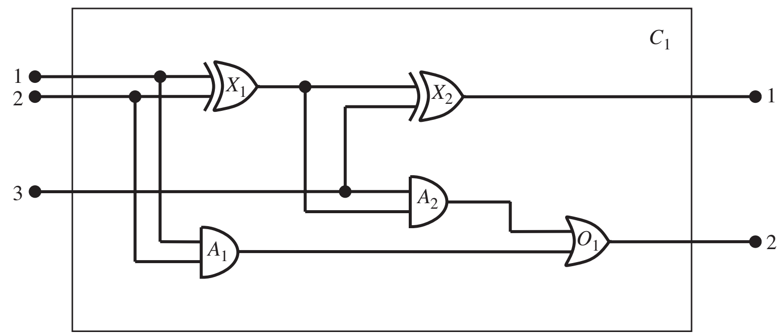


Figure 8.6 FILES: figures/adder.eps (Tue Nov 3 16:22:18 2009). A digital circuit C1, purporting to be a one-bit full adder. The first two inputs are the two bits to be added, and the third input is a carry bit. The first output is the sum, and the second output is a carry bit for the next adder. The circuit contains two XOR gates, two AND gates, and one OR gate.

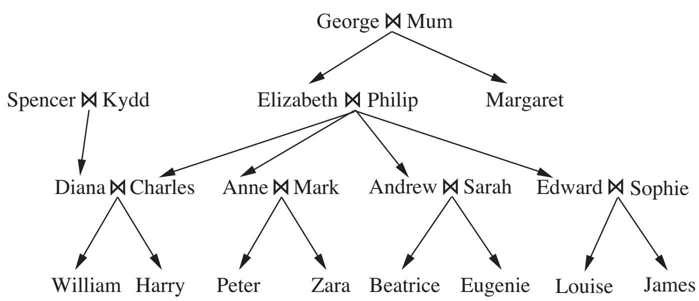


Figure 8.7 **FILES: figures/family1.eps (Tue Nov 3 16:22:46 2009).** A typical family tree. The symbol "⋈" connects spouses and arrows point to children.

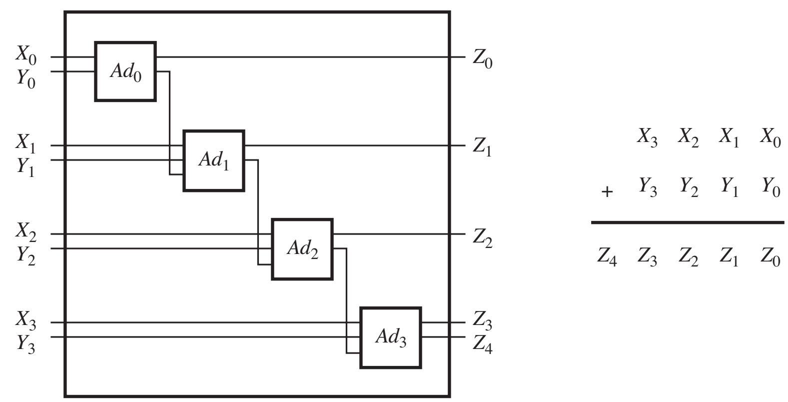


Figure 8.8 **FILES: figures/4bit-adder.eps (Tue Nov 3 16:22:10 2009).** A four-bit adder. Each Ad_i is a one-bit adder, as in Figure 8.5 on page 97.

9 INFERENCE IN FIRST-ORDER LOGIC

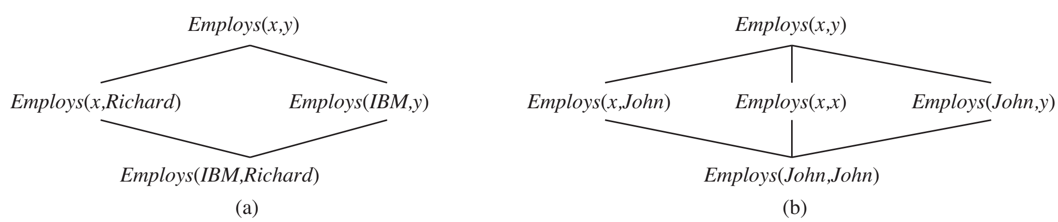


Figure 9.2 FILES: figures/subsumption-lattices.eps (Tue Nov 3 16:23:50 2009). (a) The subsumption lattice whose lowest node is $Employs(IBM, Richard)$. (b) The subsumption lattice for the sentence $Employs(John, John)$.

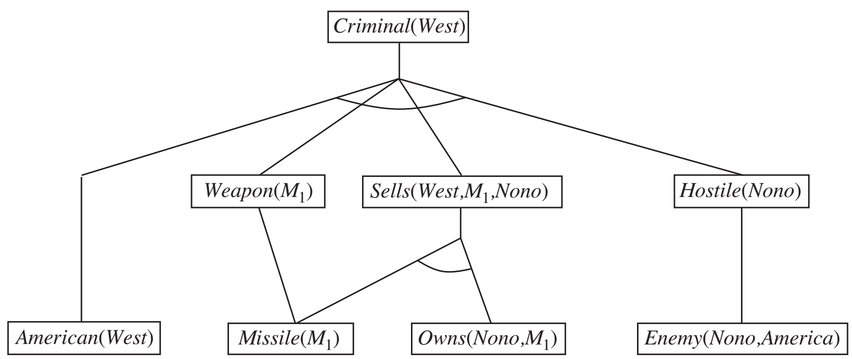
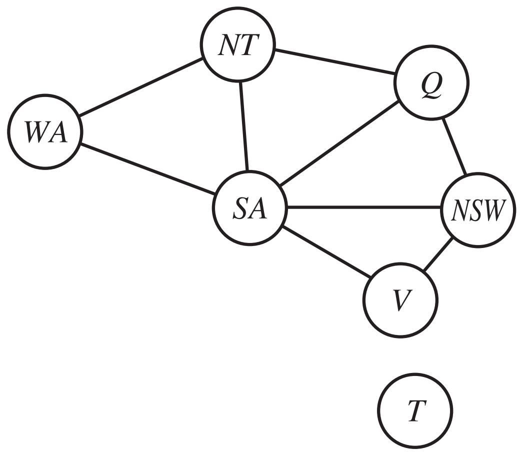


Figure 9.4 **FILES: figures/crime-fc.eps (Tue Nov 3 16:22:35 2009).** The proof tree generated by forward chaining on the crime example. The initial facts appear at the bottom level, facts inferred on the first iteration in the middle level, and facts inferred on the second iteration at the top level.



(a)

$$
\begin{aligned}
&\mathit{Diff}(wa, nt) \land \mathit{Diff}(wa, sa) \land \\
&\quad \mathit{Diff}(nt, q) \land \mathit{Diff}(nt, sa) \land \\
&\quad \mathit{Diff}(q, nsw) \land \mathit{Diff}(q, sa) \land \\
&\quad \mathit{Diff}(nsw, v) \land \mathit{Diff}(nsw, sa) \land \\
&\quad \mathit{Diff}(v, sa) \;\Rightarrow\; \mathit{Colorable}()
\end{aligned}
$$

$$
\begin{array}{ll}
\mathit{Diff}(Red, Blue) & \mathit{Diff}(Red, Green) \\
\mathit{Diff}(Green, Red) & \mathit{Diff}(Green, Blue) \\
\mathit{Diff}(Blue, Red) & \mathit{Diff}(Blue, Green)
\end{array}
$$

(b)

Figure 9.5 FILES: figures/australia-csp.eps (Tue Nov 3 16:22:25 2009). (a) Constraint graph for coloring the map of Australia. (b) The map-coloring CSP expressed as a single definite clause. Each map region is represented as a variable whose value can be one of the constants *Red*, *Green* or *Blue*.

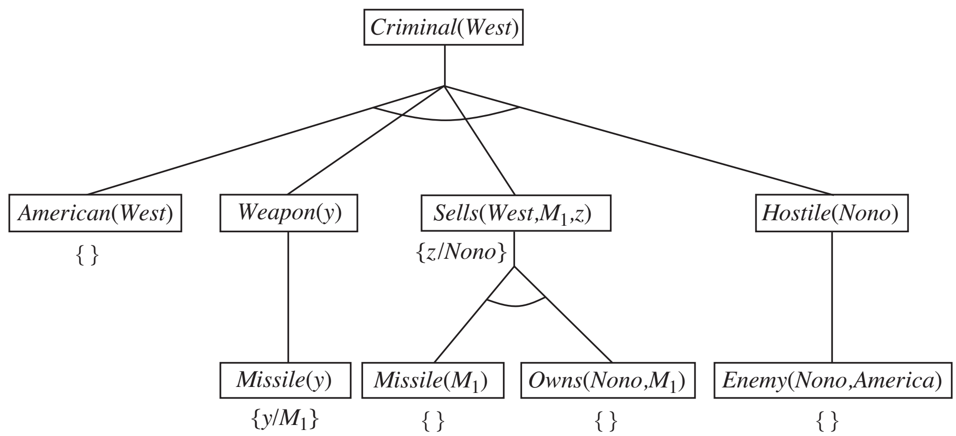


Figure 9.7 FILES: figures/crime-bc.eps (Tue Nov 3 16:22:34 2009). Proof tree constructed by backward chaining to prove that West is a criminal. The tree should be read depth first, left to right. To prove $Criminal(West)$, we have to prove the four conjuncts below it. Some of these are in the knowledge base, and others require further backward chaining. Bindings for each successful unification are shown next to the corresponding subgoal. Note that once one subgoal in a conjunction succeeds, its substitution is applied to subsequent subgoals. Thus, by the time FOL-BC-ASK gets to the last conjunct, originally $Hostile(z)$, z is already bound to $Nono$.

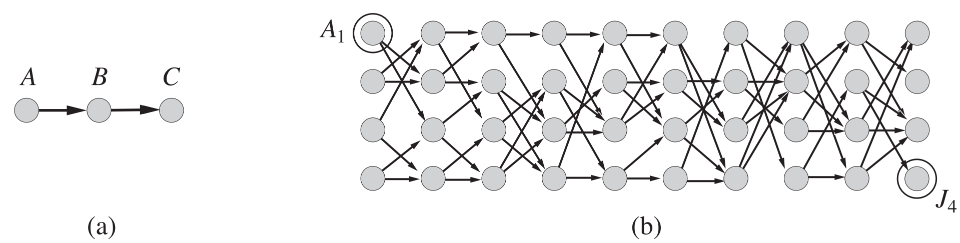


Figure 9.9 FILES: figures/prolog-ribbon.eps (Tue Nov 3 16:23:22 2009). (a) Finding a path from A to C can lead Prolog into an infinite loop. (b) A graph in which each node is connected to two random successors in the next layer. Finding a path from A_1 to J_4 requires 877 inferences.

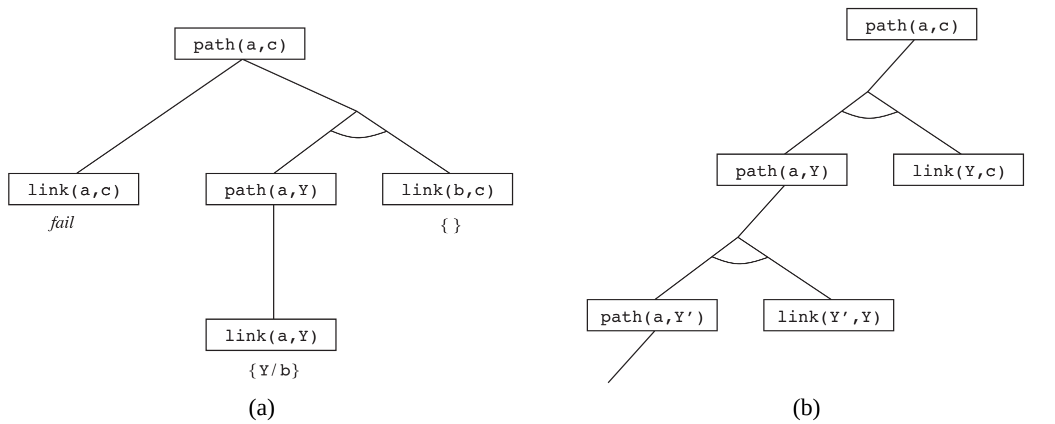


Figure 9.10 FILES: figures/proof-abc1.eps (Tue Nov 3 16:23:22 2009) figures/proof-abc2.eps (Tue Nov 3 16:23:22 2009). (a) Proof that a path exists from A to C. (b) Infinite proof tree generated when the clauses are in the "wrong" order.

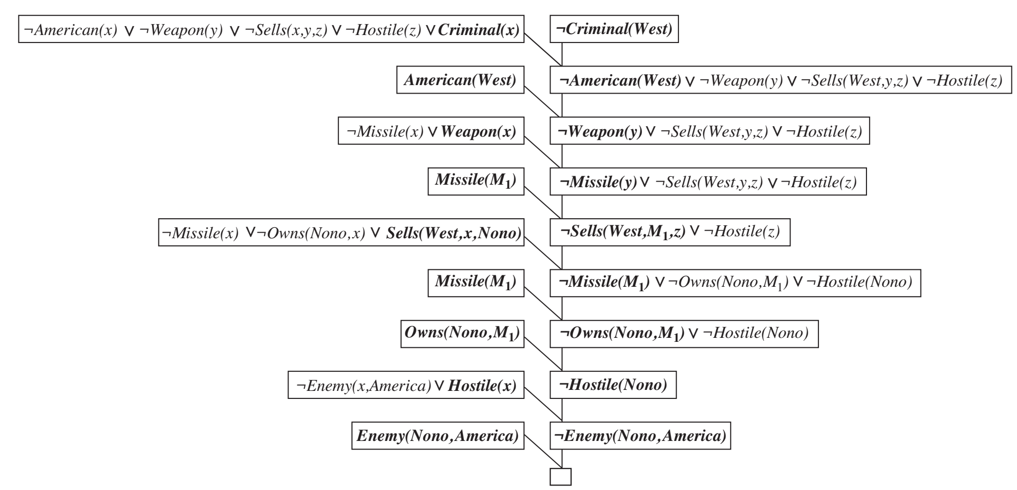


Figure 9.11 **FILES: figures/crime-resolution.eps (Tue Nov 3 16:22:35 2009).** A resolution proof that West is a criminal. At each step, the literals that unify are in bold.

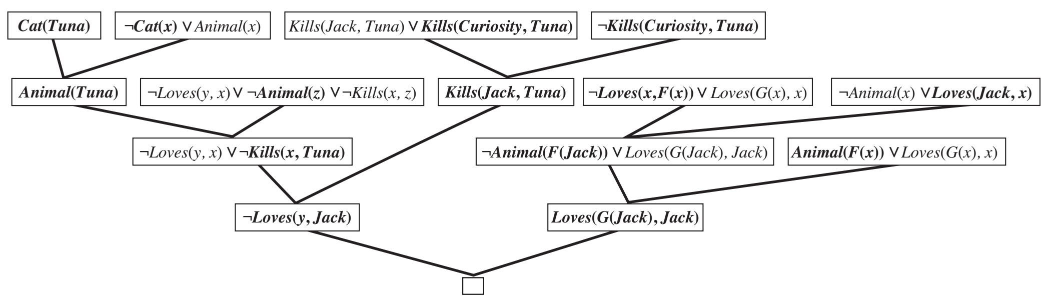


Figure 9.12 FILES: figures/curiosity.eps (Tue Nov 3 16:22:36 2009). A resolution proof that Curiosity killed the cat. Notice the use of factoring in the derivation of the clause $Loves(G(Jack), Jack)$. Notice also in the upper right, the unification of $Loves(x, F(x))$ and $Loves(Jack, x)$ can only succeed after the variables have been standardized apart.

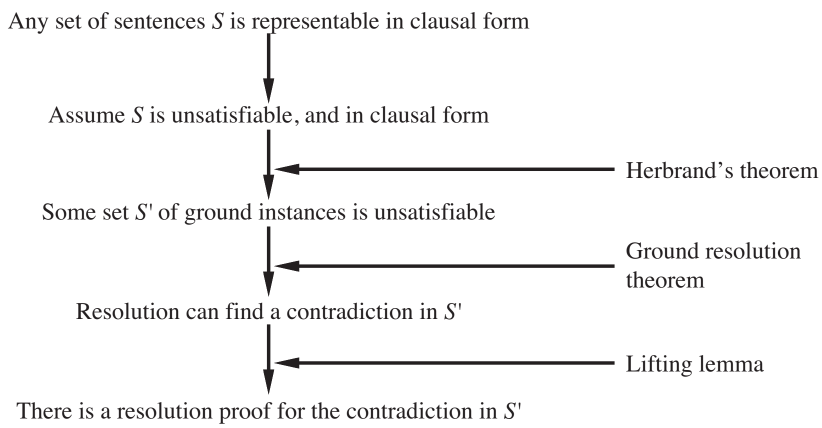


Figure 9.13 FILES: figures/resolution-completeness.eps (Tue Nov 3 16:23:28 2009). Structure of a completeness proof for resolution.

10 CLASSICAL PLANNING

Figure 10.4 **FILES: figures/sussman-anomaly.eps (Tue Nov 3 16:23:50 2009).** Diagram of the blocks-world problem in Figure **??**.

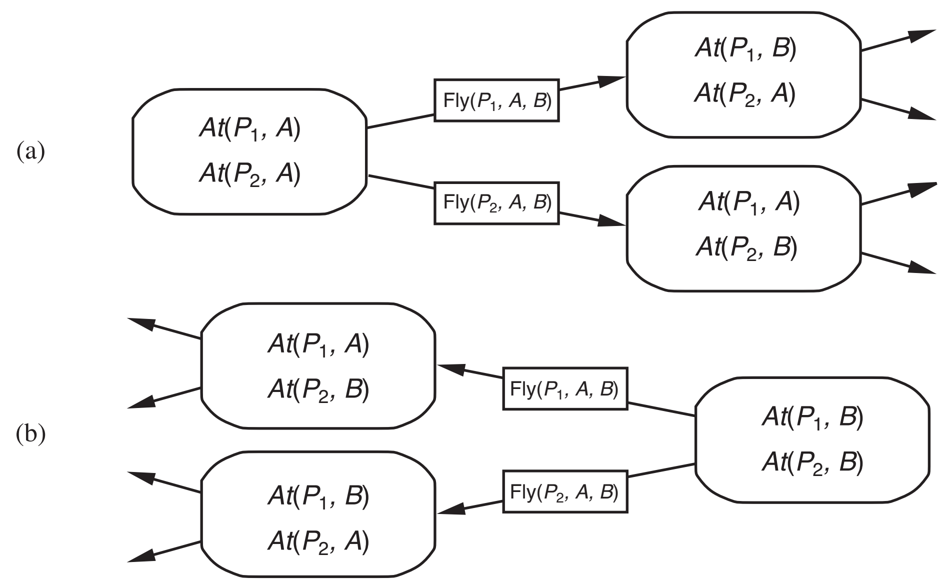


Figure 10.5 FILES: figures/two-plan-searches.eps (Tue Nov 3 16:23:58 2009). Two approaches to searching for a plan. (a) Forward (progression) search through the space of states, starting in the initial state and using the problem's actions to search forward for a member of the set of goal states. (b) Backward (regression) search through sets of relevant states, starting at the set of states representing the goal and using the inverse of the actions to search backward for the initial state.

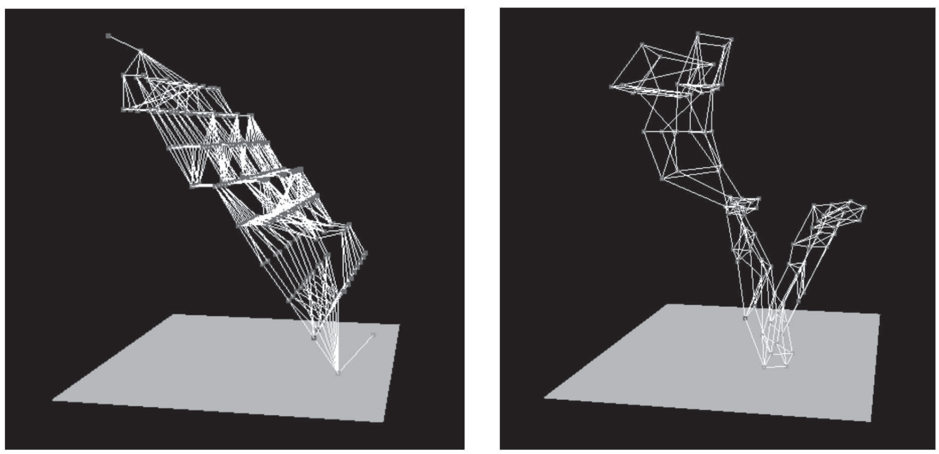


Figure 10.6 FILES: figures/ignore-del.eps (Tue Nov 3 16:23:04 2009). Two state spaces from planning problems with the ignore-delete-lists heuristic. The height above the bottom plane is the heuristic score of a state; states on the bottom plane are goals. There are no local minima, so search for the goal is straightforward. From ? (?).

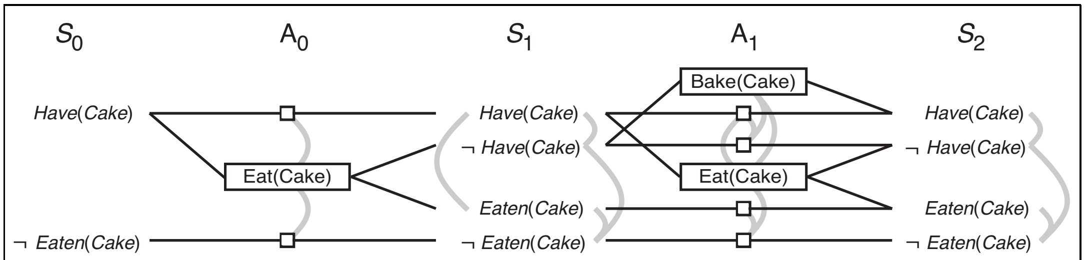


Figure 10.8 **FILES: figures/eatcake-graphplan2.eps (Tue Nov 3 16:22:41 2009).** The planning graph for the "have cake and eat cake too" problem up to level S_2. Rectangles indicate actions (small squares indicate persistence actions), and straight lines indicate preconditions and effects. Mutex links are shown as curved gray lines. Not all mutex links are shown, because the graph would be too cluttered. In general, if two literals are mutex at S_i, then the persistence actions for those literals will be mutex at A_i and we need not draw that mutex link.

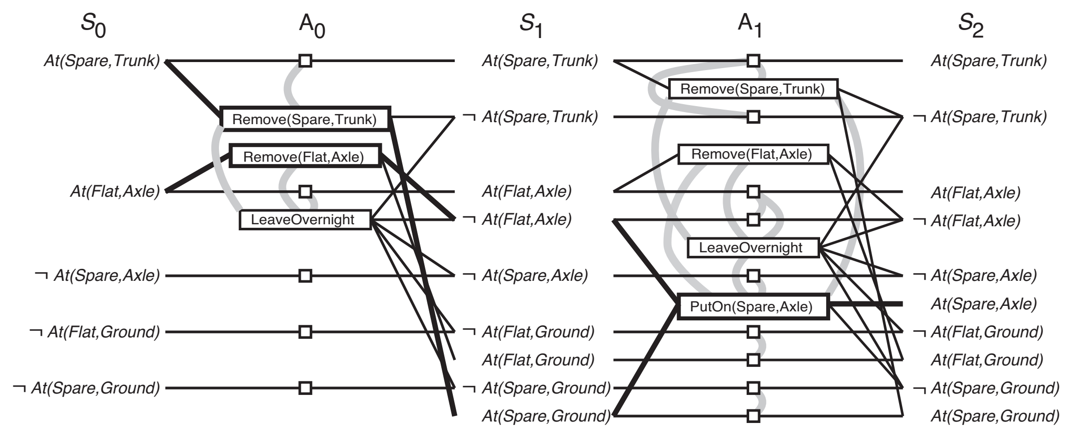


Figure 10.10 FILES: figures/tire-graphplan2.eps (Tue Nov 3 16:23:55 2009). The planning graph for the spare tire problem after expansion to level S_2. Mutex links are shown as gray lines. Not all links are shown, because the graph would be too cluttered if we showed them all. The solution is indicated by bold lines and outlines.

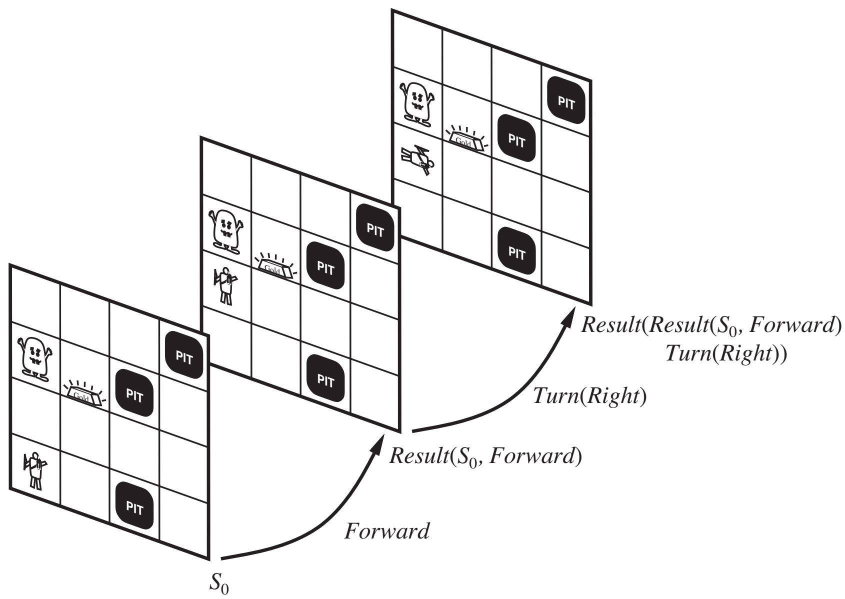


Figure 10.12 FILES: figures/situations.eps (Tue Nov 3 16:23:45 2009). Situations as the results of actions in the wumpus world.

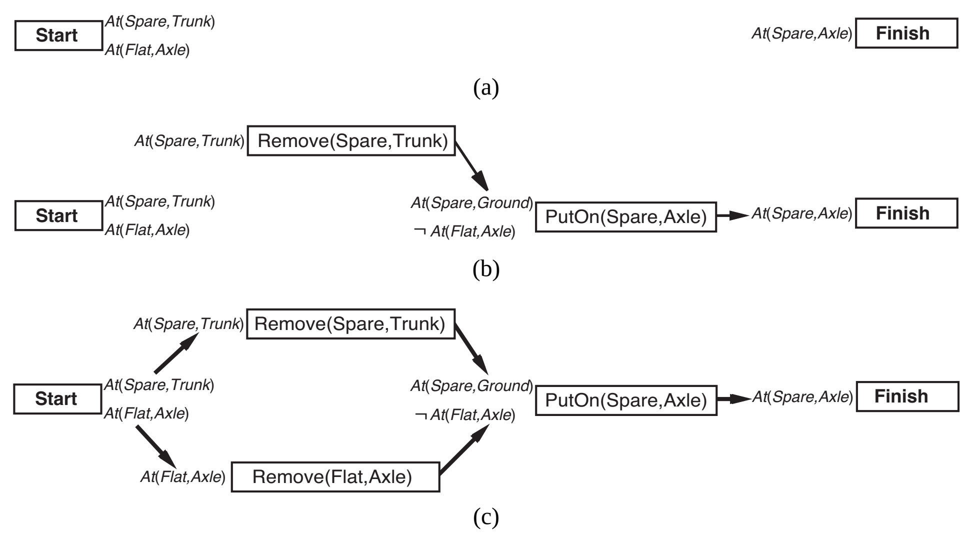


Figure 10.13 FILES: figures/tire-empty.eps (Wed Nov 4 14:41:01 2009) figures/tire0.eps (Wed Nov 4 14:40:52 2009) figures/tire2.eps (Wed Nov 4 14:40:38 2009). (a) the tire problem expressed as an empty plan. (b) an incomplete partially ordered plan for the tire problem. Boxes represent actions and arrows indicate that one action must occur before another. (c) a complete partially-ordered solution.

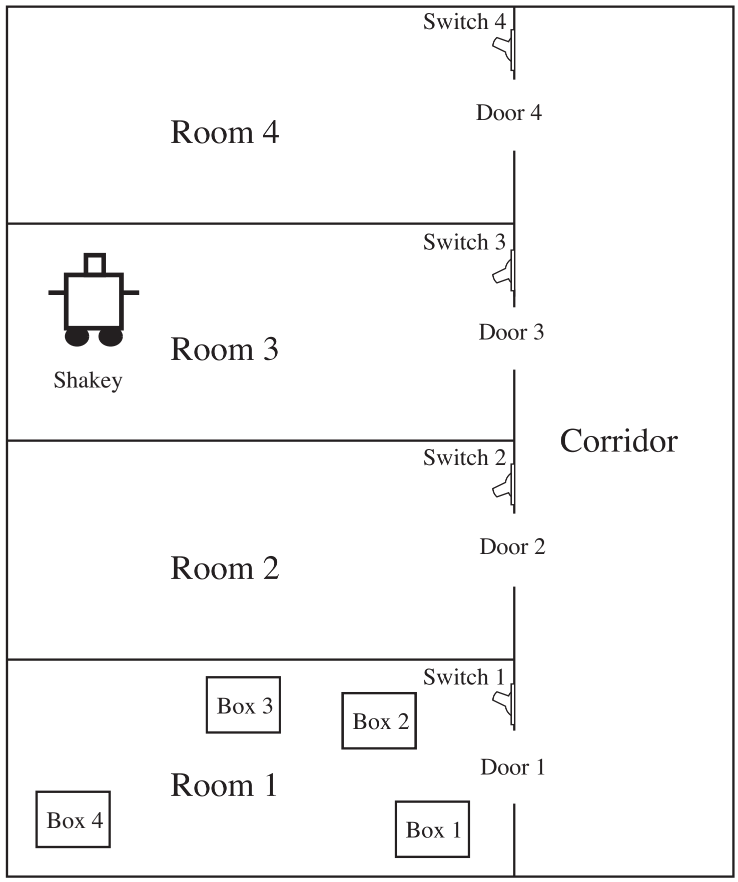


Figure 10.14 **FILES: figures/shakey2.eps (Tue Nov 3 16:23:43 2009).** Shakey's world. Shakey can move between landmarks within a room, can pass through the door between rooms, can climb climbable objects and push pushable objects, and can flip light switches.

11 PLANNING AND ACTING IN THE REAL WORLD

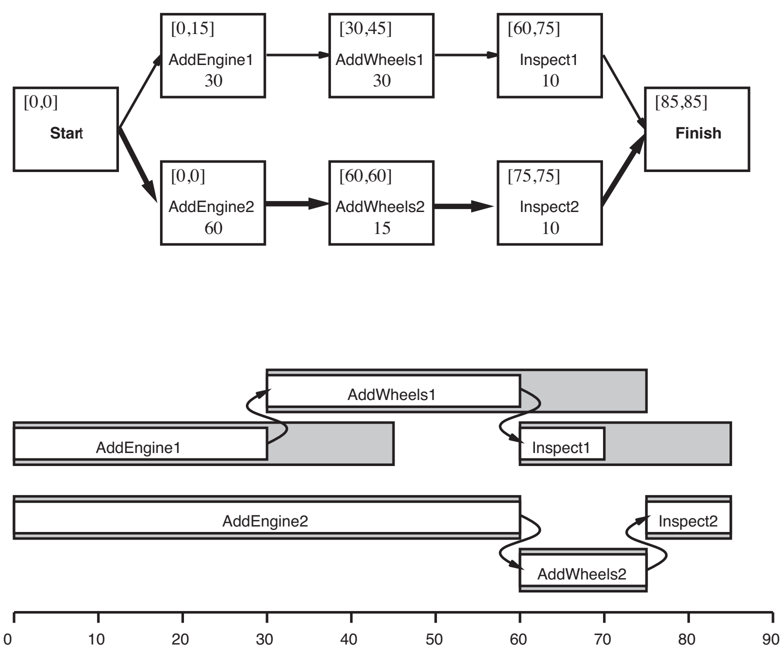


Figure 11.2 **FILES: figures/jobshop-cpm.eps (Tue Nov 3 16:23:05 2009).** Top: a representation of the temporal constraints for the job-shop scheduling problem of Figure **??**. The duration of each action is given at the bottom of each rectangle. In solving the problem, we compute the earliest and latest start times as the pair $[ES, LS]$, displayed in the upper left. The difference between these two numbers is the *slack* of an action; actions with zero slack are on the critical path, shown with bold arrows. Bottom: the same solution shown as a timeline. Grey rectangles represent time intervals during which an action may be executed, provided that the ordering constraints are respected. The unoccupied portion of a gray rectangle indicates the slack.

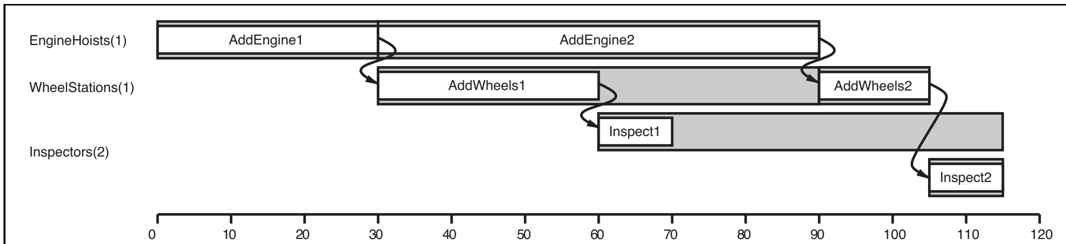


Figure 11.3 FILES: figures/jobshop-resources.eps (Tue Nov 3 16:23:05 2009). A solution to the job-shop scheduling problem from Figure **??**, taking into account resource constraints. The left-hand margin lists the three reusable resources, and actions are shown aligned horizontally with the resources they use. There are two possible schedules, depending on which assembly uses the engine hoist first; we've shown the shortest-duration solution, which takes 115 minutes.

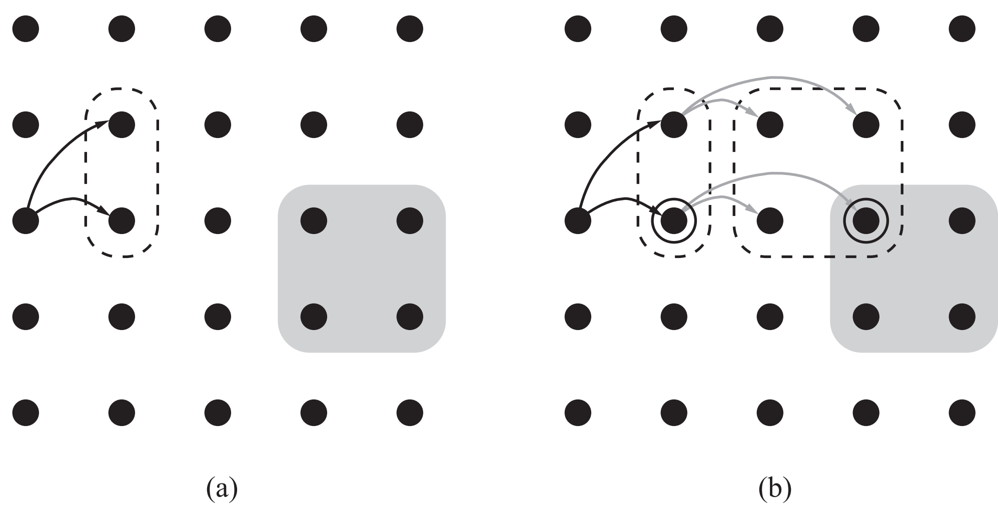


Figure 11.6 FILES: figures/reachable-sets.eps (Tue Nov 3 13:47:29 2009). Schematic examples of reachable sets. The set of goal states is shaded. Black and gray arrows indicate possible implementations of h_1 and h_2, respectively. (a) The reachable set of an HLA h_1 in a state s. (b) The reachable set for the sequence $[h_1, h_2]$. Because this intersects the goal set, the sequence achieves the goal.

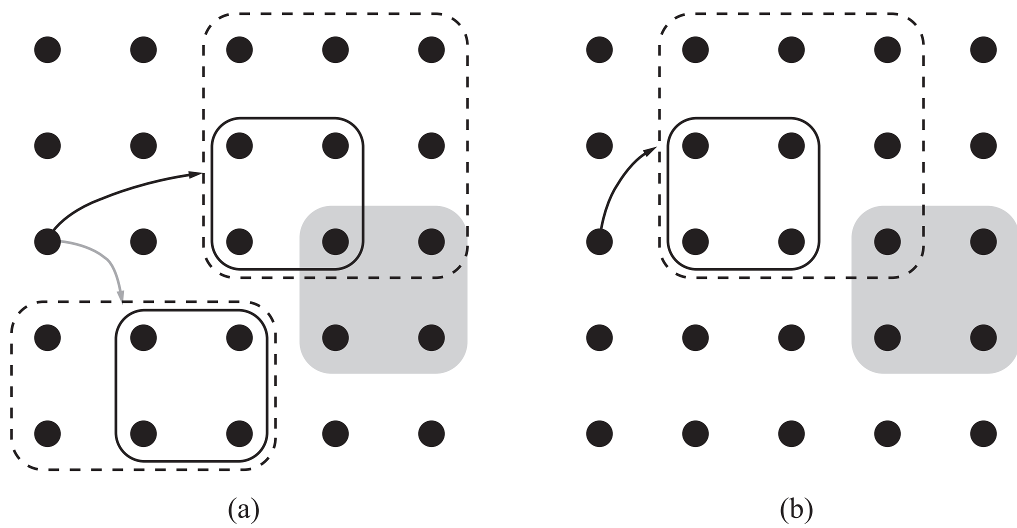


Figure 11.7 FILES: figures/approximate-HLA.eps (Tue Nov 3 13:23:08 2009). Goal achievement for high-level plans with approximate descriptions. The set of goal states is shaded. For each plan, the pessimistic (solid lines) and optimistic (dashed lines) reachable sets are shown. (a) The plan indicated by the black arrow definitely achieves the goal, while the plan indicated by the gray arrow definitely doesn't. (b) A plan that would need to be refined further to determine if it really does achieve the goal.

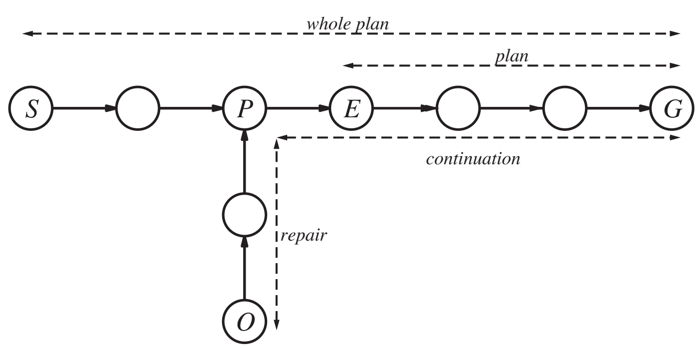


Figure 11.9 **FILES: figures/plan-repair.eps (Tue Nov 3 16:23:19 2009).** Before execution, the planner comes up with a plan, here called $whole\ plan$, to get from S to G. The agent executes steps of the plan until it expects to be in state E, but observes it is actually in O. The agent then replans for the minimal repair plus continuation to reach G.

(a) (b) (c)

Figure 11.11 FILES: figures/boids-neurogame.eps (Thu Nov 5 22:33:01 2009). (a) A simulated flock of birds, using Reynold's boids model. Image courtesy Giuseppe Randazzo, novastructura.net. (b) An actual flock of starlings. Image by Eduardo (pastaboy sleeps on flickr). (c) Two competitive teams of agents attempting to capture the towers in the NERO game. Image courtesy Risto Miikkulainen.

12 KNOWLEDGE REPRESENTATION

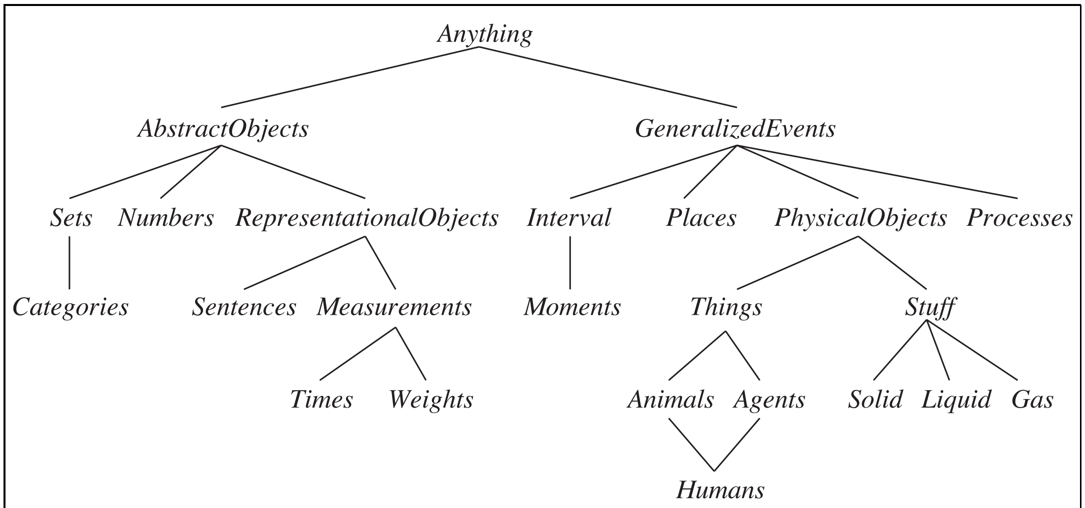


Figure 12.1 FILES: figures/everything.eps (Tue Nov 3 16:22:41 2009). The upper ontology of the world, showing the topics to be covered later in the chapter. Each link indicates that the lower concept is a specialization of the upper one. Specializations are not necessarily disjoint; a human is both an animal and an agent, for example. We will see in Section **??** why physical objects come under generalized events.

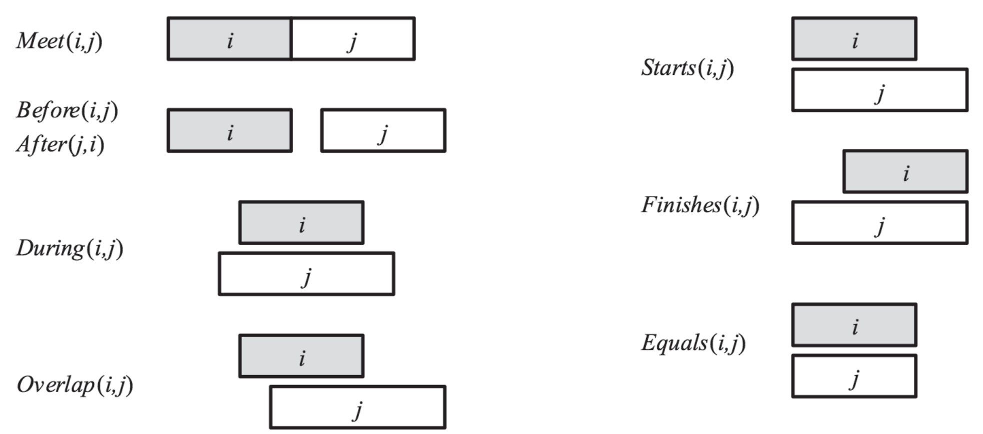


Figure 12.2 FILES: figures/allen-time-interval.eps (Tue Nov 3 16:22:20 2009). Predicates on time intervals.

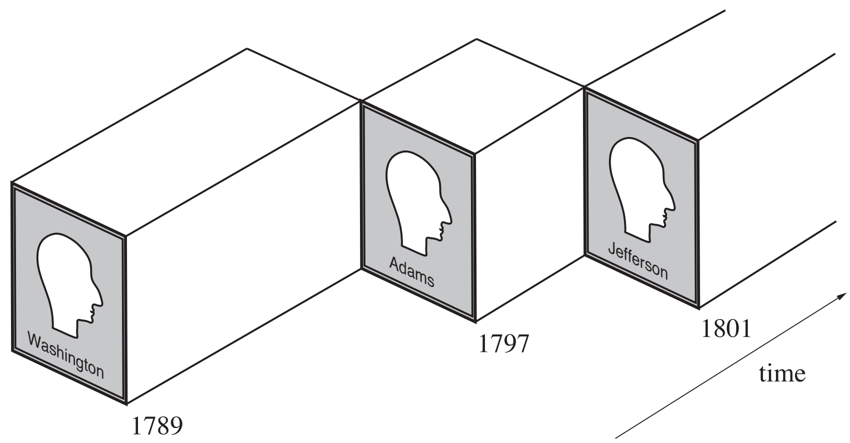


Figure 12.3 FILES: figures/president-usa.eps (Tue Nov 3 16:23:22 2009). A schematic view of the object $President(USA)$ for the first 15 years of its existence.

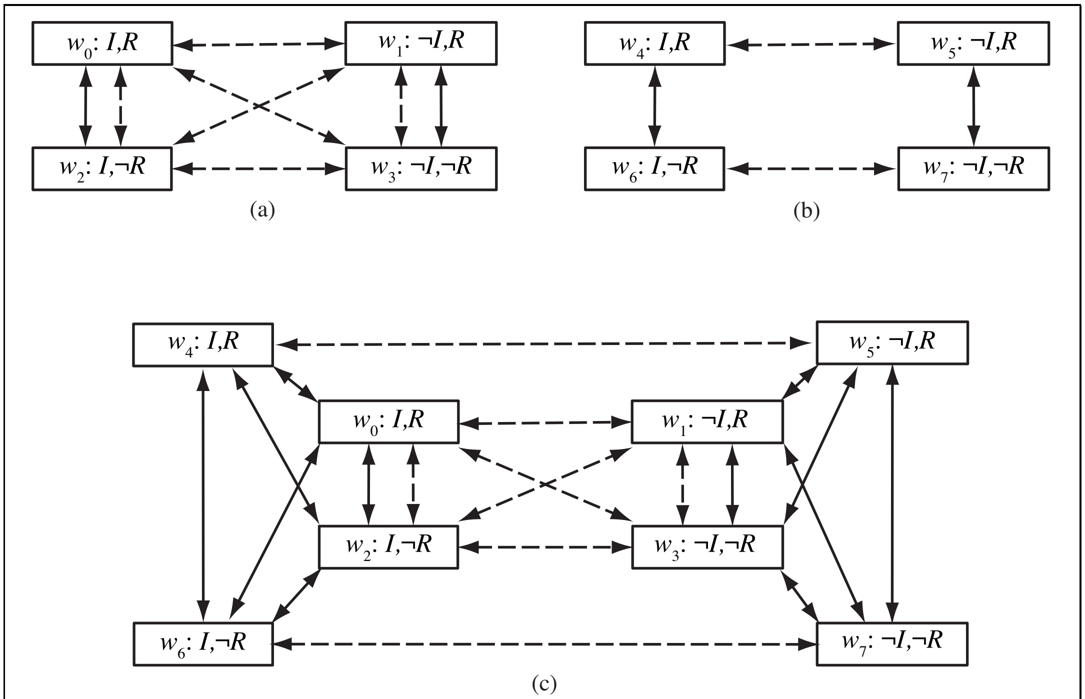


Figure 12.4 FILES: figures/possible-worlds2.eps (Wed Nov 4 11:06:34 2009). Possible worlds with accessibility relations $\mathbf{K}_{Superman}$ (solid arrows) and $\mathbf{K}_{Lois}$ (dotted arrows). The proposition R means "the weather report for tomorrow is rain" and I means "Superman's secret identity is Clark Kent." All worlds are accessible to themselves; the arrows from a world to itself are not shown.

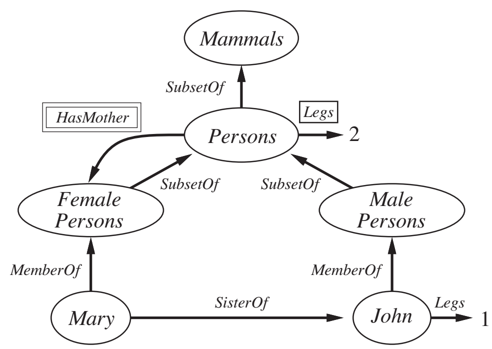


Figure 12.5 FILES: figures/semantic-network.eps (Tue Nov 3 16:23:41 2009). A semantic network with four objects (John, Mary, 1, and 2) and four categories. Relations are denoted by labeled links.

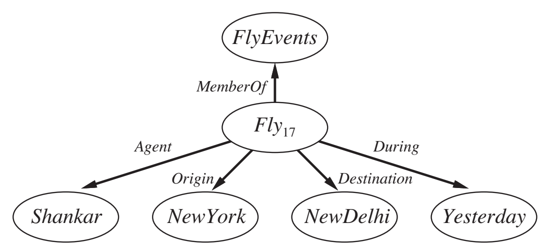


Figure 12.6 FILES: figures/flying-network.eps (Tue Nov 3 16:22:52 2009). A fragment of a semantic network showing the representation of the logical assertion $Fly(Shankar, NewYork, NewDelhi, Yesterday)$.

13 QUANTIFYING UNCERTAINTY

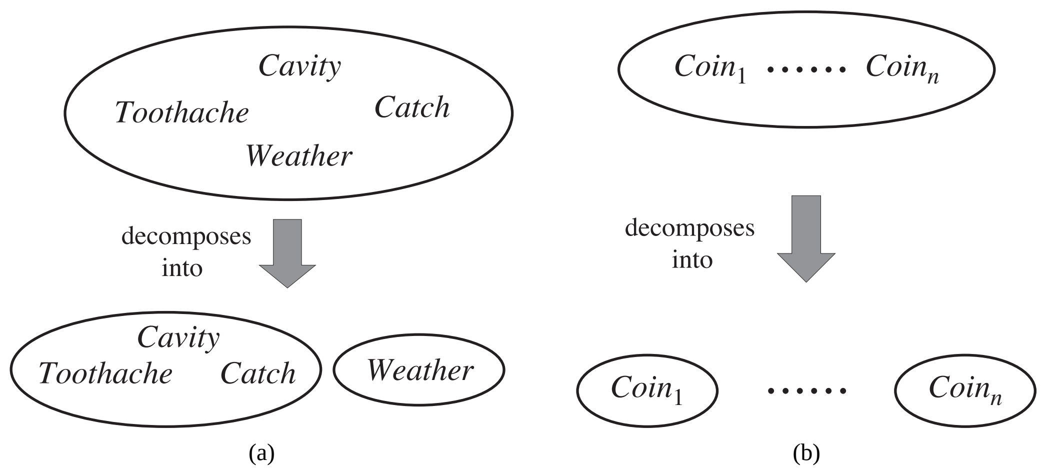


Figure 13.4 FILES: figures/weather-independence.eps (Tue Nov 3 16:24:08 2009) figures/coin-independence.eps (Tue Nov 3 16:22:33 2009). Two examples of factoring a large joint distribution into smaller distributions, using absolute independence. (a) Weather and dental problems are independent. (b) Coin flips are independent.

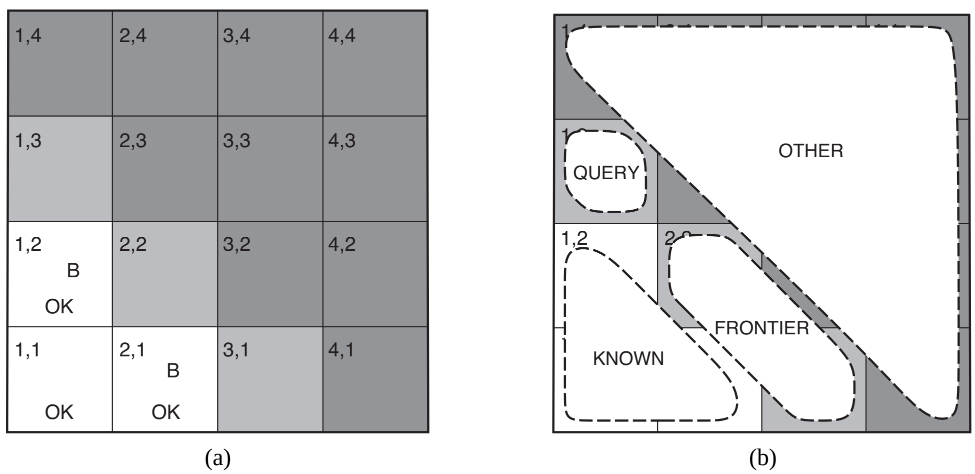


Figure 13.5 FILES: figures/wumpus-stuck.eps (Tue Nov 3 16:24:12 2009) figures/wumpus-variables.eps (Tue Nov 3 16:24:13 2009). (a) After finding a breeze in both [1,2] and [2,1], the agent is stuck—there is no safe place to explore. (b) Division of the squares into $Known$, $Frontier$, and $Other$, for a query about [1,3].

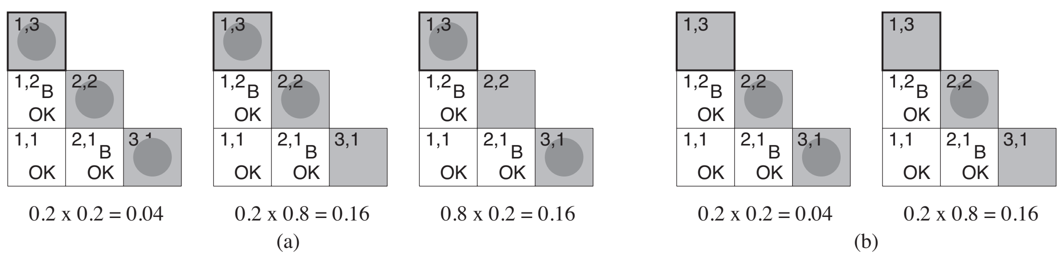


Figure 13.6 FILES: figures/wumpus-fringe-models.eps (Tue Nov 3 16:24:09 2009). Consistent models for the frontier variables $P_{2,2}$ and $P_{3,1}$, showing $P(frontier)$ for each model: (a) three models with $P_{1,3} = true$ showing two or three pits, and (b) two models with $P_{1,3} = false$ showing one or two pits.

14 PROBABILISTIC REASONING

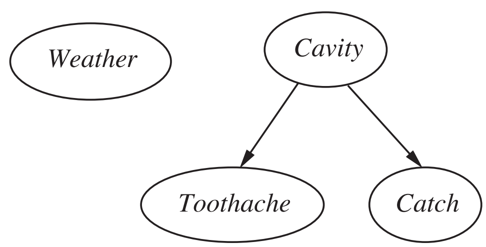


Figure 14.1 **FILES: figures/dentist-network.eps (Tue Nov 3 16:22:37 2009).** A simple Bayesian network in which $Weather$ is independent of the other three variables and $Toothache$ and $Catch$ are conditionally independent, given $Cavity$.

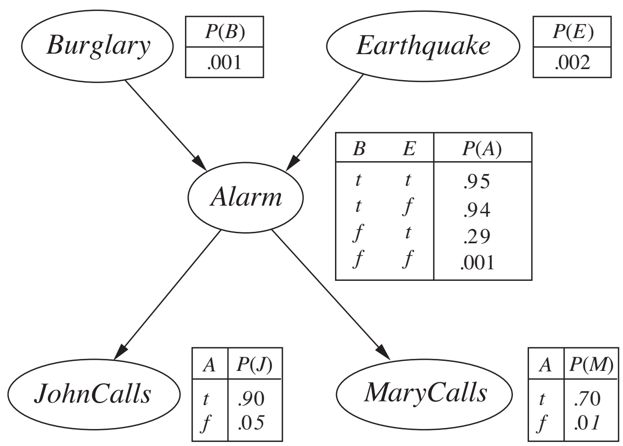


Figure 14.2 FILES: figures/burglary2.eps (Tue Nov 3 16:22:29 2009). A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for $Burglary$, $Earthquake$, $Alarm$, $JohnCalls$, and $MaryCalls$, respectively.

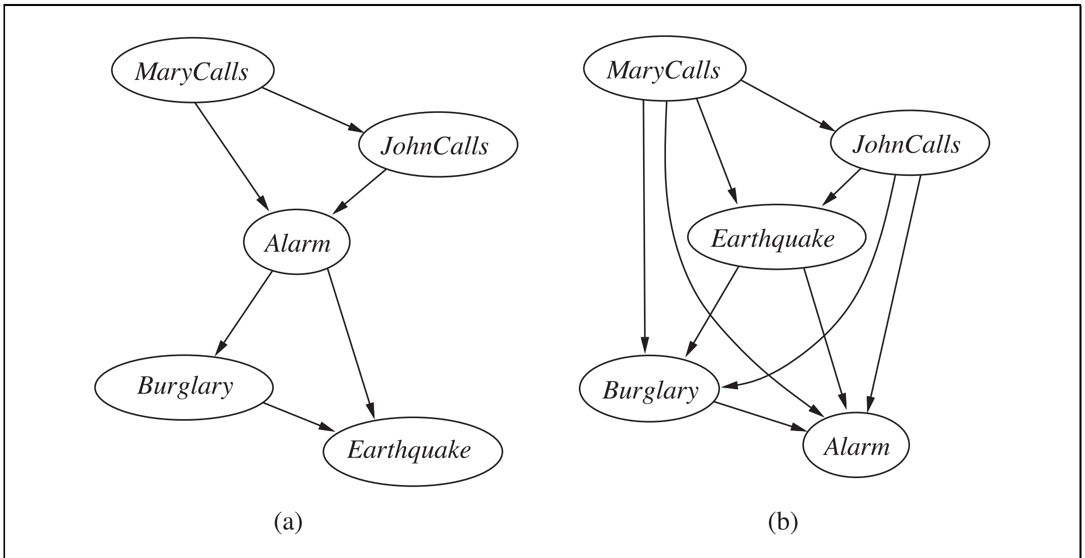


Figure 14.3 FILES: figures/burglary-mess.eps (Tue Nov 3 16:22:29 2009). Network structure depends on order of introduction. In each network, we have introduced nodes in top-to-bottom order.

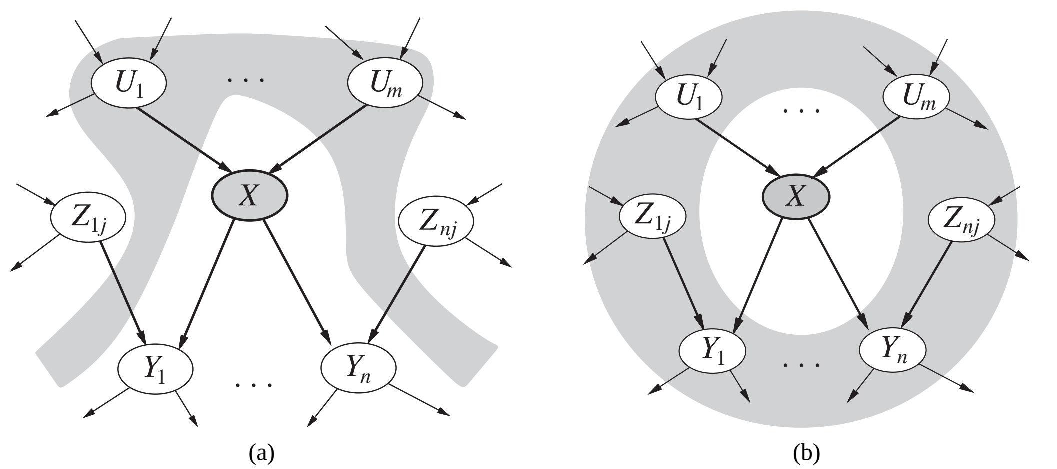


Figure 14.4 FILES: figures/nondescendants.eps (Tue Nov 3 16:23:15 2009) figures/markov-blanket.eps (Tue Nov 3 16:23:08 2009). (a) A node X is conditionally independent of its non-descendants (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area). (b) A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

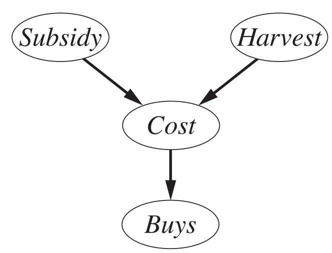


Figure 14.5 FILES: figures/continuous-net.eps (Tue Nov 3 16:22:34 2009). A simple network with discrete variables ($Subsidy$ and $Buys$) and continuous variables ($Harvest$ and $Cost$).

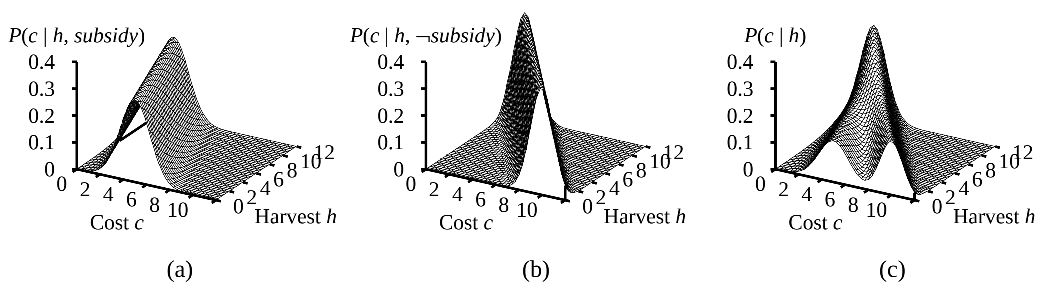


Figure 14.6 **FILES: .** The graphs in (a) and (b) show the probability distribution over $Cost$ as a function of $Harvest$ size, with $Subsidy$ true and false, respectively. Graph (c) shows the distribution $P(Cost \mid Harvest)$, obtained by summing over the two subsidy cases.

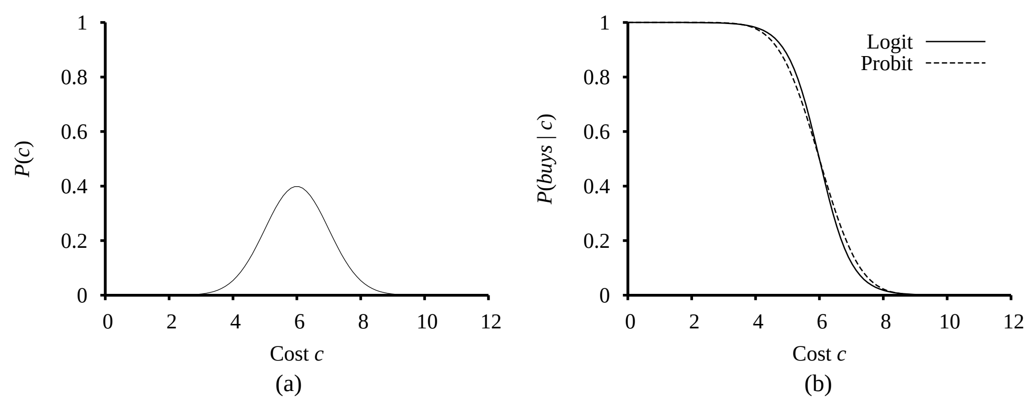


Figure 14.7 **FILES: .** (a) A normal (Gaussian) distribution for the cost threshold, centered on $\mu = 6.0$ with standard deviation $\sigma = 1.0$. (b) Logit and probit distributions for the probability of $buys$ given $cost$, for the parameters $\mu = 6.0$ and $\sigma = 1.0$.

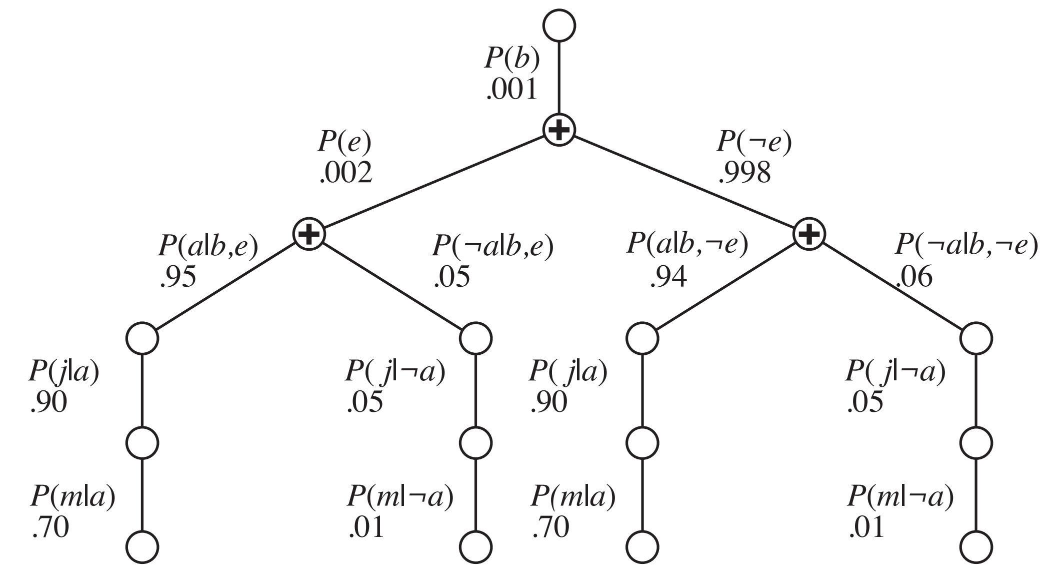


Figure 14.8 FILES: figures/enumeration-tree.eps (Tue Nov 3 16:22:41 2009). The structure of the expression shown in Equation (**??**). The evaluation proceeds top down, multiplying values along each path and summing at the "+" nodes. Notice the repetition of the paths for j and m.

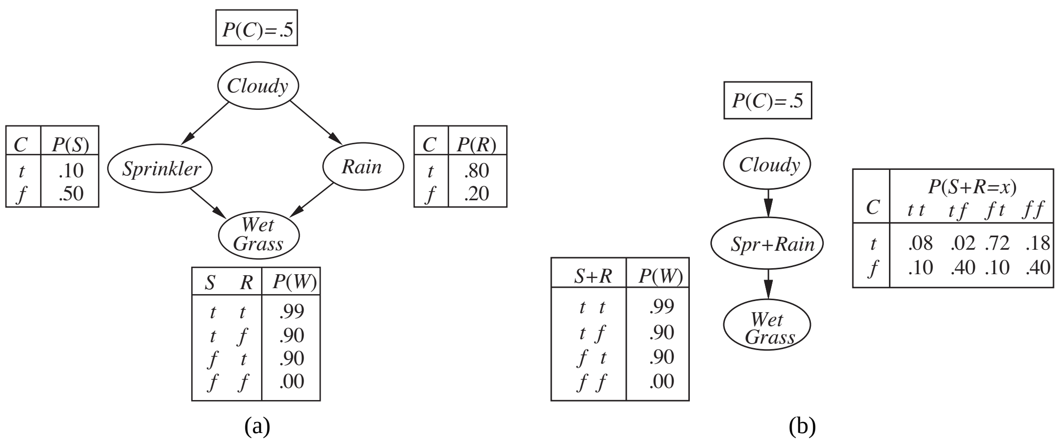


Figure 14.12 FILES: figures/rain-clustering1.eps (Tue Nov 3 16:23:27 2009) figures/rain-clustering2.eps (Tue Nov 3 16:23:27 2009). (a) A multiply connected network with conditional probability tables. (b) A clustered equivalent of the multiply connected network.

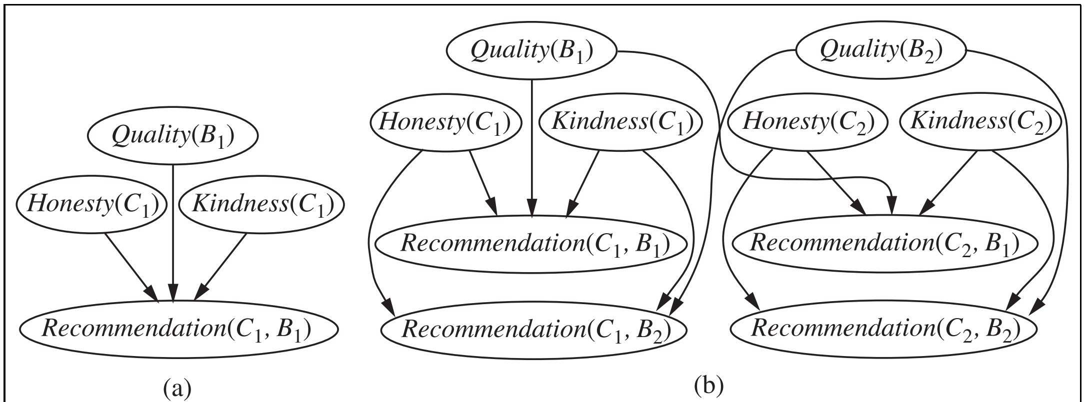


Figure 14.17 FILES: figures/new-14-16-1.eps (Tue Nov 3 16:23:14 2009) figures/new-14-16-2.eps (Tue Nov 3 16:23:14 2009) figures/new-14-16-1.eps (Tue Nov 3 16:23:14 2009). (a) Bayes net for a single customer C_1 recommending a single book B_1. $Honest(C_1)$ is Boolean, while the other variables have integer values from 1 to 5. (b) Bayes net with two customers and two books.

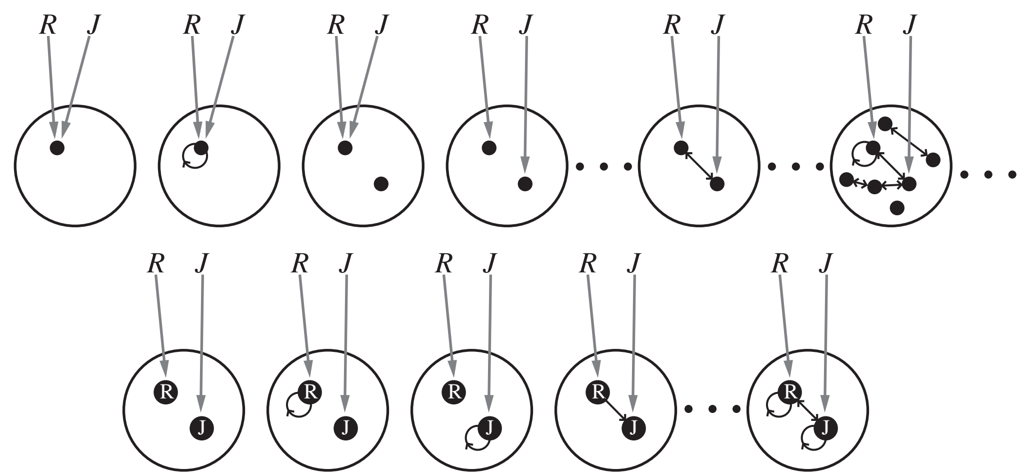


Figure 14.18 FILES: figures/all-models-both.eps (Tue Nov 3 16:22:20 2009). Top: Some members of the set of all possible worlds for a language with two constant symbols, R and J, and one binary relation symbol, under the standard semantics for first-order logic. Bottom: the possible worlds under database semantics. The interpretation of the constant symbols is fixed, and there is a distinct object for each constant symbol.

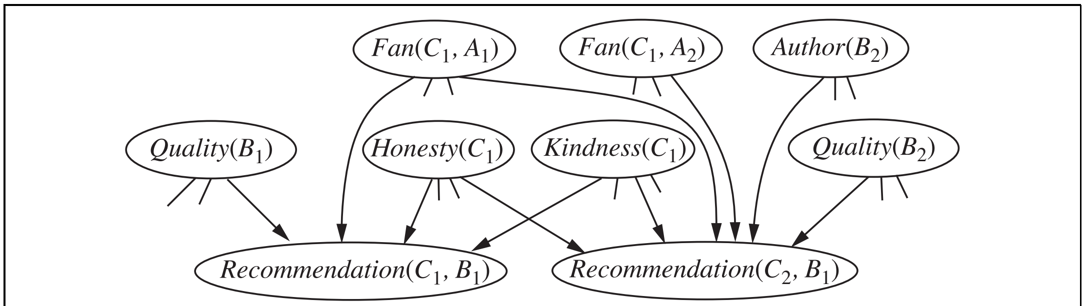


Figure 14.19 FILES: figures/new-14-18.eps (Tue Nov 3 16:23:14 2009). Fragment of the equivalent Bayes net when $Author(B_2)$ is unknown.

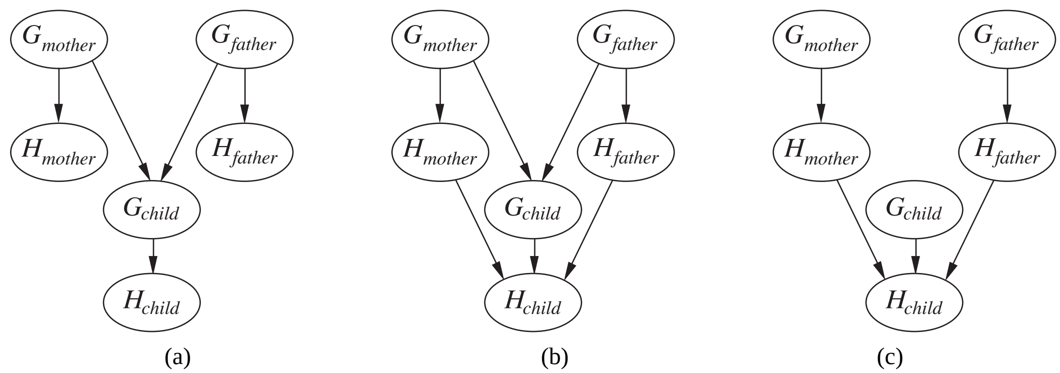


Figure 14.20 FILES: figures/handedness1.eps (Tue Nov 3 16:22:55 2009) figures/handedness2.eps (Tue Nov 3 16:22:56 2009) figures/handedness3.eps (Tue Nov 3 16:22:56 2009). Three possible structures for a Bayesian network describing genetic inheritance of handedness.

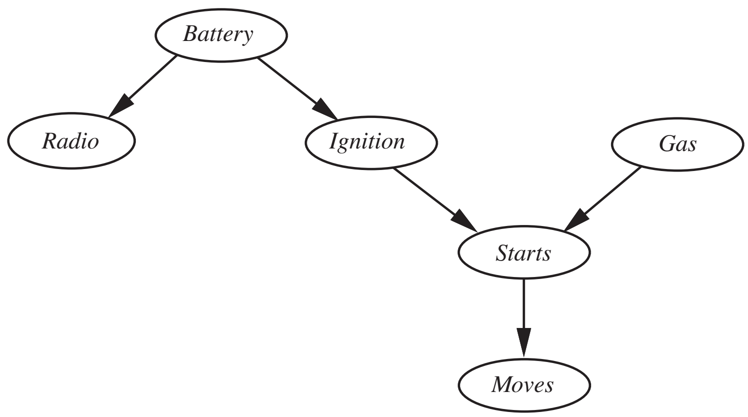


Figure 14.21 FILES: figures/car-starts.eps (Tue Nov 3 16:22:32 2009). A Bayesian network describing some features of a car's electrical system and engine. Each variable is Boolean, and the $true$ value indicates that the corresponding aspect of the vehicle is in working order.

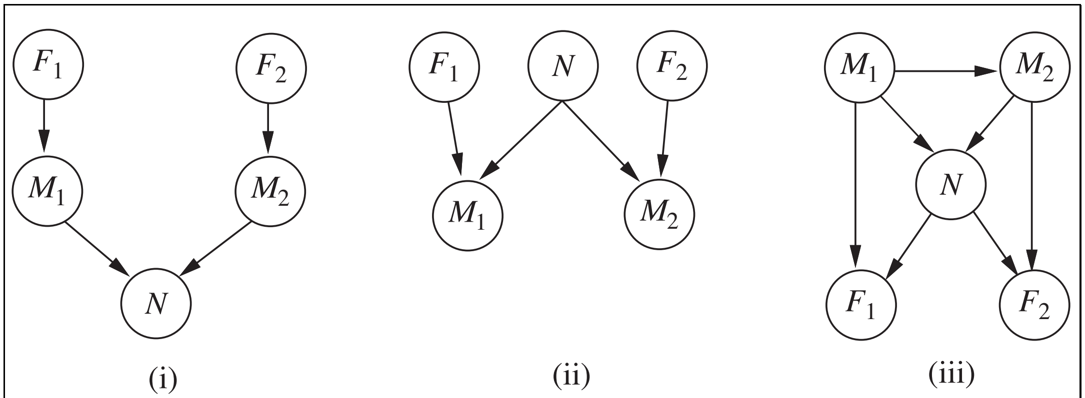
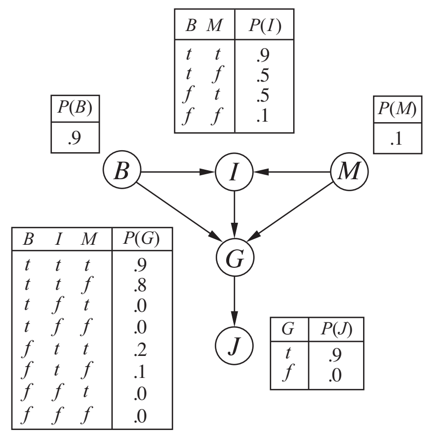


Figure 14.22 **FILES: figures/telescope-nets.eps (Tue Nov 3 16:23:51 2009).** Three possible networks for the telescope problem.



$P(B)$
.9

B	M	$P(I)$
t	t	.9
t	f	.5
f	t	.5
f	f	.1

$P(M)$
.1

B	I	M	$P(G)$
t	t	t	.9
t	t	f	.8
t	f	t	.0
t	f	f	.0
f	t	t	.2
f	t	f	.1
f	f	t	.0
f	f	f	.0

G	$P(J)$
t	.9
f	.0

Figure 14.23 FILES: figures/politics.eps (Tue Nov 3 16:23:20 2009). A simple Bayes net with Boolean variables $B = BrokeElectionLaw$, $I = Indicted$, $M = PoliticallyMotivatedProsecutor$, $G = FoundGuilty$, $J = Jailed$.

15 PROBABILISTIC REASONING OVER TIME

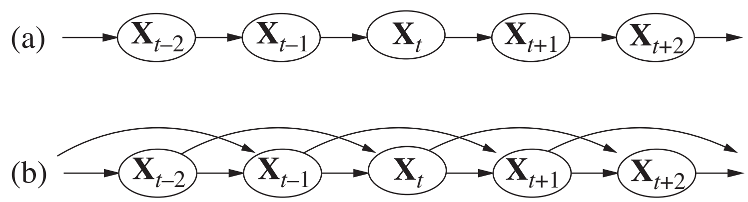


Figure 15.1 FILES: figures/markov-processes.eps (Tue Nov 3 16:23:08 2009). (a) Bayesian network structure corresponding to a first-order Markov process with state defined by the variables $\mathbf{X}_t$. (b) A second-order Markov process.

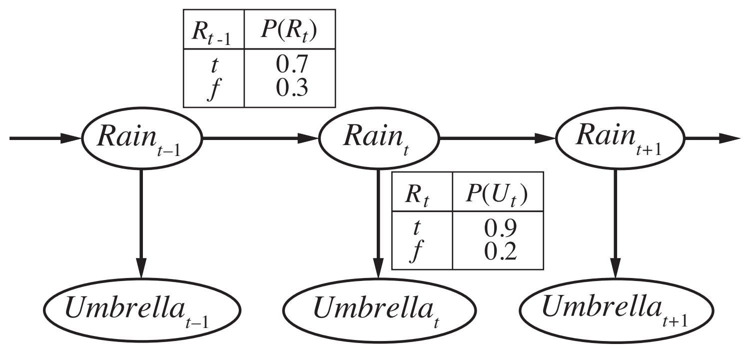


Figure 15.2 FILES: figures/umbrella-dbn.eps (Tue Nov 3 16:23:58 2009). Bayesian network structure and conditional distributions describing the umbrella world. The transition model is $P(Rain_t \mid Rain_{t-1})$ and the sensor model is $P(Umbrella_t \mid Rain_t)$.

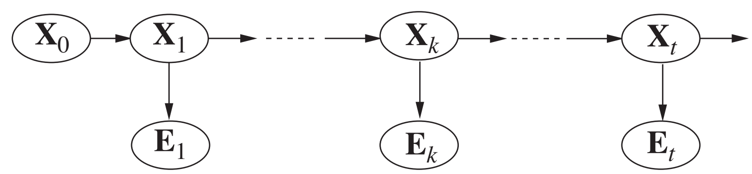


Figure 15.3 FILES: figures/smoothing-dbn.eps (Tue Nov 3 16:23:45 2009). Smoothing computes $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$, the posterior distribution of the state at some past time k given a complete sequence of observations from 1 to t.

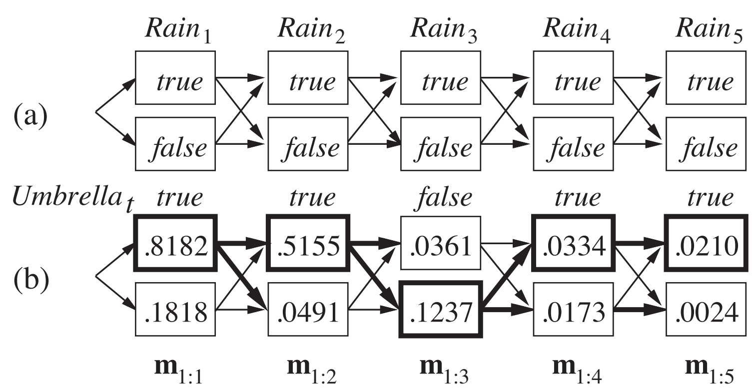
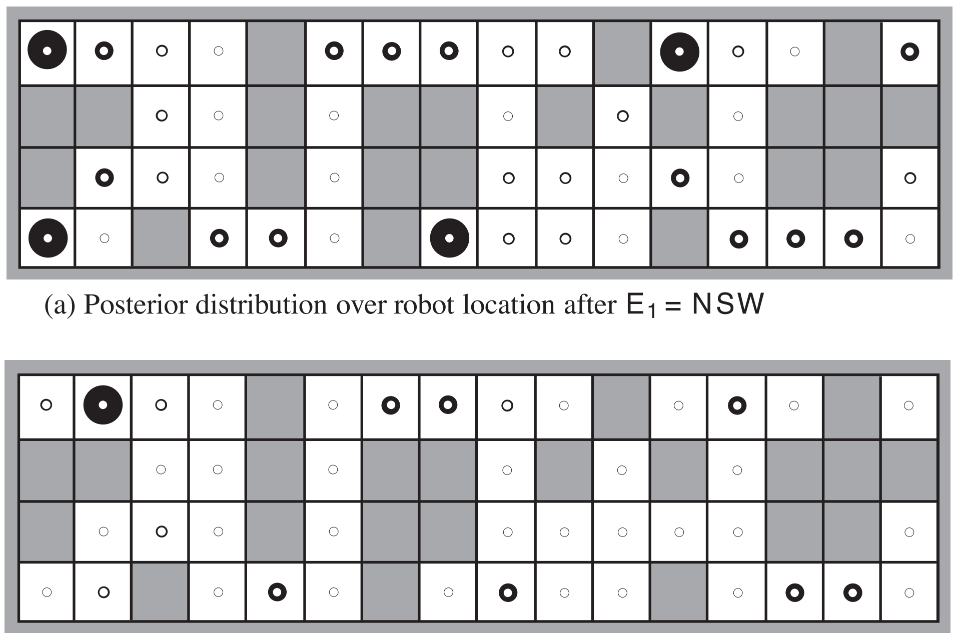


Figure 15.5 FILES: figures/umbrella-paths.eps (Tue Nov 3 16:23:59 2009). (a) Possible state sequences for $Rain_t$ can be viewed as paths through a graph of the possible states at each time step. (States are shown as rectangles to avoid confusion with nodes in a Bayes net.) (b) Operation of the Viterbi algorithm for the umbrella observation sequence $[true, true, false, true, true]$. For each t, we have shown the values of the message $\mathbf{m}_{1:t}$, which gives the probability of the best sequence reaching each state at time t. Also, for each state, the bold arrow leading into it indicates its best predecessor as measured by the product of the preceding sequence probability and the transition probability. Following the bold arrows back from the most likely state in $\mathbf{m}_{1:5}$ gives the most likely sequence.



(a) Posterior distribution over robot location after $E_1 = NSW$

(b) Posterior distribution over robot location after $E_1 = NSW, E_2 = NS$

Figure 15.7 FILES: figures/localization-figures-b.eps (Tue Nov 3 16:23:07 2009). Posterior distribution over robot location: (a) one observation $E_1 = NSW$; (b) after a second observation $E_2 = NS$. The size of each disk corresponds to the probability that the robot is at that location. The sensor error rate is $\epsilon = 0.2$.

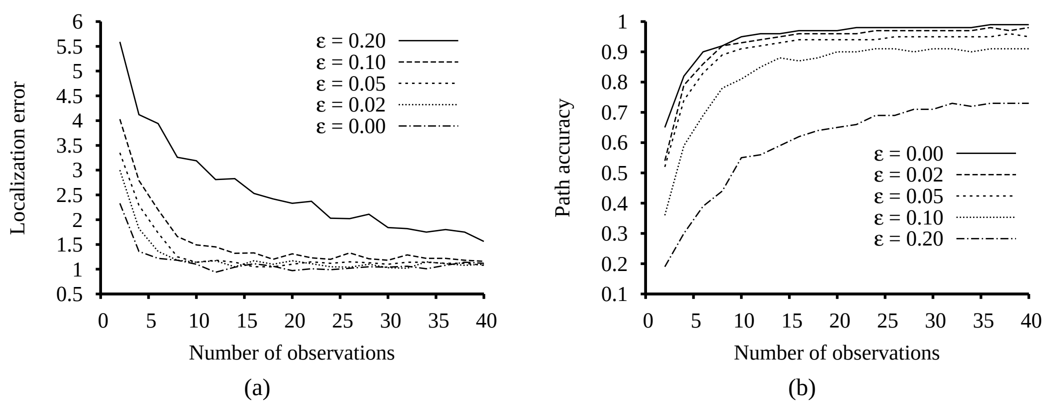


Figure 15.8 **FILES: .** Performance of HMM localization as a function of the length of the observation sequence for various different values of the sensor error probability ϵ; data averaged over 400 runs. (a) The localization error, defined as the Manhattan distance from the true location. (b) The Viterbi path accuracy, defined as the fraction of correct states on the Viterbi path.

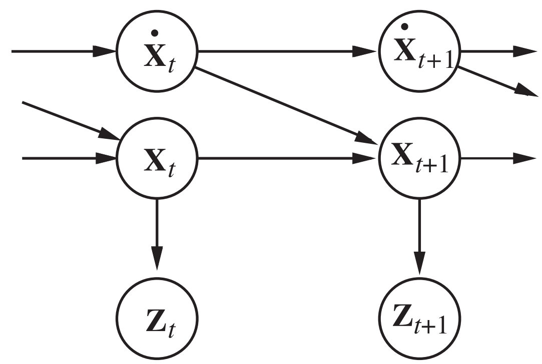


Figure 15.9 FILES: figures/kalman-network.eps (Tue Nov 3 16:23:06 2009). Bayesian network structure for a linear dynamical system with position $\mathbf{X}_t$, velocity $\dot{\mathbf{X}}_t$, and position measurement $\mathbf{Z}_t$.

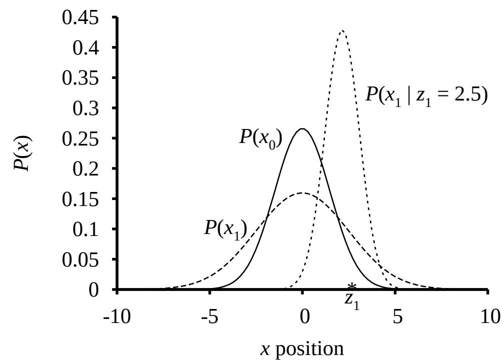
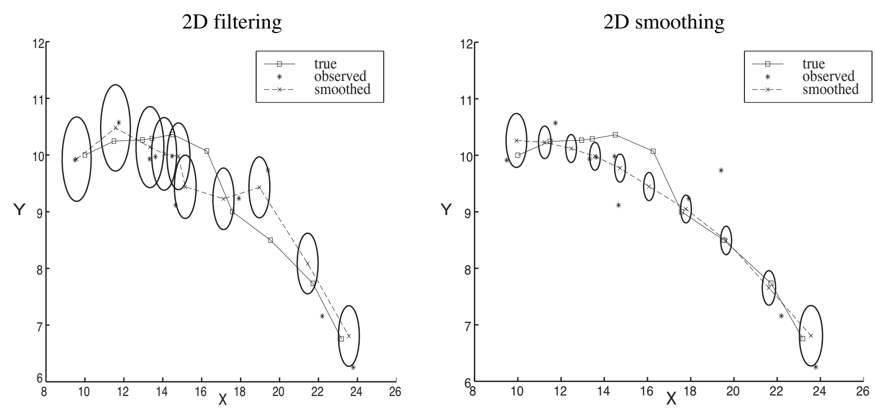


Figure 15.10 **FILES: .** Stages in the Kalman filter update cycle for a random walk with a prior given by $\mu_0 = 0.0$ and $\sigma_0 = 1.0$, transition noise given by $\sigma_x = 2.0$, sensor noise given by $\sigma_z = 1.0$, and a first observation $z_1 = 2.5$ (marked on the x-axis). Notice how the prediction $P(x_1)$ is flattened out, relative to $P(x_0)$, by the transition noise. Notice also that the mean of the posterior distribution $P(x_1 \mid z_1)$ is slightly to the left of the observation z_1 because the mean is a weighted average of the prediction and the observation.



(a) (b)

Figure 15.11 FILES: figures/kalman-2D.eps (Tue Nov 3 16:23:06 2009). (a) Results of Kalman filtering for an object moving on the X–Y plane, showing the true trajectory (left to right), a series of noisy observations, and the trajectory estimated by Kalman filtering. Variance in the position estimate is indicated by the ovals. (b) The results of Kalman smoothing for the same observation sequence.

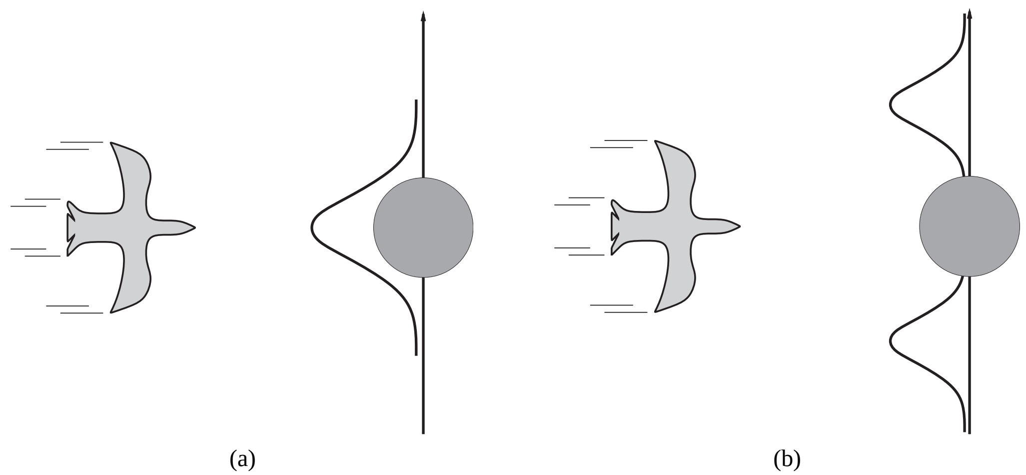


Figure 15.12 FILES: figures/kalman-bird1.eps (Tue Nov 3 16:23:06 2009) figures/kalman-bird2.eps (Tue Nov 3 16:23:06 2009). A bird flying toward a tree (top views). (a) A Kalman filter will predict the location of the bird using a single Gaussian centered on the obstacle. (b) A more realistic model allows for the bird's evasive action, predicting that it will fly to one side or the other.

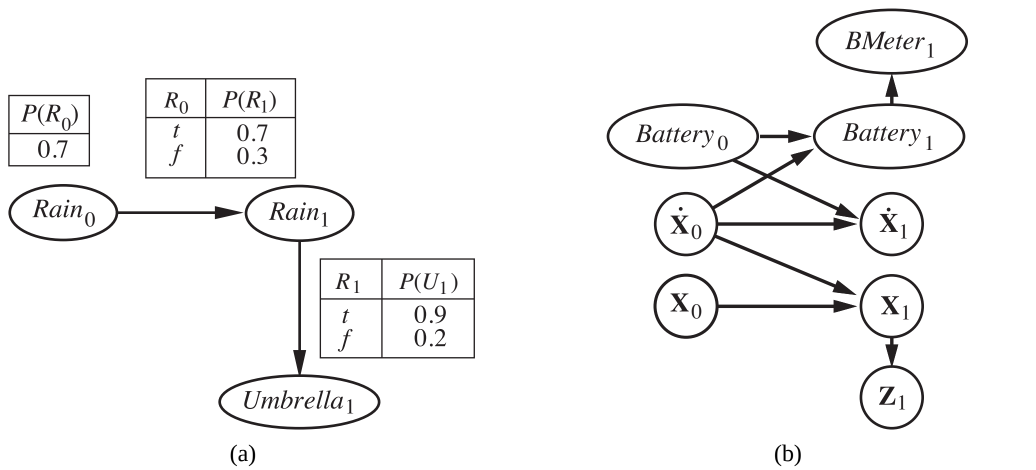


Figure 15.13 FILES: figures/umbrella-1slice.eps (Tue Nov 3 16:23:58 2009) figures/robot-dbn1.eps (Tue Nov 3 16:23:33 2009). (a) Specification of the prior, transition model, and sensor model for the umbrella DBN. All subsequent slices are assumed to be copies of slice 1. (b) A simple DBN for robot motion in the X–Y plane.

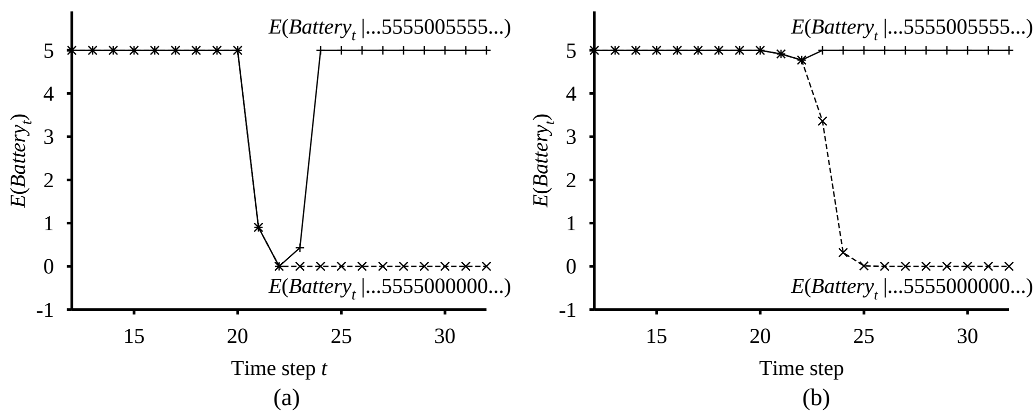


Figure 15.14 **FILES: .** (a) Upper curve: trajectory of the expected value of $Battery_t$ for an observation sequence consisting of all 5s except for 0s at $t = 21$ and $t = 22$, using a simple Gaussian error model. Lower curve: trajectory when the observation remains at 0 from $t = 21$ onwards. (b) The same experiment run with the transient failure model. Notice that the transient failure is handled well, but the persistent failure results in excessive pessimism about the battery charge.

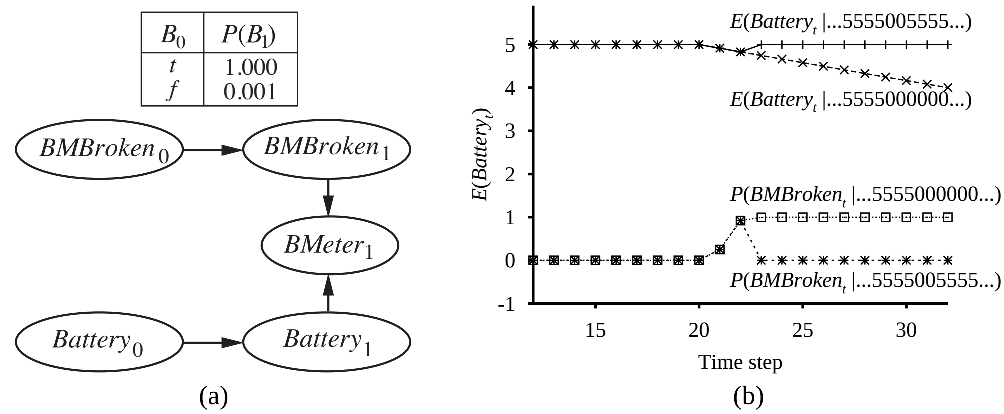


Figure 15.15 FILES: figures/battery-persistence.eps (Tue Nov 3 16:22:26 2009). (a) A DBN fragment showing the sensor status variable required for modeling persistent failure of the battery sensor. (b) Upper curves: trajectories of the expected value of $Battery_t$ for the "transient failure" and "permanent failure" observations sequences. Lower curves: probability trajectories for $BMBroken$ given the two observation sequences.

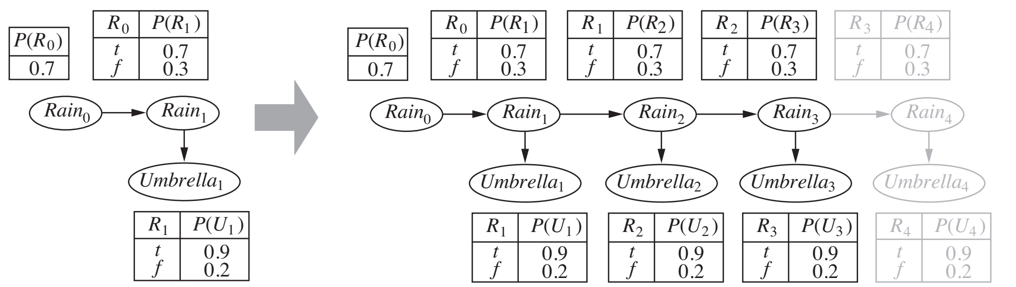


Figure 15.16 **FILES: figures/dbn-unrolling.eps (Tue Nov 3 16:22:36 2009).** Unrolling a dynamic Bayesian network: slices are replicated to accommodate the observation sequence $Umbrella_{1:3}$. Further slices have no effect on inferences within the observation period.

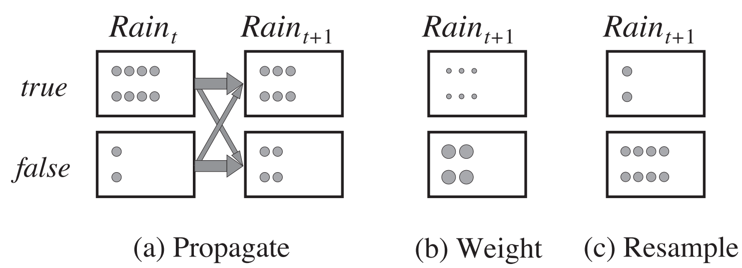


Figure 15.18 FILES: figures/umbrella-particle.eps (Tue Nov 3 16:23:59 2009). The particle filtering update cycle for the umbrella DBN with $N = 10$, showing the sample populations of each state. (a) At time t, 8 samples indicate $rain$ and 2 indicate $\neg rain$. Each is propagated forward by sampling the next state through the transition model. At time $t+1$, 6 samples indicate $rain$ and 4 indicate $\neg rain$. (b) $\neg umbrella$ is observed at $t+1$. Each sample is weighted by its likelihood for the observation, as indicated by the size of the circles. (c) A new set of 10 samples is generated by weighted random selection from the current set, resulting in 2 samples that indicate $rain$ and 8 that indicate $\neg rain$.

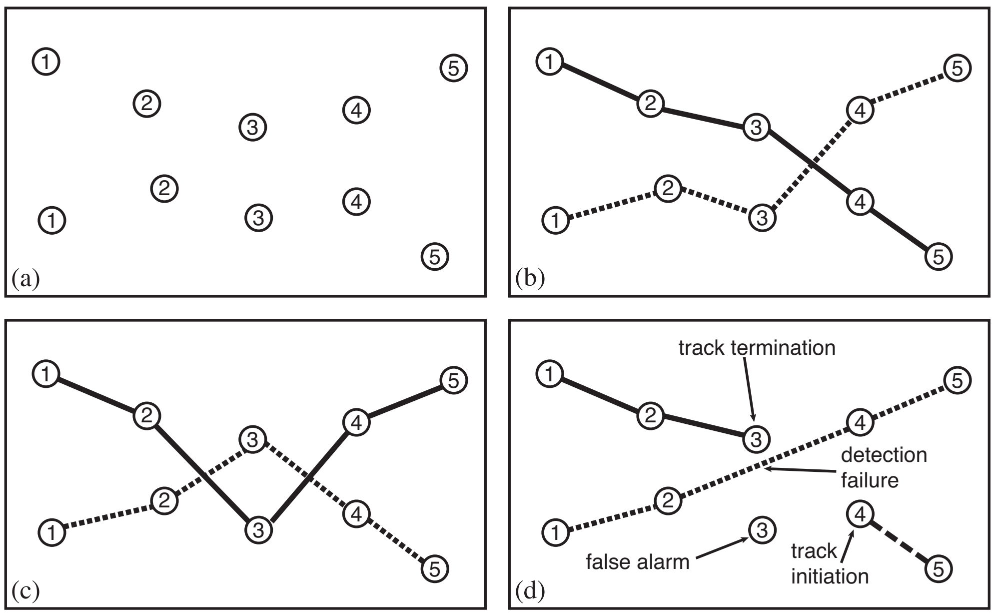


Figure 15.19 **FILES: figures/classical-DA.eps (Tue Nov 3 16:22:33 2009).** (a) Observations made of object locations in 2D space over five time steps. Each observation is labeled with the time step but does not identify the object that produced it. (b–c) Possible hypotheses about the underlying object tracks. (d) A hypothesis for the case in which false alarms, detection failures, and track initiation/termination are possible.

(a) (b)

Figure 15.20 FILES: figures/traffic-upstream.eps (Tue Nov 3 16:23:58 2009) figures/traffic-downstream.eps (Tue Nov 3 16:23:57 2009). Images from (a) upstream and (b) downstream surveillance cameras roughly two miles apart on Highway 99 in Sacramento, California. The boxed vehicle has been identified at both cameras.

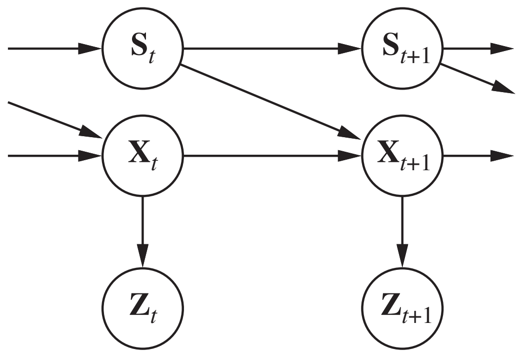


Figure 15.21 FILES: figures/switching-kf.eps (Tue Nov 3 16:23:50 2009). A Bayesian network representation of a switching Kalman filter. The switching variable S_t is a discrete state variable whose value determines the transition model for the continuous state variables $\mathbf{X}_t$. For any discrete state i, the transition model $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{X}_t, S_t = i)$ is a linear Gaussian model, just as in a regular Kalman filter. The transition model for the discrete state, $\mathbf{P}(S_{t+1}|S_t)$, can be thought of as a matrix, as in a hidden Markov model.

16 MAKING SIMPLE DECISIONS

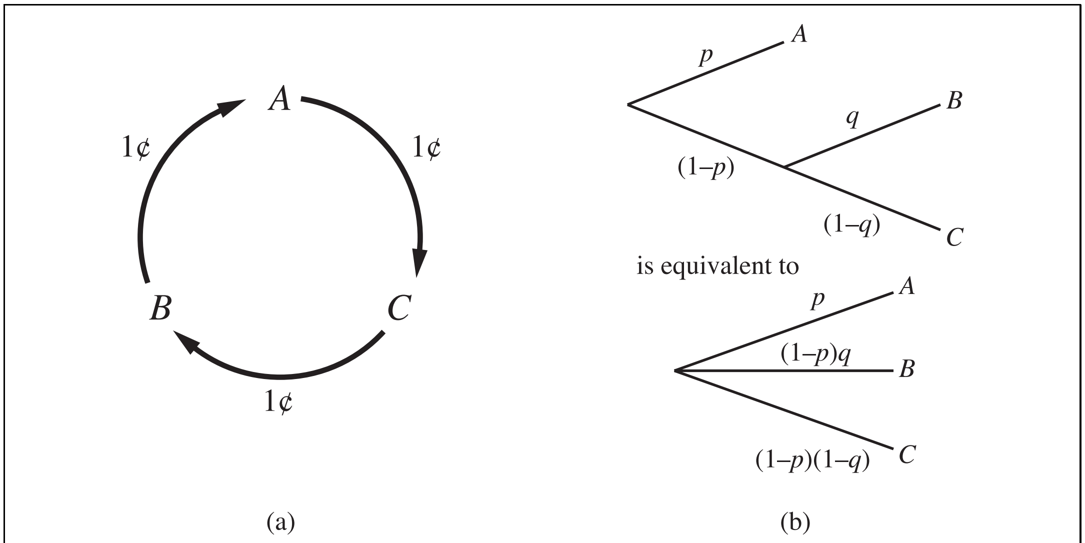


Figure 16.1 **FILES: figures/cash-machine-and-decomposability.eps (Tue Nov 3 13:30:24 2009).** (a) A cycle of exchanges showing that the nontransitive preferences $A \succ B \succ C \succ A$ result in irrational behavior. (b) The decomposability axiom.

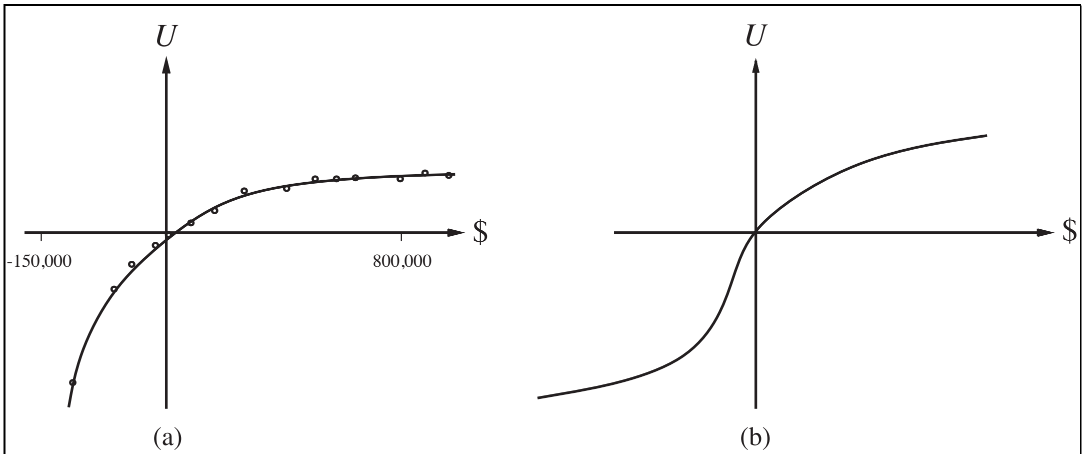


Figure 16.2 FILES: figures/utility-curve.eps (Tue Nov 3 16:24:00 2009). The utility of money. (a) Empirical data for Mr. Beard over a limited range. (b) A typical curve for the full range.

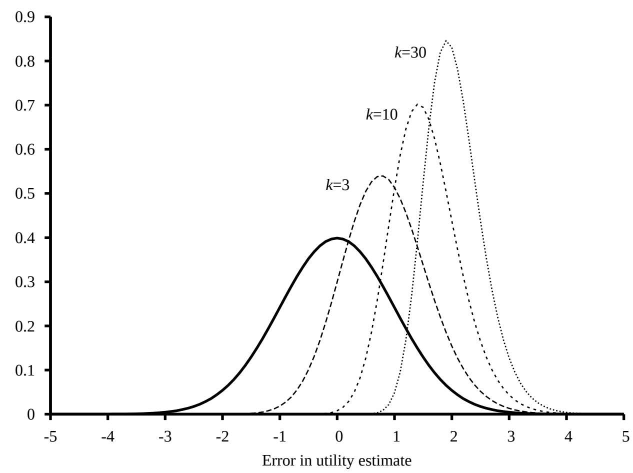


Figure 16.3 **FILES: .** Plot of the error in each of k utility estimates and of the distribution of the maximum of k estimates for $k = 3$, 10, and 30.

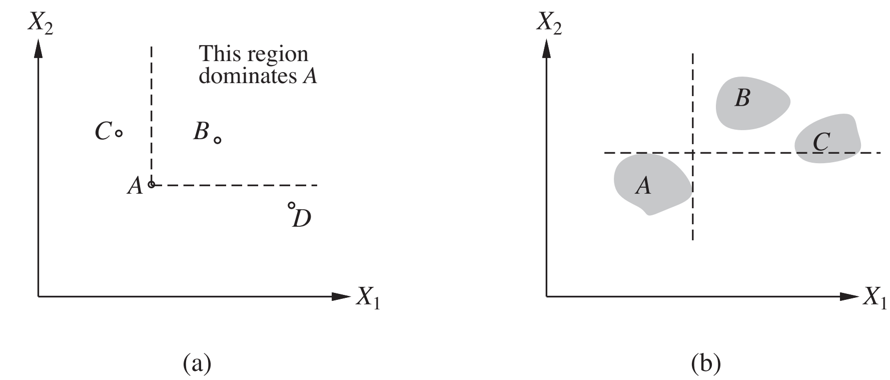


Figure 16.4 FILES: figures/strict-dominance.eps (Tue Nov 3 13:49:56 2009). Strict dominance. (a) Deterministic: Option A is strictly dominated by B but not by C or D. (b) Uncertain: A is strictly dominated by B but not by C.

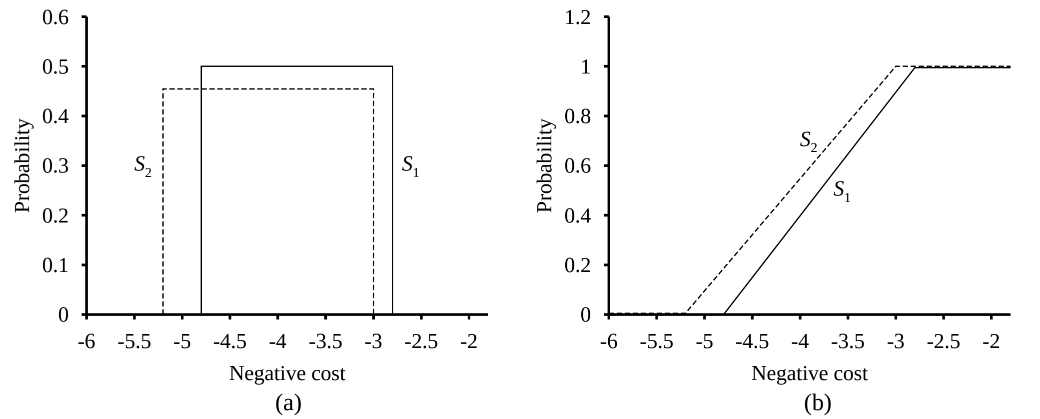


Figure 16.5 **FILES: .** Stochastic dominance. (a) S_1 stochastically dominates S_2 on cost. (b) Cumulative distributions for the negative cost of S_1 and S_2.

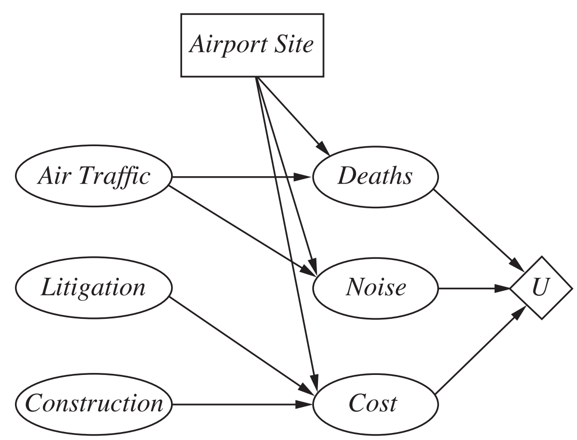


Figure 16.6 FILES: figures/airport-id.eps (Tue Nov 3 16:22:19 2009). A simple decision network for the airport-siting problem.

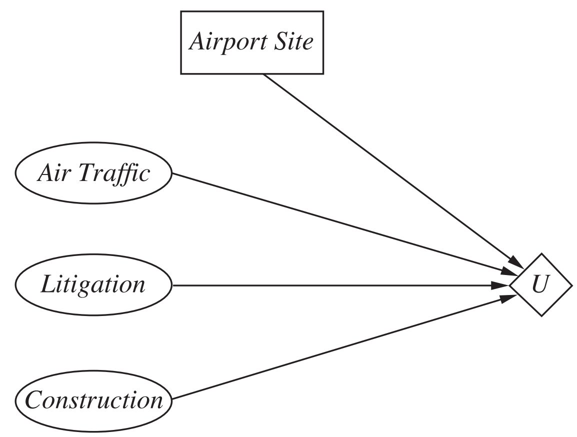


Figure 16.7 FILES: figures/airport-au-id.eps (Tue Nov 3 16:22:19 2009). A simplified representation of the airport-siting problem. Chance nodes corresponding to outcome states have been factored out.

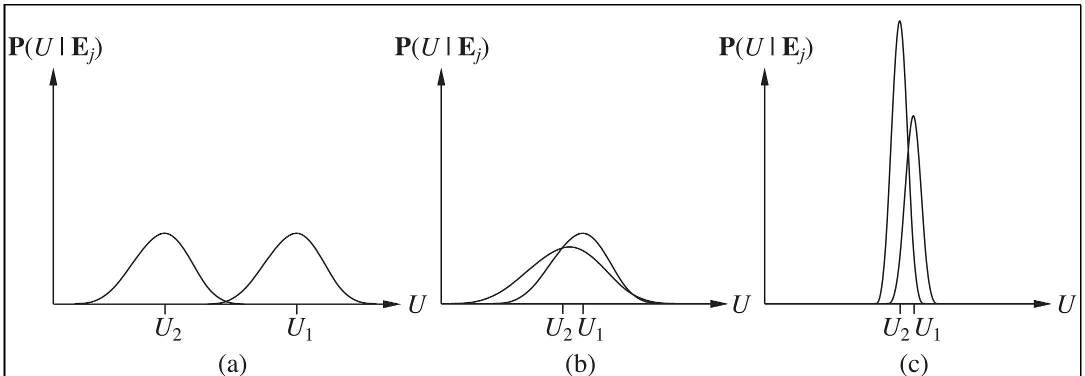


Figure 16.8 **FILES: figures/3cases.eps (Tue Nov 3 16:22:10 2009).** Three generic cases for the value of information. In (a), a_1 will almost certainly remain superior to a_2, so the information is not needed. In (b), the choice is unclear and the information is crucial. In (c), the choice is unclear, but because it makes little difference, the information is less valuable. (Note: The fact that U_2 has a high peak in (c) means that its expected value is known with higher certainty than U_1.)

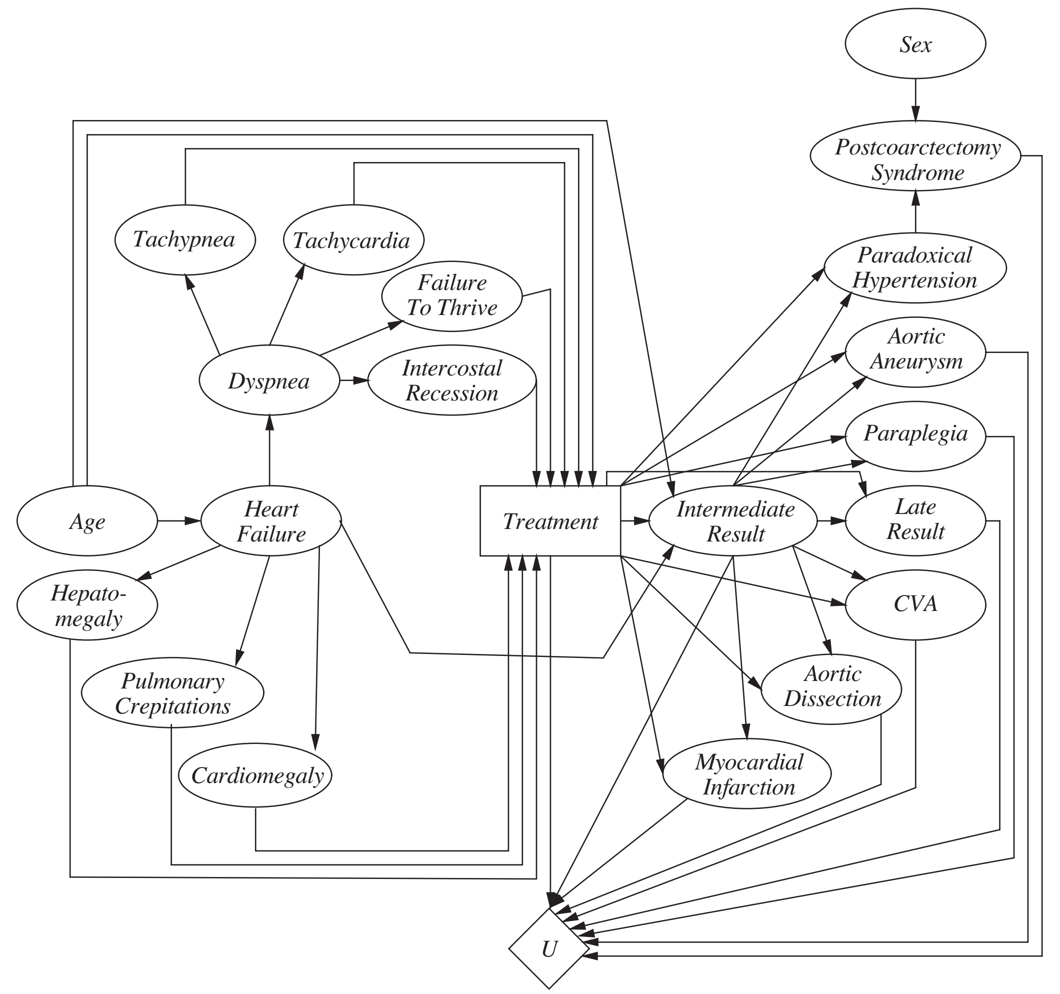


Figure 16.10 **FILES: figures/heart-infl-diagram.eps (Tue Nov 3 16:23:01 2009).** Influence diagram for aortic coarctation (courtesy of Peter Lucas).

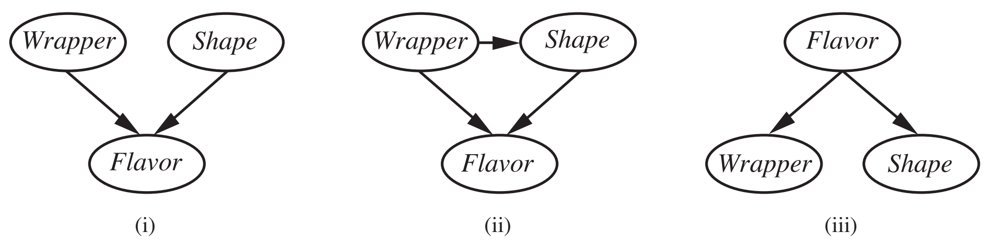


Figure 16.11 FILES: figures/3candy.eps (Tue Nov 3 16:22:10 2009). Three proposed Bayes nets for the Surprise Candy problem, Exercise **??**.

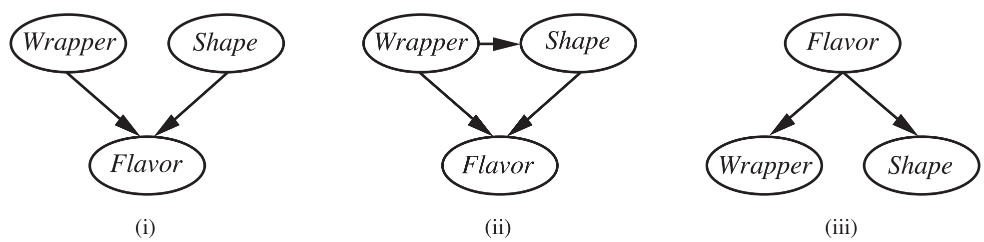


Figure 16.12 FILES: figures/3candy.eps (Tue Nov 3 16:22:10 2009). Three proposed Bayes nets for the Surprise Candy problem, Exercise **??**.

17 MAKING COMPLEX DECISIONS

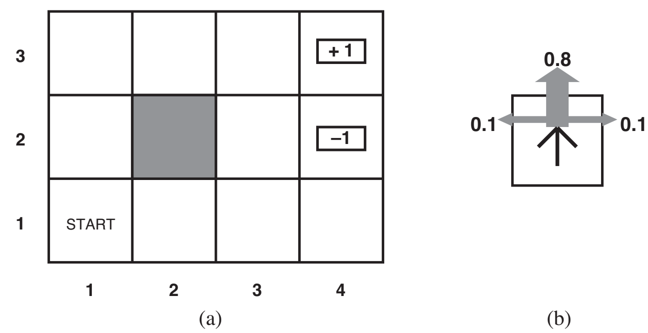


Figure 17.1 FILES: figures/sequential-decision-world.eps (Tue Nov 3 16:23:43 2009). (a) A simple 4×3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. The two terminal states have reward +1 and –1, respectively, and all other states have a reward of –0.04.

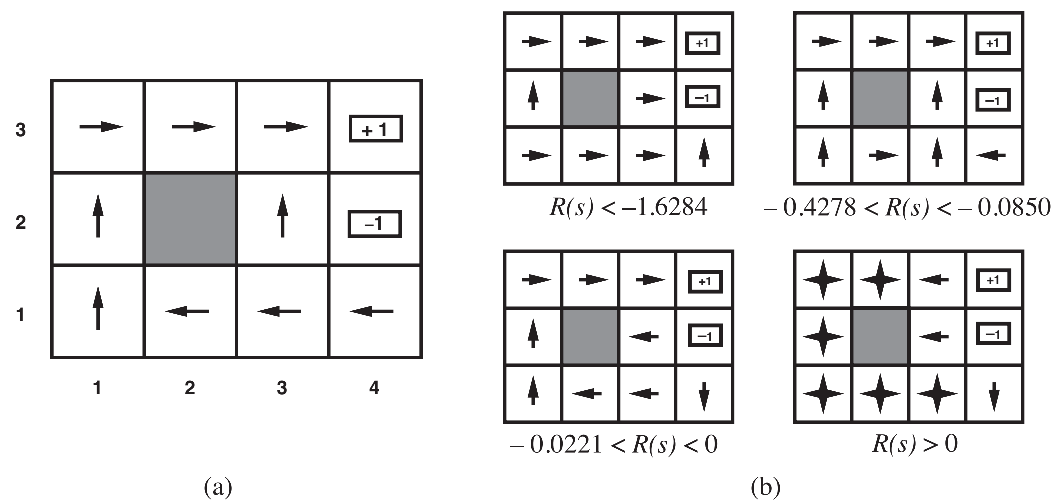


Figure 17.2 FILES: figures/sequential-decision-policies.eps (Tue Nov 3 16:23:42 2009). (a) An optimal policy for the stochastic environment with $R(s) = -0.04$ in the nonterminal states. (b) Optimal policies for four different ranges of $R(s)$.

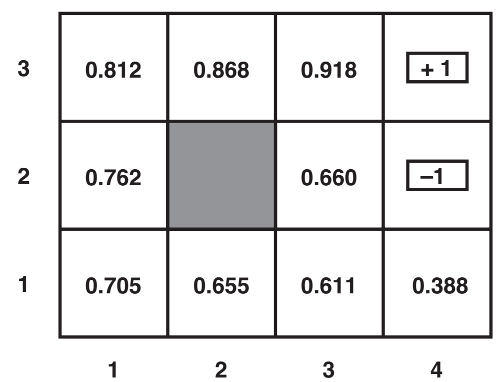


Figure 17.3 FILES: figures/sequential-decision-values.eps (Tue Nov 3 16:23:42 2009). The utilities of the states in the 4×3 world, calculated with $\gamma = 1$ and $R(s) = -0.04$ for nonterminal states.

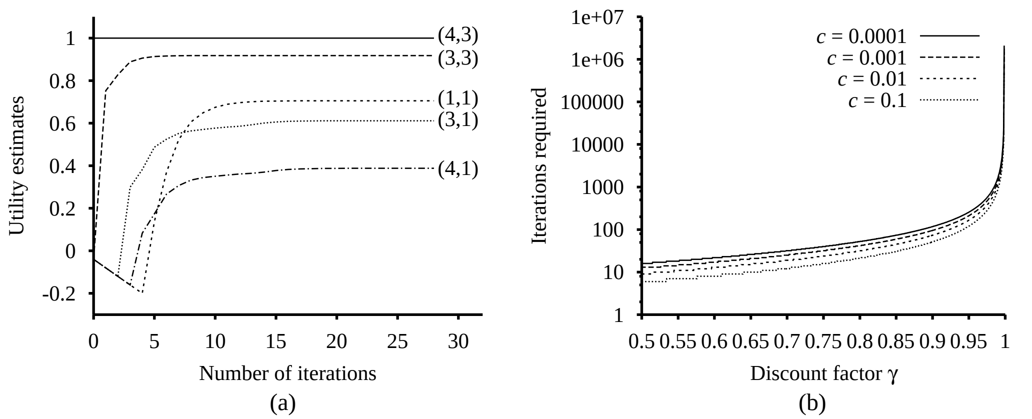


Figure 17.5 **FILES: .** (a) Graph showing the evolution of the utilities of selected states using value iteration. (b) The number of value iterations k required to guarantee an error of at most $\epsilon = c \cdot R_{\max}$, for different values of c, as a function of the discount factor γ.

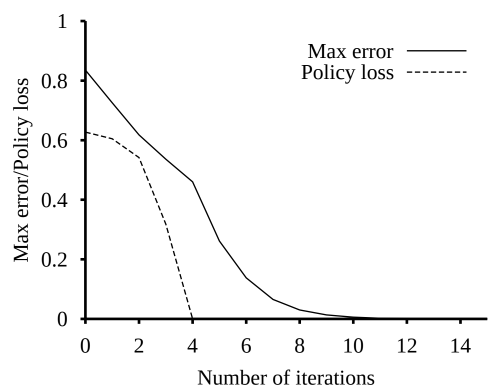


Figure 17.6 **FILES: .** The maximum error $||U_i - U||$ of the utility estimates and the policy loss $||U^{\pi_i} - U||$, as a function of the number of iterations of value iteration.

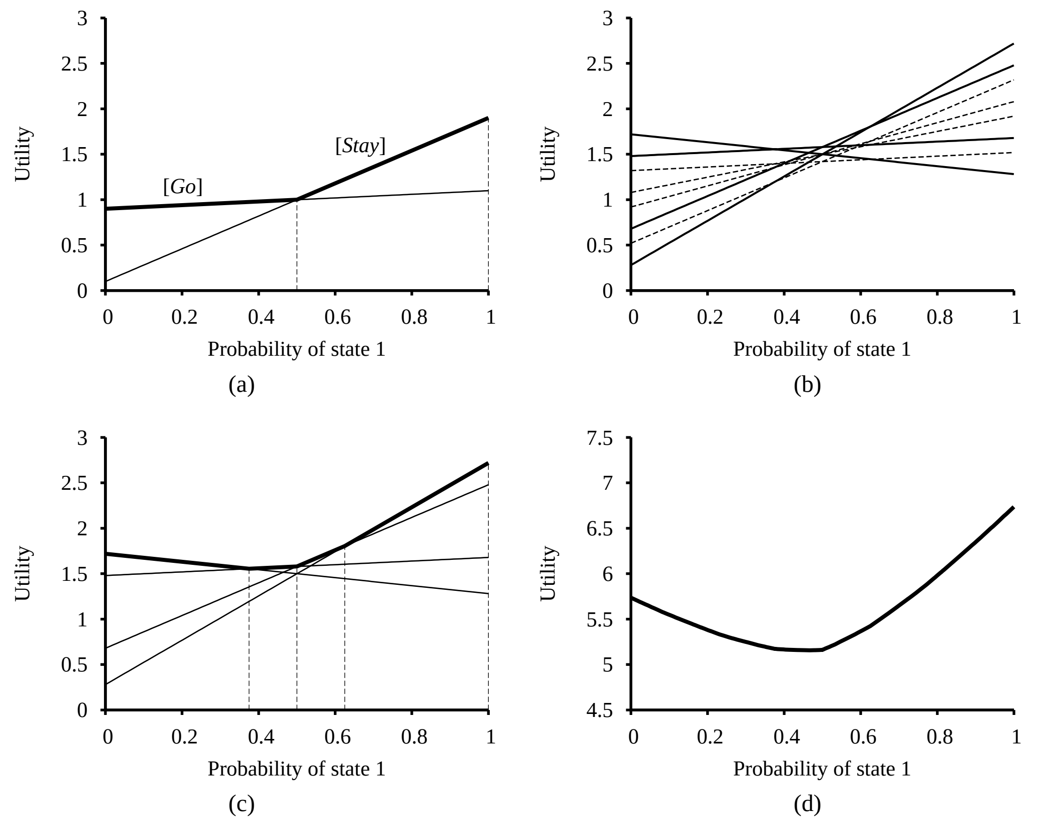


Figure 17.8 **FILES: .** (a) Utility of two one-step plans as a function of the initial belief state $b(1)$ for the two-state world, with the corresponding utility function shown in bold. (b) Utilities for 8 distinct two-step plans. (c) Utilities for four undominated two-step plans. (d) Utility function for optimal eight-step plans.

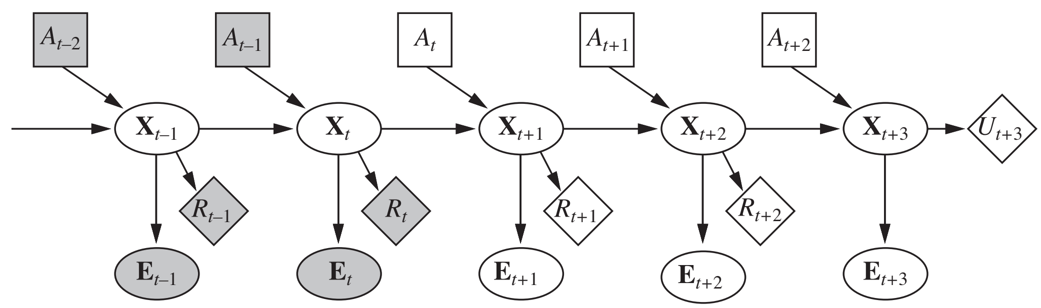


Figure 17.10 FILES: figures/generic-ddn.eps (Tue Nov 3 16:22:53 2009). The generic structure of a dynamic decision network. Variables with known values are shaded. The current time is t and the agent must decide what to do—that is, choose a value for A_t. The network has been unrolled into the future for three steps and represents future rewards, as well as the utility of the state at the look-ahead horizon.

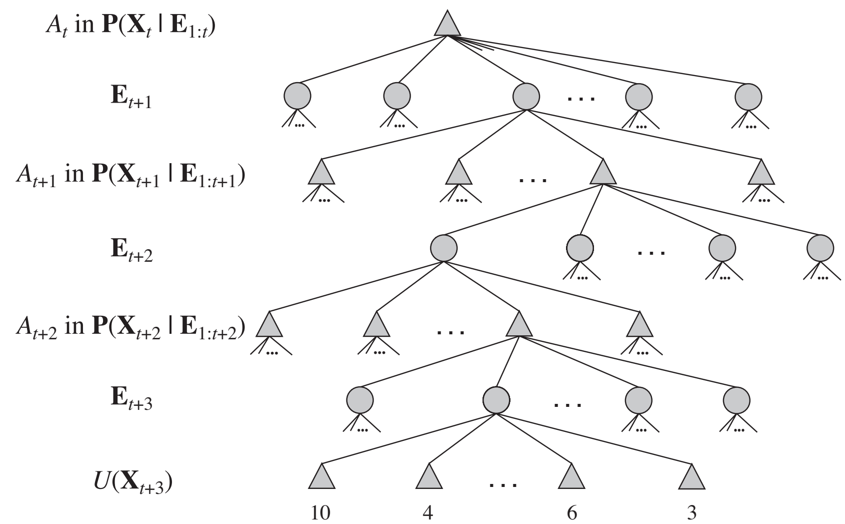


Figure 17.11 FILES: figures/pomdp-tree.eps (Tue Nov 3 16:23:20 2009). Part of the look-ahead solution of the DDN in Figure 17.10. Each decision will be taken in the belief state indicated.

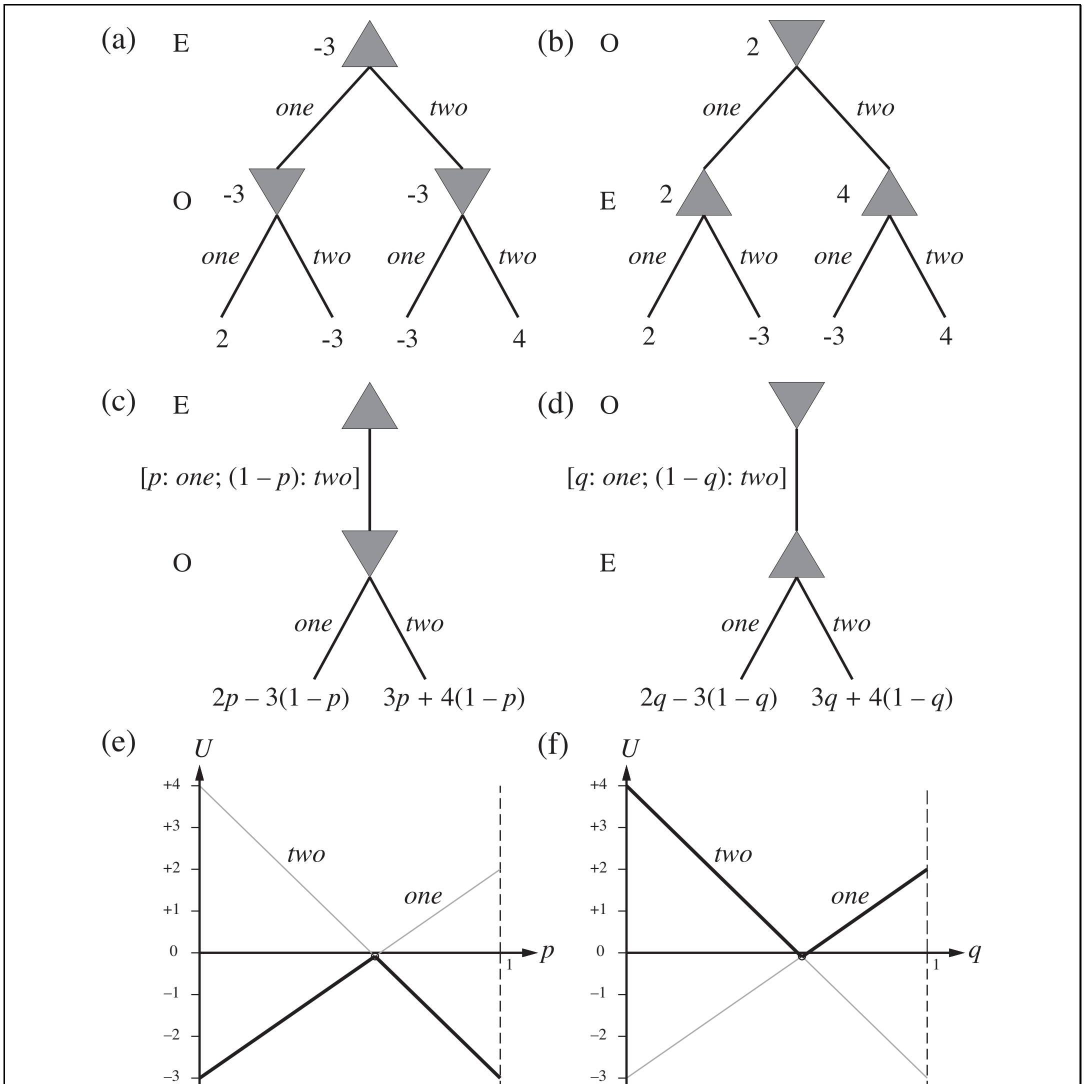


Figure 17.12 FILES: figures/morra-trees.eps (Tue Nov 3 16:23:11 2009). (a) and (b): Minimax game trees for two-finger Morra if the players take turns playing pure strategies. (c) and (d): Parameterized game trees where the first player plays a mixed strategy. The payoffs depend on the probability parameter (p or q) in the mixed strategy. (e) and (f): For any particular value of the probability parameter, the second player will choose the "better" of the two actions, so the value of the first player's mixed strategy is given by the heavy lines. The first player will choose the probability parameter for the mixed strategy at the intersection point.

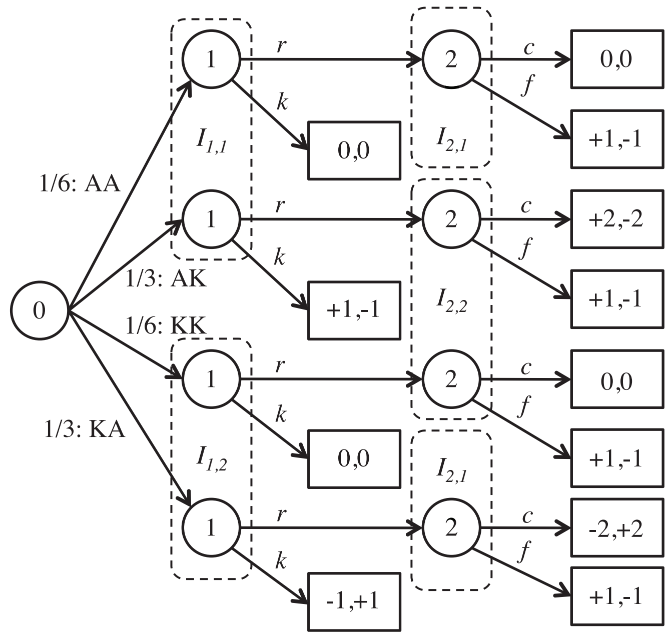


Figure 17.13 FILES: figures/extensive-game.eps (Tue Nov 3 16:22:45 2009). Extensive form of a simplified version of poker.

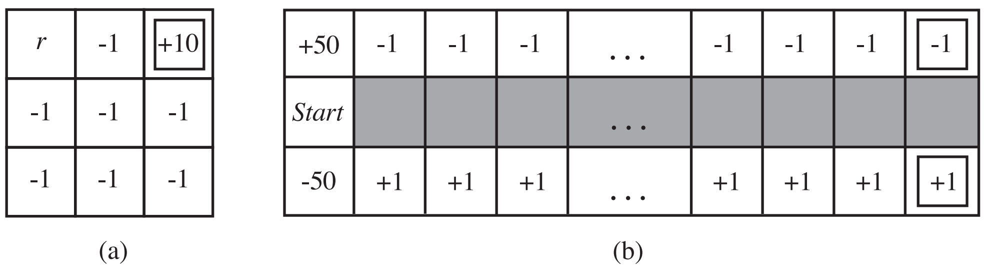


Figure 17.14 FILES: figures/grid-mdp-figure.eps (Tue Nov 3 16:22:55 2009). (a) 3×3 world for Exercise **??**. The reward for each state is indicated. The upper right square is a terminal state. (b) 101×3 world for Exercise **??** (omitting 93 identical columns in the middle). The start state has reward 0.

18 LEARNING FROM EXAMPLES

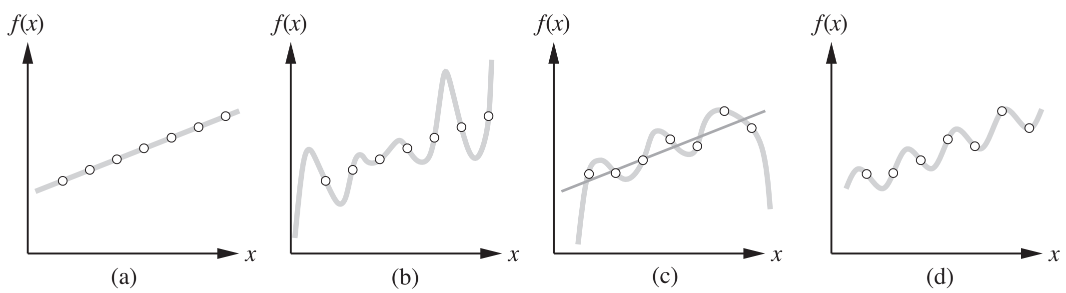


Figure 18.1 FILES: figures/xy-plot.eps (Tue Nov 3 16:24:13 2009). (a) Example $(x, f(x))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data set.

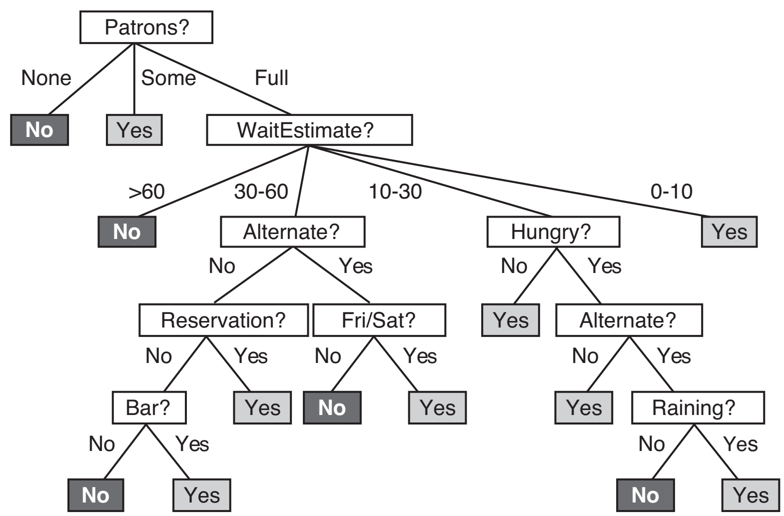


Figure 18.2 FILES: figures/restaurant-tree.eps (Tue Nov 3 16:23:29 2009). A decision tree for deciding whether to wait for a table.

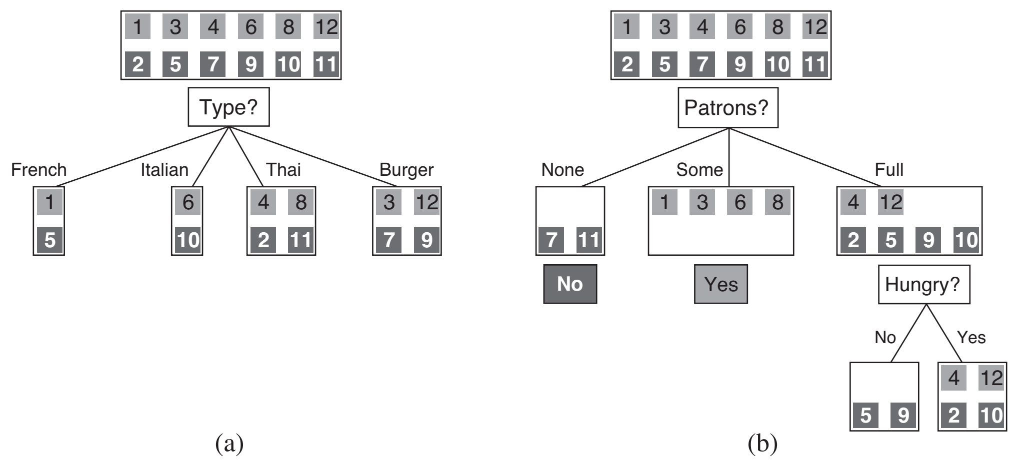


Figure 18.4 **FILES: figures/restaurant-stub.eps (Tue Nov 3 16:23:28 2009).** Splitting the examples by testing on attributes. At each node we show the positive (light boxes) and negative (dark boxes) examples remaining. (a) Splitting on *Type* brings us no nearer to distinguishing between positive and negative examples. (b) Splitting on *Patrons* does a good job of separating positive and negative examples. After splitting on *Patrons*, *Hungry* is a fairly good second test.

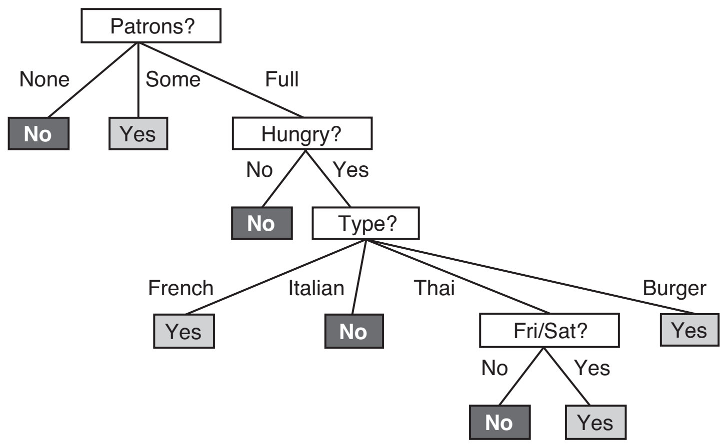


Figure 18.6 FILES: figures/induced-restaurant-tree.eps (Tue Nov 3 16:23:04 2009). The decision tree induced from the 12-example training set.

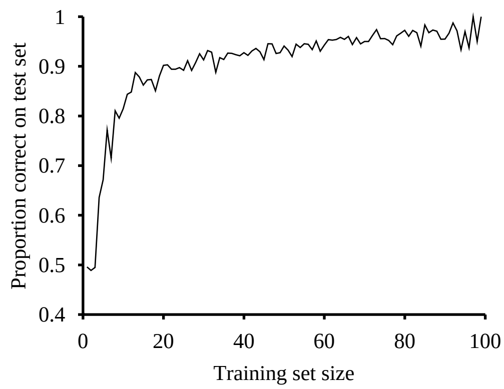


Figure 18.7 **FILES: .** A learning curve for the decision tree learning algorithm on 100 randomly generated examples in the restaurant domain. Each data point is the average of 20 trials.

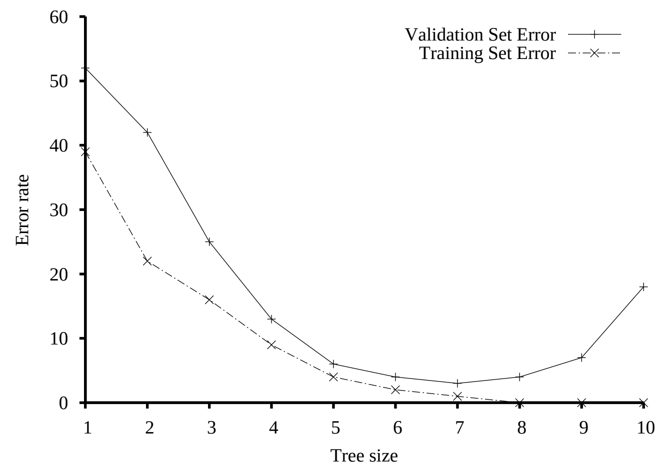


Figure 18.9 **FILES: .** Error rates on training data (lower, dashed line) and validation data (upper, solid line) for different size decision trees. We stop when the training set error rate asymptotes, and then choose the tree with minimal error on the validation set; in this case the tree of size 7 nodes.

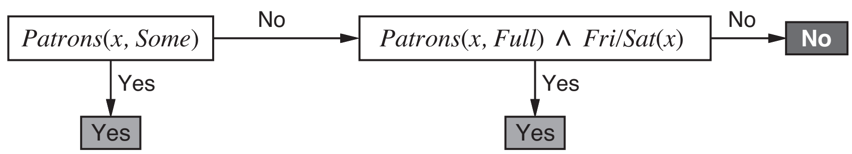


Figure 18.10 **FILES: figures/decision-list.eps (Tue Nov 3 16:22:37 2009).** A decision list for the restaurant problem.

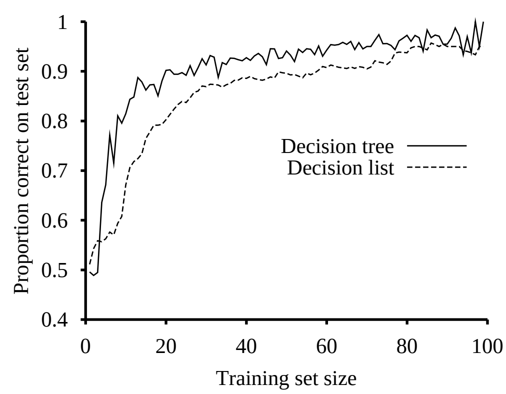


Figure 18.12 **FILES: .** Learning curve for DECISION-LIST-LEARNING algorithm on the restaurant data. The curve for DECISION-TREE-LEARNING is shown for comparison.

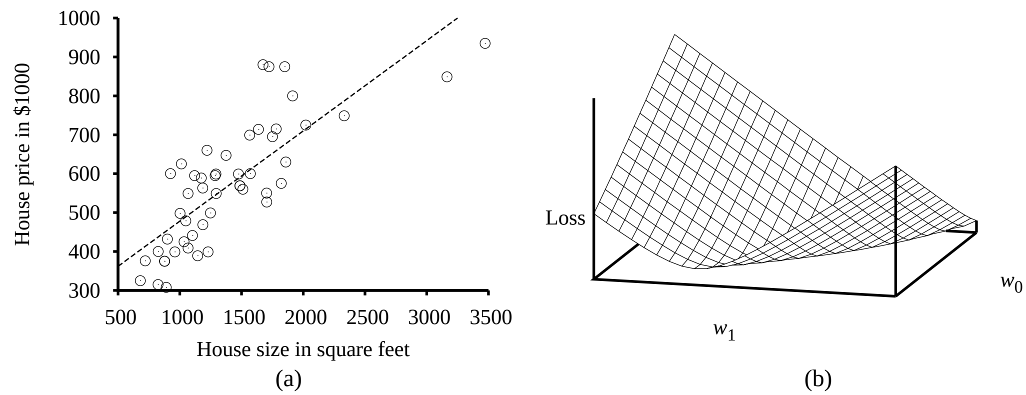


Figure 18.13 **FILES: .** (a) Data points of price versus floor space of houses for sale in Berkeley, CA, in July 2009, along with the linear function hypothesis that minimizes squared error loss: $y = 0.232x + 246$. (b) Plot of the loss function $\sum_j (w_1 x_j + w_0 - y_j)^2$ for various values of w_0, w_1. Note that the loss function is convex, with a single global minimum.

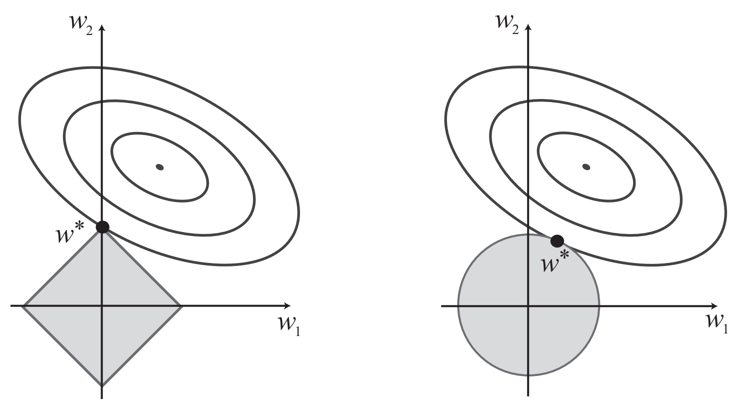


Figure 18.14 FILES: figures/diamond.eps (Wed Nov 4 14:45:53 2009). Why L_1 regularization tends to produce a sparse model. (a) With L_1 regularization (box), the minimal achievable loss (concentric contours) often occurs on an axis, meaning a weight of zero. (b) With L_2 regularization (circle), the minimal loss is likely to occur anywhere on the circle, giving no preference to zero weights.

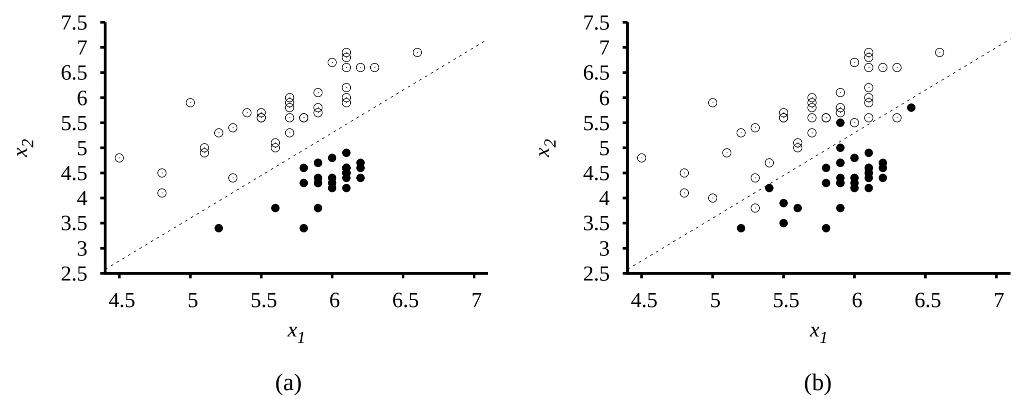


Figure 18.15 **FILES: .** (a) Plot of two seismic data parameters, body wave magnitude x_1 and surface wave magnitude x_2, for earthquakes (white circles) and nuclear explosions (black circles) occurring between 1982 and 1990 in Asia and the Middle East (?). Also shown is a decision boundary between the classes. (b) The same domain with more data points. The earthquakes and explosions are no longer linearly separable.

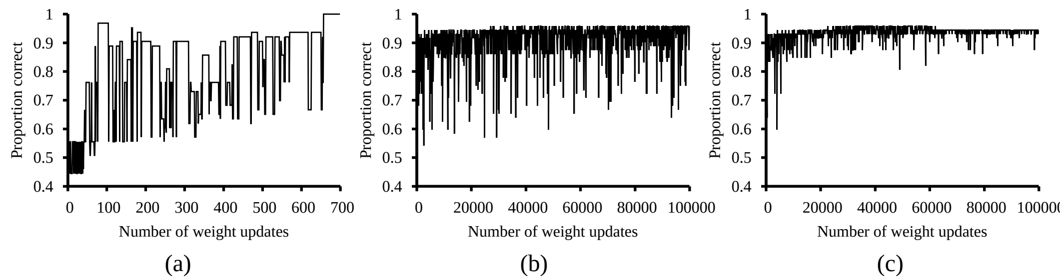


Figure 18.16 **FILES: .** (a) Plot of total training-set accuracy vs. number of iterations through the training set for the perceptron learning rule, given the earthquake/explosion data in Figure 18.14(a). (b) The same plot for the noisy, non-separable data in Figure 18.14(b); note the change in scale of the x-axis. (c) The same plot as in (b), with a learning rate schedule $\alpha(t) = 1000/(1000 + t)$.

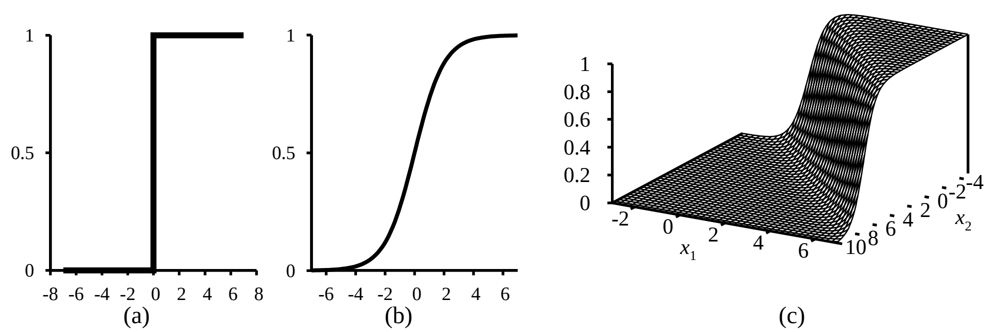


Figure 18.17 **FILES: .** (a) The hard threshold function $Threshold(z)$ with 0/1 output. Note that the function is nondifferentiable at $z = 0$. (b) The logistic function, $Logistic(z) = \frac{1}{1+e^{-z}}$, also known as the sigmoid function. (c) Plot of a logistic regression hypothesis $h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x})$ for the data shown in Figure 18.14(b).

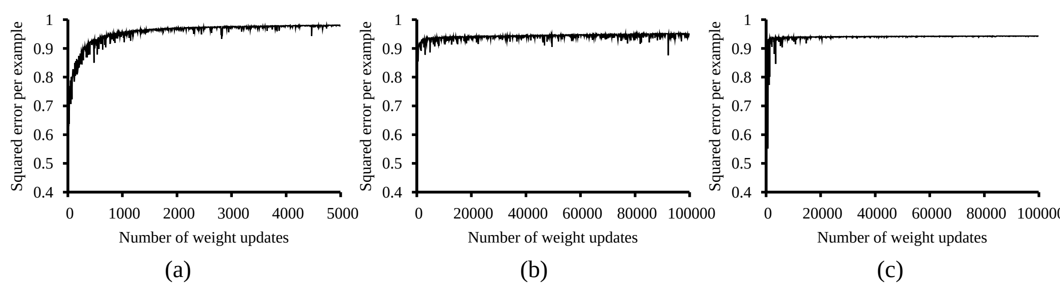


Figure 18.18 **FILES: .** Repeat of the experiments in Figure 18.15 using logistic regression and squared error. The plot in (a) covers 5000 iterations rather than 1000, while (b) and (c) use the same scale.

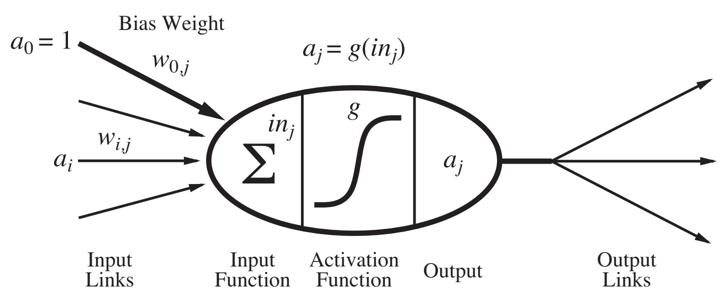


Figure 18.19 **FILES: figures/neuron-unit.eps (Wed Nov 4 11:23:13 2009).** A simple mathematical model for a neuron. The unit's output activation is $a_j = g(\sum_{i=0}^{n} w_{i,j} a_i)$, where a_i is the output activation of unit i and $w_{i,j}$ is the weight on the link from unit i to this unit.

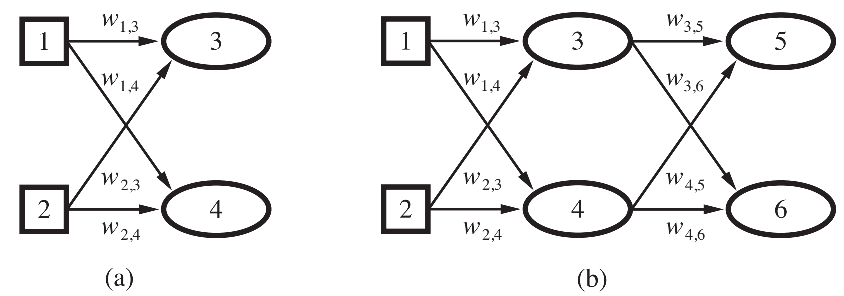


Figure 18.20 FILES: figures/neural-net.eps (Wed Nov 4 11:08:22 2009). (a) A perceptron network with two inputs and two output units. (b) A neural network with two inputs, one hidden layer of two units, and one output unit. Not shown are the dummy inputs and their associated weights.

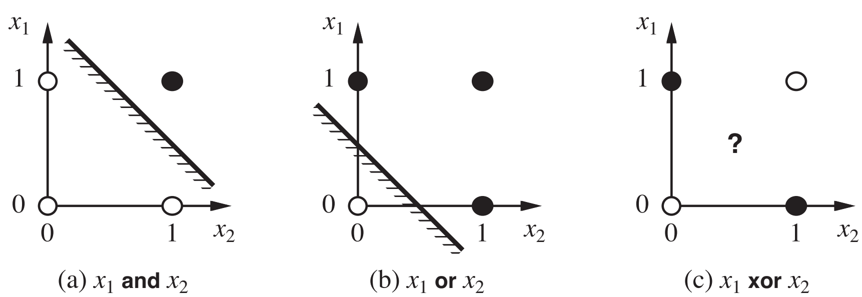


Figure 18.21 FILES: figures/perceptron-linear.eps (Tue Nov 3 16:23:17 2009). Linear separability in threshold perceptrons. Black dots indicate a point in the input space where the value of the function is 1, and white dots indicate a point where the value is 0. The perceptron returns 1 on the region on the non-shaded side of the line. In (c), no such line exists that correctly classifies the inputs.

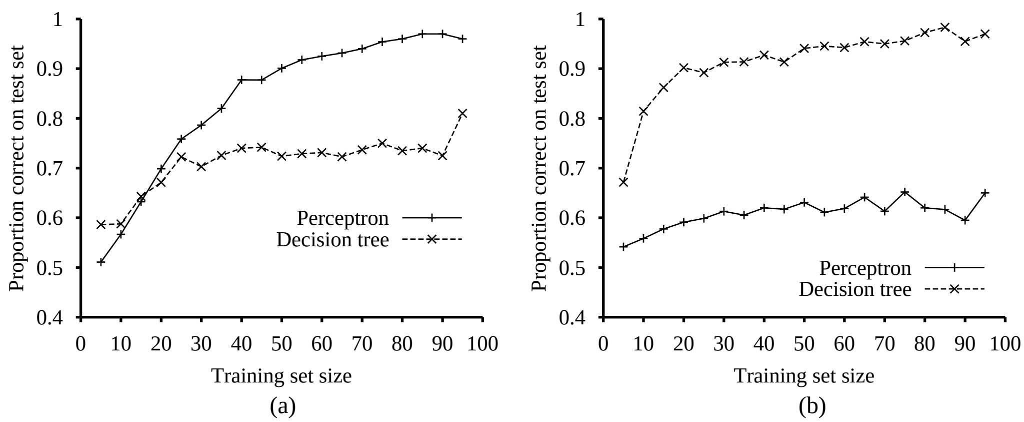


Figure 18.22 **FILES: .** Comparing the performance of perceptrons and decision trees. (a) Perceptrons are better at learning the majority function of 11 inputs. (b) Decision trees are better at learning the $WillWait$ predicate in the restaurant example.

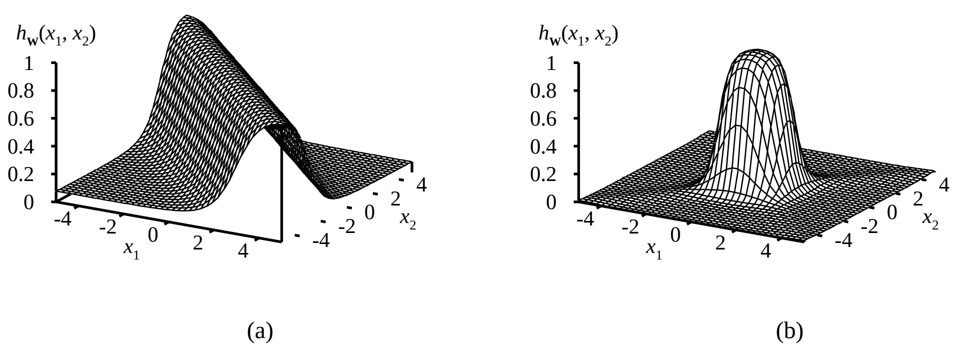


Figure 18.23 **FILES: .** (a) The result of combining two opposite-facing soft threshold functions to produce a ridge. (b) The result of combining two ridges to produce a bump.

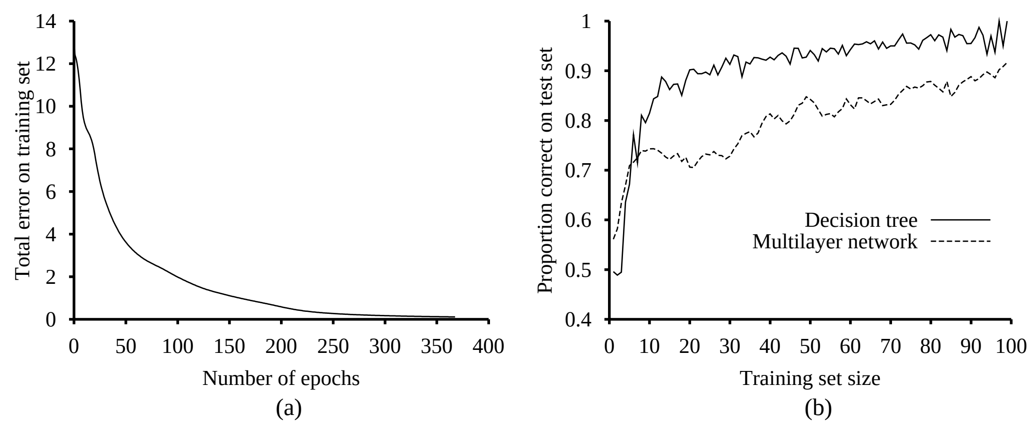
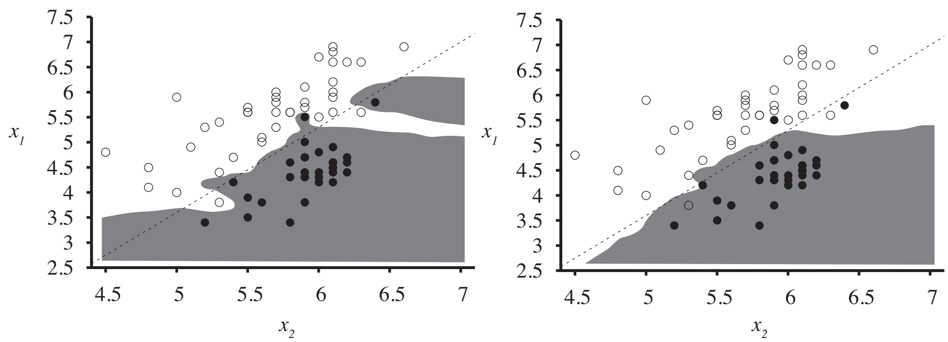


Figure 18.25 **FILES: .** (a) Training curve showing the gradual reduction in error as weights are modified over several epochs, for a given set of examples in the restaurant domain. (b) Comparative learning curves showing that decision-tree learning does slightly better on the restaurant problem than back-propagation in a multilayer network.



$(k=1)$ $(k=5)$

Figure 18.26 FILES: figures/earthquake-nn1.eps (Tue Nov 3 16:22:38 2009) figures/earthquake-nn5.eps (Tue Nov 3 16:22:38 2009). (a) A k-nearest-neighbor model showing the extent of the explosion class for the data in Figure 18.14, with $k=1$. Overfitting is apparent. (b) With $k=5$, the overfitting problem goes away for this data set.

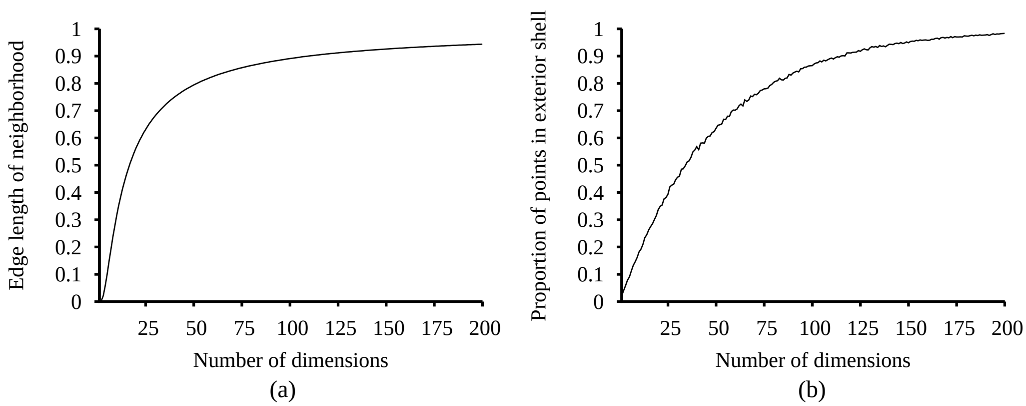


Figure 18.27 **FILES: .** The curse of dimensionality: (a) The length of the average neighborhood for 10-nearest-neighbors in a unit hypercube with 1,000,000 points, as a function of the number of dimensions. (b) The proportion of points that fall within a thin shell consisting of the outer 1% of the hypercube, as a function of the number of dimensions. Sampled from 10,000 randomly distributed points.

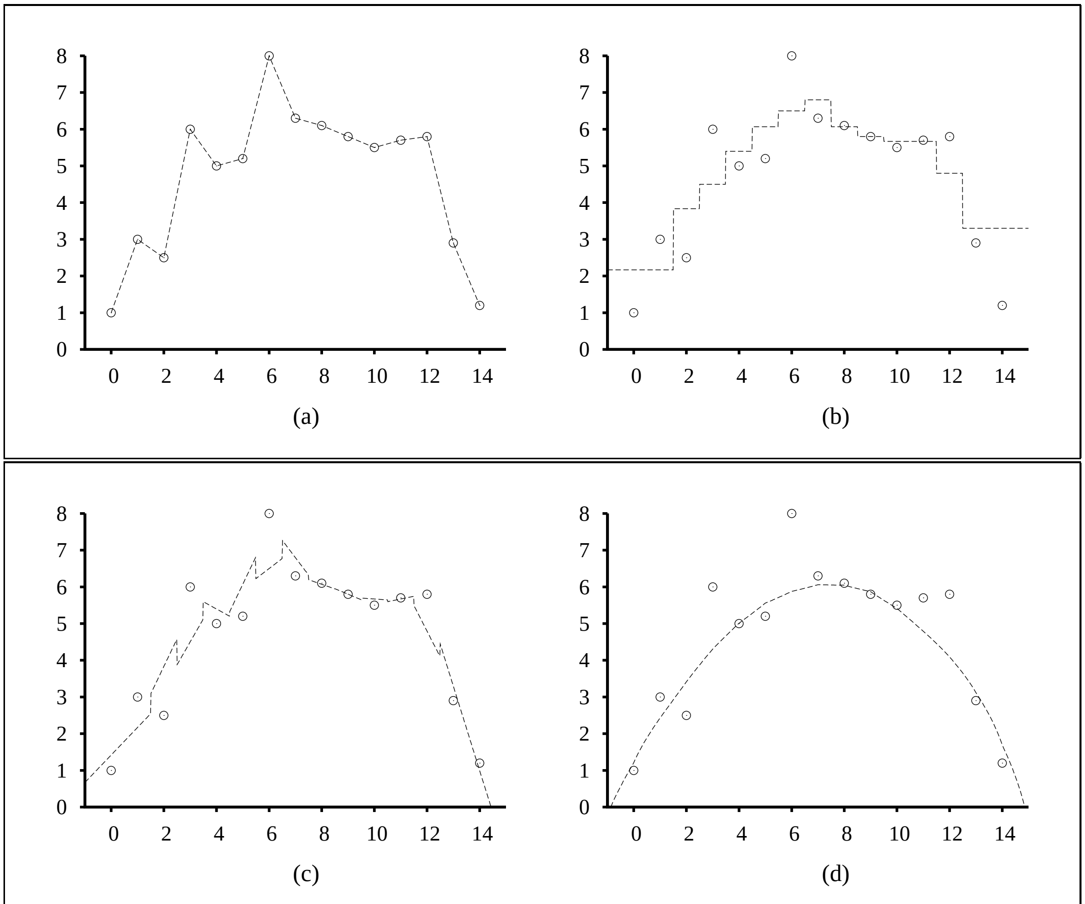


Figure 18.28 **FILES: .** Nonparametric regression models: (a) connect the dots, (b) 3-nearest neighbors average, (c) 3-nearest-neighbors linear regression, (d) locally weighted regression with a quadratic kernel of width $k = 10$.

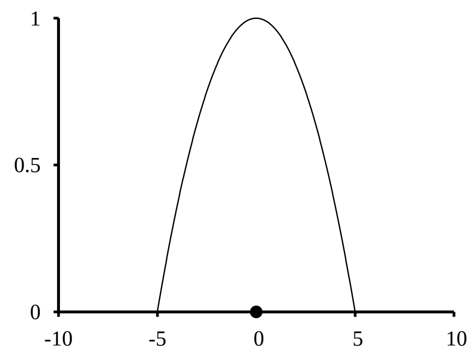


Figure 18.29 **FILES: .** A quadratic kernel, $\mathcal{K}(d) = \max(0, 1 - (2|x|/k)^2)$, with kernel width $k = 10$, centered on the query point $x = 0$.

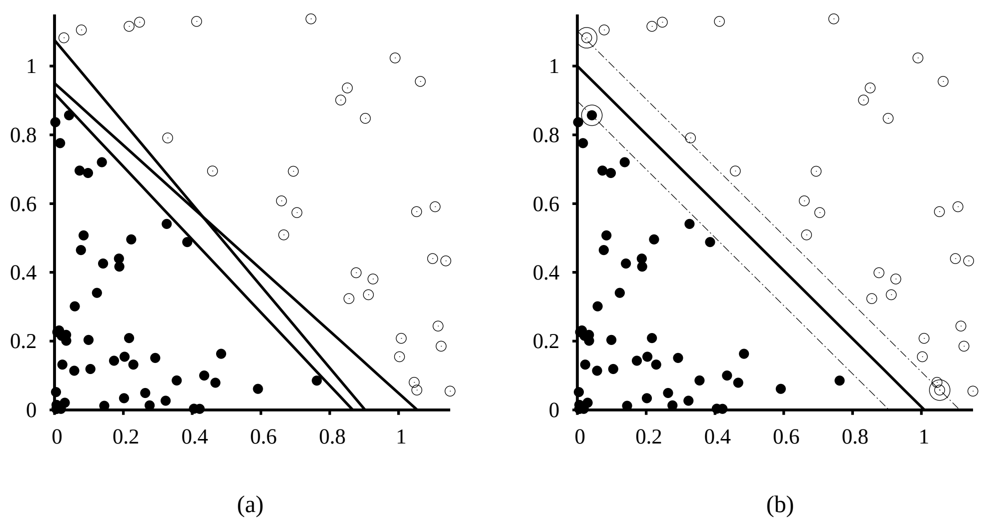


Figure 18.30 **FILES: .** Support vector machine classification: (a) Two classes of points (black and white circles) and three candidate linear separators. (b) The maximum margin separator (heavy line), is at the midpoint of the **margin** (area between dashed lines). The **support vectors** (points with large circles) are the examples closest to the separator.

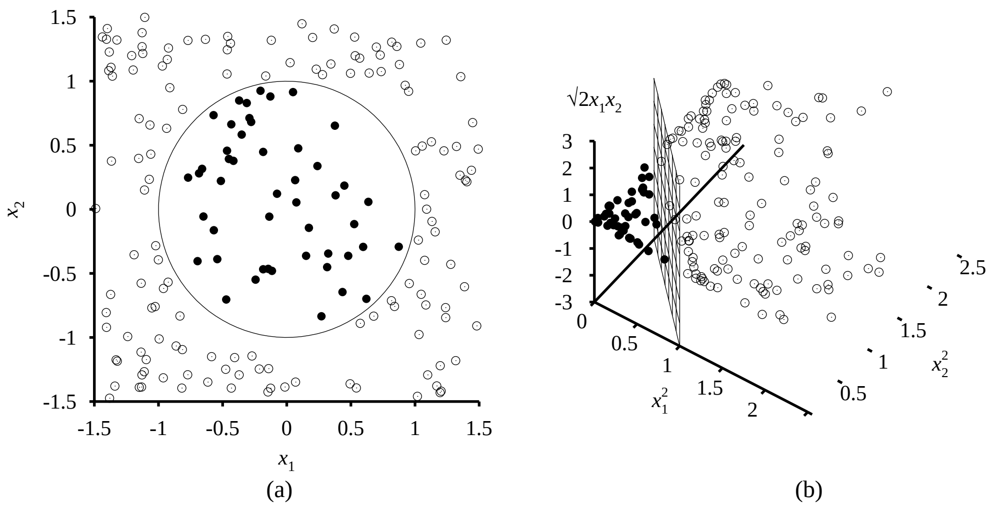


Figure 18.31 **FILES: .** (a) A two-dimensional training set with positive examples as black circles and negative examples as white circles. The true decision boundary, $x_1^2 + x_2^2 \leq 1$, is also shown. (b) The same data after mapping into a three-dimensional input space $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$. The circular decision boundary in (a) becomes a linear decision boundary in three dimensions. Figure 18.29(b) gives a closeup of the separator in (b).

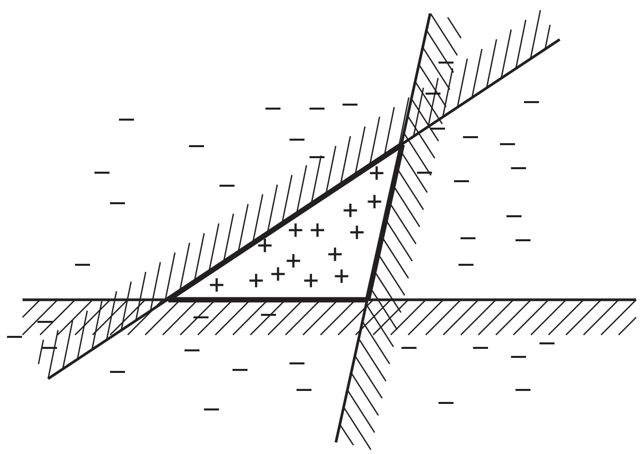


Figure 18.32 FILES: figures/ensemble-expressiveness.eps (Tue Nov 3 16:22:41 2009). Illustration of the increased expressive power obtained by ensemble learning. We take three linear threshold hypotheses, each of which classifies positively on the unshaded side, and classify as positive any example classified positively by all three. The resulting triangular region is a hypothesis not expressible in the original hypothesis space.

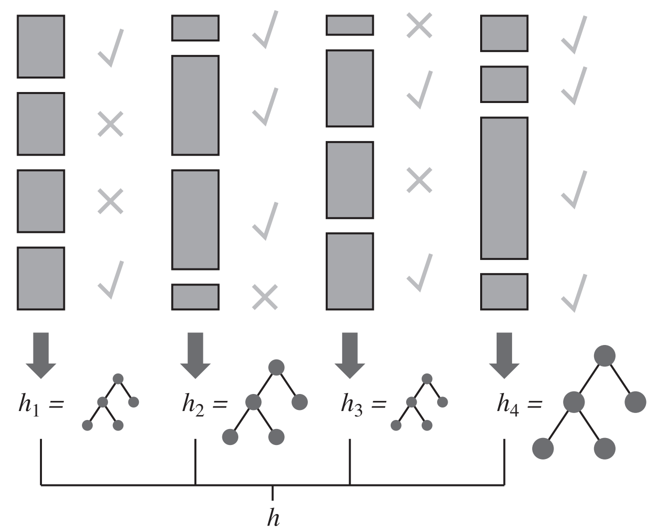


Figure 18.33 FILES: figures/boosting.eps (Tue Nov 3 16:22:28 2009). How the boosting algorithm works. Each shaded rectangle corresponds to an example; the height of the rectangle corresponds to the weight. The checks and crosses indicate whether the example was classified correctly by the current hypothesis. The size of the decision tree indicates the weight of that hypothesis in the final ensemble.

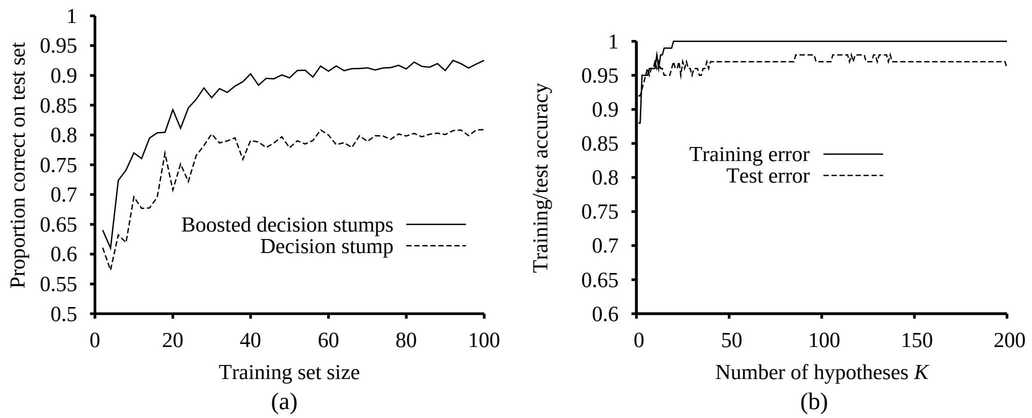


Figure 18.35 **FILES: .** (a) Graph showing the performance of boosted decision stumps with $K = 5$ versus unboosted decision stumps on the restaurant data. (b) The proportion correct on the training set and the test set as a function of K, the number of hypotheses in the ensemble. Notice that the test set accuracy improves slightly even after the training accuracy reaches 1, i.e., after the ensemble fits the data exactly.

Figure 18.36 FILES: figures/easy-hard.eps (Wed Nov 4 15:38:34 2009). Examples from the NIST database of handwritten digits. Top row: examples of digits 0–9 that are easy to identify. Bottom row: more difficult examples of the same digits.

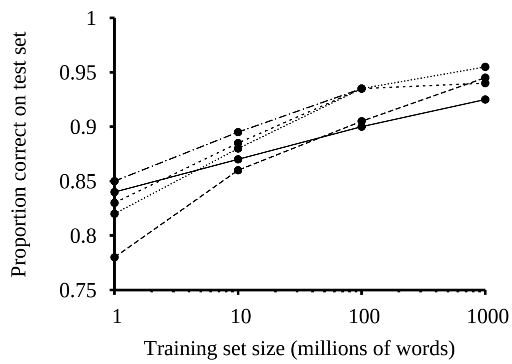
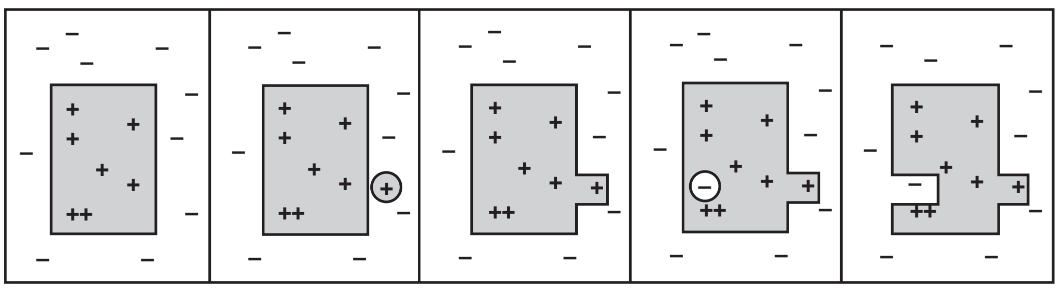


Figure 18.37 **FILES: .** Learning curves for five learning algorithms on a common task. Note that there appears to be more room for improvement in the horizontal direction (more training data) than in the vertical direction (different machine learning algorithm). Adapted from ? (?).

19 KNOWLEDGE IN LEARNING



(a) (b) (c) (d) (e)

Figure 19.1 FILES: figures/cbh.eps (Tue Nov 3 16:22:32 2009). (a) A consistent hypothesis. (b) A false negative. (c) The hypothesis is generalized. (d) A false positive. (e) The hypothesis is specialized.

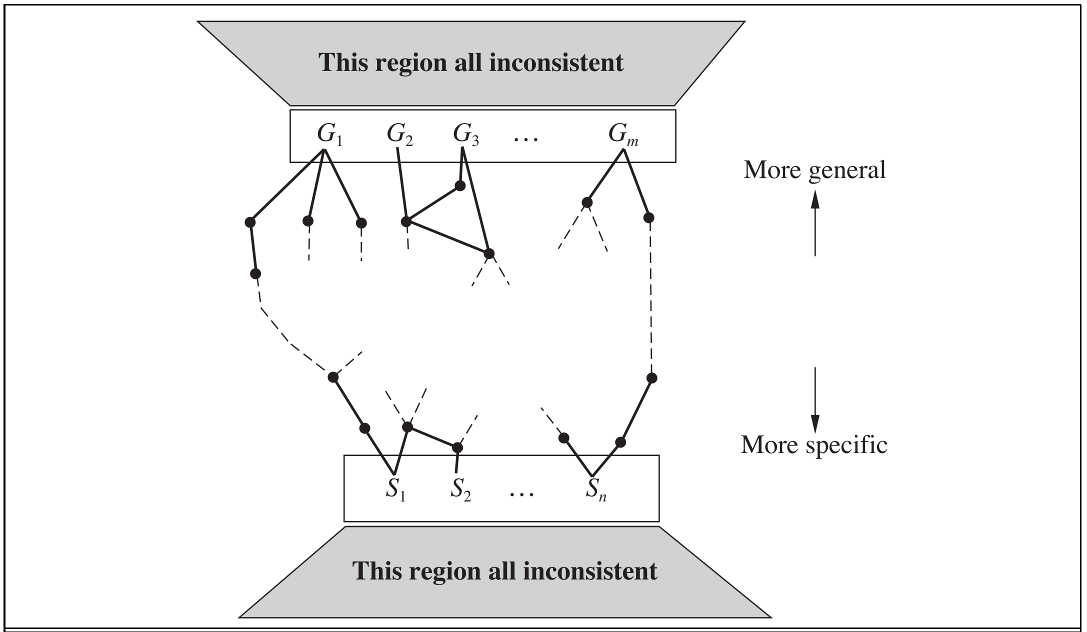


Figure 19.4 FILES: figures/version-space.eps (Tue Nov 3 16:24:02 2009). The version space contains all hypotheses consistent with the examples.

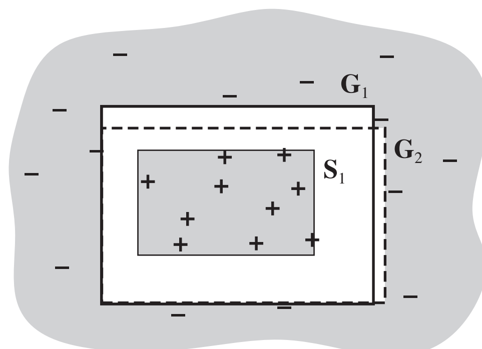


Figure 19.5 FILES: figures/vs-proof.eps (Tue Nov 3 16:24:05 2009). The extensions of the members of G and S. No known examples lie in between the two sets of boundaries.

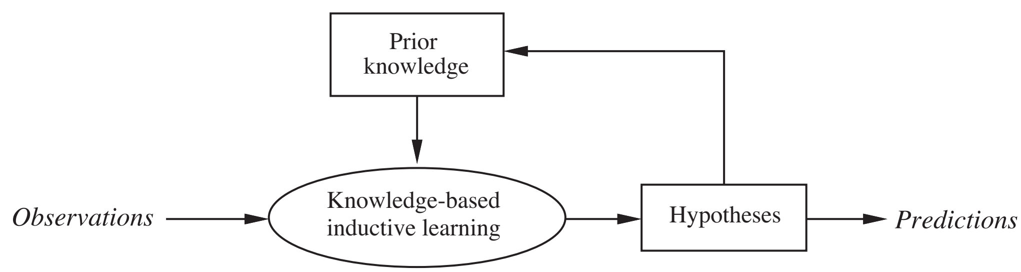


Figure 19.6 FILES: figures/cumulative-learning.eps (Tue Nov 3 16:22:36 2009). A cumulative learning process uses, and adds to, its stock of background knowledge over time.

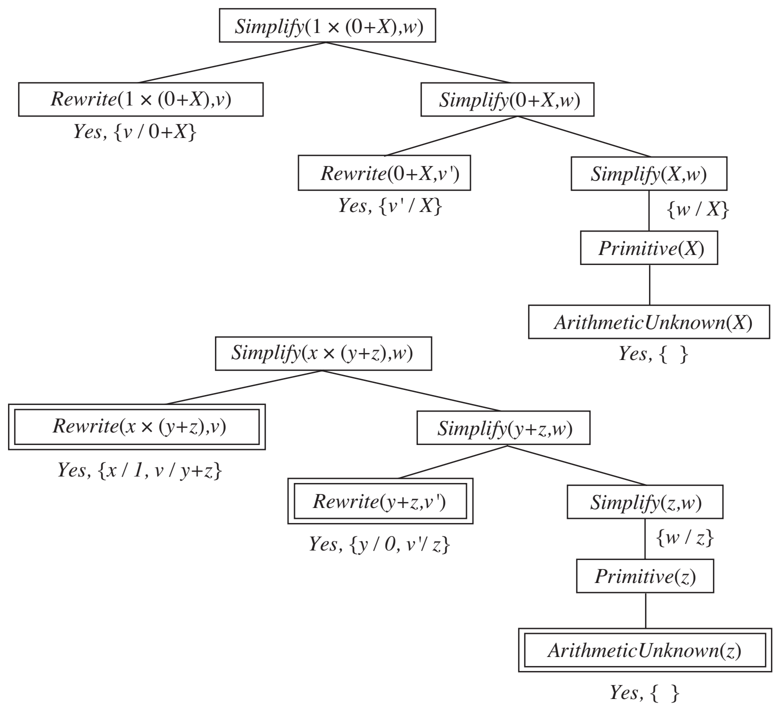


Figure 19.7 **FILES: figures/simplify-proof2.eps (Tue Nov 3 16:23:44 2009).** Proof trees for the simplification problem. The first tree shows the proof for the original problem instance, from which we can derive

$$ArithmeticUnknown(z) \Rightarrow Simplify(1 \times (0 + z), z) .$$

The second tree shows the proof for a problem instance with all constants replaced by variables, from which we can derive a variety of other rules.

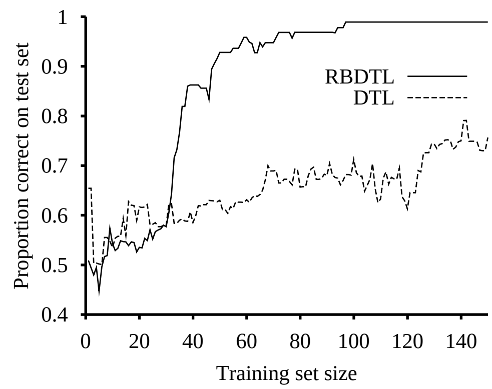
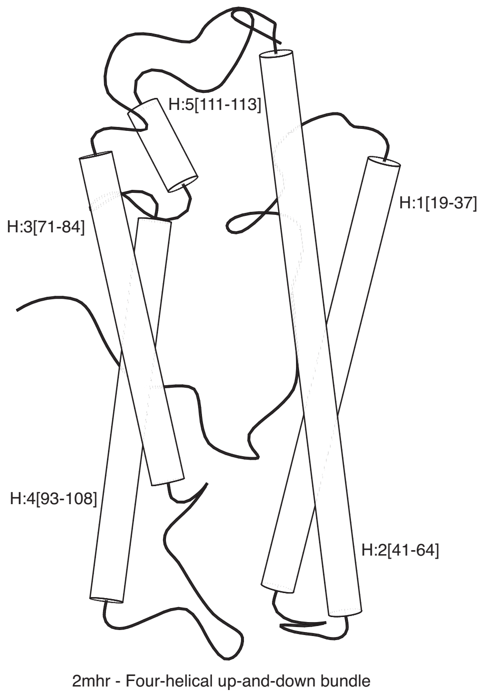
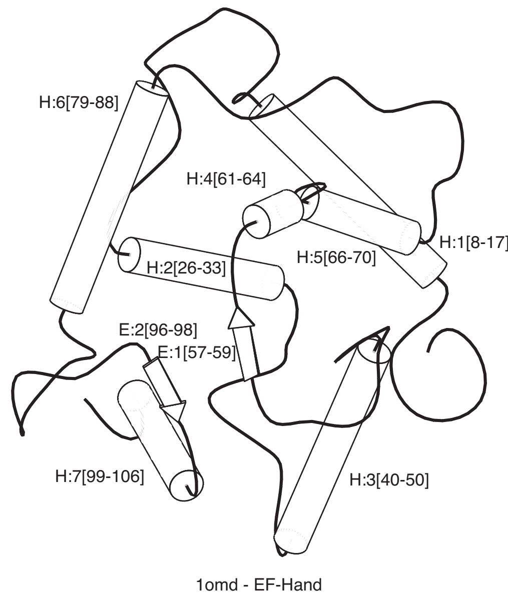


Figure 19.9 FILES: . A performance comparison between DECISION-TREE-LEARNING and RBDTL on randomly generated data for a target function that depends on only 5 of 16 attributes.



(a)



(b)

Figure 19.10 FILES: figures/pdb2mhr.eps (Tue Nov 3 16:23:15 2009) figures/pdb1omd.eps (Tue Nov 3 16:23:15 2009). (a) and (b) show positive and negative examples, respectively, of the "four-helical up-and-down bundle" concept in the domain of protein folding. Each example structure is coded into a logical expression of about 100 conjuncts such as $TotalLength(D2mhr, 118) \wedge NumberHelices(D2mhr, 6) \wedge \ldots$. From these descriptions and from classifications such as $Fold(\text{FOUR-HELICAL-UP-AND-DOWN-BUNDLE}, D2mhr)$, the ILP system PROGOL (?) learned the following rule:

$$\begin{aligned} & Fold(\text{FOUR-HELICAL-UP-AND-DOWN-BUNDLE}, p) \Leftarrow \\ & \quad Helix(p, h_1) \wedge Length(h_1, \text{HIGH}) \wedge Position(p, h_1, n) \\ & \quad \wedge (1 \leq n \leq 3) \wedge Adjacent(p, h_1, h_2) \wedge Helix(p, h_2) \, . \end{aligned}$$

This kind of rule could not be learned, or even represented, by an attribute-based mechanism such as we saw in previous chapters. The rule can be translated into English as " Protein p has fold class "Four-helical up-and-down-bundle" if it contains a long helix h_1 at a secondary structure position between 1 and 3 and h_1 is next to a second helix."

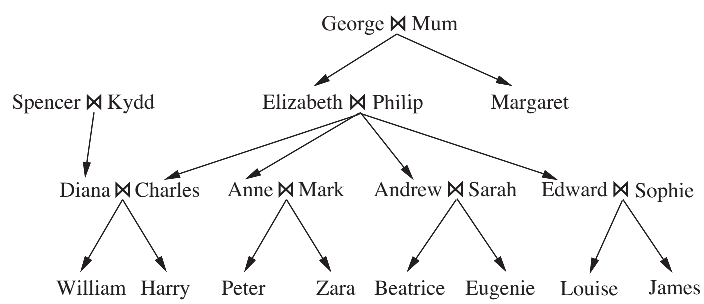


Figure 19.11 FILES: figures/family1.eps (Tue Nov 3 16:22:46 2009). A typical family tree.

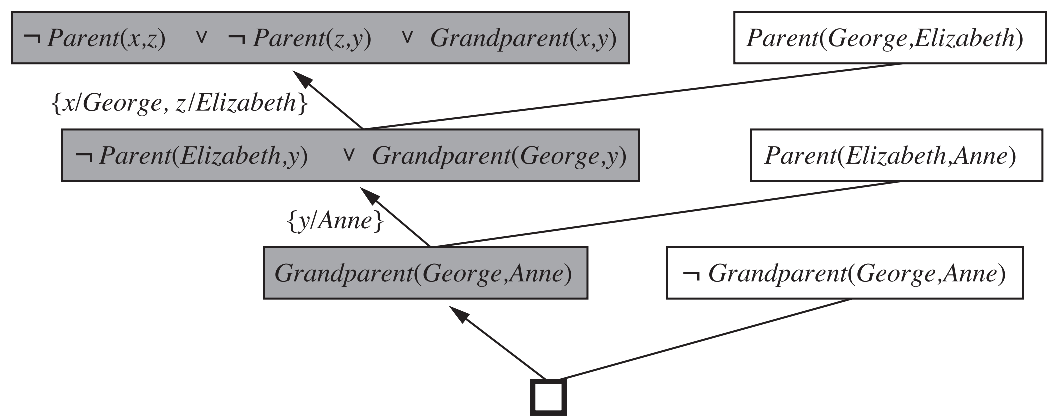


Figure 19.13 FILES: figures/inverse-proof.eps (Tue Nov 3 16:23:05 2009). Early steps in an inverse resolution process. The shaded clauses are generated by inverse resolution steps from the clause to the right and the clause below. The unshaded clauses are from the *Descriptions* and *Classifications* (including negated *Classifications*).

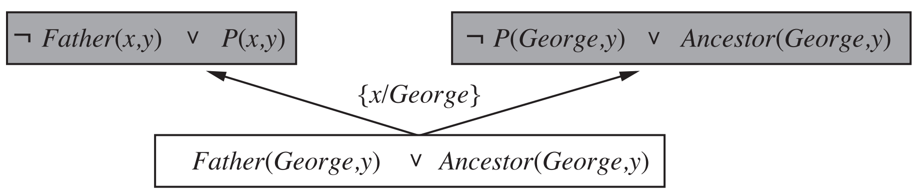


Figure 19.14 FILES: figures/new-predicate.eps (Tue Nov 3 16:23:14 2009). An inverse resolution step that generates a new predicate P.

20 LEARNING PROBABILISTIC MODELS

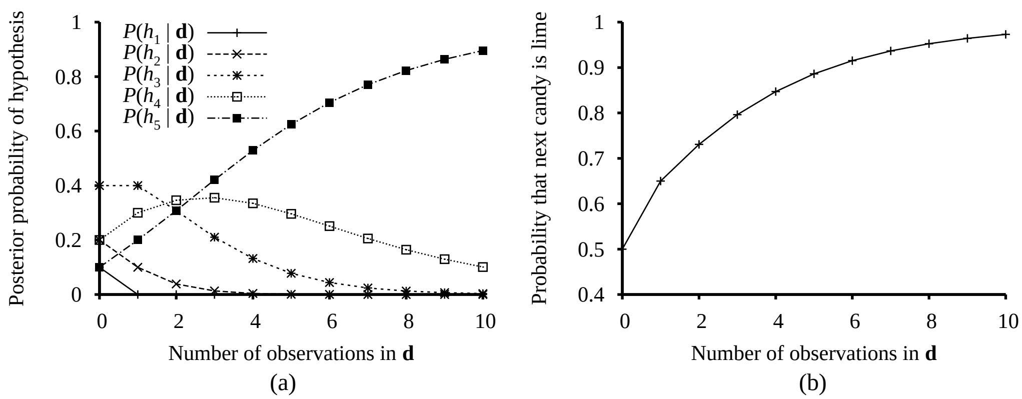


Figure 20.1 **FILES: .** (a) Posterior probabilities $P(h_i \mid d_1, \ldots, d_N)$ from Equation (**??**). The number of observations N ranges from 1 to 10, and each observation is of a lime candy. (b) Bayesian prediction $P(d_{N+1} = lime \mid d_1, \ldots, d_N)$ from Equation (**??**).

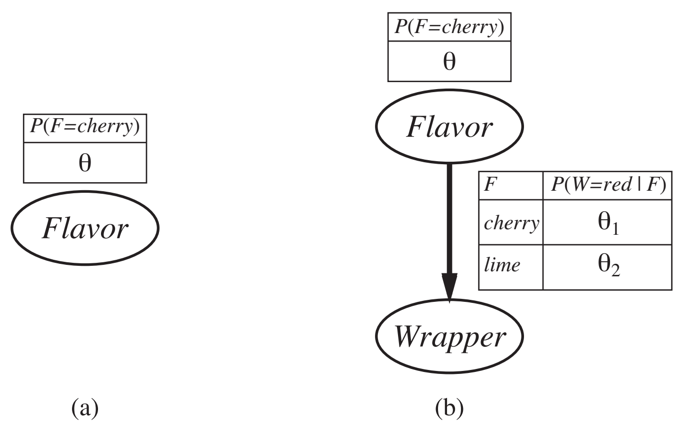


Figure 20.2 FILES: figures/ml-networks.eps (Tue Nov 3 16:23:11 2009). (a) Bayesian network model for the case of candies with an unknown proportion of cherries and limes. (b) Model for the case where the wrapper color depends (probabilistically) on the candy flavor.

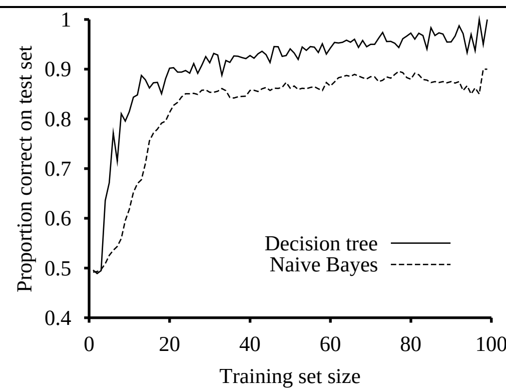


Figure 20.3 **FILES: .** The learning curve for naive Bayes learning applied to the restaurant problem from Chapter 18; the learning curve for decision-tree learning is shown for comparison.

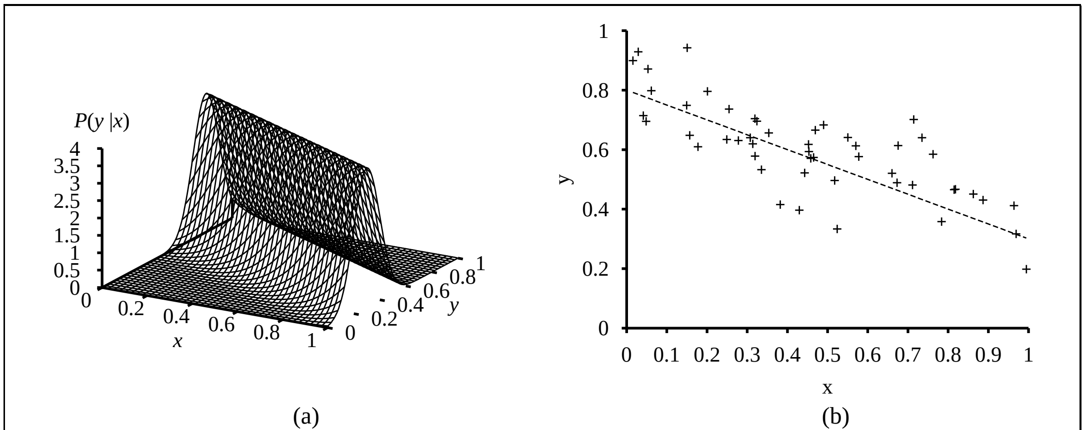


Figure 20.4 **FILES: .** (a) A linear Gaussian model described as $y = \theta_1 x + \theta_2$ plus Gaussian noise with fixed variance. (b) A set of 50 data points generated from this model.

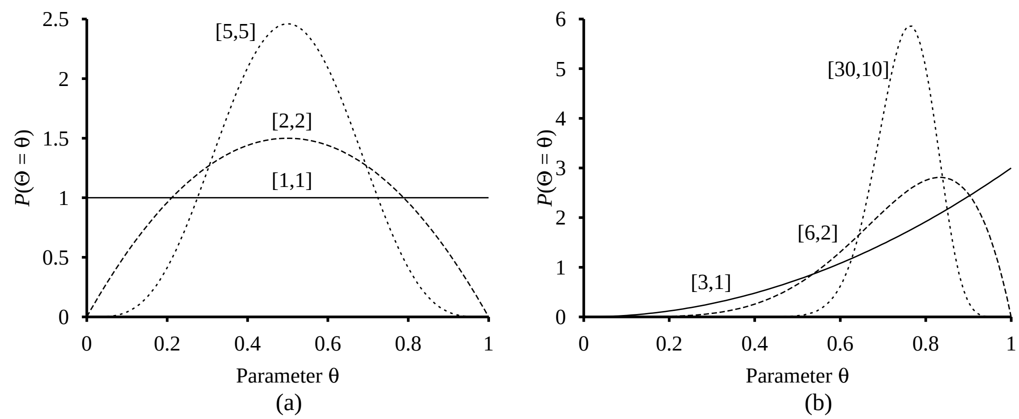


Figure 20.5 **FILES: .** Examples of the $\mathrm{beta}[a, b]$ distribution for different values of $[a, b]$.

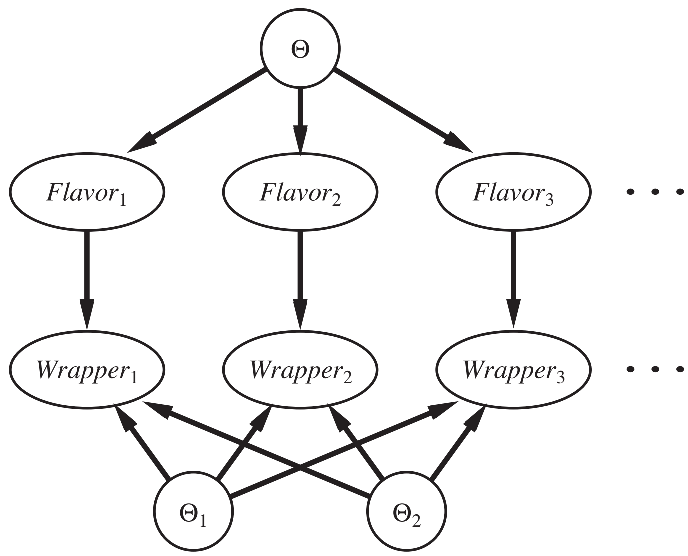


Figure 20.6 FILES: figures/bayesian-learning-network.eps (Tue Nov 3 16:22:26 2009). A Bayesian network that corresponds to a Bayesian learning process. Posterior distributions for the parameter variables Θ, Θ_1, and Θ_2 can be inferred from their prior distributions and the evidence in the $Flavor_i$ and $Wrapper_i$ variables.

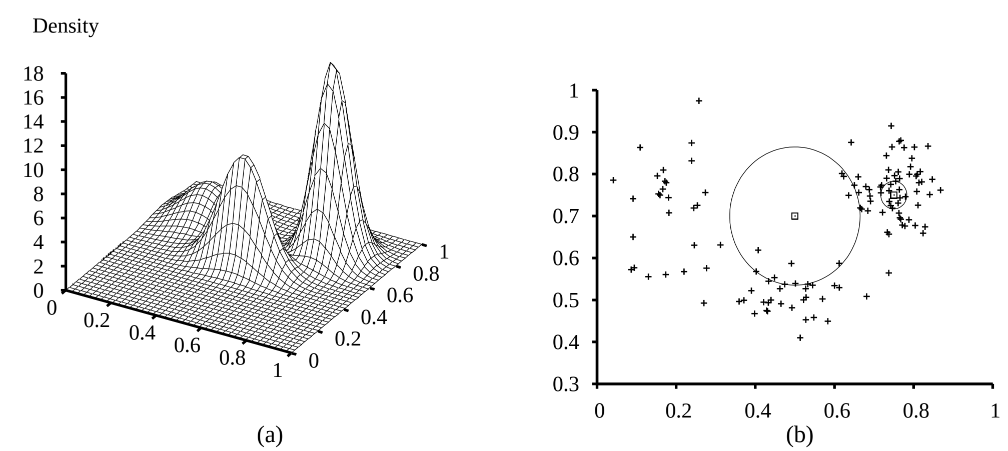


Figure 20.7 **FILES: .** (a) A 3D plot of the mixture of Gaussians from Figure 20.11(a). (b) A 128-point sample of points from the mixture, together with two query points (small squares) and their 10-nearest-neighborhoods (medium and large circles).

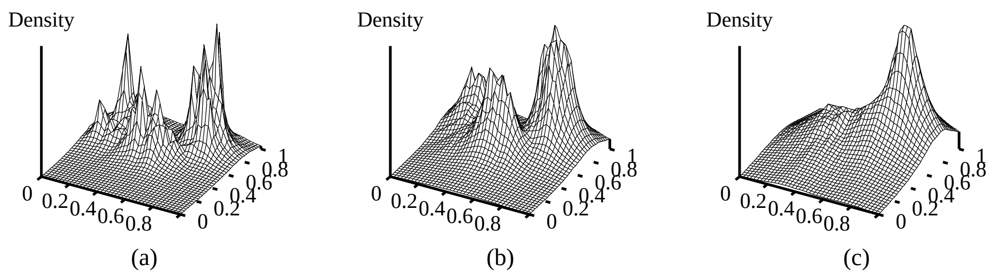


Figure 20.8 **FILES: .** Density estimation using k-nearest-neighbors, applied to the data in Figure 20.7(b), for $k = 3$, 10, and 40 respectively. $k = 3$ is too spiky, 40 is too smooth, and 10 is just about right. The best value for k can be chosen by cross-validation.

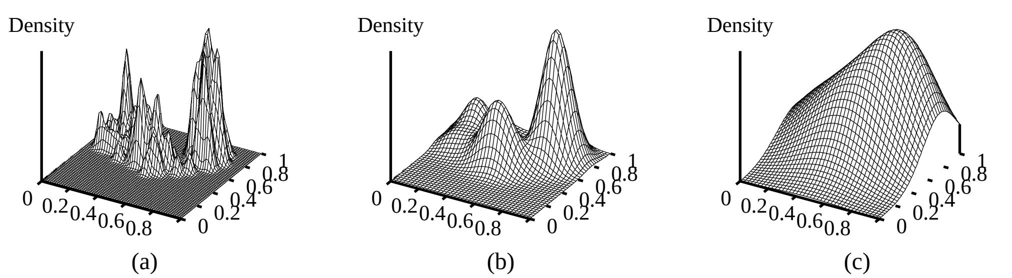


Figure 20.9 FILES: . Kernel density estimation for the data in Figure 20.7(b), using Gaussian kernels with $w = 0.02$, 0.07, and 0.20 respectively. $w = 0.07$ is about right.

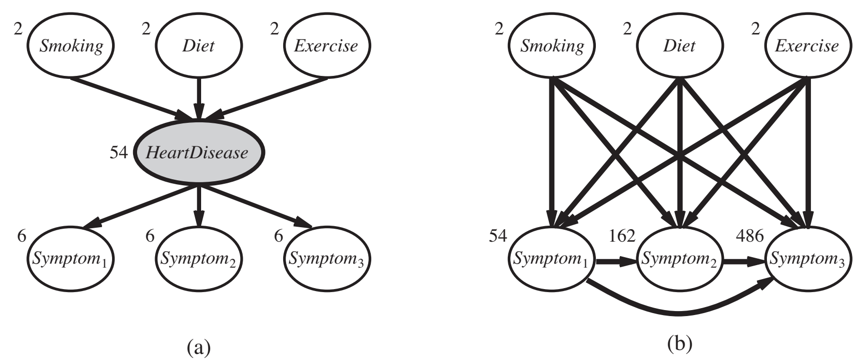


Figure 20.10 FILES: figures/313-heart-disease.eps (Tue Nov 3 16:22:09 2009). (a) A simple diagnostic network for heart disease, which is assumed to be a hidden variable. Each variable has three possible values and is labeled with the number of independent parameters in its conditional distribution; the total number is 78. (b) The equivalent network with $HeartDisease$ removed. Note that the symptom variables are no longer conditionally independent given their parents. This network requires 708 parameters.

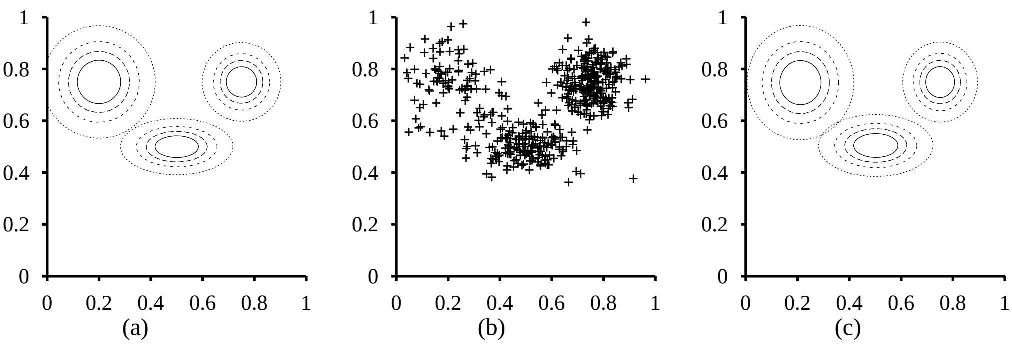


Figure 20.11 **FILES: .** (a) A Gaussian mixture model with three components; the weights (left-to-right) are 0.2, 0.3, and 0.5. (b) 500 data points sampled from the model in (a). (c) The model reconstructed by EM from the data in (b).

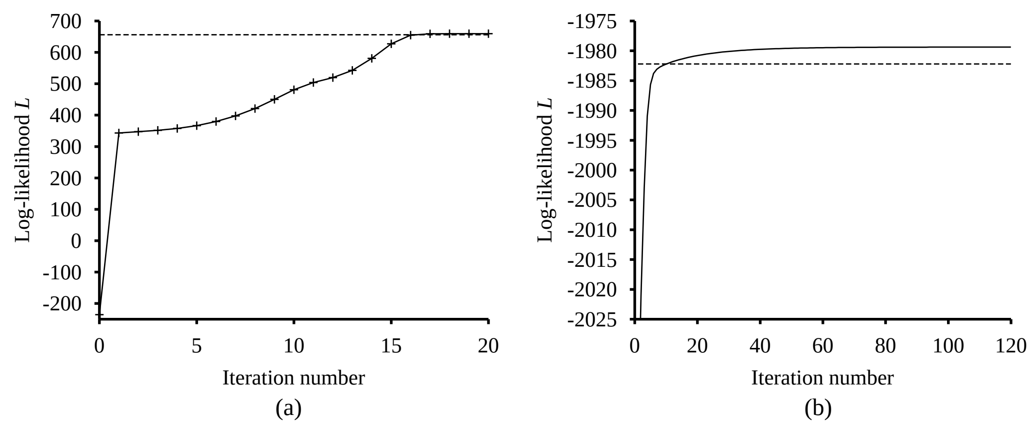


Figure 20.12 **FILES: .** Graphs showing the log likelihood of the data, L, as a function of the EM iteration. The horizontal line shows the log likelihood according to the true model. (a) Graph for the Gaussian mixture model in Figure 20.11. (b) Graph for the Bayesian network in Figure 20.13(a).

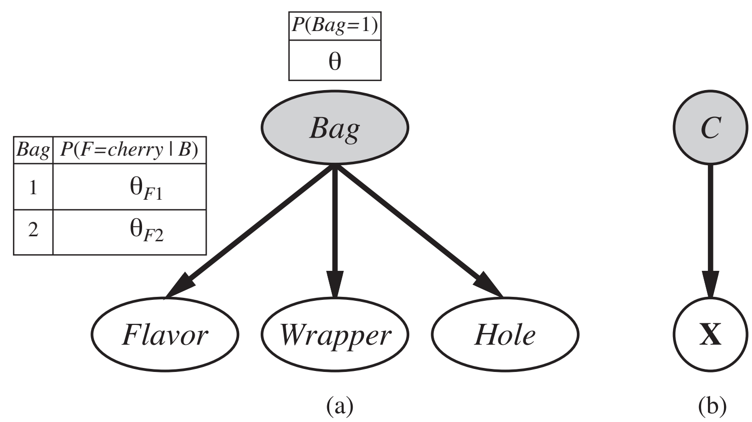


Figure 20.13 FILES: figures/mixture-networks.eps (Tue Nov 3 16:23:11 2009). (a) A mixture model for candy. The proportions of different flavors, wrappers, presence of holes depend on the bag, which is not observed. (b) Bayesian network for a Gaussian mixture. The mean and covariance of the observable variables $\mathbf{X}$ depend on the component C.

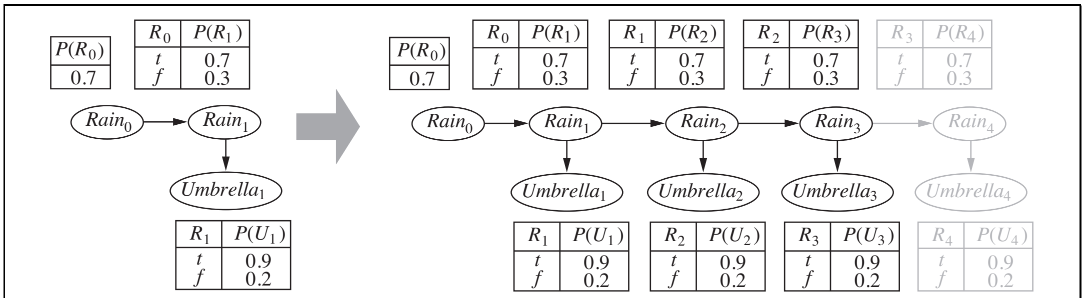


Figure 20.14 FILES: figures/dbn-unrolling.eps (Tue Nov 3 16:22:36 2009). An unrolled dynamic Bayesian network that represents a hidden Markov model (repeat of Figure 15.16).

21 REINFORCEMENT LEARNING

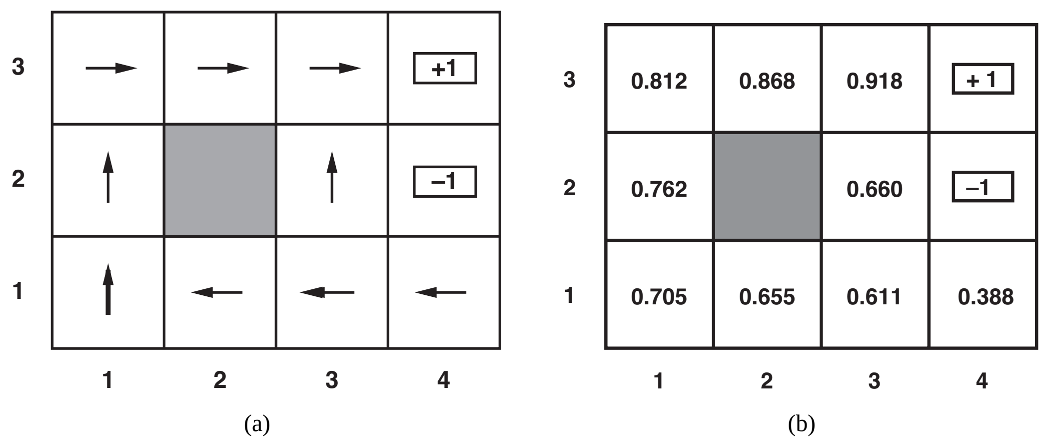


Figure 21.1 FILES: figures/4x3-optimal-policy.eps (Tue Nov 3 16:22:11 2009) figures/sequential-decision-values.eps (Tue Nov 3 16:23:42 2009). (a) A policy π for the 4×3 world; this policy happens to be optimal with rewards of $R(s) = -0.04$ in the nonterminal states and no discounting. (b) The utilities of the states in the 4×3 world, given policy π.

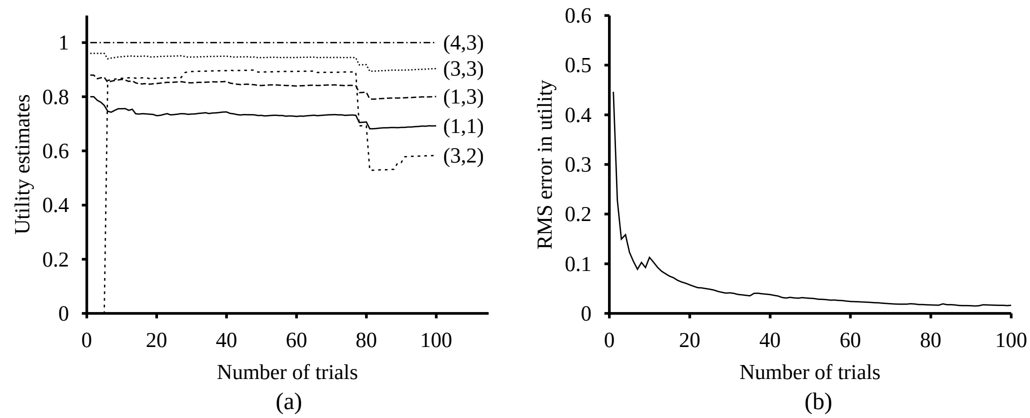


Figure 21.3 **FILES: .** The passive ADP learning curves for the 4×3 world, given the optimal policy shown in Figure 21.1. (a) The utility estimates for a selected subset of states, as a function of the number of trials. Notice the large changes occurring around the 78th trial—this is the first time that the agent falls into the -1 terminal state at (4,2). (b) The root-mean-square error (see Appendix A) in the estimate for $U(1,1)$, averaged over 20 runs of 100 trials each.

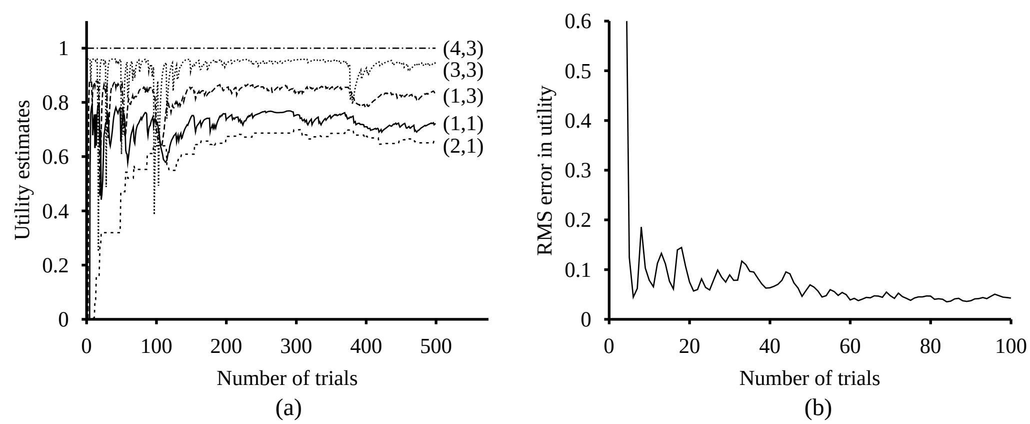


Figure 21.5 **FILES: .** The TD learning curves for the 4×3 world. (a) The utility estimates for a selected subset of states, as a function of the number of trials. (b) The root-mean-square error in the estimate for $U(1,1)$, averaged over 20 runs of 500 trials each. Only the first 100 trials are shown to enable comparison with Figure 21.3.

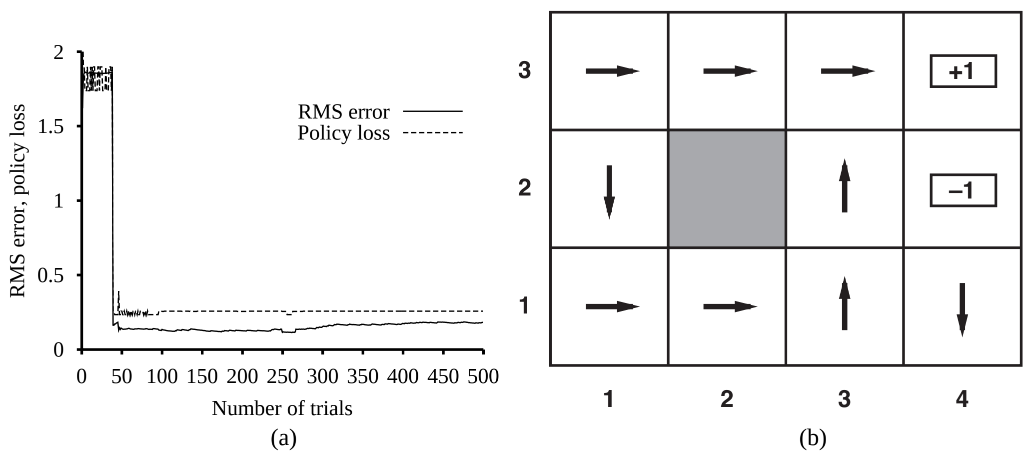


Figure 21.6 FILES: figures/4x3-greedy-adp-policy.eps (Tue Nov 3 16:22:10 2009). Performance of a greedy ADP agent that executes the action recommended by the optimal policy for the learned model. (a) RMS error in the utility estimates averaged over the nine nonterminal squares. (b) The suboptimal policy to which the greedy agent converges in this particular sequence of trials.

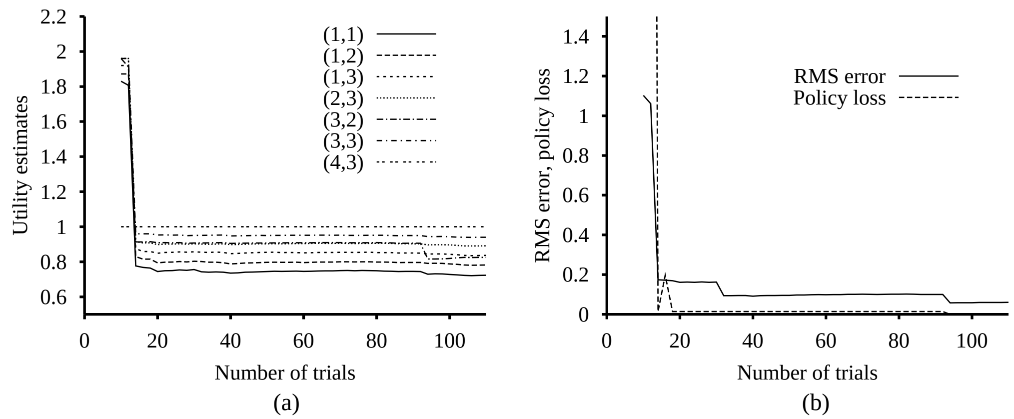


Figure 21.7 **FILES: .** Performance of the exploratory ADP agent. using $R^+ = 2$ and $N_e = 5$. (a) Utility estimates for selected states over time. (b) The RMS error in utility values and the associated policy loss.

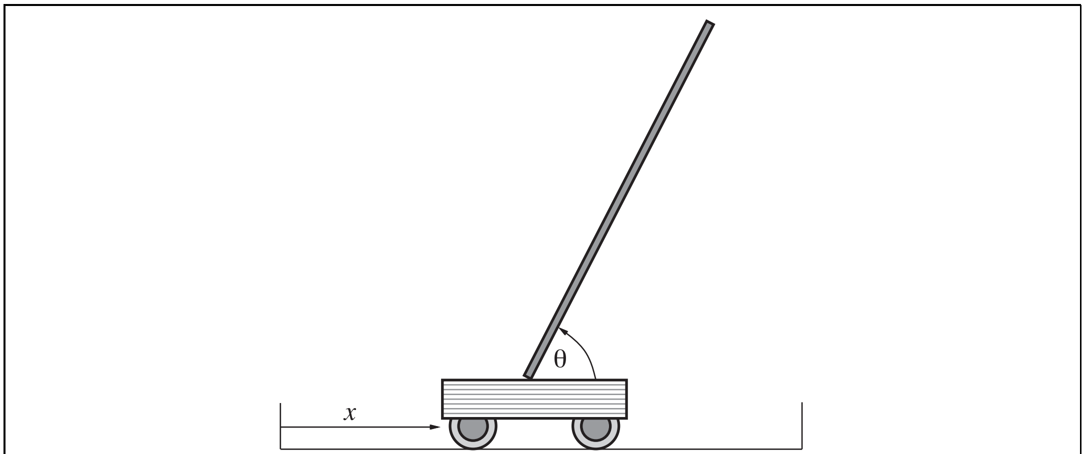


Figure 21.9 FILES: figures/cart-pole.eps (Tue Nov 3 16:22:32 2009). Setup for the problem of balancing a long pole on top of a moving cart. The cart can be jerked left or right by a controller that observes x, θ, $\dot{x}$, and $\dot{\theta}$.

Figure 21.10 FILES: figures/heliComposite.eps (Tue Nov 3 16:23:02 2009). Superimposed time-lapse images of an autonomous helicopter performing a very difficult "nose-in circle" maneuver. The helicopter is under the control of a policy developed by the PEGASUS policy-search algorithm. A simulator model was developed by observing the effects of various control manipulations on the real helicopter; then the algorithm was run on the simulator model overnight. A variety of controllers were developed for different maneuvers. In all cases, performance far exceeded that of an expert human pilot using remote control. (Image courtesy of Andrew Ng.)

22 NATURAL LANGUAGE PROCESSING

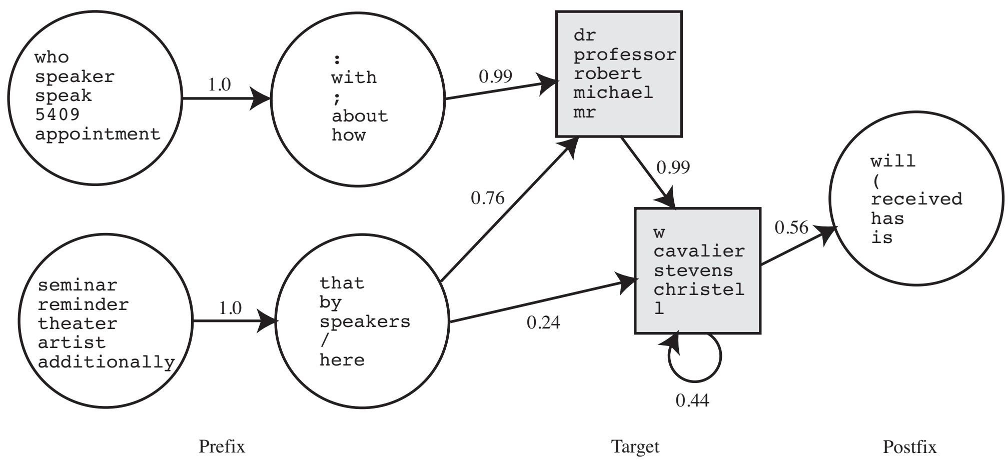


Figure 22.2 FILES: figures/freitag.eps (Tue Nov 3 16:22:53 2009). Hidden Markov model for the **speaker** of a talk announcement. The two square states are the target (note the second target state has a self-loop, so the target can match a string of any length), the four circles to the left are the prefix, and the one on the right is the postfix. For each state, only a few of the high-probability words are shown. From ? (?).

23 NATURAL LANGUAGE FOR COMMUNICATION

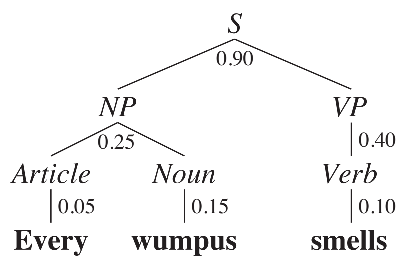


Figure 23.3 FILES: figures/parse-pcfg.eps (Tue Nov 3 16:23:15 2009). Parse tree for the sentence "Every wumpus smells" according to the grammar $\mathcal{E}_0$. Each interior node of the tree is labeled with its probability. The probability of the tree as a whole is $0.9 \times 0.25 \times 0.05 \times 0.15 \times 0.40 \times 0.10 = 0.0000675$. Since this tree is the only parse of the sentence, that number is also the probability of the sentence. The tree can also be written in linear form as $[S\ [NP\ [Article$ **every**$]\ [Noun$ **wumpus**$]][VP\ [Verb$ **smells**$]]]$.

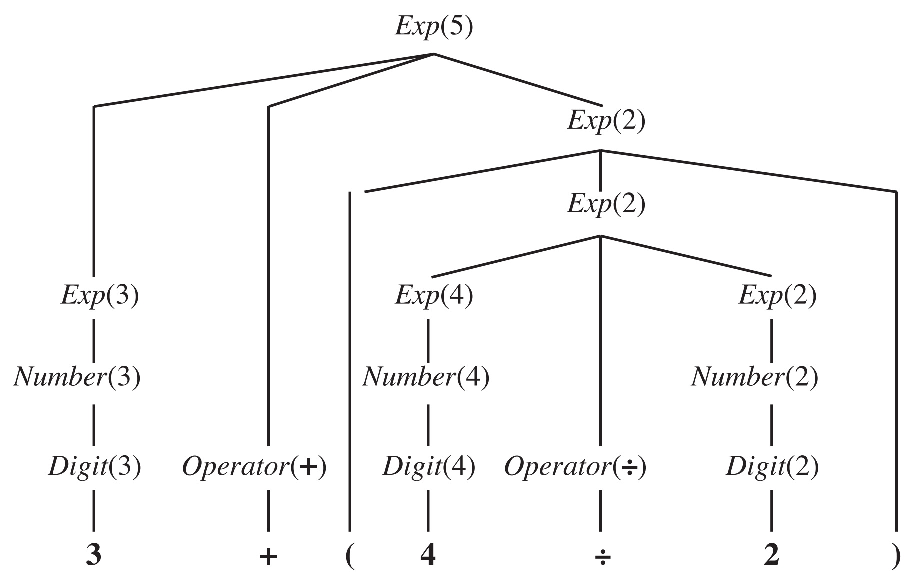


Figure 23.9 FILES: figures/parse2.eps (Tue Nov 3 16:23:15 2009). Parse tree with semantic interpretations for the string “$3 + (4 \div 2)$”.

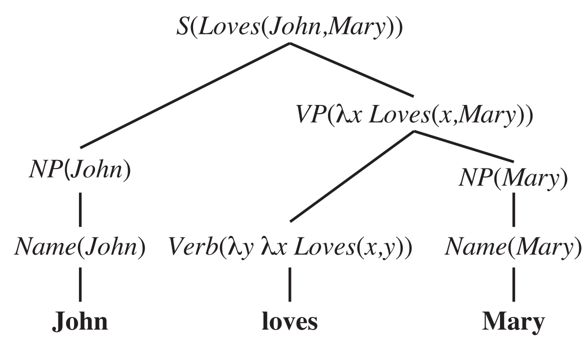


Figure 23.11 FILES: figures/john-mary-semantics.eps (Tue Nov 3 16:23:05 2009). A parse tree with semantic interpretations for the string "John loves Mary".

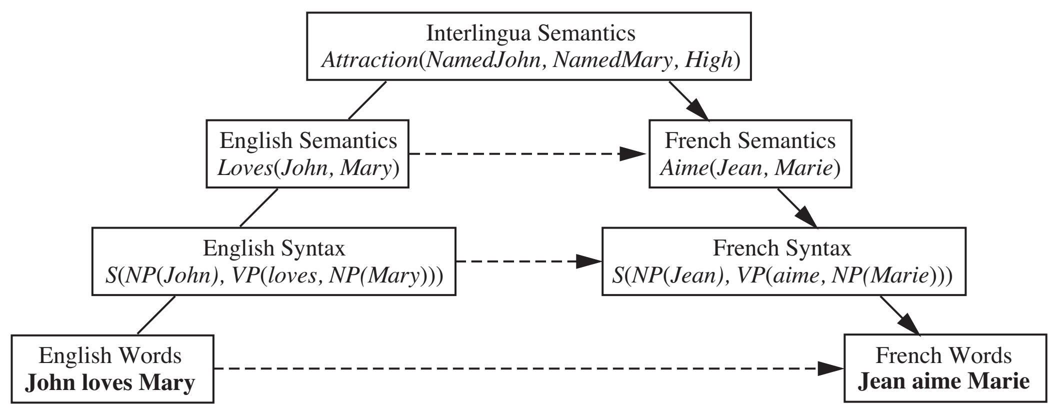


Figure 23.12 **FILES: figures/mt-interlingua.eps (Tue Nov 3 16:23:11 2009).** The Vauquois triangle: schematic diagram of the choices for a machine translation system (?). We start with English text at the top. An interlingua-based system follows the solid lines, parsing English first into a syntactic form, then into a semantic representation and an interlingua representation, and then through generation to a semantic, syntactic, and lexical form in French. A transfer-based system uses the dashed lines as a shortcut. Different systems make the transfer at different points; some make it at multiple points.

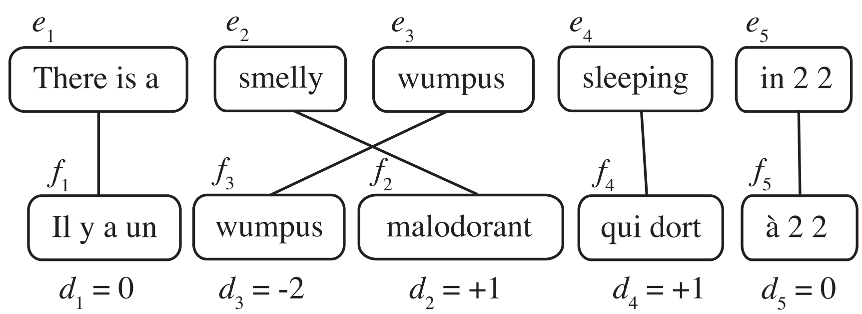


Figure 23.13 FILES: figures/mt-alignment3.eps (Wed Nov 4 11:23:52 2009). Candidate French phrases for each phrase of an English sentence, with distortion (d) values for each French phrase.

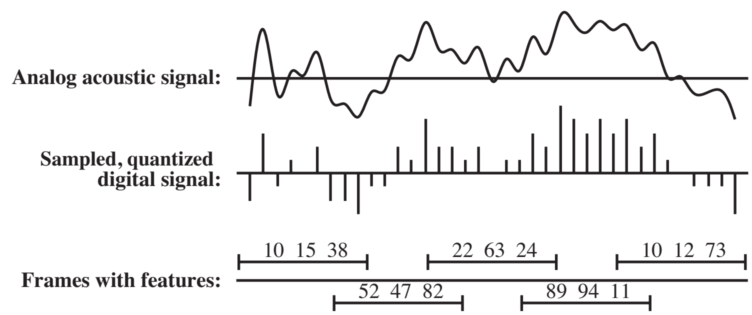
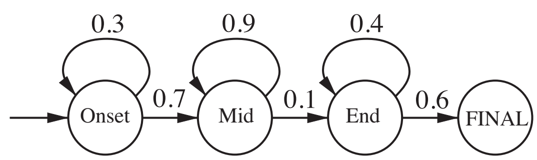


Figure 23.15 FILES: figures/sr-acoustic-frames.eps (Tue Nov 3 16:23:46 2009). Translating the acoustic signal into a sequence of frames. In this diagram each frame is described by the discretized values of three acoustic features; a real system would have dozens of features.

Phone HMM for [m]:



Output probabilities for the phone HMM:

Onset:	Mid:	End:
C_1: 0.5	C_3: 0.2	C_4: 0.1
C_2: 0.2	C_4: 0.7	C_6: 0.5
C_3: 0.3	C_5: 0.1	C_7: 0.4

Figure 23.16 FILES: figures/sr-hmm.eps (Tue Nov 3 16:23:46 2009). An HMM for the three-state phone [m]. Each state has several possible outputs, each with its own probability. The MFCC feature labels C_1 through C_7 are arbitrary, standing for some combination of feature values.

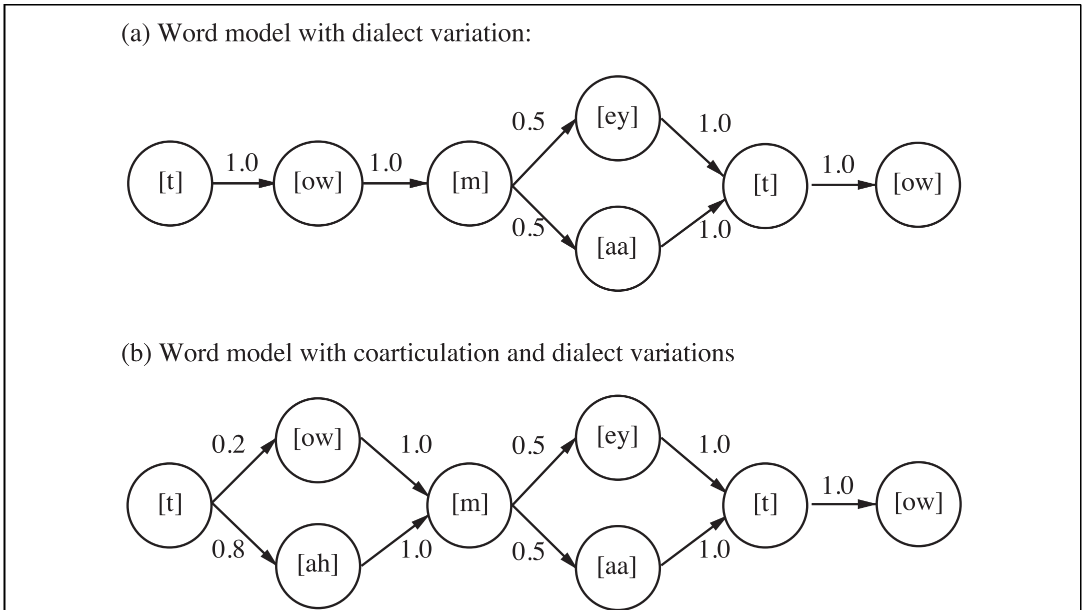


Figure 23.17 FILES: figures/sr-tomato.eps (Tue Nov 3 16:23:46 2009). Two pronunciation models of the word “tomato.” Each model is shown as a transition diagram with states as circles and arrows showing allowed transitions with their associated probabilities. (a) A model allowing for dialect differences. The 0.5 numbers are estimates based on the two authors’ preferred pronunciations. (b) A model with a coarticulation effect on the first vowel, allowing either the [ow] or the [ah] phone.

24 PERCEPTION

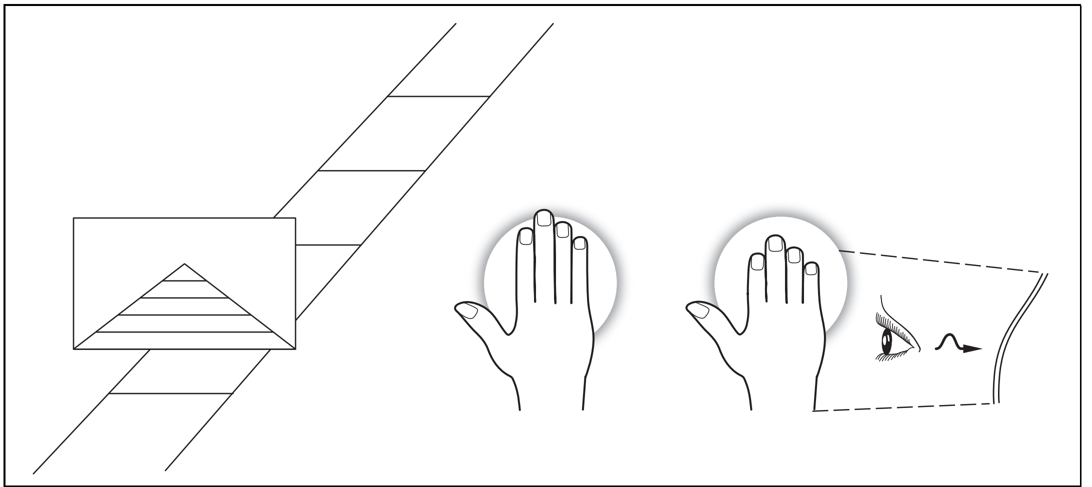


Figure 24.1 FILES: figures/c24f001.eps (Tue Nov 3 16:22:30 2009). Imaging distorts geometry. Parallel lines appear to meet in the distance, as in the image of the railway tracks on the left. In the center, a small hand blocks out most of a large moon. On the right is a foreshortening effect: the hand is tilted away from the eye, making it appear shorter than in the center figure.

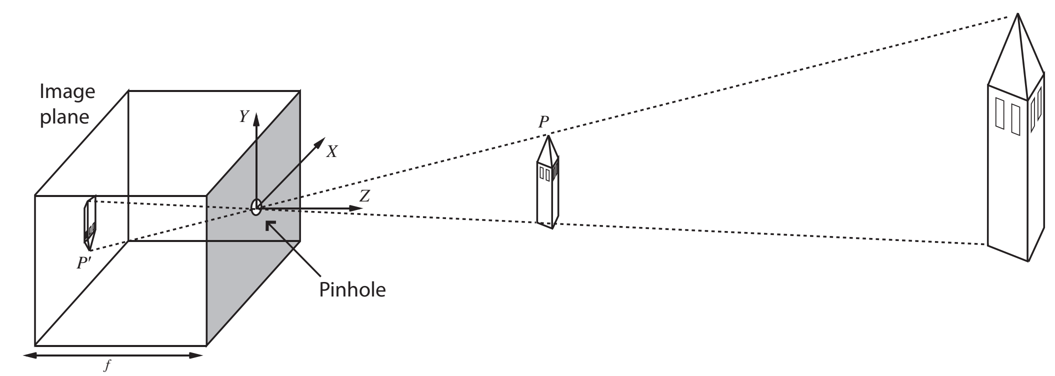


Figure 24.2 **FILES: figures/newpinhole.eps (Tue Nov 3 16:23:14 2009).** Each light-sensitive element in the image plane at the back of a pinhole camera receives light from a the small range of directions that passes through the pinhole. If the pinhole is small enough, the result is a focused image at the back of the pinhole. The process of projection means that large, distant objects look the same as smaller, nearby objects. Note that the image is projected upside down.

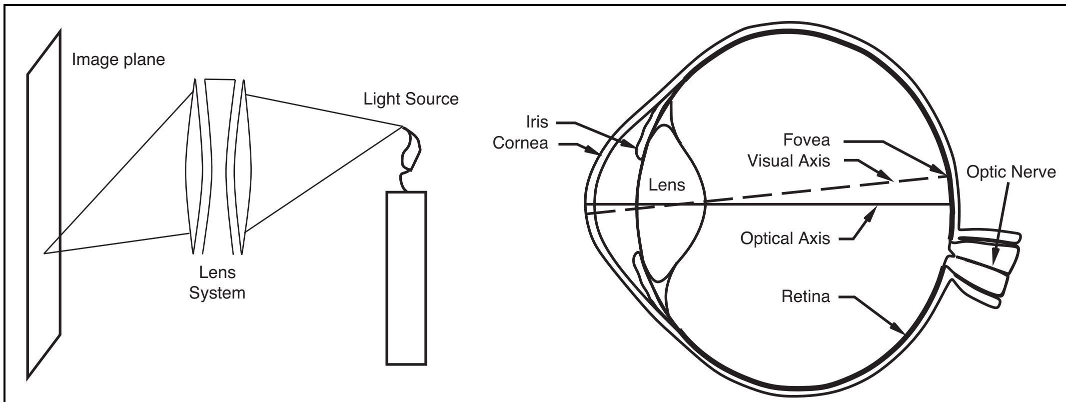


Figure 24.3 FILES: figures/lens-eye.eps (Tue Nov 3 16:23:06 2009). Lenses collect the light leaving a scene point in a range of directions, and steer it all to arrive at a single point on the image plane. Focusing works for points lying close to a focal plane in space; other points will not be focused properly. In cameras, elements of the lens system move to change the focal plane, whereas in the eye, the shape of the lens is changed by specialized muscles.

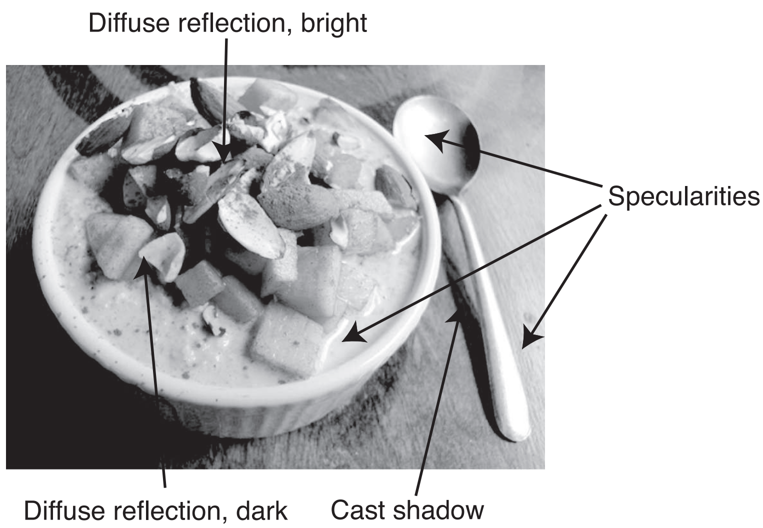


Figure 24.4 FILES: figures/illumination.eps (Tue Nov 3 16:23:04 2009). A variety of illumination effects. There are specularities on the metal spoon and on the milk. The bright diffuse surface is bright because it faces the light direction. The dark diffuse surface is dark because it is tangential to the illumination direction. The shadows appear at surface points that cannot see the light source. Photo by Mike Linksvayer (mlinksva on flickr).

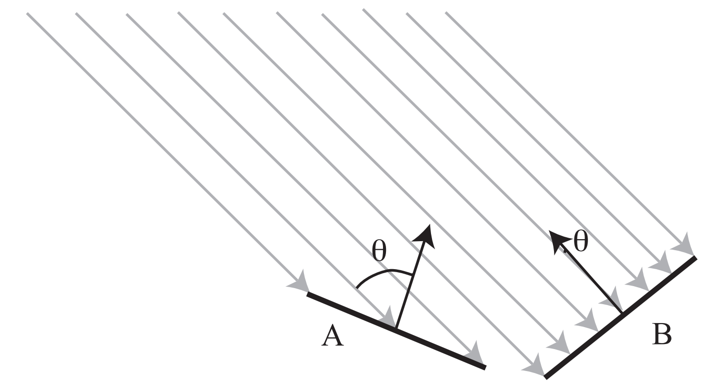


Figure 24.5 FILES: figures/lambert.eps (Tue Nov 3 13:41:38 2009). Two surface patches are illuminated by a distant point source, whose rays are shown as gray arrowheads. Patch A is tilted away from the source (θ is close to 90^0) and collects less energy, because it cuts fewer light rays per unit surface area. Patch B, facing the source (θ is close to 0^0), collects more energy.

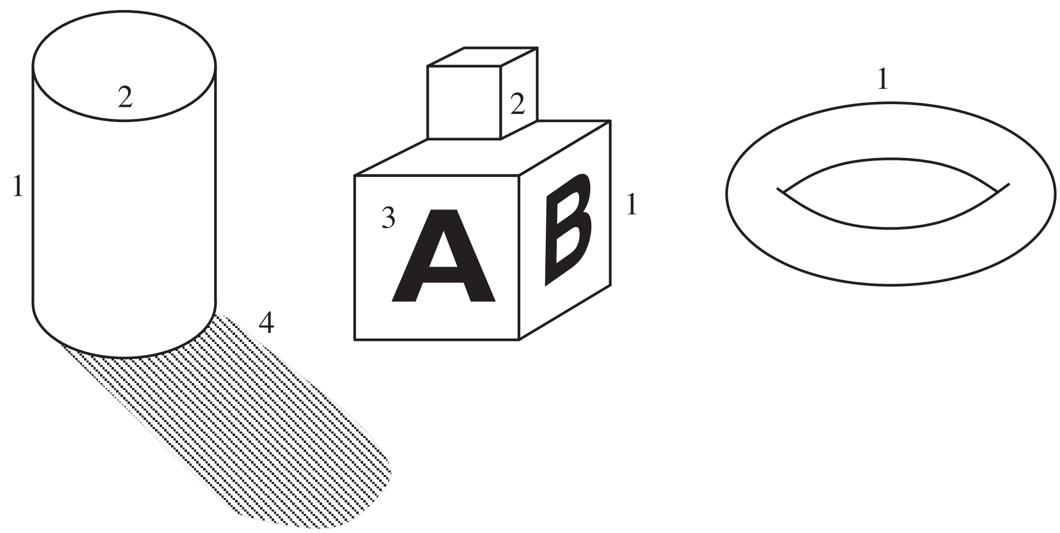
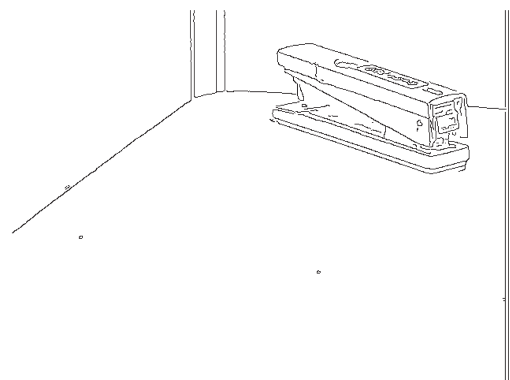


Figure 24.6 FILES: figures/diff-edges.eps (Tue Nov 3 16:22:37 2009). Different kinds of edges: (1) depth discontinuities; (2) surface orientation discontinuities; (3) reflectance discontinuities; (4) illumination discontinuities (shadows).

(a)



(b)

Figure 24.7 FILES: figures/stapler1-test.eps (Tue Nov 3 16:23:47 2009) figures/stapler1.edge-test.eps (Tue Nov 3 16:23:47 2009). (a) Photograph of a stapler. (b) Edges computed from (a).

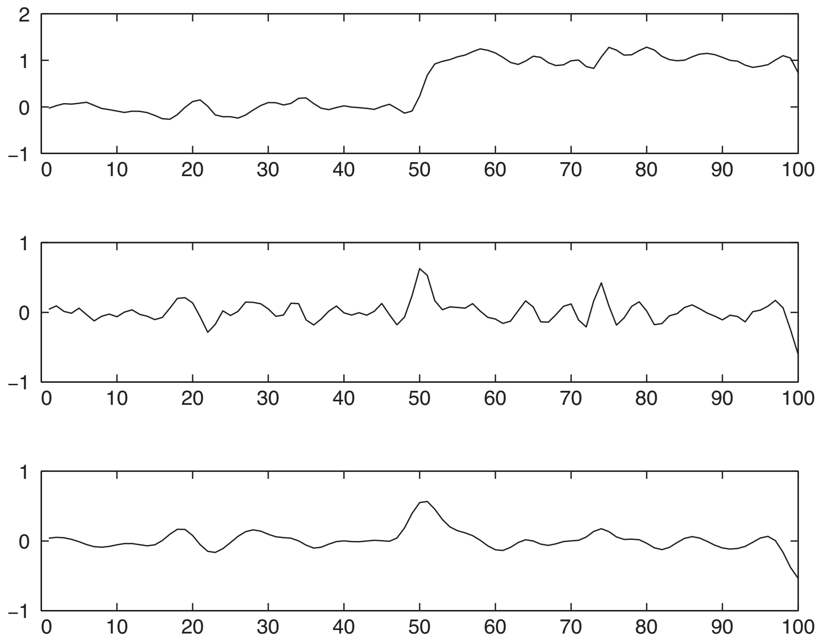
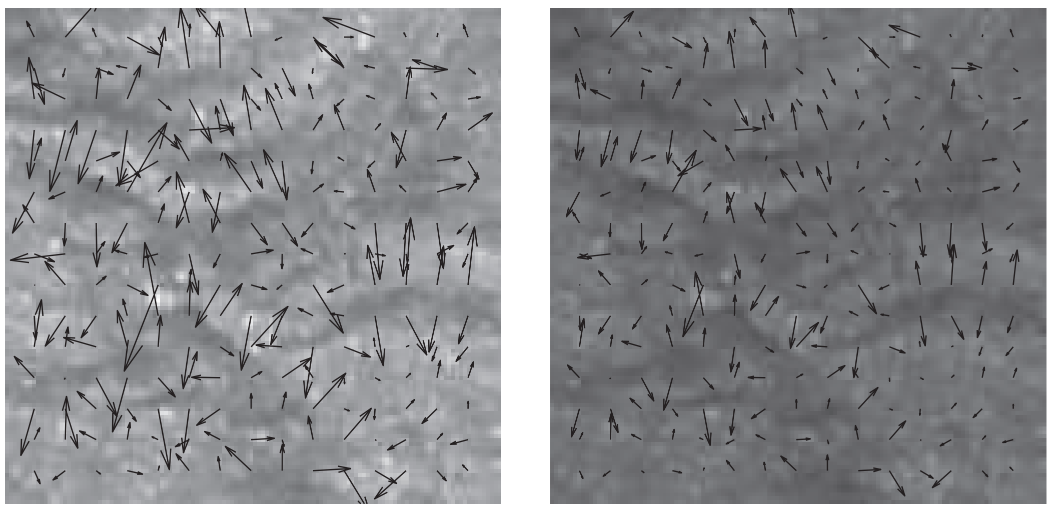


Figure 24.8 **FILES: figures/edgewderiv.eps (Tue Nov 3 16:22:41 2009).** Top: Intensity profile $I(x)$ along a one-dimensional section across an edge at $x=50$. Middle: The derivative of intensity, $I'(x)$. Large values of this function correspond to edges, but the function is noisy. Bottom: The derivative of a smoothed version of the intensity, $(I * G_\sigma)'$, which can be computed in one step as the convolution $I * G'_\sigma$. The noisy candidate edge at $x=75$ has disappeared.



(a) (b)

Figure 24.9 FILES: figures/lightricegrad.eps (Wed Nov 4 15:09:20 2009) figures/darkricegrad.eps (Wed Nov 4 15:09:26 2009). Two images of the same texture of crumpled rice paper, with different illumination levels. The gradient vector field (at every eighth pixel) is plotted on top of each one. Notice that, as the light gets darker, all the gradient vectors get shorter. The vectors do not rotate, so the gradient orientations do not change.

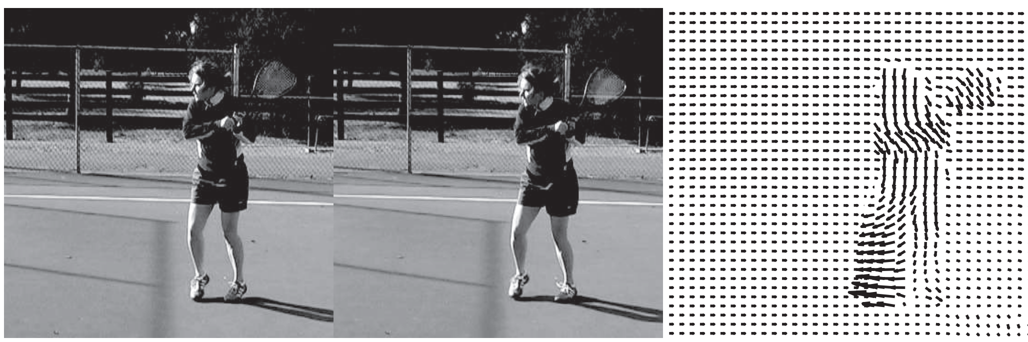
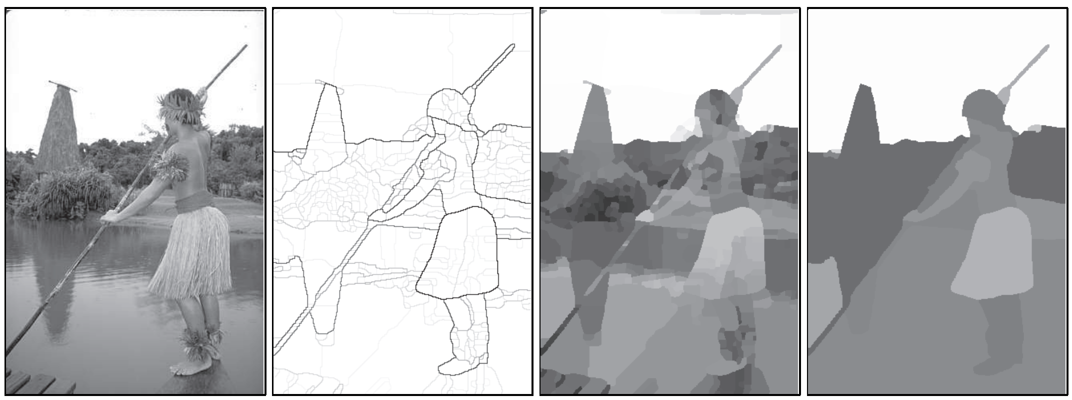


Figure 24.10 FILES: figures/broxrevised.eps (Tue Nov 3 16:22:29 2009) figures/broxIn1.eps (not found) figures/broxIn2.eps (not found) figures/broxFlow.eps (not found). Two frames of a video sequence. On the right is the optical flow field corresponding to the displacement from one frame to the other. Note how the movement of the tennis racket and the front leg is captured by the directions of the arrows. (Courtesy of Thomas Brox.)



(a) (b) (c) (d)

Figure 24.11 FILES: figures/101087.eps (Tue Nov 3 16:22:07 2009) figures/101087-ucm-th0.eps (not found) figures/101087-seg-th0.eps (not found) figures/101087-seg-th0-5.eps (not found). (a) Original image. (b) Boundary contours, where the higher the P_b value, the darker the contour. (c) Segmentation into regions, corresponding to a fine partition of the image. Regions are rendered in their mean colors. (d) Segmentation into regions, corresponding to a coarser partition of the image, resulting in fewer regions. (Courtesy of Pablo Arbelaez, Michael Maire, Charles Fowlkes, and Jitendra Malik)

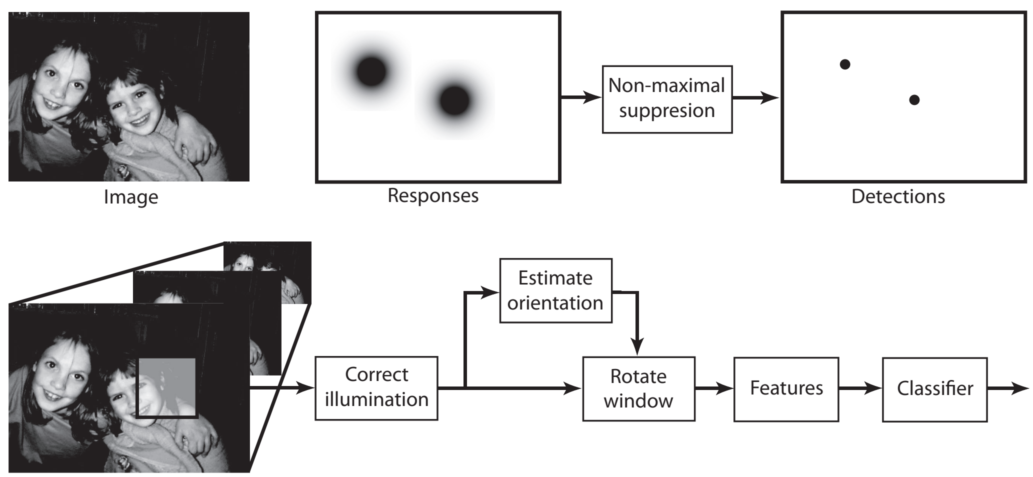


Figure 24.12 FILES: figures/facesys.eps (not found) figures/facesys2.eps (Tue Nov 3 16:22:46 2009). Face finding systems vary, but most follow the architecture illustrated in two parts here. On the top, we go from images to responses, then apply non-maximum suppression to find the strongest local response. The responses are obtained by the process illustrated on the bottom. We sweep a window of fixed size over larger and smaller versions of the image, so as to find smaller or larger faces, respectively. The illumination in the window is corrected, and then a regression engine (quite often, a neural net) predicts the orientation of the face. The window is corrected to this orientation and then presented to a classifier. Classifier outputs are then postprocessed to ensure that only one face is placed at each location in the image.

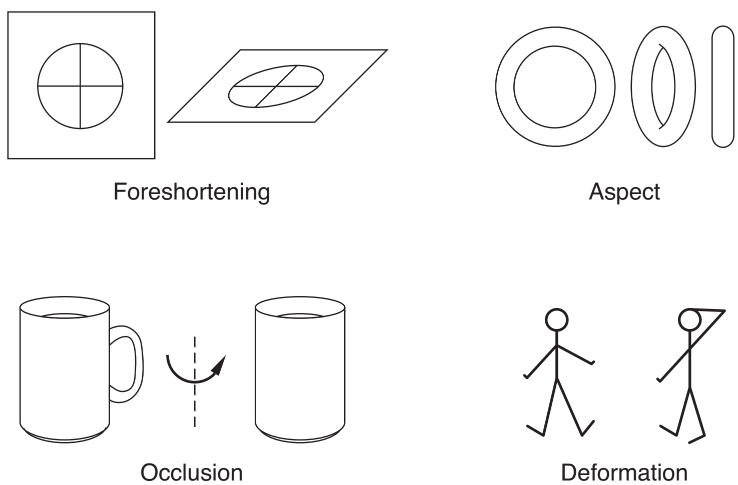


Figure 24.13 FILES: figures/c24f013.eps (Tue Nov 3 16:22:30 2009). Sources of appearance variation. First, elements can foreshorten, like the circular patch on the top left. This patch is viewed at a slant, and so is elliptical in the image. Second, objects viewed from different directions can change shape quite dramatically, a phenomenon known as aspect. On the top right are three different aspects of a doughnut. Occlusion causes the handle of the mug on the bottom left to disappear when the mug is rotated. In this case, because the body and handle belong to the same mug, we have self-occlusion. Finally, on the bottom right, some objects can deform dramatically.

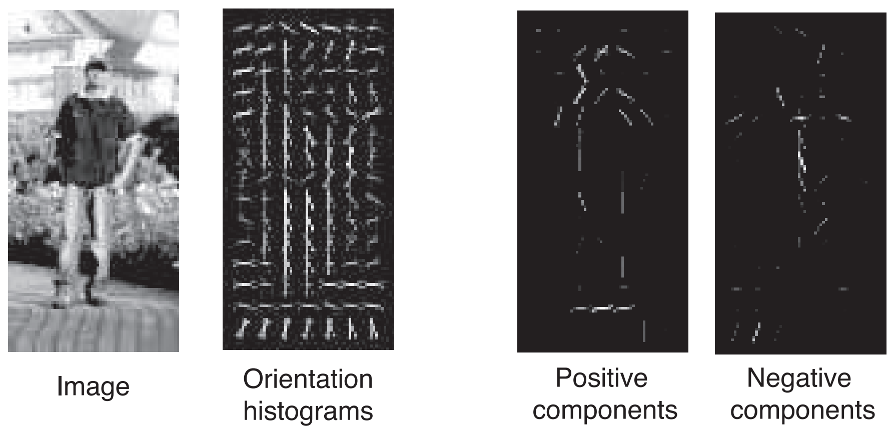


Figure 24.14 FILES: figures/hogfig.eps (Tue Nov 3 16:23:03 2009). Local orientation histograms are a powerful feature for recognizing even quite complex objects. On the left, an image of a pedestrian. On the center left, local orientation histograms for patches. We then apply a classifier such as a support vector machine to find the weights for each histogram that best separate the positive examples of pedestrians from non-pedestrians. We see that the positively weighted components look like the outline of a person. The negative components are less clear; they represent all the patterns that are not pedestrians. Figure from ? (?) © IEEE.

Figure 24.15 FILES: figures/lowefig.eps (Wed Nov 4 14:48:27 2009). Another example of object recognition, this one using the SIFT feature (Scale Invariant Feature Transform), an earlier version of the HOG feature. On the **left**, images of a shoe and a telephone that serve as object models. In the **center**, a test image. On the **right**, the shoe and the telephone have been detected by: finding points in the image whose SIFT feature descriptions match a model; computing an estimate of pose of the model; and verifying that estimate. A strong match is usually verified with rare false positives. Images from ? (?) © IEEE.

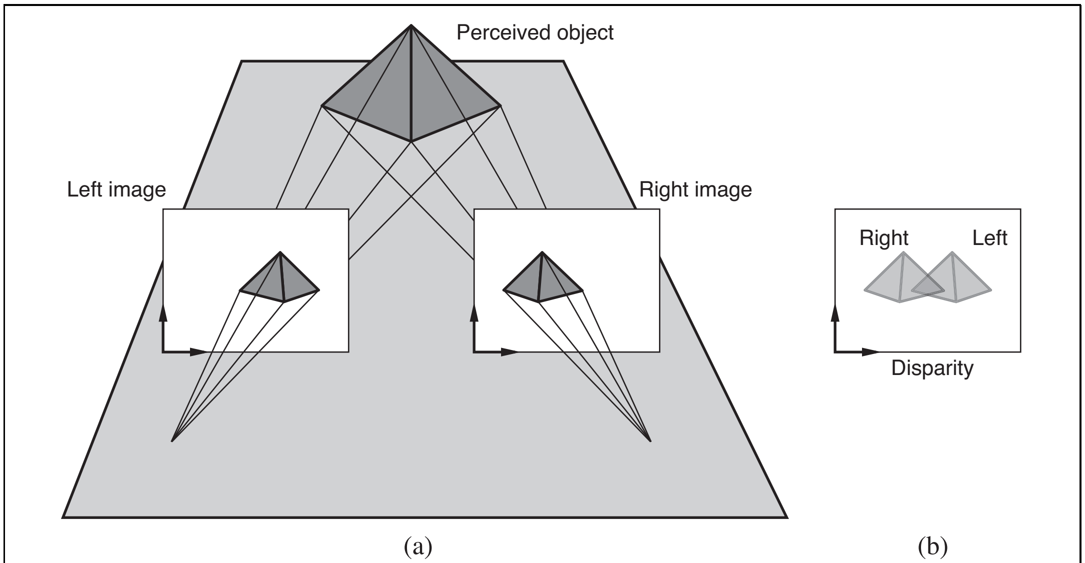


Figure 24.16 FILES: figures/c24f017.eps (Tue Nov 3 16:22:31 2009) figures/stereo-1.eps (not found) figures/stereo-2.eps (not found). Translating a camera parallel to the image plane causes image features to move in the camera plane. The disparity in positions that results is a cue to depth. If we superimpose left and right image, as in (b), we see the disparity.

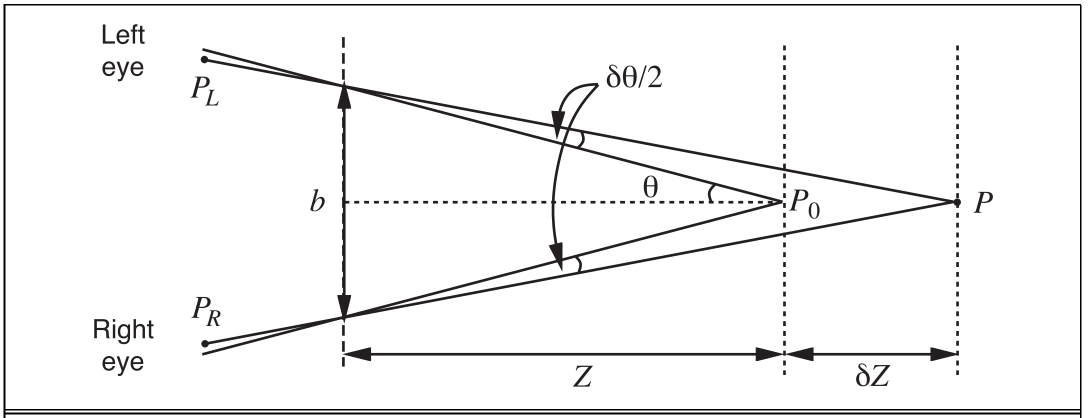


Figure 24.17 FILES: figures/stereopsis.eps (Tue Nov 3 16:23:49 2009). The relation between disparity and depth in stereopsis. The centers of projection of the two eyes are b apart, and the optical axes intersect at the fixation point P_0. The point P in the scene projects to points P_L and P_R in the two eyes. In angular terms, the disparity between these is $\delta\theta$. See text.

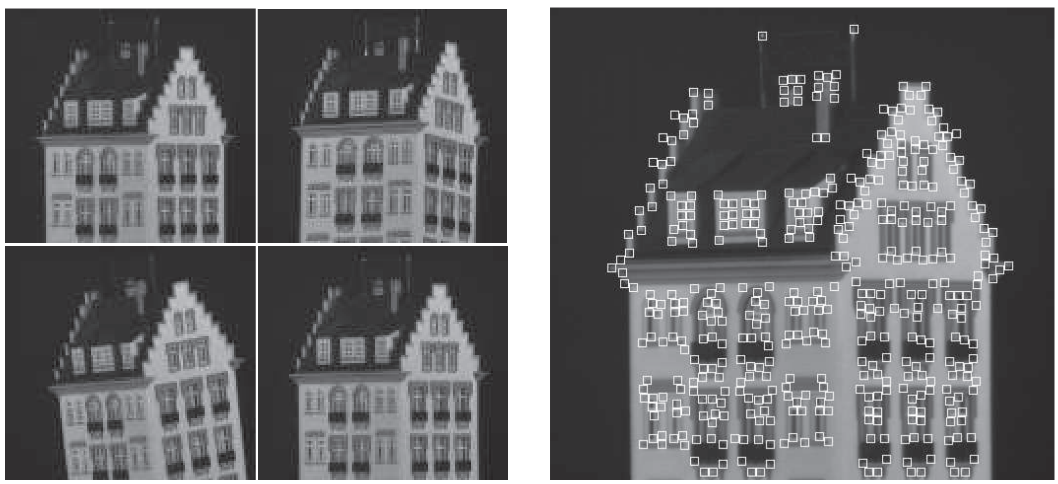
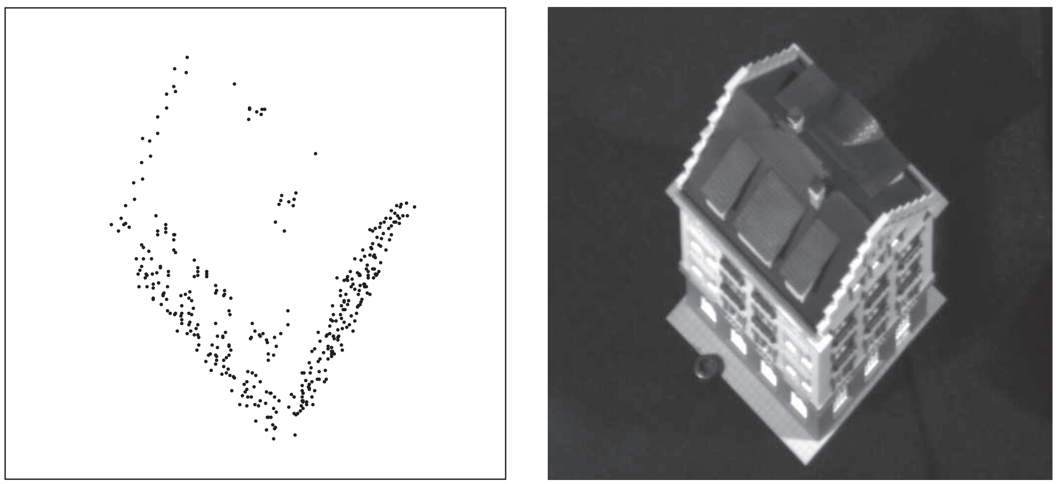
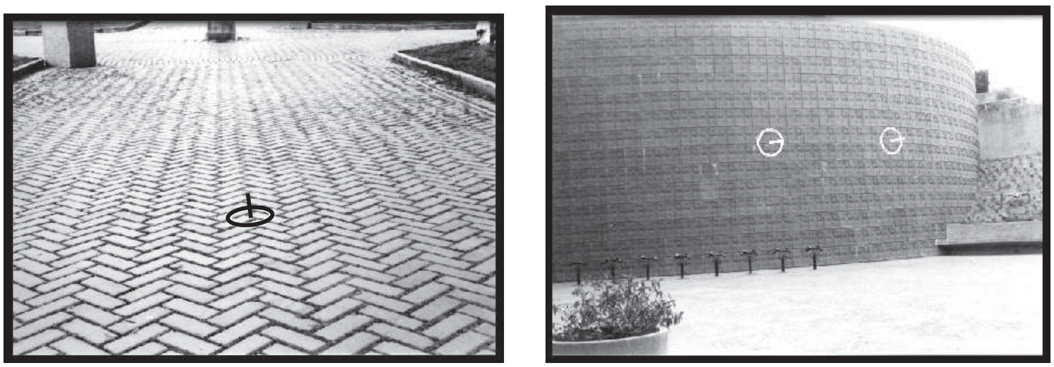


Figure 24.18 FILES: figures/frame1.eps (Tue Nov 3 16:22:52 2009) figures/frame60.eps (Tue Nov 3 16:22:53 2009) figures/frame120.eps (Tue Nov 3 16:22:53 2009) figures/frame150.eps (Tue Nov 3 16:22:53 2009) figures/features.eps (Tue Nov 3 16:22:47 2009). (a) Four frames from a video sequence in which the camera is moved and rotated relative to the object. (b) The first frame of the sequence, annotated with small boxes highlighting the features found by the feature detector. (Courtesy of Carlo Tomasi.)



(a) (b)

Figure 24.19 FILES: figures/topview-dots.eps (Tue Nov 3 16:23:56 2009) figures/topview-real.eps (Tue Nov 3 16:23:57 2009). (a) Three-dimensional reconstruction of the locations of the image features in Figure 24.18, shown from above. (b) The real house, taken from the same position.



(a) (b)

Figure 24.20 FILES: figures/camp-test.eps (Tue Nov 3 16:22:32 2009) figures/chem-test.eps (Tue Nov 3 16:22:32 2009). (a) A textured scene. Assuming that the real texture is uniform allows recovery of the surface orientation. The computed surface orientation is indicated by overlaying a black circle and pointer, transformed as if the circle were painted on the surface at that point. (b) Recovery of shape from texture for a curved surface (white circle and pointer this time). Images courtesy of Jitendra Malik and Ruth Rosenholtz (?).

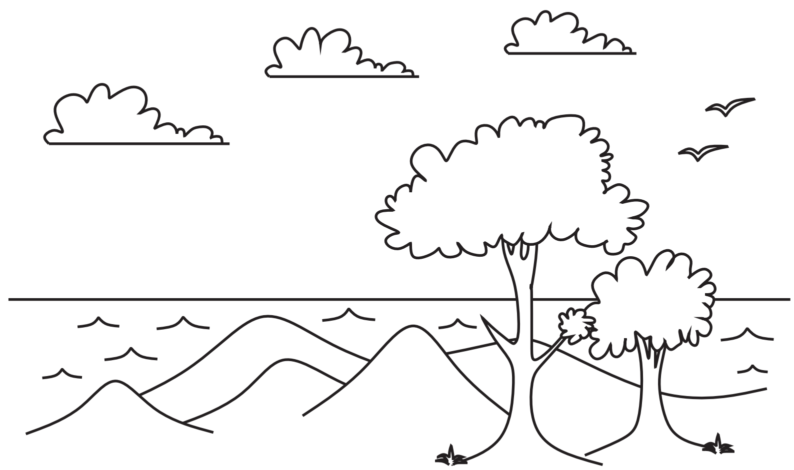


Figure 24.21 FILES: figures/isha.eps (Tue Nov 3 16:23:05 2009). An evocative line drawing. (Courtesy of Isha Malik.)

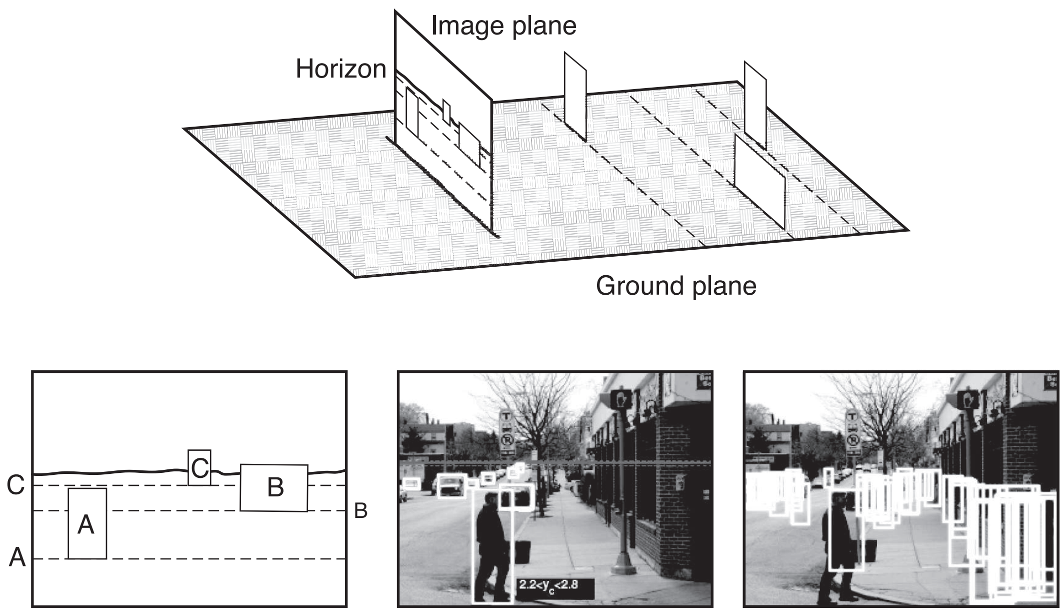


Figure 24.22 FILES: figures/c24f022-a.eps (Tue Nov 3 16:22:31 2009). In an image of people standing on a ground plane, the people whose feet are closer to the horizon in the image must be farther away (top drawing). This means they must look smaller in the image (left lower drawing). This means that the size and location of real pedestrians in an image depend upon one another and on the location of the horizon. To exploit this, we need to identify the ground plane, which is done using shape-from-texture methods. From this information, and from some likely pedestrians, we can recover a horizon as shown in the center image. On the right, acceptable pedestrian boxes given this geometric context. Notice that pedestrians who are higher in the scene must be smaller. If they are not, then they are false positives. Images from ? (?) © IEEE.

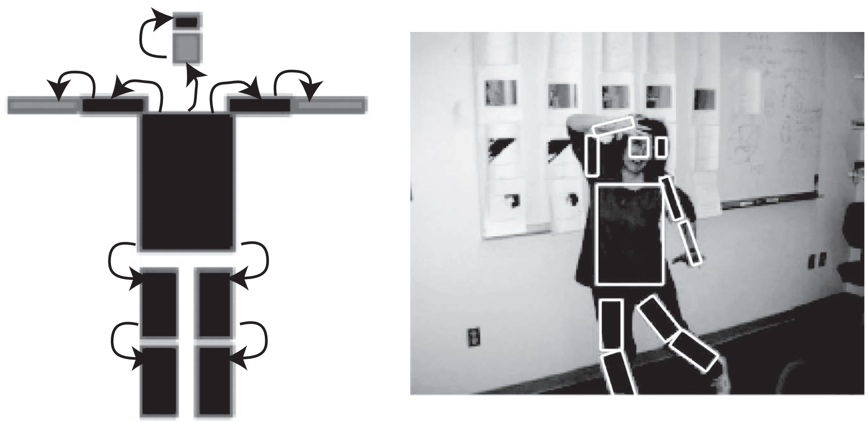


Figure 24.23 FILES: figures/armslegs.eps (Tue Nov 3 16:22:24 2009). A pictorial structure model evaluates a match between a set of image rectangles and a cardboard person (shown on the left) by scoring the similarity in appearance between body segments and image segments and the spatial relations between the image segments. Generally, a match is better if the image segments have about the right appearance and are in about the right place with respect to one another. The appearance model uses average colors for hair, head, torso, and upper and lower arms and legs. The relevant relations are shown as arrows. On the right, the best match for a particular image, obtained using dynamic programming. The match is a fair estimate of the configuration of the body. Figure from ? (?) © IEEE.

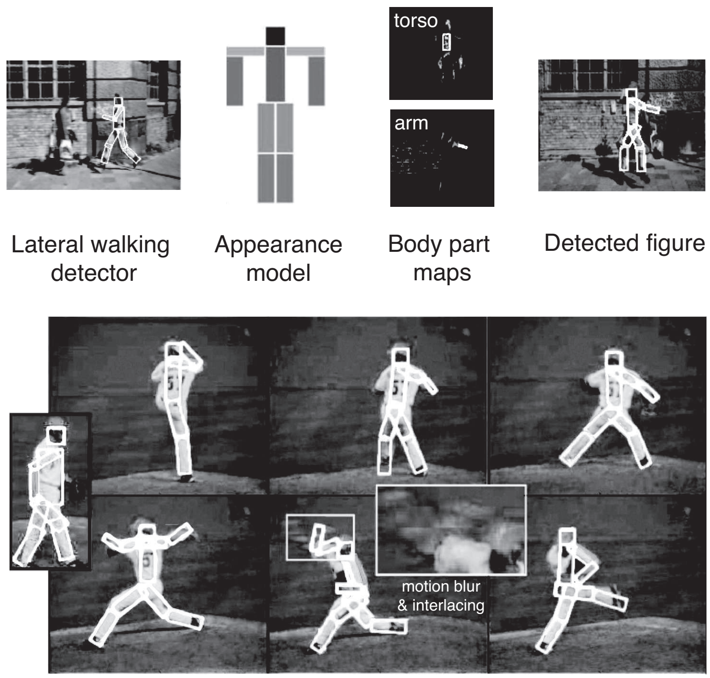


Figure 24.24 FILES: figures/tracker.eps (Tue Nov 3 16:23:57 2009). We can track moving people with a pictorial structure model by first obtaining an appearance model, then applying it. To obtain the appearance model, we scan the image to find a lateral walking pose. The detector does not need to be very accurate, but should produce few false positives. From the detector response, we can read off pixels that lie on each body segment, and others that do not lie on that segment. This makes it possible to build a discriminative model of the appearance of each body part, and these are tied together into a pictorial structure model of the person being tracked. Finally, we can reliably track by detecting this model in each frame. As the frames in the lower part of the image suggest, this procedure can track complicated, fast-changing body configurations, despite degradation of the video signal due to motion blur. Figure from ? (?) © IEEE.

Figure 24.25 FILES: figures/drinking-2.eps (Tue Nov 3 16:22:38 2009). Some complex human actions produce consistent patterns of appearance and motion. For example, drinking involves movements of the hand in front of the face. The first three images are correct detections of drinking; the fourth is a false-positive (the cook is looking into the coffee pot, but not drinking from it). Figure from ? (?) © IEEE.

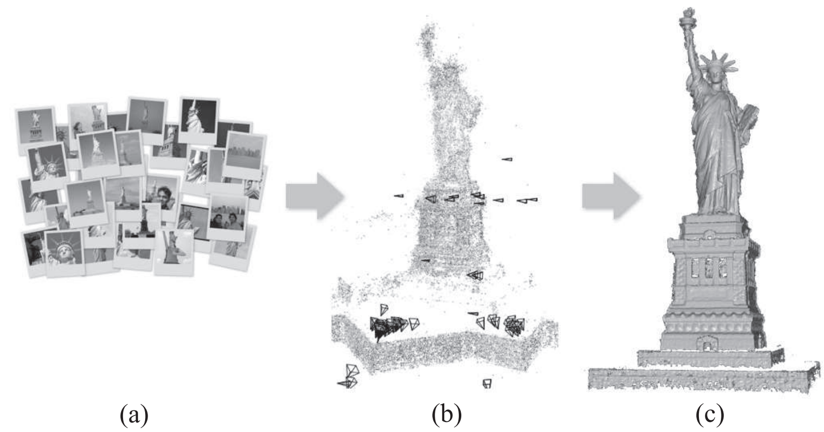


Figure 24.26 FILES: figures/liberty-new.eps (Tue Nov 3 16:23:06 2009). The state of the art in multiple-view reconstruction is now highly advanced. This figure outlines a system built by Michael Goesele and colleagues from the University of Washington, TU Darmstadt, and Microsoft Research. From a collection of pictures of a monument taken by a large community of users and posted on the Internet (a), their system can determine the viewing directions for those pictures, shown by the small black pyramids in (b) and a comprehensive 3D reconstruction shown in (c).

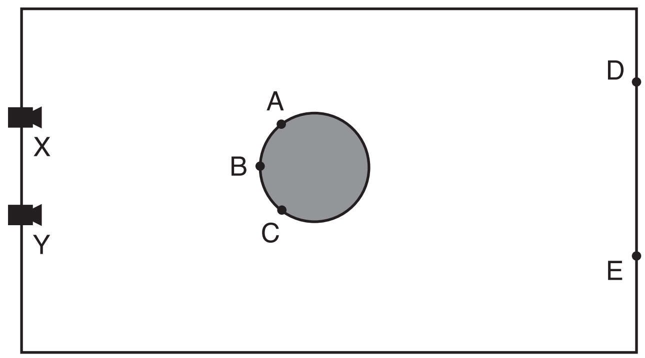
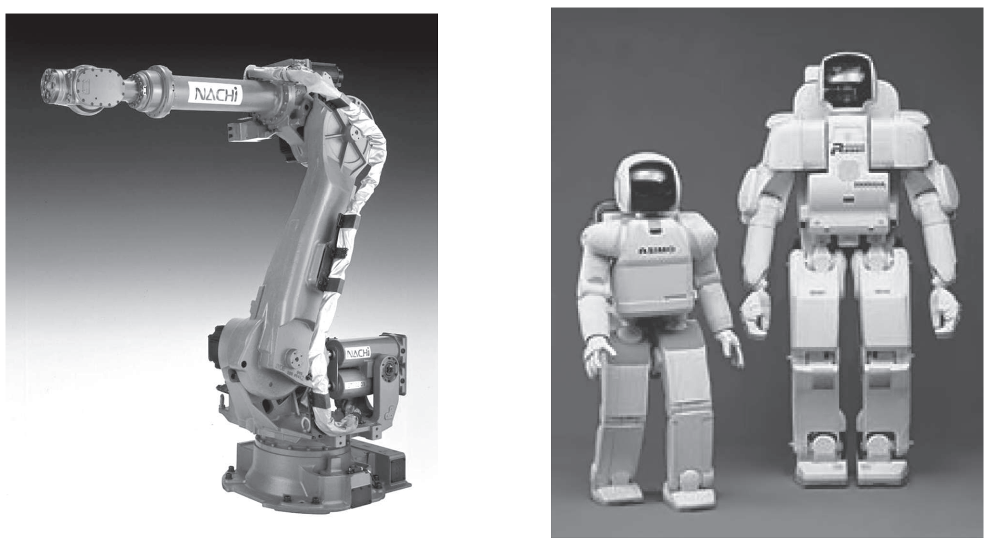


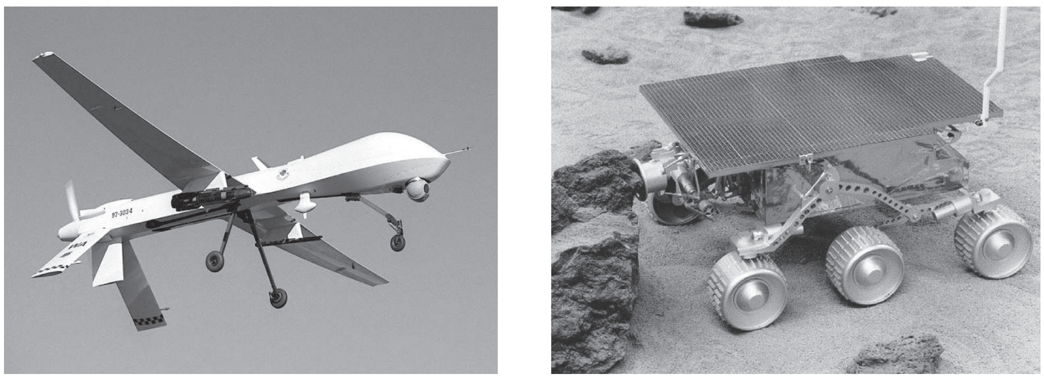
Figure 24.27 FILES: figures/bottle-stereo.eps (Tue Nov 3 16:22:28 2009). Top view of a two-camera vision system observing a bottle with a wall behind it.

25 ROBOTICS



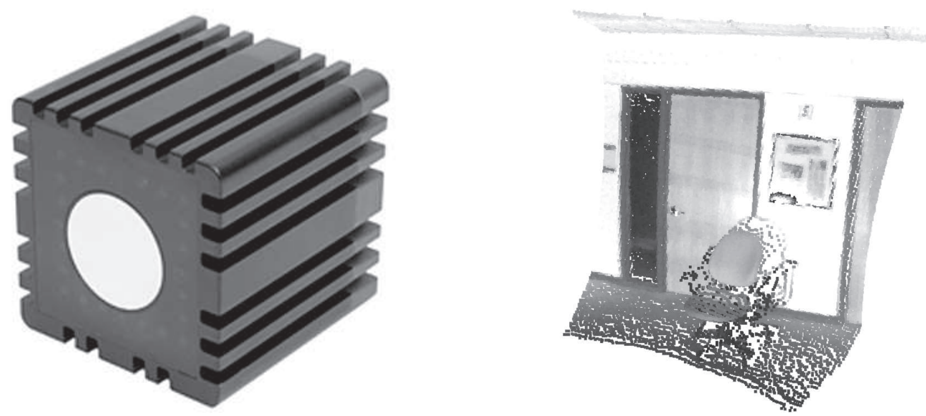
(a) (b)

Figure 25.1 FILES: figures/nachi.eps (Wed Nov 4 15:11:08 2009) figures/honda-asimo-robot.eps (not found). (a) An industrial robotic manipulator for stacking bags on a pallet. Image courtesy of Nachi Robotic Systems. (b) Honda's P3 and Asimo humanoid robots.



(a) (b)

Figure 25.2 FILES: figures/predator.eps (Wed Nov 4 15:12:28 2009) figures/Sojourner.eps (Wed Nov 4 15:12:23 2009). (a) Predator, an unmanned aerial vehicle (UAV) used by the U.S. Military. Image courtesy of General Atomics Aeronautical Systems. (b) NASA's Sojourner, a mobile robot that explored the surface of Mars in July 1997.



(a) (b)

Figure 25.3 FILES: figures/R317-SR4000-CW.eps (Wed Nov 4 15:16:09 2009) figures/wall-chair2.eps (Wed Nov 4 15:16:06 2009). (a) Time of flight camera; image courtesy of Mesa Imaging GmbH. (b) 3D range image obtained with this camera. The range image makes it possible to detect obstacles and objects in a robot's vicinity.

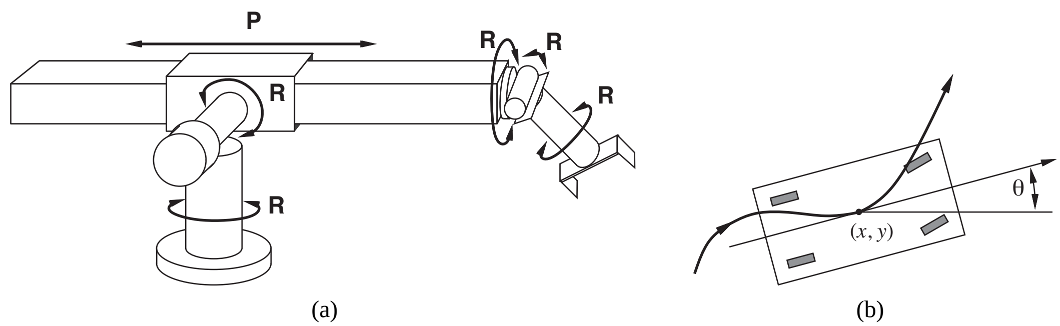
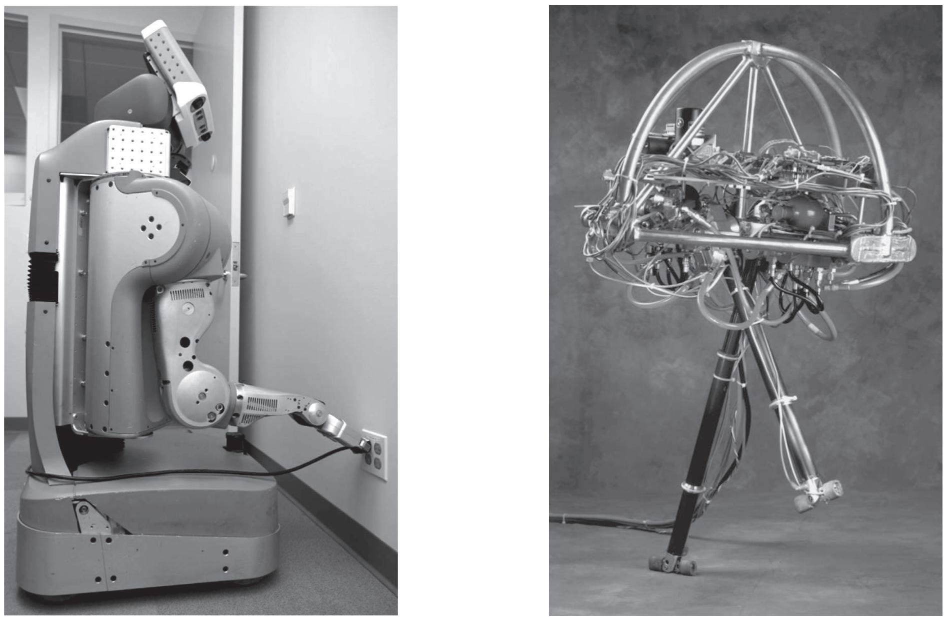
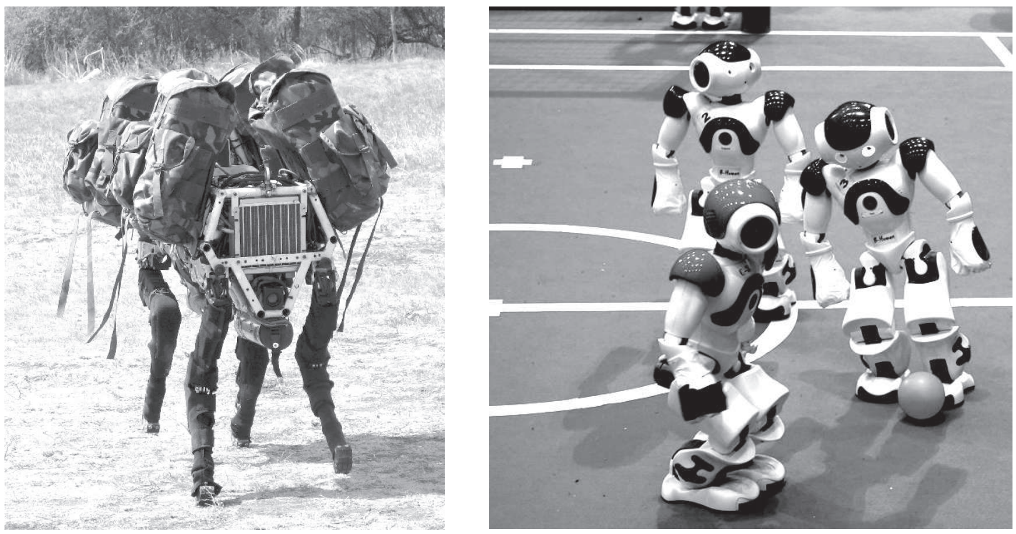


Figure 25.4 FILES: figures/stanford-arm.eps (Tue Nov 3 16:23:46 2009) figures/car-like.eps (Tue Nov 3 16:22:32 2009). (a) The Stanford Manipulator, an early robot arm with five revolute joints (R) and one prismatic joint (P), for a total of six degrees of freedom. (b) Motion of a nonholonomic four-wheeled vehicle with front-wheel steering.



(a) (b)

Figure 25.5 FILES: figures/RobotPlugInSkin.eps (Wed Nov 4 14:50:50 2009) figures/raibert-1leg.eps (Tue Nov 3 16:23:27 2009). (a) Mobile manipulator plugging its charge cable into a wall outlet. Image courtesy of Willow Garage, © 2009. (b) One of Marc Raibert's legged robots in motion.



(a) (b)

Figure 25.6 FILES: figures/BDI-DKFI.eps (Tue Nov 3 16:22:14 2009). (a) Four-legged dynamically-stable robot "Big Dog." Image courtesy Boston Dynamics, © 2009. (b) 2009 RoboCup Standard Platform League competition, showing the winning team, B-Human, from the DFKI center at the University of Bremen. Throughout the match, B-Human outscored their opponents 64:1. Their success was built on probabilistic state estimation using particle filters and Kalman filters; on machine-learning models for gait optimization; and on dynamic kicking moves. Image courtesy DFKI, © 2009.

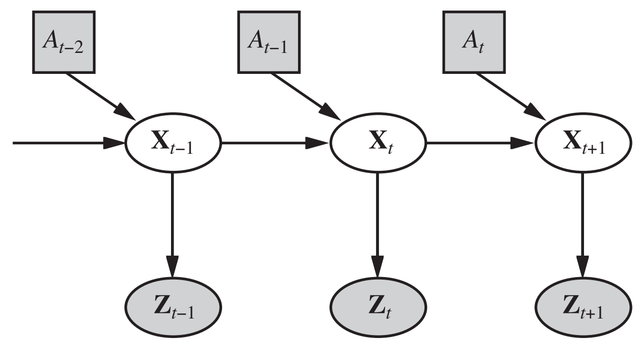


Figure 25.7 **FILES: figures/robotics-ddn.eps (Tue Nov 3 16:23:33 2009).** Robot perception can be viewed as temporal inference from sequences of actions and measurements, as illustrated by this dynamic Bayes network.

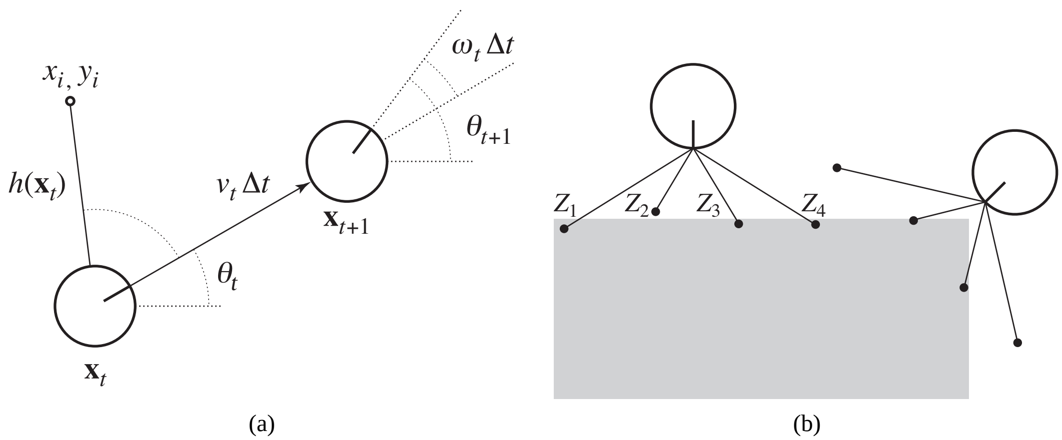


Figure 25.8 FILES: figures/robotics-pic2.eps (Tue Nov 3 16:23:34 2009) figures/range-scan-model.eps (Tue Nov 3 16:23:27 2009). (a) A simplified kinematic model of a mobile robot. The robot is shown as a circle with an interior line marking the forward direction. The state $\mathbf{x}_t$ consists of the (x_t, y_t) position (shown implicitly) and the orientation θ_t. The new state $\mathbf{x}_{t+1}$ is obtained by an update in position of $v_t\Delta_t$ and in orientation of $\omega_t\Delta_t$. Also shown is a landmark at (x_i, y_i) observed at time t. (b) The range-scan sensor model. Two possible robot poses are shown for a given range scan (z_1, z_2, z_3, z_4). It is much more likely that the pose on the left generated the range scan than the pose on the right.

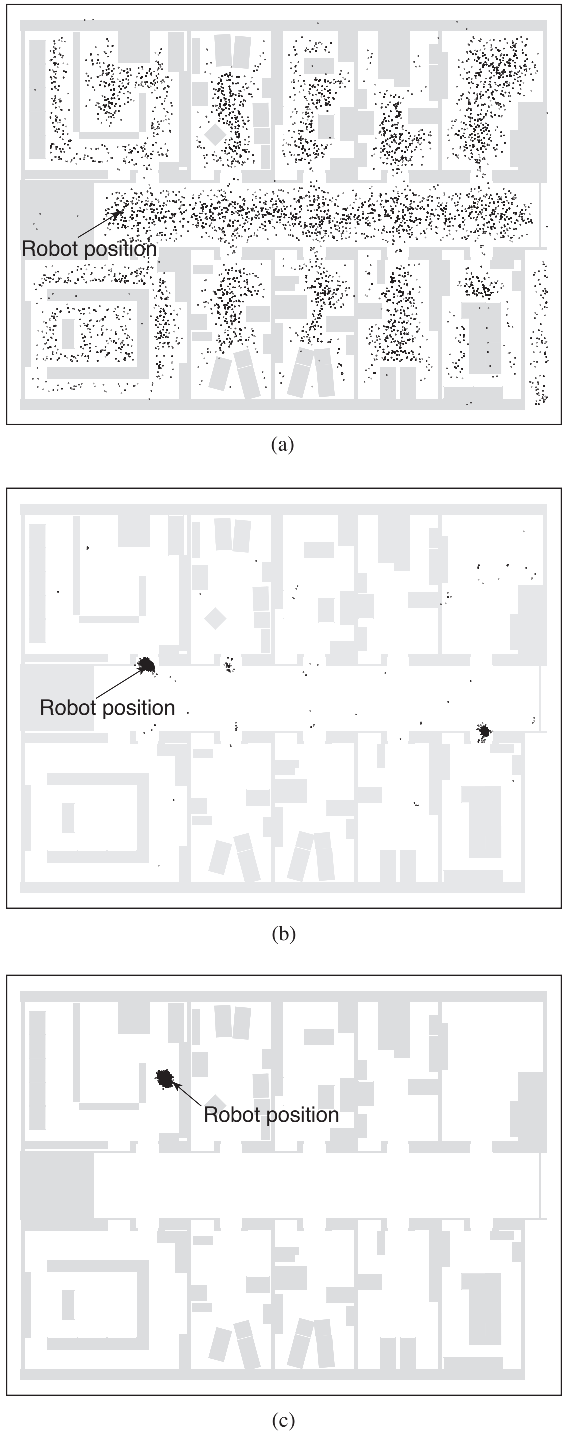


Figure 25.10 FILES: figures/first.eps (Tue Nov 3 16:22:51 2009) figures/second.eps (Tue Nov 3 16:23:41 2009) figures/third.eps (Tue Nov 3 16:23:54 2009). Monte Carlo localization, a particle filtering algorithm for mobile robot localization. (a) Initial, global uncertainty. (b) Approximately bi-modal uncertainty after navigating in the (symmetric) corridor. (c) Unimodal uncertainty after entering a room and finding it to be distinctive.

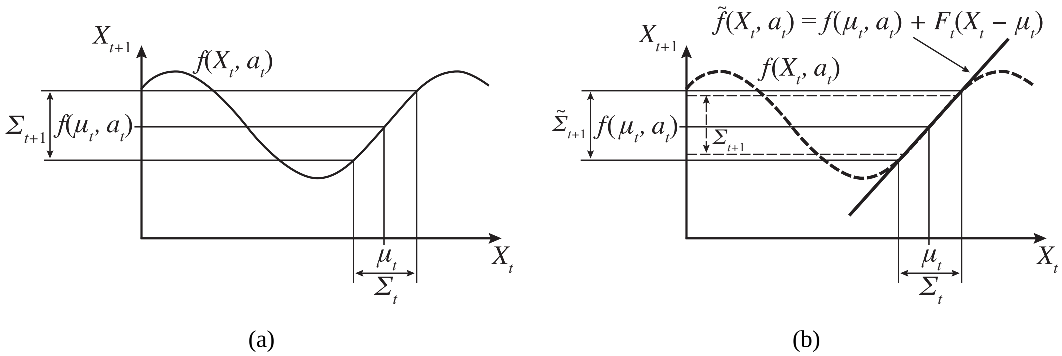


Figure 25.11 FILES: figures/robotics-pic3.eps (Tue Nov 3 16:23:34 2009) figures/robotics-pic4.eps (Tue Nov 3 16:23:34 2009). One-dimensional illustration of a linearized motion model: (a) The function f, and the projection of a mean $\boldsymbol{\mu}_t$ and a covariance interval (based on $\boldsymbol{\Sigma}_t$) into time $t+1$. (b) The linearized version is the tangent of f at $\boldsymbol{\mu}_t$. The projection of the mean $\boldsymbol{\mu}_t$ is correct. However, the projected covariance $\tilde{\boldsymbol{\Sigma}}_{t+1}$ differs from $\boldsymbol{\Sigma}_{t+1}$.

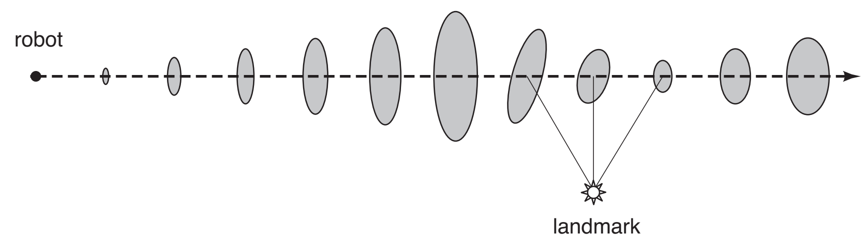


Figure 25.12 FILES: figures/robotics-pic6.eps (Tue Nov 3 16:23:35 2009). Example of localization using the extended Kalman filter. The robot moves on a straight line. As it progresses, its uncertainty increases gradually, as illustrated by the error ellipses. When it observes a landmark with known position, the uncertainty is reduced.

(a) (b) (c)

Figure 25.13 FILES: figures/visimg1.eps (Wed Nov 4 15:06:58 2009) figures/visimg2.eps (Wed Nov 4 15:07:03 2009) figures/visimg3.eps (Wed Nov 4 15:07:11 2009). Sequence of "drivable surface" classifier results using adaptive vision. In (a) only the road is classified as drivable (striped area). The V-shaped dark line shows where the vehicle is heading. In (b) the vehicle is commanded to drive off the road, onto a grassy surface, and the classifier is beginning to classify some of the grass as drivable. In (c) the vehicle has updated its model of drivable surface to correspond to grass as well as road.

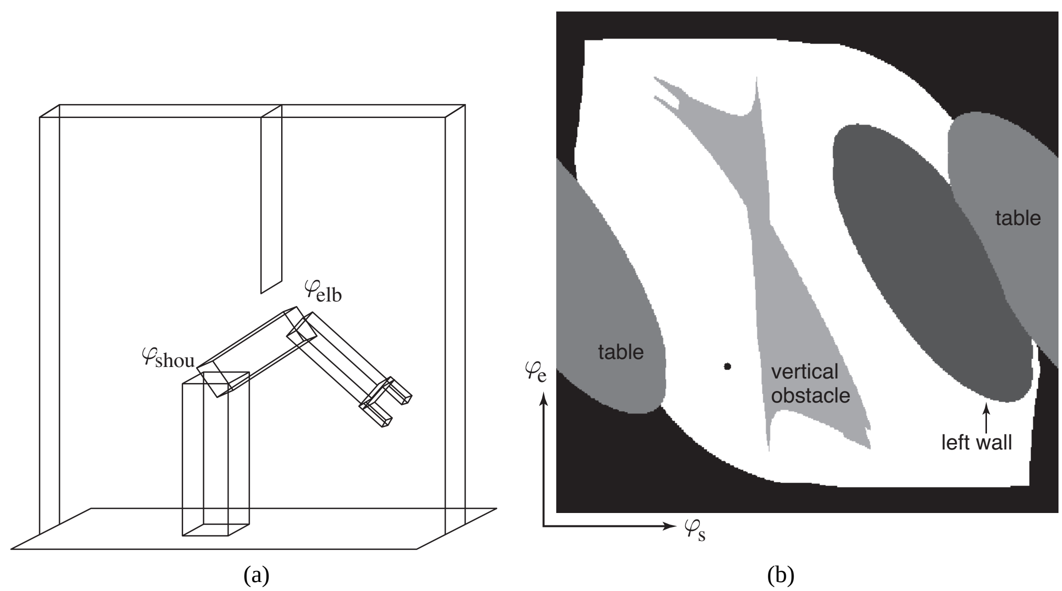


Figure 25.14 FILES: figures/armPlain.eps (Tue Nov 3 16:22:22 2009) figures/armPlainConfSpace.eps (Tue Nov 3 16:22:23 2009). (a) Workspace representation of a robot arm with 2 DOFs. The workspace is a box with a flat obstacle hanging from the ceiling. (b) Configuration space of the same robot. Only white regions in the space are configurations that are free of collisions. The dot in this diagram corresponds to the configuration of the robot shown on the left.

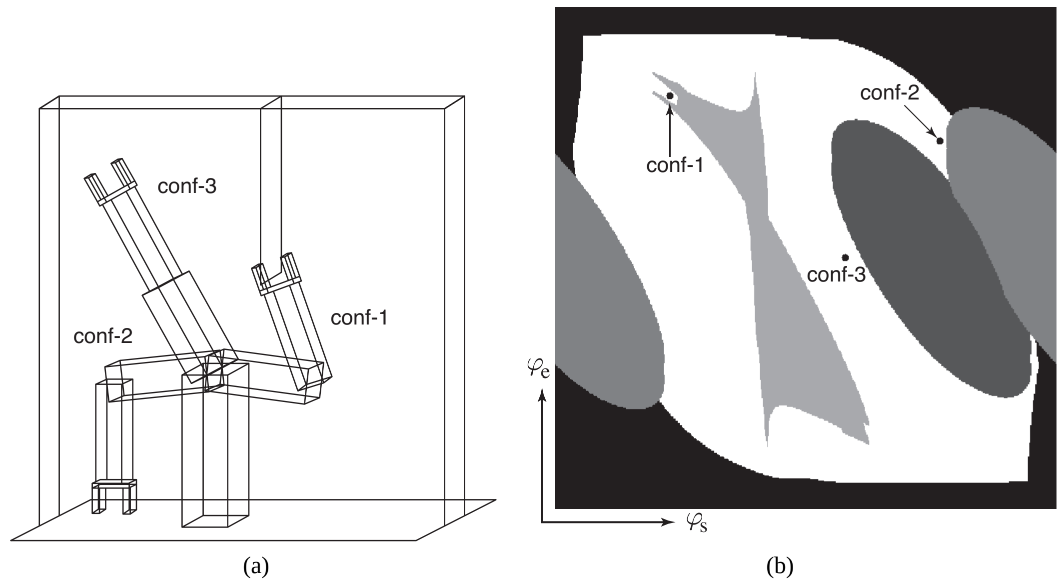


Figure 25.15 FILES: figures/armExampleWorkSpace.eps (Tue Nov 3 16:22:22 2009) figures/armExampleConfSpace.eps (Tue Nov 3 16:22:22 2009). Three robot configurations, shown in workspace and configuration space.

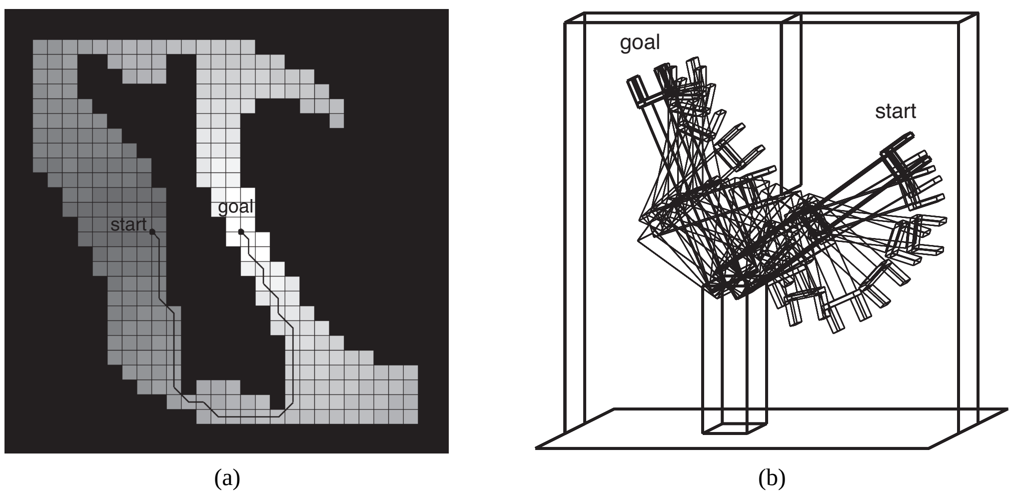
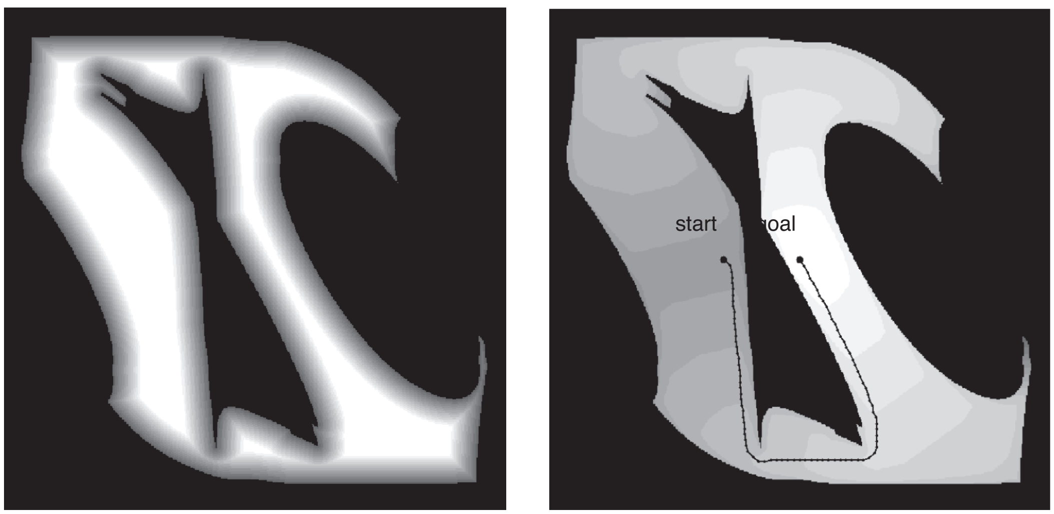
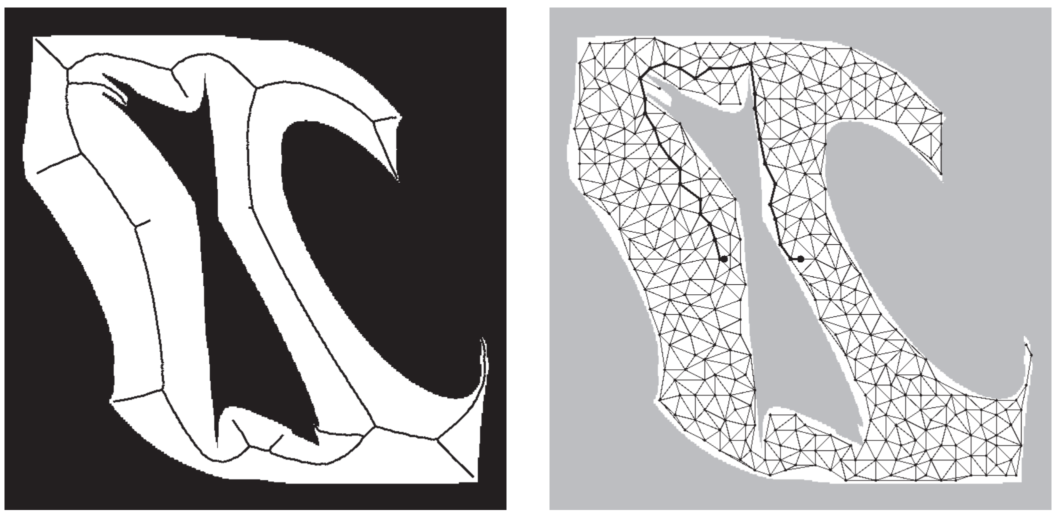


Figure 25.16 FILES: figures/armDPwithoutPotentialCoarse.eps (Wed Nov 4 15:51:42 2009) figures/armDPwithoutPotentialWorkspaceCoarse.eps (Tue Nov 3 16:22:22 2009). (a) Value function and path found for a discrete grid cell approximation of the configuration space. (b) The same path visualized in workspace coordinates. Notice how the robot bends its elbow to avoid a collision with the vertical obstacle.



(a) (b)

Figure 25.17 FILES: figures/armPotentialField.eps (Tue Nov 3 16:22:23 2009) figures/armDPwithPotential.eps (Tue Nov 3 16:22:21 2009). (a) A repelling potential field pushes the robot away from obstacles. (b) Path found by simultaneously minimizing path length and the potential.



(a) (b)

Figure 25.18 FILES: figures/armVoronoi.eps (Tue Nov 3 16:22:23 2009) figures/armRoadmap.eps (Tue Nov 3 16:22:23 2009). (a) The Voronoi graph is the set of points equidistant to two or more obstacles in configuration space. (b) A probabilistic roadmap, composed of 400 randomly chosen points in free space.

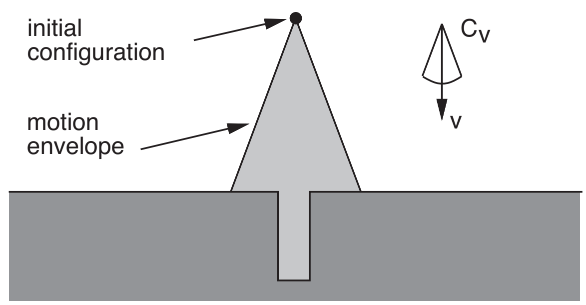


Figure 25.19 FILES: figures/peg-in-hole.eps (Tue Nov 3 16:23:17 2009). A two-dimensional environment, velocity uncertainty cone, and envelope of possible robot motions. The intended velocity is v, but with uncertainty the actual velocity could be anywhere in C_v, resulting in a final configuration somewhere in the motion envelope, which means we wouldn't know if we hit the hole or not.

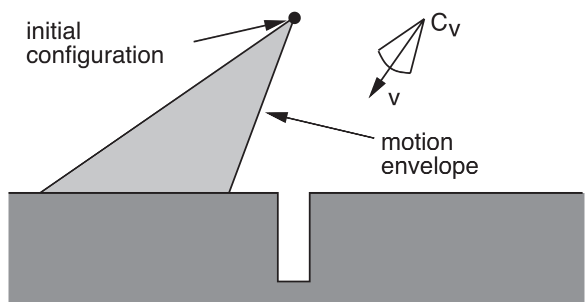


Figure 25.20 **FILES: figures/peg-in-hole-step1.eps (Tue Nov 3 16:23:16 2009).** The first motion command and the resulting envelope of possible robot motions. No matter what the error, we know the final configuration will be to the left of the hole.

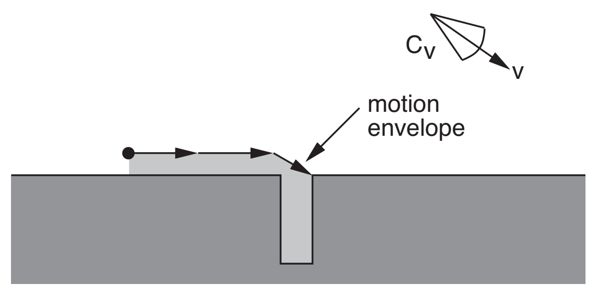
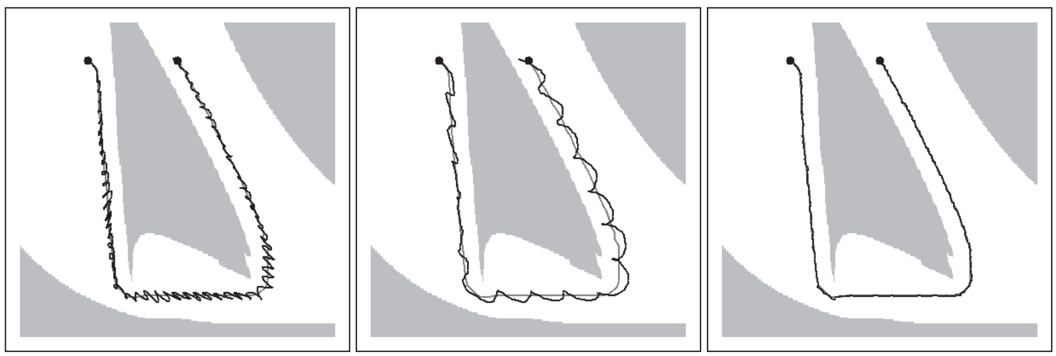


Figure 25.21 **FILES: figures/peg-in-hole-step2.eps (Tue Nov 3 16:23:16 2009).** The second motion command and the envelope of possible motions. Even with error, we will eventually get into the hole.



(a) (b) (c)

Figure 25.22 FILES: figures/armControlP1.0.eps (Tue Nov 3 16:22:21 2009) figures/armControlP0.1.eps (Tue Nov 3 16:22:21 2009) figures/armControlP0.3D0.8.eps (Tue Nov 3 16:22:21 2009). Robot arm control using (a) proportional control with gain factor 1.0, (b) proportional control with gain factor 0.1, and (c) PD (proportional derivative) control with gain factors 0.3 for the proportional component and 0.8 for the differential component. In all cases the robot arm tries to follow the path shown in gray.

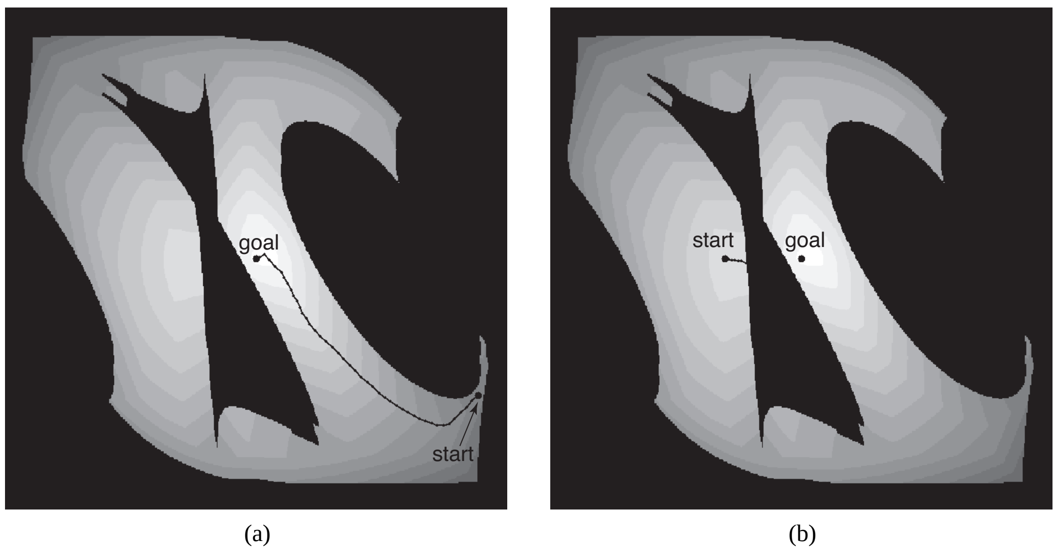
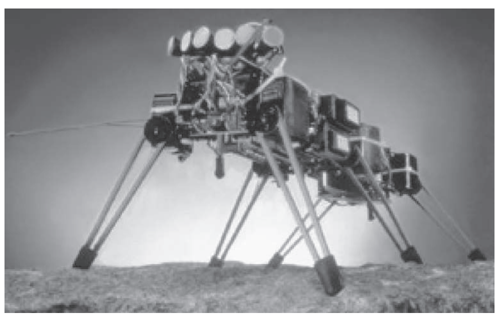
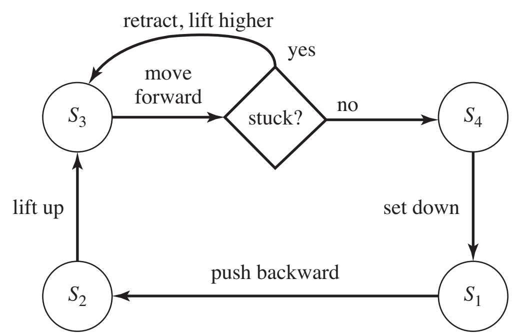


Figure 25.23 FILES: figures/armSimplePotentialAlt.eps (Tue Nov 3 16:22:23 2009) figures/armSimplePotential.eps (Tue Nov 3 16:22:23 2009). Potential field control. The robot ascends a potential field composed of repelling forces asserted from the obstacles and an attracting force that corresponds to the goal configuration. (a) Successful path. (b) Local optimum.



(a)



(b)

Figure 25.24 FILES: figures/genghis.eps (Tue Nov 3 13:34:49 2009) figures/robotics-pic5.eps (Tue Nov 3 16:23:34 2009). (a) Genghis, a hexapod robot. (b) An augmented finite state machine (AFSM) for the control of a single leg. Notice that this AFSM reacts to sensor feedback: if a leg is stuck during the forward swinging phase, it will be lifted increasingly higher.

Figure 25.25 **FILES: figures/flip-mosaic.eps (not found).** Multiple exposures of an RC helicopter executing a flip based on a policy learned with reinforcement learning. Images courtesy of Andrew Ng, Stanford University.

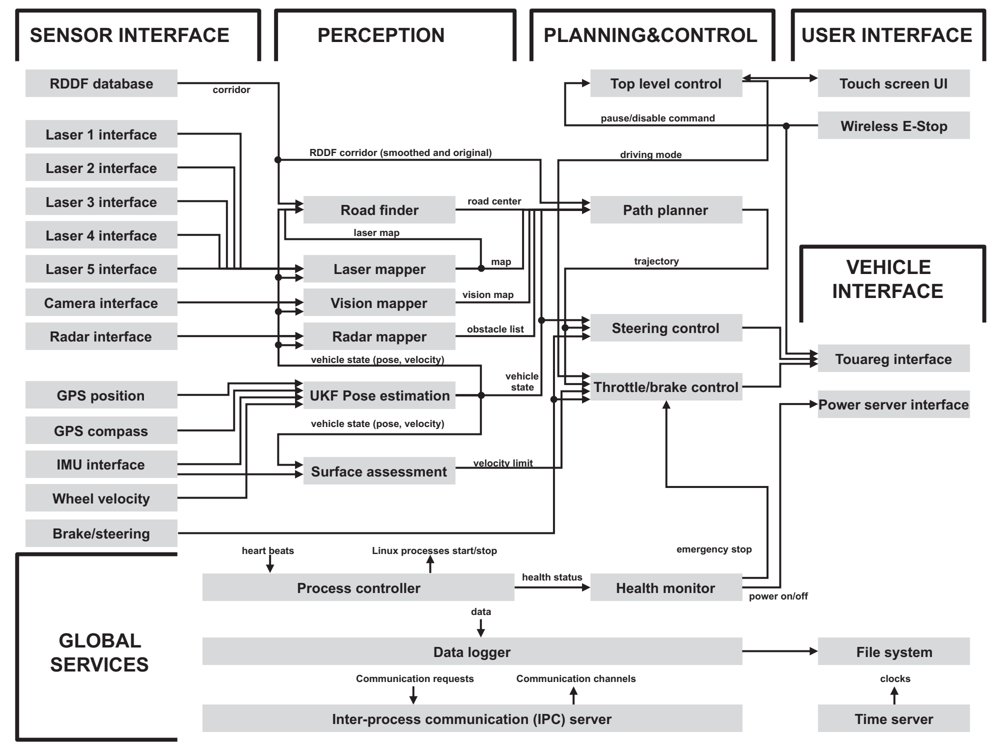


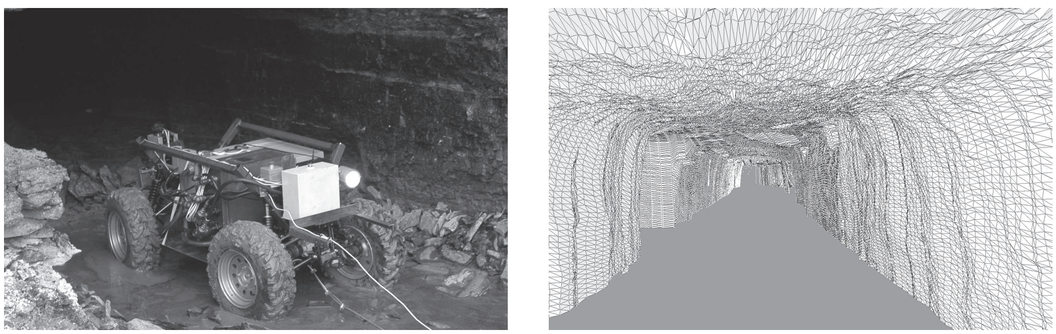
Figure 25.26 **FILES: figures/stanley-processes.eps (Wed Nov 4 11:17:26 2009).** Software architecture of a robot car. This software implements a data pipeline, in which all modules process data simultaneously.

(a) (b)

Figure 25.27 FILES: figures/helpmate.eps (Tue Nov 3 15:26:49 2009) figures/DenvierStation.eps (Tue Nov 3 16:22:14 2009). (a) The Helpmate robot transports food and other medical items in dozens of hospitals worldwide. (b) Kiva robots are part of a material-handling system for moving shelves in fulfillment centers. Image courtesy of Kiva Systems.

(a) (b)

Figure 25.28 FILES: figures/race12.eps (Wed Nov 4 15:18:14 2009) figures/munich-ORsmall.eps (not found). (a) Robotic car BOSS, which won the DARPA Urban Challenge. Courtesy of Carnegie Mellon University. (b) Surgical robots in the operating room. Image courtesy of da Vinci Surgical Systems.



(a) (b)

Figure 25.29 FILES: figures/mine-robot.eps (Tue Nov 3 16:23:11 2009) figures/mine-data.eps (Tue Nov 3 16:23:09 2009). (a) A robot mapping an abandoned coal mine. (b) A 3D map of the mine acquired by the robot.

(a) (b)

Figure 25.30 FILES: figures/roomba1.eps (Wed Nov 4 15:22:23 2009) figures/icra-cordless-phone-gray3.eps (not found). (a) Roomba, the world's best-selling mobile robot, vacuums floors. Image courtesy of iRobot, © 2009. (b) Robotic hand modeled after human hand. Image courtesy of University of Washington and Carnegie Mellon University.

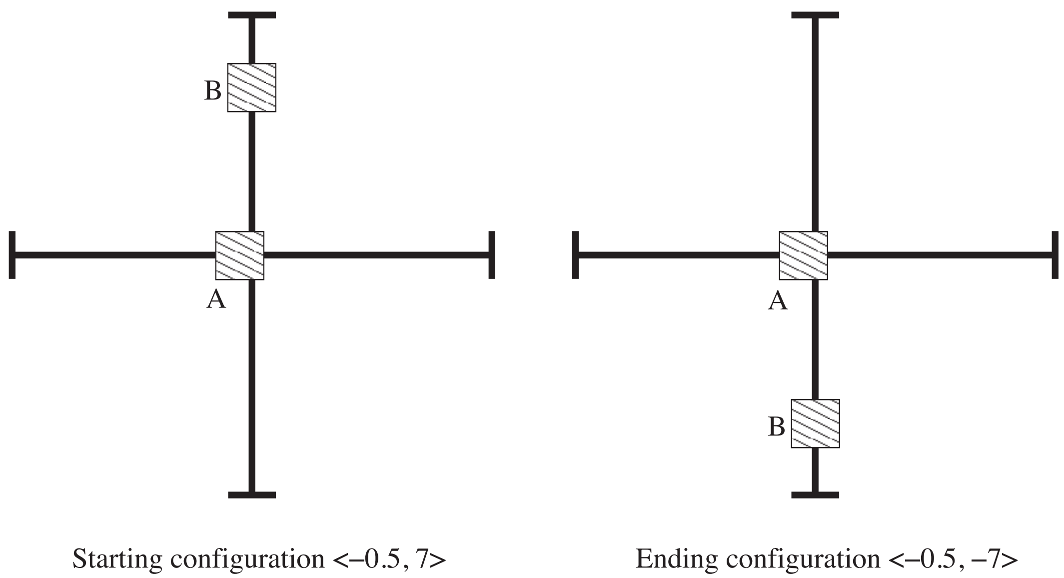
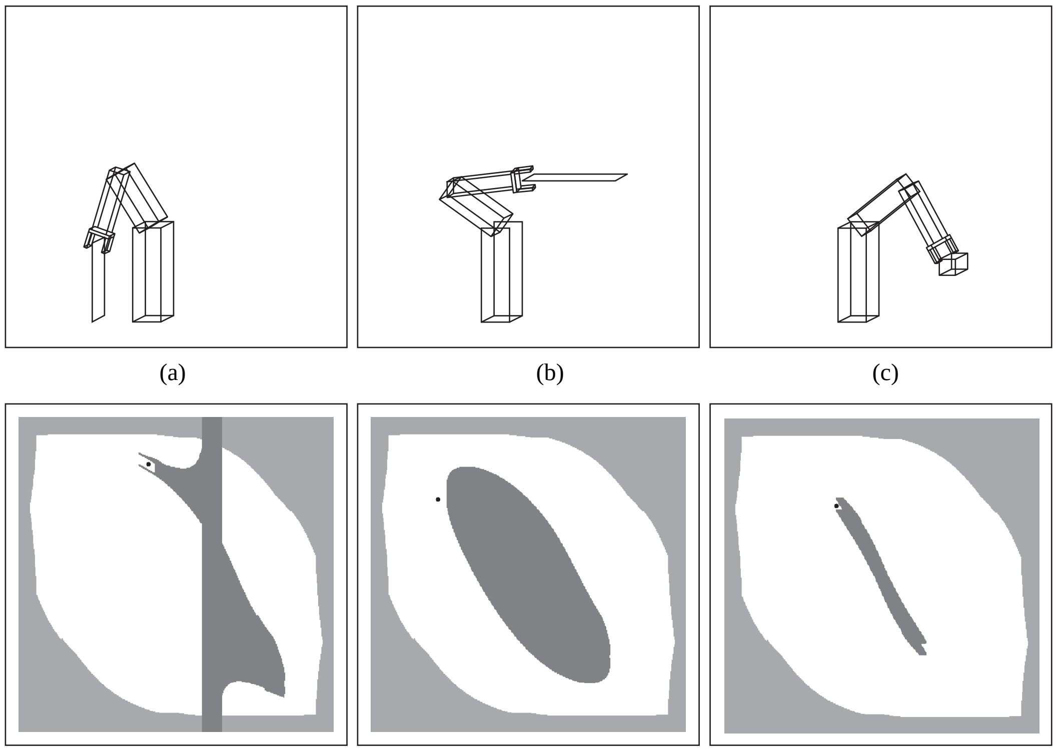


Figure 25.31 FILES: figures/figRobot2.eps (Tue Nov 3 16:22:47 2009). A Robot manipulator in two of its possible configurations.



(a) (b) (c)

(d) (e) (f)

Figure 25.32 FILES: figures/exerciseRobot1.eps (Tue Nov 3 16:22:42 2009) figures/exerciseRobot3.eps (Tue Nov 3 16:22:43 2009) figures/exerciseRobot6.eps (Tue Nov 3 16:22:44 2009) figures/exerciseConf2.eps (Tue Nov 3 16:22:41 2009) figures/exerciseConf4.eps (Tue Nov 3 16:22:42 2009) figures/exerciseConf5.eps (Tue Nov 3 16:22:42 2009). Diagrams for Exercise **??**.

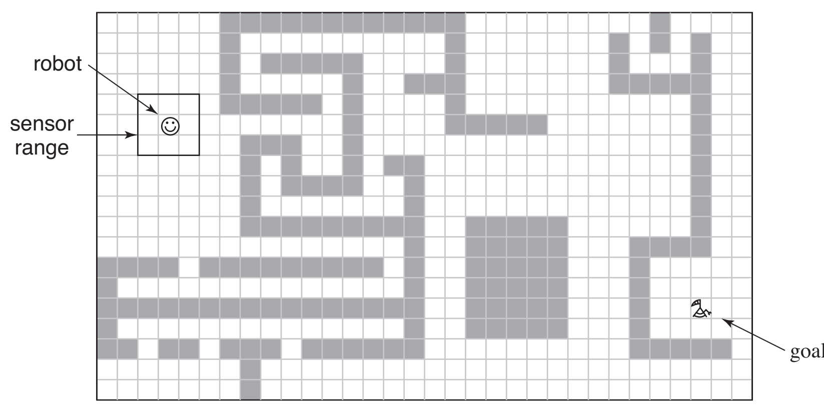


Figure 25.33 FILES: figures/robotics-pic7.eps (Tue Nov 3 16:23:35 2009). Simplified robot in a maze. See Exercise **??**.

26 PHILOSOPHICAL FOUNDATIONS

27 AI: THE PRESENT AND FUTURE

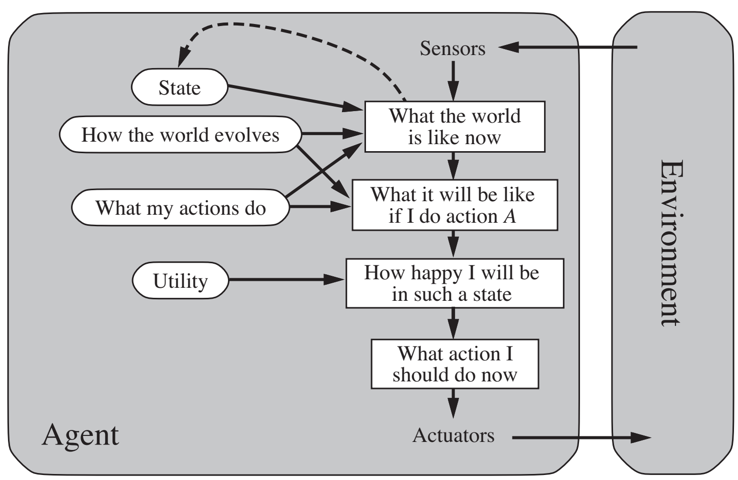


Figure 27.1 FILES: figures/utility-based-agent.eps (Tue Nov 3 16:23:59 2009). A model-based, utility-based agent, as first presented in Figure 2.10.

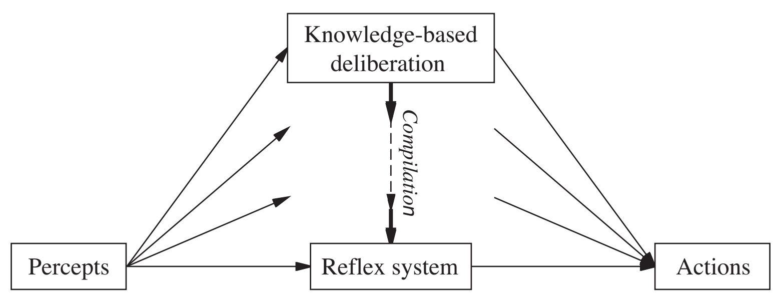


Figure 27.2 FILES: figures/compilation.eps (Tue Nov 3 16:22:34 2009). Compilation serves to convert deliberative decision making into more efficient, reflexive mechanisms.

28 MATHEMATICAL BACKGROUND

29 NOTES ON LANGUAGES AND ALGORITHMS