Commonsense Knowledge Retrieval

Preliminary Report

Phil Oertel
Computer Science Department
University of Illinois at Urbana-Champaign
Urbana, IL 61801
oertel@uiuc.edu

Eyal Amir
Computer Science Department
University of Illinois at Urbana-Champaign
Urbana, IL 61801
eyal@cs.uiuc.edu

Abstract

An autonomous agent that explores and acts in a rich world needs knowledge to act effectively. This agent can use knowledge that is available in *Commonsense Knowledge Bases* (CSKBs), when the agent designer cannot encode all the information the agent might need. CSKBs include general-purpose information about the everyday world in a formal language, but this information is not always correct, relevant, or useful for the agent's purpose.

In this paper we present an approach to retrieving commonsense knowledge for autonomous decision making. We consider agents whose formal language is different from that of the CSKB, and can use multiple CSKBs of various expressivity and coverage. We present a complete retrieval framework with algorithms for mapping languages and selection of knowledge. We report on preliminary experimental results of these algorithms for the ConceptNet CSKB.

1 Introduction

There is growing interest in using knowledge about the world to aid autonomous decision making, e.g., [Bacchus and Kabanza, 2000; Doherty and Kvarnström, 2001]. This knowledge is crafted carefully by a knowledge engineer to fit the domain and goal. However, autonomous agents that explore and acts in rich world cannot receive all their information from the agent designer because he does not know the target environment in design time or the environment is too complex. In these cases this agent can use knowledge that is available in *Commonsense Knowledge Bases* (CSKBs).

Typically, CSKBs comprise of ontologies and hierarchical concept information together with many thousands of axioms that bring these concept together. Examples include CYC [Lenat, 1995], SUMO [Niles and Pease, 2001], the HPKB project at SRI [Cohen *et al.*, 1998], the HPKB project at Stanford's KSL [Fikes and Farquhar, 1999], ConceptNet [Singh *et al.*, 2002], and WordNet [Miller, 1995].

The challenges posed by decision making are not addressed by current CSKB research. Research on CSKBs has focused so far on applications for natural language processing and for question answering. The first typically uses the concept hierarchy information that is embedded in the CSKB, while the second uses axioms of the CSKB that are tuned carefully for the specific topic and queries together with a theorem prover. In contrast, decision making requires knowledge that is versatile and can aid in answering questions like *"what will action a do in my situation?"* or *"how do I get closer to X?"*. The two challenges posed by such applications are selecting knowledge that is correct, relevant, and useful for the current task, and then using this knowledge. We cannot expect fine tuning of axioms or manual selection for an agent that is expected to explore an unknown territory. Furthermore, our agent has to be able to select and use knowledge that is not complete or accurate.

In this paper we present an approach to retrieving commonsense knowledge for autonomous decision making, and a complete retrieval framework with several algorithms for mapping languages and selection of knowledge. We consider agents whose formal language is different from that of the CSKB, and can use multiple CSKBs of various expressivity and coverage. Our algorithms translate an agent's knowledge base (AKB) to the language of a CSKB, even when the CSKB is sparse. We show how an agent may use the AKB and knowledge from a CSKB to choose actions.

The usage of knowledge by our agent gives rise to two types of queries that we can ask from a CSKB. These are *region queries*, which find relevant concepts and axioms given a set of concepts that the agent considers *active*, and *path queries*, which find relevant concepts and axioms given the current situation and a goal description. The corresponding queries are processed using a graph that we associate with the CSKB, and using methods from information retrieval [Salton and Buckley, 1988] and automated reasoning. We report on preliminary experimental results of these algorithms for the ConceptNet CSKB.

This is work in progress, and some details are omitted with only little experimental evaluation.

2 Knowledge and Decision Making

Our running example is that of an agent playing an adventure game [Amir and Doyle, 2002]. This problem isolates the commonsense reasoning problem from external challenges like vision and motor control. Far from the arrows and breezy pits of the wumpus world, these adventure games are intended

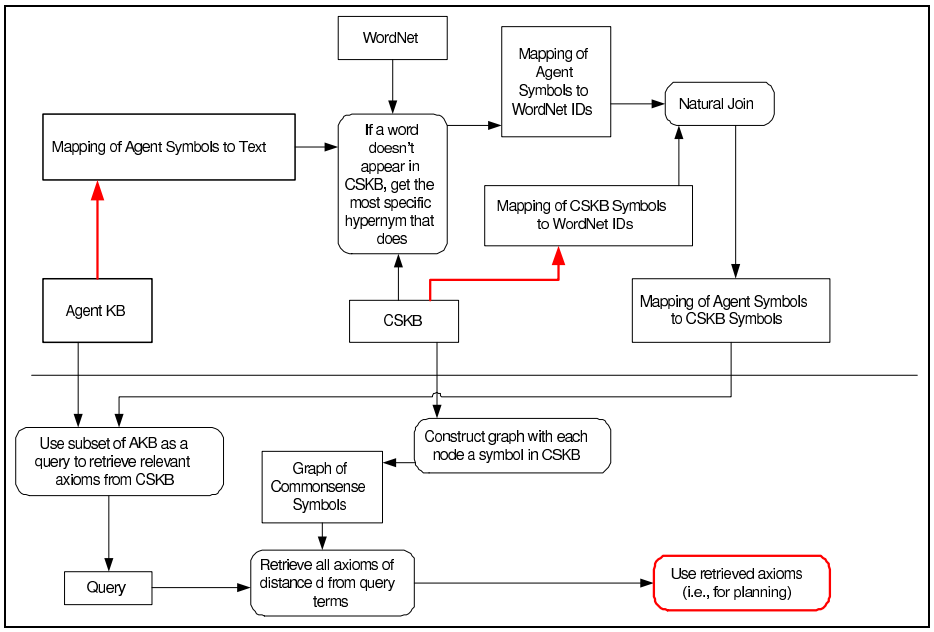

Figure 1: A high-level perspective of the retrieval system

to be abstractions of real-world scenarios. The agent is designed to track its world in a compact belief state, use commonsense to reason about the semantics of its knowledge, learn the effects of actions through observation, and use its knowledge to progress toward some goal. Importantly, the agent is required to learn the action model, which is unknown initially[1].

2.1 Problem Setting

We assume that there is a single agent in the partially observable world, with knowledge about the current state of the world (and its effects of actions) that is represented by a set of logical sentences. We also assume that the agent can observe its environment using its sensors and act upon it. The world conforms to the assumptions of the basic Situation Calculus [McCarthy and Hayes, 1969; Reiter, 2001]: the Markov property holds, actions have explicit preconditions and effects and are deterministic, and there is a set of unique names axioms. Simple adventure games already have these properties, so they are not too restrictive, and they allow us to represent actions using a well-understood language.

The agent is given a goal that is achievable from any state via a finite sequence of executable actions. (In many adventure games, a state satisfying the goal is not specified initially but must be *discovered* through exploration (e.g., by reading a note), but we ignore this case here.) Thus, in such environments the agent can perform some default exploratory behavior until a goal is reached. However, this exploration is infeasible because it may take time that is linear (or worse) in the number of states in our system, which is in $\Omega(2^n)$ for n propositional features in the domain.

Thus, our decision maker is faced with the choice of a action policy A such that executing A in the current state s produces the goal state g. Methods that address this problem to some degree include planning under partial observability [Bertoli and Pistore, 2004] and reinforcement learning in partially observable domains [Even-Dar *et al.*, 2004]. They maintain a belief state and compute a plan or a policy for a given problem domain and goal.

2.2 Using Knowledge

There are four ways in which we wish to use knowledge: (1) control knowledge for planning and search, (2) information about the way actions change the world (eliminate some of the uncertainty about the transition model), (3) better guidance to exploration on the way to the goal (using information gain), and (4) better guidance to acting on the way to the goal (using projection paths).

Control knowledge for planning (e.g., [Bacchus and Kabanza, 2000; Doherty and Kvarnström, 2001]) can be represented as a restriction strategy on the choices that the agent can make. For example, one can state that if our agent is faced by a locked door, then it should not try to unlock the door with a bannana (this may jam the door).

Information about actions' effects and preconditions can help in making our planning problem more feasible, and may allow us to reach islands in our search space. For example, if we know that unlocking the door with the correct key will have the effect that the door is unlocked, then we will try this action before others, to find if the key that we have is the right one, and possibly get into the room on the other side of the door.

Knowledge about the way our actions change (or do not change) the world, and knowledge about the way they are

[1]This paper does not concern learning action models or maintaining a compact belief state; the focus is on knowledge retrieval only.

likely or *unlikely* to change the world is important for exploration. Information gain [Mitchell, 1997] is a measure that is typically used to perform this exploration. It is particularly attractive in adventure and exploration games because there one can use the information gain measure with hill climbing to reach the goal (the goal is achieved exactly when we discover an action that achieves it).

Finally, a chain of plausible, high-level concept landmarks or actions (e.g., a chain of concepts connecting a house with a grocery store) can serve as guidance for both planning and reinforcement learning. Planning techniques can use such chains to guide a rough forward plan search with a subsequent process that corrects the plan where needed (e.g., [anb Bernhard Nebel, 2001]). Reinforcement learning can use such information to re-shape the reward structure by transferring some of the final reward (of reaching the goal) to various positions in the state space [Ng *et al.*, 1999].

2.3 Representation of Agent's Knowledge and Interaction Model

In this paper we assume a logical representation for actions (e.g., in the situation calculus), having met the preconditions for doing so with the previous section's restrictions. As mentioned above, the action model is initially unknown. However, we assume that the game environment provide the agent with a list of relevant actions.

We use the intuition that action $a(\vec{x})$ is relevant if it is executable (with a non-nil effect) in some state s' that is reachable from the initial state. Such a list of relevant actions is usually available to a human agent in an adventure game, either through a help system or through feedback that notifies the player of an unrecognized action. The rest of the action model is initially hidden from the agent.

Ideally, an agent should be able to receive all observations in natural language, as humans do, but converting natural language sentences to logical ones is tantamount to a solution to the general knowledge extraction problem. Thus, we assume that observations contain only a limited natural language component, and we associate this natural language component with a set of logical sentences that represent the observations of the agent. We use the text that is associated with nonlogical symbols in our First-Order Logical language L as semantic information that allows us to connect L with the language of any CSKB.

3 System Overview

Figure 1 shows how the commonsense retrieval system augments an agent's knowledge base with relevant commonsense axioms. The upper and lower halves of the diagram represent the two broad tasks being performed. The first task is to create a mapping from symbols in the AKB to those in the CSKB. The second task is to use a subset of the AKB (now mapped to the language of the CSKB) as a query with which to find useful axioms in the CSKB. To determine relevance as generally as possible given the unstructured nature of some CSKBs, relevance is only calculated based on distance in a graph representation of the CSKB. Every CSKB has ground terms and axioms over those terms, so a simple graph representation has a node for each term and an edge between two nodes iff there is an axiom that mentions both of those terms. The astute reader will note that the algorithms as presented retrieve concepts rather than axioms. These concepts are are intended to be used along with the supplied query concepts as endpoints of a path along which all axioms are retrieved.

3.1 Mapping Agent Language to CSKB Language

To reason with commonsense knowledge, a mapping must be made from the symbols in the agent's language to those in the CSKB. The most important thing to realize when constructing the mapping is that it won't be exact, because no formal ontology exists that can represent all the information in English words (even when these words occur in isolation). Worse, it is not clear that such a formalization is even possible. With that limitation in mind, we propose the baseline system seen in Figure 2.

PROCEDURE MapSymbols(AKB,$WordNetKB$,$CSKB$)
AKB Agent's KB (in FOL), $WordNetKB$ WordNet KB (Ontology and English words), $CSKB$ Commonsense KB (in FOL)

1. Set $Mapping \leftarrow \emptyset$
2. Loop for every constant symbol s_a in AKB:
   (a) Search for s_a in WordNet, putting the resulting set of synsets in $senses$
   (b) If $senses = \emptyset$, then set $Mapping \leftarrow Mapping \cup \{\langle s_a, nil\rangle\}$. Else:
      i. Set $syn \leftarrow argmax_{s \in senses} P(s|AKB)$
      ii. Set s_c to be a CSKB concept (predicate, function, or constant) symbol corresponding to syn's WordNet ID.
      iii. If s_c is non-nil, then set $Mapping \leftarrow Mapping \cup \{\langle s_a, s_c\rangle\}$. Otherwise, loop until $syn = TOP$:
         A. Set $syn \leftarrow hypernym(syn)$, syn's nearest hypernym in WordNet
         B. If $CSKB$ has a concept s_c corresponding to syn's WordNet ID, then set $Mapping \leftarrow Mapping \cup \{\langle s_a, s_c\rangle\}$, and break
3. Return $Mapping$

Figure 2: Algorithm for mapping agent's symbols to CSKB symbols

MapSymbols maps every constant symbol in the AKB to a matching entry in WordNet, based on a textual match only. It performs word sense disambiguation based only on bag-of-words co-occurrence data, which could be obtained from a WordNet sense-tagged corpus. Then it attempts to find a matching concept in the CSKB, requiring an existing mapping from CSKB symbols to WordNet synsets. If a match cannot be found in the CSKB, the next nearest hyponym in WordNet is checked for a match, and so on until the most specific match available is found.

For symbols other than constants, there may be more dimensions to match than simple textual similarity. In CYC,

for instance, functions and predicates take typed arguments, and functions themselves have types. Types and arity represent syntactic restrictions that must be checked before an agent symbol can be mapped to a CSKB symbol. To map functions and predicates using MapSymbols, we can perform unification, recursively matching terms textually or - if the terms are functions - through unification.

3.2 Retrieval Tasks

We want our knowledge retrieval system to work well on each of the CSKBs, but their differences make this goal difficult to achieve. To minimize the effects of these differences, we create a simplified but uniform representation for CSKBs.

The retrieval system is not an algorithm for deciding which axioms are relevant but a system for facilitating the application of such algorithms on different types of CSKBs. Our aim is to be able to retrieve useful information from any of them, but their differences make that a difficult task. To minimize the effects of these differences, we simplify the CSKBs, converting them to a weighted graph. The simple procedure is given in Figure 3. The following sections describe *region queries* and *path queries*, the two retrieval options allowed in the framework.

PROCEDURE RetrieveFromGraph($CSKB$, S, T, q)
$CSKB$, Commonsense KB (in FOL); $S \subseteq CSKB$, a relevant subset of the agent's KB, mapped to $CSKB$ concepts; $T \subseteq CSKB$, a set of concepts appearing in the goal state, mapped to $CSKB$ concepts; $q \in \{region, path\}$, the type of query

1. Set $A \leftarrow \emptyset$, the set of retrieved axioms.
2. Construct a weighted graph, G, from $CSKB$, with a node for every concept and an edge between two nodes iff the corresponding concepts are mentioned in any axioms together. The weight on each edge corresponds to the number of axioms it represents.
3. Set $S' \leftarrow$ the nodes in G corresponding to S
4. Set $T' \leftarrow$ the nodes in G corresponding to T
5. if $query_type = region$ then $A \leftarrow RegionQuery(G, S')$
6. else $A \leftarrow PathQuery(G, S', T')$
7. Return A

Figure 3: Algorithm for retrieving relevant commonsense axioms

Region Query

This type of search is intended to help the retriever find out more about its current knowledge. The intent is for an agent to select a small number of concepts from its KB, and the result should be the set of axioms in the "contextual region" of the given concepts. This region is identified using spreading activation, a technique that has been used successfully in similar applications in which some seed was used to retrieve a collection of related items from a graph .

Conceptually, spreading activation starts with a set of nodes with some *activation weight* and proceeds to activate neighboring nodes recursively over a series of time steps. The activation weight $a_i^{(t)}$ of a node i at time t is $f(\sum_j w_{ij} a_j^{(t-1)})$, where j varies over the neighbors of i, and w_{ij} is the edge weight from j to i. $f()$ is usually a decay function that has the effect of decreasing activation as distance increases from the activation sources. Activation weights are ranked after a halting condition is reached, based either on time or nearness to some asymptotic distribution. Spreading activation is only a general procedure, but the "leaky capacitor" model used in has been analyzed parametrically in and can be used on our network with little modification. $RegionQuery(G, S)$, then, is a straightforward application of the leaky capacitor model, parameterized for a rich, connected network.

Path Query

This type of search is intended only to find paths between two concepts (or sets of concepts). They might represent two regions of the belief state that the agent wants to connect (a key and a locked door, for instance), or one of them might represent the goal, or part of the goal. Since it is assumed that the retrieving agent is trying to find the shortest path to its goal, a goal-directed search returns axioms lying on the shortest paths between the supplied concept sets. The algorithm is given in Figure 4.

PROCEDURE PathQuery(G, S, T)
G, a weighted graph; S, the set of source nodes; T, the set of destination nodes

1. Remove all edges between nodes in S.
2. Create a new node, s.
3. For each edge (p, q) where $p \in S$, create an edge (s, q) of equal weight and remove (p, q). if (s, q) already exists, add to its weight the weight of (p, q).
4. Remove all nodes in S.
5. Repeat this procedure for T, to produce a node t.
6. Assign a score of 0 to each node in the graph.
7. Find the k shortest paths between s and t, where k is a parameter.
8. for each path p_i $(i = 1, ..., k)$
   (a) for each node n_j on p_i (not including s and t)
      i. $n_j.score+ = \frac{1}{length(p_i)}$
9. Return the α (another parameter) nodes with the highest scores

Figure 4: Algorithm for returning nodes on paths between two regions

4 Discovering Interesting Actions in ConceptNet

We have been able to implement a simple version of the retrieval system on ConceptNet first because of that CSKB's simple graph structure and ready availability. This version approximates spreading activation in $RegionQuery$ with ConceptNet's own $GetContext$ function. This function measures two nodes' relatedness as the number of paths between them, weighted to favor shorter paths. As mentioned previously, all axioms on the paths between query concepts and retrieved concepts are returned.

What is the right amount of information? This question has not been answered. Any upper bound is established by performance concerns: inference is ultimately exponential in the size of the underlying knowledge base, so too much information will be useless to any agent. The ideal result of a retrieval is one fact or statement: the one that will determine an agent's next correct action. But the generality that makes common sense useful will generally necessitate returning a broader scope of facts than simply the correct next action. Keeping in mind the hard limits posed by exponential time, this retrieval system must be able to operate on a precise context.

4.1 Concept Expansion

The benefit of having a ranked contextual neighborhood is the same as in any ranked result set: the ability to select the most relevant subset of results. In the examples used while building and testing this system, usually only the top five or ten results have actual real-world relevance to the supplied context. As with any ranking heuristic, there were some anomalies - instances of irrelevant concepts appearing much higher in the ranking than would be reasonably expected.

The most relevant concepts in the list, in no particular order, are "unlock door", "open lock", and "lock". Some other concepts' positions at the top of the list could be argued, but it is unlikely anyone would argue that "metal" should even appear in the top ten. Its appearance here is a likely indicator that this contextual neighborhood is somewhat sparse, meaning there are relatively few relations involving doors, keys, and locks.

Results like this demonstrate the need for normalization in relevance scoring. Intuitively, if one concept is of broad general interest and the database contains many relations referring to it, that concept will be ranked highly in many queries. If normalization had been used in the "door, "key" example, it is unlikely that "metal" would have ranked so highly. But whether normalization will improve ranking in the general case is a question best answered empirically. To perform such a test, there will need to be a set of test cases, each a pair consisting of a query (given as a set of concepts) and a desired result (either a set of relevant results, or a perfect ranking). Any test set will be dependent on the nature of the agent using the system.

4.2 Additions to the ConceptNet Retrieval System

The earlier system was retrieving all concepts and relations in ConceptNet related to the given context. Our current problem definition refines the results we want. Now it is sufficient only to retrieve those actions semantically related to a context. Any more information would be useless because we have no way of knowing how to reason with it. So we add an extra step to the end of the procedure: select the intersection of the concepts retrieved by GetContext and the universe of actions.

The original system to retrieve data from ConceptNet used all the nonlogical symbols from the AKB as arguments to the GetContext function. That approach is fine for small KBs, but for a long-running agent, only a small subset of the symbols will define its present context. We restrict the set of symbols sent to the context query to be only those appearing in the agent's current location. Clearly this only allows objects to be used in the query.

Procedure GetRelevantAction, presented in Figure 5, takes the agent's knowledge base as input and returns a ranked list of interesting actions. It is a specialization of Procedure RetrieveFromGraph from Figure 3.

PROCEDURE GetRelevantAction(AKB,$CSKB$)
AKB Agent's KB (in FOL), $CSKB$ Commonsense KB (in FOL)

1. Let $C \leftarrow null$
2. For every nonlogical symbol s in AKB for which there is a relation $ObjectAtLocation(s, l)$ where l is the agent's current location, add s to C
3. Return $GetContext(C, CSKB) \cap A$, where A is the universe of actions

Figure 5: Getting relevant actions from a commonsense KB

The algorithm does not make use of common sense markup from the previous algorithm that mapped the agent language to common sense.

5 Related Work

Background knowledge is increasingly important for decision making. Work on reinforcement learning [Kaelbling *et al.*, 1996; Andre and Russell, 2000; Ng and Russell, 2000; Ng *et al.*, 1999] uses background knowledge to structure the state space, update the reward function, and approximate the value function. Also, work on planning uses background knowledge to guide the search for a plan [Levesque *et al.*, 1997; Bacchus and Kabanza, 2000; Doherty and Kvarnström, 2001]. Finally, there has been some work on discovering knowledge to aid planning , and also about using nonmonotonic reasoning to speed up reasoning [Ginsberg, 1991]. However, no work known to us has approached the problem of retrieving the right knowledge for a specific task.

The topic of matching symbols between KBs has attracted much attention in recent years. There has been some successes in the case of matching database schemas that have common underlying instances and language [Doan, 2002; Doan *et al.*, 2003], and some investigation was made in the case of more general AI knowledge bases . However, work on

merging KBs and matching ontologies between KBs remains manually driven [Noy *et al.*, 2001].

References

[Amir and Doyle, 2002] Eyal Amir and Patrick Doyle. Adventure games: A challenge for cognitive robotics (full version). AAAI'02 workshop on Cognitive Robotics. Also, available at the author's website (http://www.cs.uiuc.edu/~eyal/papers), 2002.

[anb Bernhard Nebel, 2001] Jörg Hoffmann anb Bernhard Nebel. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[Andre and Russell, 2000] David Andre and Stuart J. Russell. Programmable reinforcement learning agents. In *Proceedings of the 13th Conference on Neural Information Processing Systems (NIPS'00)*, pages 1019–1025. MIT Press, 2000.

[Bacchus and Kabanza, 2000] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1-2):123–191, 2000.

[Bertoli and Pistore, 2004] Piergiorgio Bertoli and Marco Pistore. Planning with extended goals and partial observability. In *Proceedings of the 14th Int'l Conf. on Automated Planning and Scheduling (ICAPS'04)*, 2004.

[Cohen *et al.*, 1998] Paul Cohen, Robert Schrag, Eric Jones, Adam Pease, Albert Lin, Barbara Starr, David Gunning, and Murray Burke. The darpa high-performance knowledge bases project. *AI Magazine*, 19(4):25–49, 1998.

[Doan *et al.*, 2003] AnHai Doan, Pedro Domingo, and Alon Halevi. Learning to match the schemas of databases: A multistrategy approach. *Machine Learning*, 50:279–301, 2003.

[Doan, 2002] AnHai Doan. *Learning to map between structured representations of data*. PhD thesis, University of Washington, Seatle, 2002.

[Doherty and Kvarnström, 2001] Patrick Doherty and Jonas Kvarnström. Talplanner: A temporal logic based planner. *AI Magazine*, 2001. Accepted for publication.

[Even-Dar *et al.*, 2004] Eyal Even-Dar, Sham M. Kakade, and Yishay Mansour. Reinforcement learning in pomdps. In *Proceedings of the 17th Conference on Neural Information Processing Systems (NIPS'04)*, 2004.

[Fikes and Farquhar, 1999] Richard Fikes and Adam Farquhar. Large-scale repositories of highly expressive reusable knowledge. *IEEE Intelligent Systems*, 14(2), 1999.

[Ginsberg, 1991] Matthew L. Ginsberg. The computational value of nonmonotonic reasoning. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 262–268, San Mateo, CA, USA, April 1991. Morgan Kaufmann Publishers.

[Kaelbling *et al.*, 1996] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Lenat, 1995] Douglas B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[Levesque *et al.*, 1997] H.J. Levesque, R. Reiter, Y. Lesprance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.

[McCarthy and Hayes, 1969] John McCarthy and Patrick J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[Miller, 1995] George A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[Mitchell, 1997] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[Ng and Russell, 2000] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, 2000.

[Ng *et al.*, 1999] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: theory and application to reward shaping. In *Proc. 16th International Conf. on Machine Learning*, pages 278–287. Morgan Kaufmann, 1999.

[Niles and Pease, 2001] Ian Niles and Adam Pease. Towards a standard upper ontology. In *International conference on Formal Ontology in Information Systems*, pages 2–9. ACM Press, 2001.

[Noy *et al.*, 2001] Natalya F. Noy, Michael Sintek, Stefan Decker, Monica Crubzy, Ray W. Fergerson, and Mark A. Musen. Creating semantic web contents with protege-2000. *IEEE Intelligent Systems special issue on Semantic Web Teconology*, 16(2):60–71, 2001.

[Reiter, 2001] Raymod Reiter. *Knowledge In Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press, 2001.

[Salton and Buckley, 1988] Gerard Salton and Chris Buckley. On the use of spreading activation methods in automatic information retrieval. In *SIGIR '88: Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 147–160. ACM Press, 1988.

[Singh *et al.*, 2002] Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. Open mind common sense: Knowledge acquisition from the general public. In *Proceedings of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems*, LNCS. Springer-Verlag, 2002.