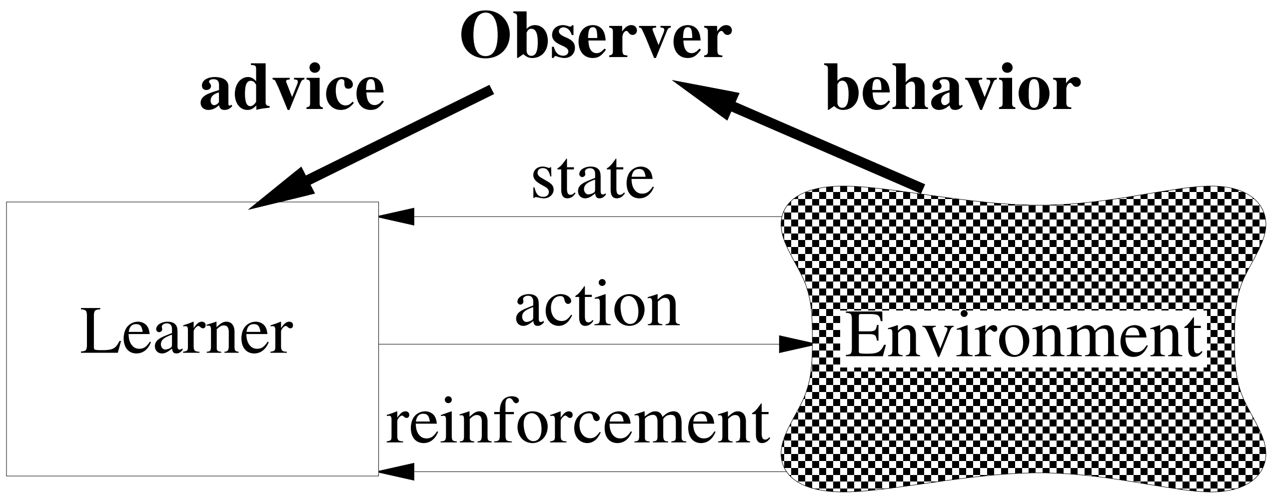

Figure 1: In basic reinforcement learning the learner receives a description of the current environment (the state), selects an action to choose, and receives a reinforcement as a consequence of selecting that action. We augment this with a process that allows an observer to watch the learner and suggest advice based on the learner's behavior.

Table 1: Samples of advice in our advice language (left column).

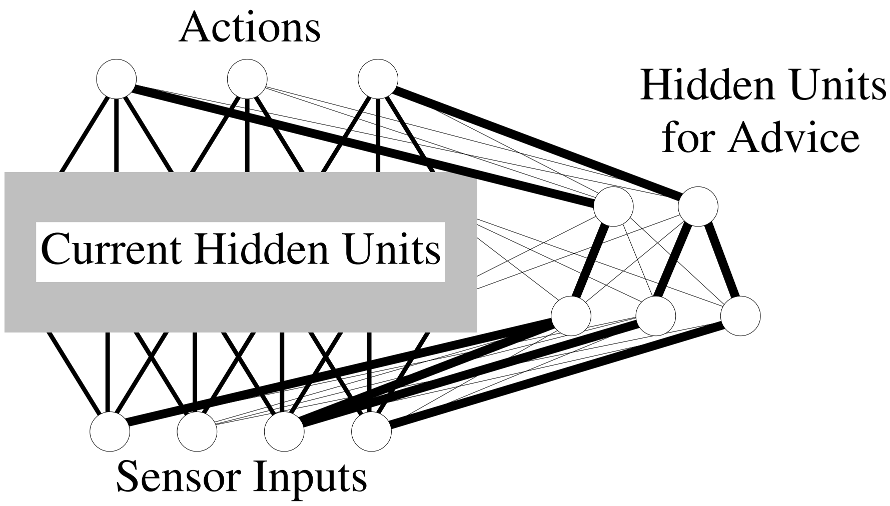

Figure 2: Adding advice to the RL agent's neural network by creating new hidden units that represent the advice. The thick links on the right capture the semantics of the advice. The added thin links initially have near-zero weight; during subsequent backpropagation training the magnitude of their weights can change, thereby refining the original advice. Details and an example appear in Section 3.

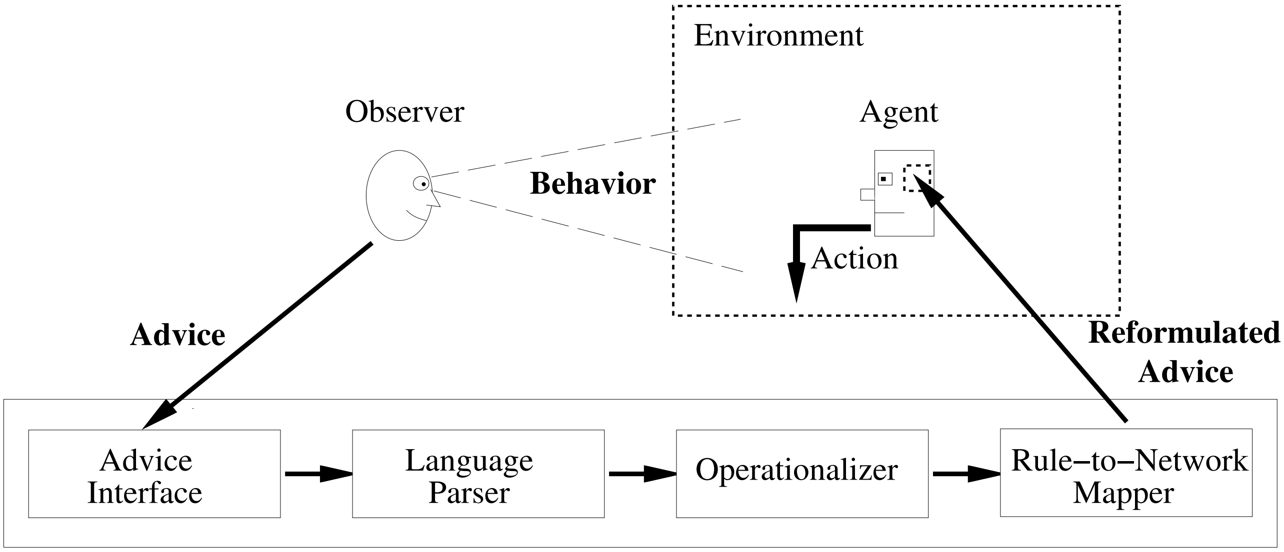

Figure 3: Interaction of the observer, agent, and our advice-taking system. The process is a cycle: the observer watches the agent's behavior to determine what advice to give, the advice-taking system processes the advice and inserts it into the agent, which changes the agent's behavior, thus possibly causing the observer to provide more advice. The agent operates as a normal Q-learning agent when not presented with advice.

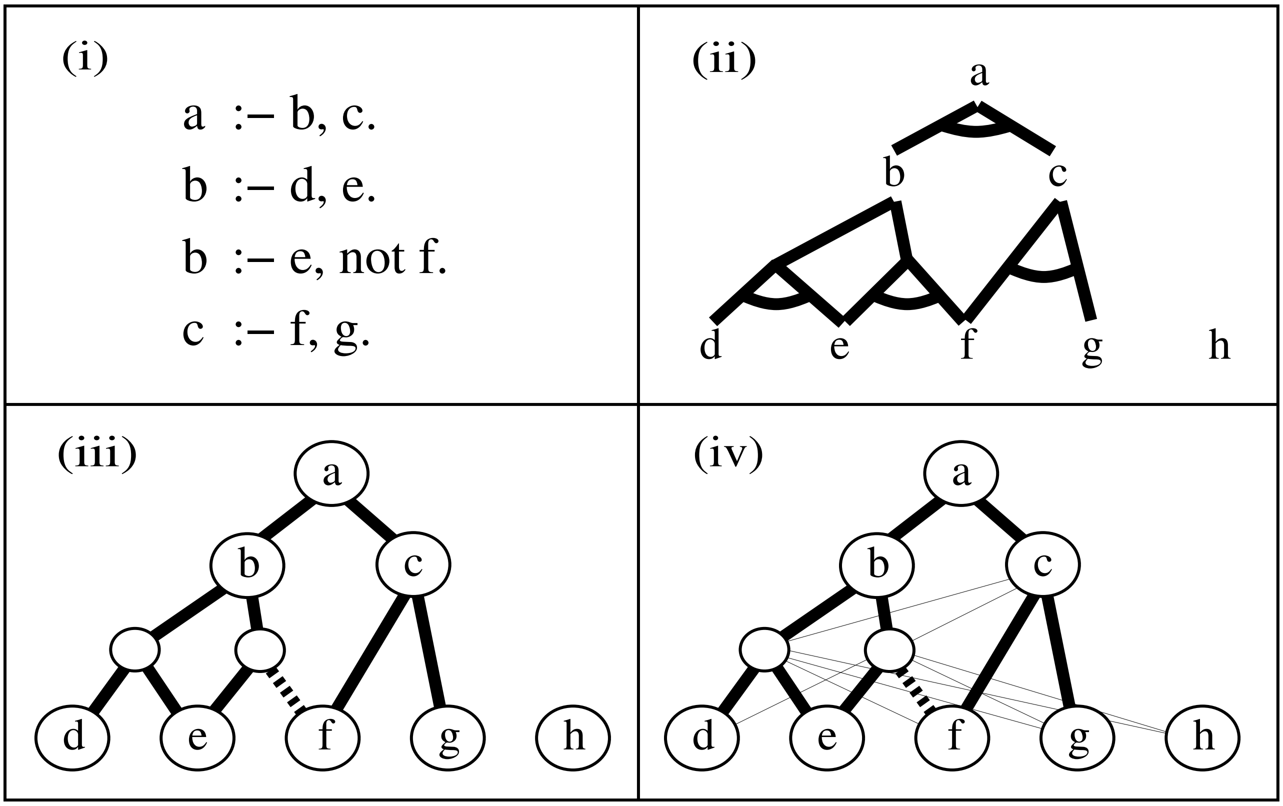

Figure 4: Sample of the KBANN algorithm: (i) a propositional rule set in Prolog notation; (ii) the rules viewed as an AND-OR dependency graph; (iii) each proposition is represented as a unit (extra units are also added to represent disjunctive definitions, e.g., b), and their weights and biases are set so that they implement AND or OR gates, e.g, the weights $b \rightarrow a$ and $c \rightarrow a$ are set to 4 and a's bias (threshold) to 6 (the bias of an OR node is 2); (iv) low-weighted links are added between layers as a basis for future learning (e.g., an antecedent can be added to a rule by increasing one of these weights).

Figure 5: On the left is the first piece of advice from Table 1. On the right is RATLE's translation of this piece of advice. The shaded ellipse represents the original hidden units. Arcs show units and weights that are set to implement a conjunction. RATLE adds zero-weighted links (not shown here – see Figure 4d) to other parts of the current network; these links support subsequent refinement.

Figure 6: On the left is the second piece of advice from Table 1. On the right is RATLE's translation of it. Dotted lines indicate negative weights. These new units are added to the existing network (not shown).

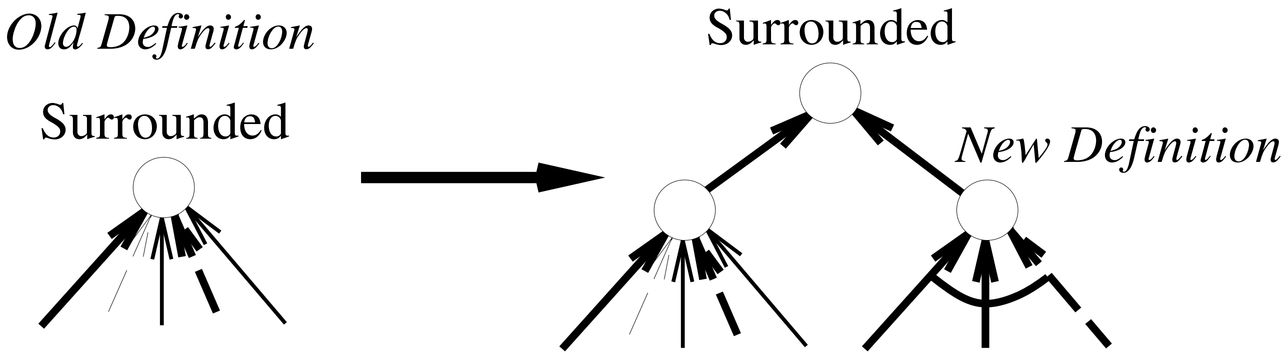

Figure 7: Incorporating the definition of a term that already exists.

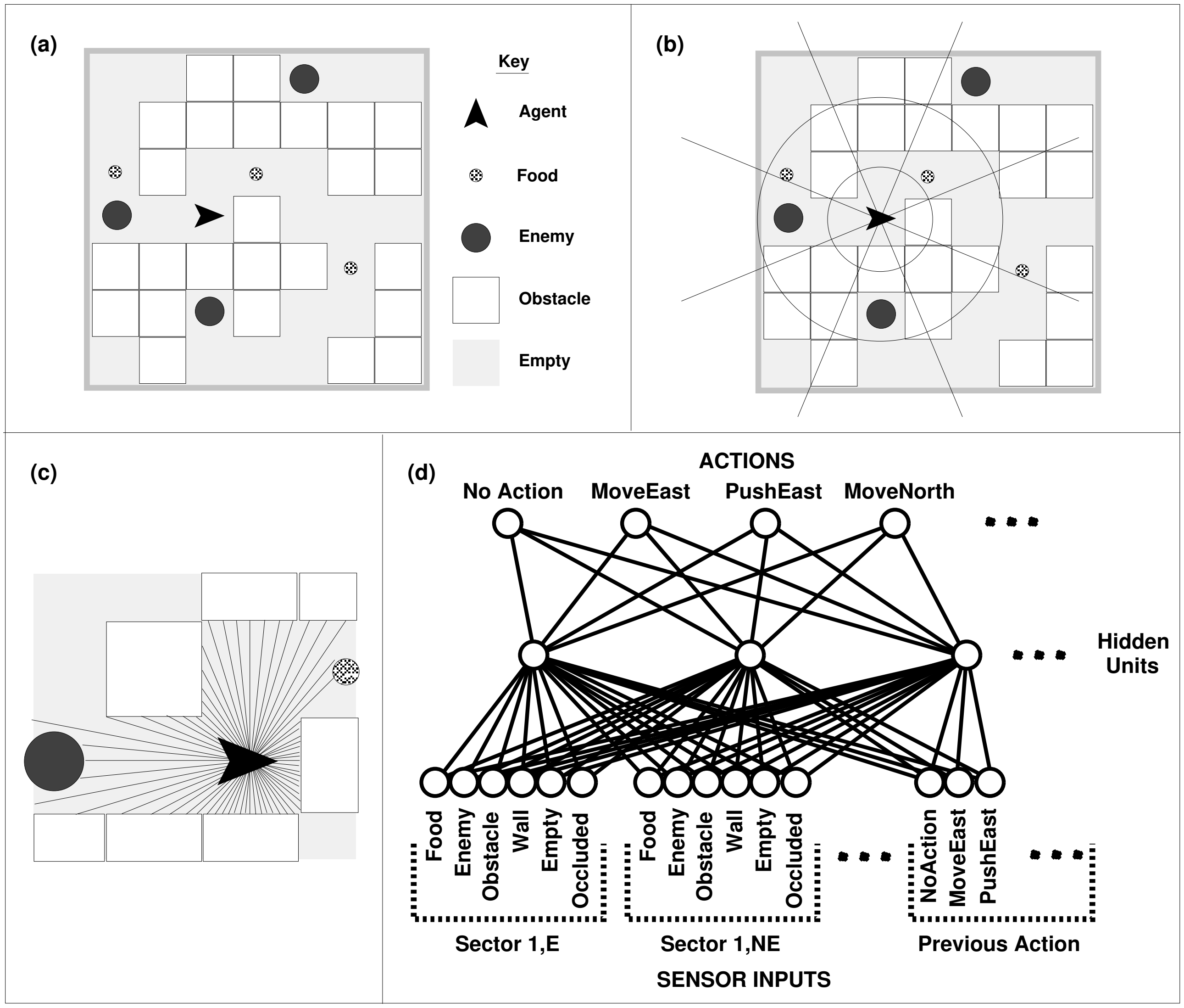

Figure 8: Our sample test environment: (a) sample configuration; (b) sample division of the environment into sectors; (c) distances to the nearest occluding object along a fixed set of arcs (measured from the agent); (d) a neural network that computes the utility of actions.

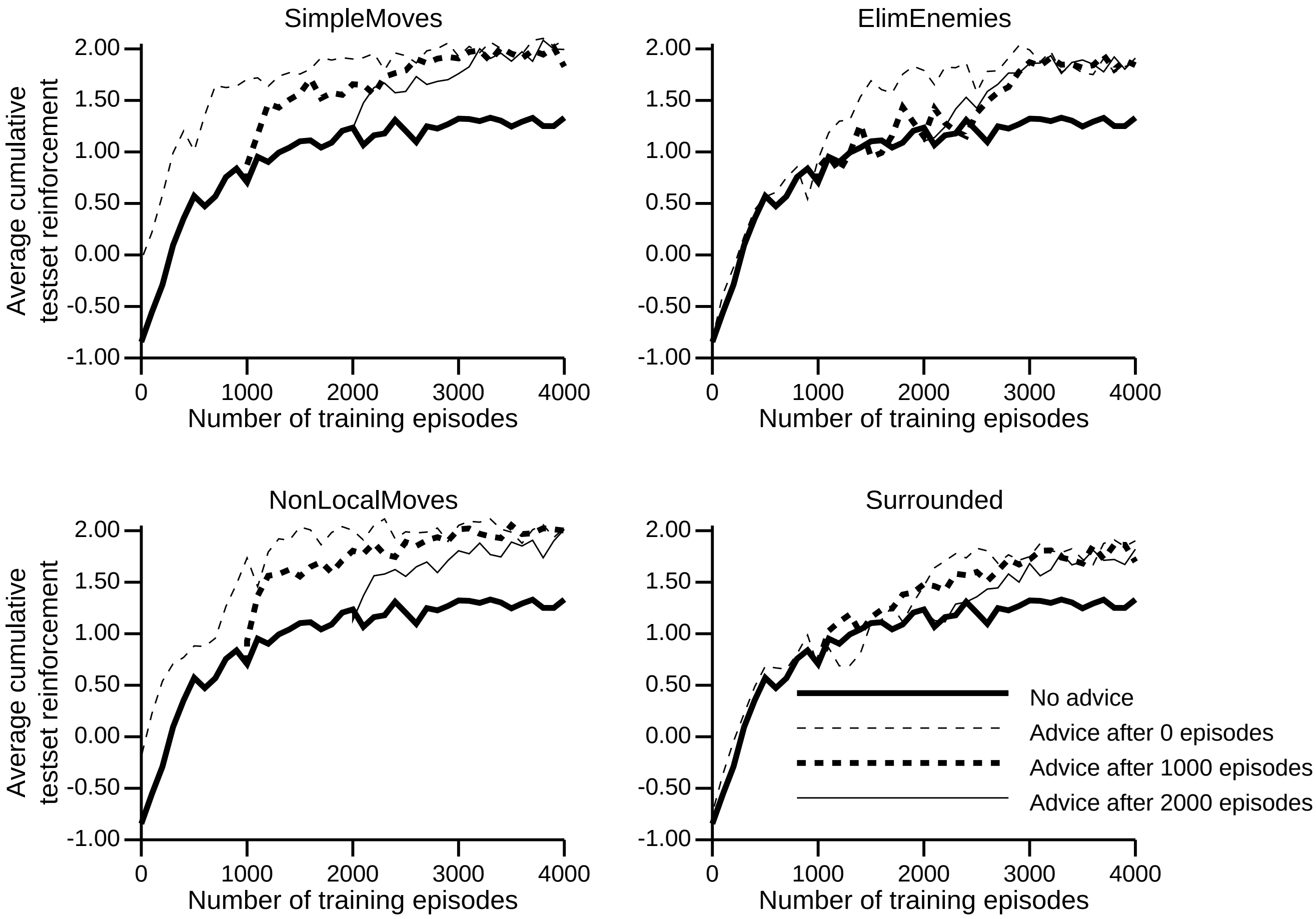

Figure 9: Average total reinforcement for our four sample pieces of advice as a function of amount of training and point of insertion of the advice.

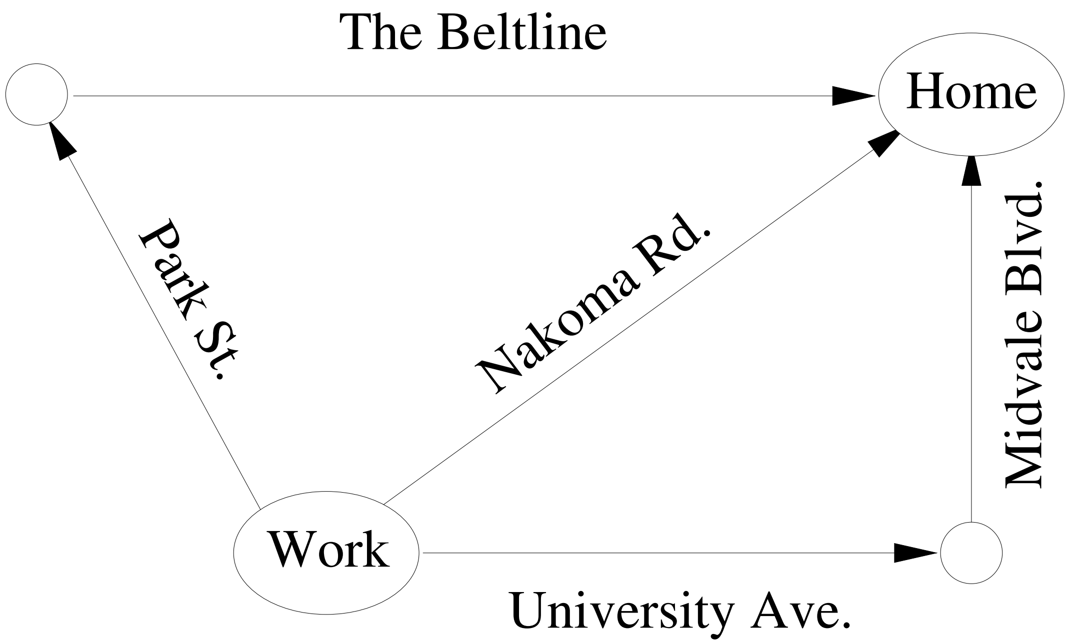

Figure 10: A sample problem where good advice can fail to enhance performance. Assume the goal of the agent is to go from Work to Home, and that the agent will receive a large reward for taking Nakoma Road, and a slightly smaller reward for taking University Avenue followed by Midvale Boulevard. If the agent receives advice that University followed by Midvale is a good plan, the agent, when confronted with the problem of going from Work to Home will likely follow this plan (even during training, since actions are selected proportional to their predicted utility). Thus it may take a long time for the learner to try Nakoma Road often enough to learn that it is a better route. A learner without advice might try both routes equally often and quickly learn the correct utility value for each route.