Graph-Based Reasoning and Reinforcement Learning for Improving Q/A Performance in Large Knowledge-Based Systems

Abhishek Sharma and Kenneth D. Forbus
Qualitative Reasoning Group, Northwestern University
Evanston, IL 60208, USA
{a-sharma, forbus}@northwestern.edu

Abstract
Learning to plausibly reason with minimal user intervention could significantly improve knowledge acquisition. We describe how to integrate graph-based heuristic generalization, higher-order knowledge, and reinforcement learning to learn to produce plausible inferences with only small amounts of user training. Experiments on ResearchCyc KB contents show significant improvement in Q/A performance with high accuracy.

Introduction and Motivation
Question answering is an important application for AI systems. Inference based Q/A systems have an advantage over information extraction systems because they can reason and can provide explanations for their answers. However, knowledge base construction is difficult and tedious. One solution to this problem is to exploit whatever learning strategies are available to populate a knowledge base and use feedback to learn to reason. For example, learning by reading systems [Matuzek et al 2005; Forbus et al 2007] are currently better at providing ground facts than correct, fully quantified logical axioms. Being able to learn plausible patterns of inference over ground facts could significantly improve the scalability of knowledge base construction. Such plausible inferences also help solve a second problem in reasoning with large KBs: Typically the set of logically quantified axioms has woefully low coverage. At this point it is far from clear that human reasoning rests on massive sets of correct logically quantified axioms, and some interesting evidence against it (e.g., the Wason task). While prior work has explored plausible inference schemes (e.g. [Collins 1978]) we are also concerned with learning to do such reasoning. Since human attention is a scarce resource, our goal is to minimize the amount of feedback users must provide.

This paper shows how to integrate graph search, higher-order knowledge representation, and reinforcement learning to learn reliable patterns of plausible reasoning from ground facts. Given a fully ground query, we show how to incrementally search the facts which mention the entities in it guided by a set of plausible inference patterns (PIPs). PIPs are similar to knowledge patterns [Clark et al 2000], but are expressed in terms of higher-order concepts in the knowledge base, specifically predicate type information. Since the number of predicate types is much smaller than the number of predicates, this greatly reduces the size of search space. We show that the quality of inference chains of PIPs can be learned by reinforcement learning.

We begin by discussing other related work. We then discuss the idea of PIPs and how they are defined in terms of predicate types. How reinforcement learning is used to learn the quality of answers is discussed next. We describe results and conclude in the final section.

Related Work
Researchers from the fields of Information Retrieval, Natural Language Processing, Databases and Logical Inference have contributed to the advancement of question answering technologies [Brill et al 2002] [Prager et al 2004]. Overviews of question answering techniques can be found in [Belduccini et al 2008, Molla and Vicedo 2007]. A comparison of challenge problems and different approaches has been discussed in a recent IBM report [Ferrucci et al 2009]. Learning Bayesian networks for WordNet relations for QA systems [Ravichandran and Hovy 2002] and surface patterns from natural language text [Molla 2006, Grois and Wilkins 2005] have been discussed. Our work is different in that we are trying to improve the performance of a plausible inference based Q/A system by learning to reason. Other learning to reason

Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
frameworks [Khardon 1999] have been explored. However, their efficacy for improving Q/A performance is not known. Reinforcement learning has been used for learning control rules for guiding inference in ResearchCyc KB [Taylor et al 2007]. To the best of our knowledge, there has not been prior work which develops a method for providing plausible explanations for queries (without using logically quantified axioms) with a learning framework.

**Representation and Reasoning**

We use conventions from Cyc in this paper since that is the major source of knowledge base contents used in our experiments. We summarize the key conventions here [Matuszek et al 2006]. Cyc represents concepts as collections. Each collection is a kind or type of thing whose instances share a certain property, attribute, or feature. For example, Cat is the collection of all and only cats. Collections are arranged hierarchically by the genus relation. (genls <sub> <super>) means that anything that is an instance of <sub> is also an instance of <super>. For example, (genls Dog Mammal) holds. Moreover, (isa <thing> <collection>) means that <thing> is an instance of collection <collection>. Predicates are also arranged in hierarchies. In Cyc terminology, (genlPreds <s> <g>) means that <g> is a generalization of <s>. For example, (genlPreds touches near) means that touching something implies being near to it. The set of genlPreds statements, like the genls statements, forms a lattice. In Cyc terminology, (argIsa <relation> <n> <col>) means that to be semantically well-formed, anything given as the <n>th argument to <relation> must be an instance of <col>. That is, ( <relation> … <arg-n> … ) is semantically well-formed only if (isa <arg-n> <col>) holds. For example, (argIsa mother 1 Animal) holds. We use Cyc’s predicate type hierarchy extensively. PredicateType is a collection of collections and each instance of PredicateType is a collection of predicates. The predicates in a given predicate category represented in the KB are typically those sharing some common feature(s) considered significant enough that the collection of all such predicates is useful to reify. Instances of PredicateType include TemporalPartPredicate, SpatialPredicate, Goals-Attitude-Topic, PhysicalPartPredicate and PropositionalAttitudeSlot. ResearchCyc can be viewed either as incorporating higher-order logic or as a first-order knowledge base with extensive reification. We take the latter perspective here.

The task of answering questions without using logically quantified axioms is difficult because it involves finding arbitrary relations between predicates which could explain the query. Therefore, we have taken the simpler approach of building a small sub-graph of relations around the entities in the query and then assessing the quality of inference chains between them. This intuition is similar to connection graphs [Faloutsos et al 2004] and relational pathfinding, where the domain is viewed as a (possibly infinite) graph of constants linked by the relations which hold between the constants [Richards & Mooney 1992]. Since prior knowledge is important for biasing learning, we leverage existing axioms in the KB to create plausible inference patterns (PIPs) which are used to keep only more likely inference chains. These PIPs are created by replacing predicates in axioms by their predicate types. PIPs are accepted if they are generated by more than N axioms. (In this work, N=5). We turn to a concrete example for illustration.

Let us assume that the system has been asked to provide a plausible inference for the query (acquaintedWith BillClinton HillaryClinton). A small section of the KB relevant for answering this query is shown in Figure 1. In the first step of the algorithm shown in Figure 3, \(e1\) is set to BillClinton and \(e2\) is set to HillaryClinton. For simplicity, let us assume that we have just one PIP:

\[
\text{FamilyRelationSlot}(?x,?y) \text{ AND } \text{FamilyRelationSlot}(?y,?z) \rightarrow \text{PersonalAssociationPredicate(?x,?z)} \quad [\text{PIP1}]
\]

This pattern represents the knowledge that two predicates of type FamilyRelationSlot can plausibly combine to infer assertions involving personal associations. This representation has been chosen because we believe that predicate types like SubEventPredicate, PhysicalPartPredicate and CausaltyPredicate provide a meaningful level of abstraction for identifying plausible inference patterns. For instance, all predicates of type SubEventPredicate can be used for proving eventPartiallyOcursAt queries. Similarly, all predicates of type PhysicalPartPredicate are relevant for proving objectFoundInLocation queries.

---

1. We use knowledge extracted from the ResearchCyc knowledge base with our own reasoning system, instead of using Cyecorp’s reasoning system.


---

3. Some examples of SubEventPredicate predicates are firstSubEvents, cotemporalSubEvents, finalSubEvents etc.

4. Some examples of PhysicalPartPredicate are physicalParts, internalParts, northernRegion etc.
Therefore, learning knowledge in terms of predicate types is easier and more natural. The relative tractability of this formulation can also be seen by noting the difference between the sizes of search spaces. Learning to distinguish between correct and incorrect derivations of length k involves searching in a space of size $N^k$, where N is the size of vocabulary. In our KB, the number of predicates is 24 times larger than the number of predicate types. Therefore, learning PIPs in terms of predicate types is significantly easier. The algorithm FPEQ (see Figure 3) constructs a graph around the entities mentioned in the query and returns explanations which plausibly entail the query. Steps 1-3 perform initialization, with the path search being handled in Step 4. It maintains a list of partial paths which are extended by retrieving facts involving the frontier entities. The algorithm terminates when all paths which match the antecedents of the available PIPs are found. In the previous example, acquaintedWith is a PersonalAssociationPredicate predicate; therefore the pattern PIP1 is relevant for this query. The algorithm shown in Figure 3 finds paths by successively expanding nodes at the frontier and keeps the partial paths in the list Paths. In step 3 of the algorithm, a new path p is created. Here, p.starting-entity, p.target-entity and p.remaining are set to BillClinton, HillaryClinton and ChelseaClinton. Another example of a PIP that would not be selected because no PIPs use predicates of type ProperNamePredicate is [FamilyRelationSlot(?x,?y), FamilyRelationSlot(?y,?z)].}

---

5 The algorithm shown in Figure 3 has been simplified for clarity. In particular, the algorithm keeps track of bindings of variables and paths can only be extended when bindings are consistent.
The FPEQ algorithm uses the predicate type hierarchy. Our inference scheme also simplifies inference by condensing inference chains. For example, wife is a PersonalAssociationPredicate; therefore the inference from wife to acquaintedWith is a one-step process. On the other hand, using the normal predicate type hierarchy involves multi-step inferences. For example, the inference chain from wife to acquaintedWith requires following axioms:

\[\text{acquaintedWith}(\text{BillClinton}, ?x) \rightarrow \text{mutualAcquaintances}(?x, ?y) \]  
\[\text{mutualAcquaintances}(?x, ?y) \rightarrow \text{mate}(?x, ?y) \]  
\[\text{mate}(?x, ?y) \rightarrow \text{spouse}(?x, ?y) \]  
\[\text{spouse}(?x, ?y) \rightarrow \text{wife}(?x, ?y) \]

As discussed above, the number of predicate types is less than the number of predicates. Therefore, the predicate type hierarchy maps the predicates to a smaller space. This phenomenon speeds up the search because the average path length between two nodes in this smaller space is less than what we encounter in a typical predicate hierarchy. This plays an important role in improving inference in resource constrained Q/A systems.

The FPEQ algorithm can be easily extended to handle queries with variables. This would entail checking that the node at the search frontier satisfies the argument constraint of the predicate. For example, let us consider the query (acquaintedWith BillClinton ?x). When the search process reaches the node labeled HillaryClinton, it would notice that the antecedents of the PIP have appropriate bindings and the entity HillaryClinton satisfies the argument constraint of the second argument position of acquaintedWith. Such inference chains will be included in the solutions.

**Learning to Reason**

Many learning systems find the correct level of generalization by trial-and-error. Our approach gets initial plausible inference patterns by replacing predicates in axioms by their predicate types. These generalizations certainly increase the deductive closure but can lead to erroneous answers. For example, the pattern PIP2 shown above would lead to an incorrect answer if we use bordersOn as an instance of SpatialPredicate in the consequent. Therefore, our aim is to design a system which could learn to identify incorrect search steps from minimal user feedback without sacrificing the gains obtained from generalization. This task is complicated by the fact that a typical user may not be able to identify the incorrect search choice(s) made during a multistep reasoning process. The learner should be able to work with delayed feedback about the correctness of the final answer and learn to find plausible inferences for queries. We believe that reinforcement learning is a reasonable method for solving this problem. Formally, the model consists of (a) a discrete set of states, S; (b) a discrete set of agent actions, A; (c) a reward function R: S x A → \{−1, 1, 0\} and (d) a state transition function T: S x A → \[\Pi(S), \text{where a member of } \Pi(S) \text{ is a probability distribution over the set } S \text{ [Kaelbling et al 1996].} \]

**Value Iteration Algorithm**

1. Initialize V(s) arbitrarily  
2. Repeat step 3 until policy good enough  
3. loop for s ∈ S  
   a. Loop for a ∈ A  
      1. Q(s, a) ← R(s, a) + \[γ \sum \text{T}(s, a, s') \] V(s')  
      b. V(s) ← max \[\text{a} \text{ } Q(s, a)\]

**Figure 4: Value Iteration algorithm**

(From Kaelbling et al 1996)

In this context, a *state* is the list of predicate types already used during the partially complete search process. At each step of the reasoning process, the inference engine has choice points at which it chooses or rejects different alternatives. It has to assess how useful a particular predicate type is for completing the proof given the predicate types already chosen in the current search path. The *actions* are the selection of a particular predicate type for completing the partial assignment of variables. The *value function* (or V(s)) is the inference engine’s current mapping from the set of possible states to its estimates of the long-term reward to be expected after visiting a state and continuing the search with the same policy. Q(s, a) represents the value of taking the action a in state s. We use the value iteration algorithm [Kaelbling et al 1996] for learning the plausibility of search paths.

For example, \[V(\{\text{Group-Topic}\})\] for proving objectFoundInLocation would represent the value of starting with a predicate of type Group-Topic while finding a solution of an objectFoundInLocation-query. Similarly, \[Q(\{\text{Group-Topic}\}, \text{Betweenness-Spatial})\] represents the quality of choosing a Betweenness-Spatial predicate when the partial search path has already chosen a Group-Topic predicate. We use a delayed reward model with user-provided rewards of +1 and -1 for
correct and incorrect answers respectively. Rewards are
generalized via the predicate hierarchy. For instance,
reward statements for spatiallySubsumes are also used
for computing V(s) values for its generalizations like
spatiallyIntersects. The computational complexity of
each iteration of the algorithm is \(O(|A||S|^2)\). In our
experiments, the policy converged in less than ten
iterations.

Experimental Method and Results
To show that these ideas generate more answers compared
to traditional deductive reasoning, we describe a set of
experiments. Five sets of questions were selected based on
the availability of ground facts in KB and their relevance in
learning by reading [Forbus et al 2007]. These questions
templates were: (1) Where did <Event> occur? (2) Who is
affected by <Event>? (3) Where is <SpatialThing>? (4)
Who performed the <Event>? and (5) Where is
GeographicalRegion? Each question template expands
to a disjunction of formal queries. The parameters in the
question template (e.g., <Event>) indicate the kind of
thing for which the question makes sense. Queries were
generated by randomly selecting facts for these questions
from the KB.

For a baseline comparison, we included all axioms for
these predicates and their subgoals through depth 3. We
used a simple backchainer working on a LTMS based
inference engine [Forbus & de Kleer, 1993]. The depth of
backchaining was limited to three and each query was
timed out after three minutes. All experiments were done
on a 3.2 GHz Pentium Xeon processor with 3GB of RAM.
25% of the queries were used as the training set for
learning the V(s) values. Answers whose V(s) values were
more than a threshold were accepted.

Table 1 compares the performance of FPEQ algorithm
and reinforcement learning against the baseline for the test
set (i.e. remaining 75% queries). Column T is the total
number of queries, with AW being the number that could
be answered given the KB contents, as determined by hand
inspection. The columns P and R indicate precision, and
recall, respectively. The user assessed 334 unique answers
(from the training set) and the feedback was used for
learning the V(s) values. The accuracy of answers provided
by FPEQ algorithm was 73%. We then removed answers
whose V(s) values were below the threshold. The total
number of new answers at this stage is 1010 and the
accuracy has improved from 73% to 94%. The FPEQ
algorithm mainly reduces false negatives whereas
reinforcement learning reduces false positives. Together,
they provide a factor of 2.2 improvements (i.e. 120%
improvement) over the baseline with an average accuracy
of 94%.

It is clear from these results that the ResearchCyc KB
contents are not uniformly distributed and different regions
have different densities of ground facts. Moreover, some
questions are easier to answer than others. For example,
the accuracy for Expt. No. 5 is significantly better than the
accuracy for Expt. No. 3. We believe that this is due to the
fact that it was possible to generate answers for queries
involved in Experiment 5 from simple reasoning on a tree-
like hierarchy. By contrast, queries involved in Experiment
3 needed more general inference.

As mentioned above, 6% of the derived answers were
incorrect. Moreover, the last column in Table 1 shows that
some answers are still not being generated by the algorithm
proposed here. Therefore, we would like to know the types
of failures associated with these missed and incorrect
answers. Knowledge of the causes of such failures would
help the researchers prioritize their research goals. The
second column of Table 2 (TC) shows the total number of
corrective actions needed to obtain all answers correctly. It
is basically the sum of false positives and false negatives
for the algorithm FPEQ. The other columns of Table 2
show our by-hand analysis of what additional learning
strategies would be required to improve performance
further. We have found that randomly chosen training set is
imbalanced and leads to redundancy. We believe that a
training set which would represent all sections of the KB
without redundancy would be smaller and lead to better
results. The third column (labeled A) in Table 2 shows the
number of problems which would be solved by a better
training set. In some cases, we found that we need to
augment the graph representation so that it could handle
functions (e.g. (industryFacilities (IndustryOfRegionFn OilIndustry SaudiArabia) Iraq-SaudiArabiaPipeline)), and quantified assertions.
The number of such cases is shown in the column labeled
B. The column labeled C shows cases when additional
knowledge would have helped. Cases where a PIP needs to
be replaced by a more specific pattern in terms of existing
predicate types are indicated by D, and cases where a new
predicate type between existing predicate types would
improve matters are indicated by E. Note that the amount
of training data is small compared to the relative
improvement for each experiment.

<table>
<thead>
<tr>
<th>Exp. No.</th>
<th>Query sets</th>
<th>T</th>
<th>AW</th>
<th>P</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Baseline</td>
<td>833</td>
<td>412</td>
<td>1.00</td>
<td>0.51</td>
</tr>
<tr>
<td></td>
<td>FPEQ</td>
<td>833</td>
<td>412</td>
<td>0.95</td>
<td>0.87</td>
</tr>
<tr>
<td>2</td>
<td>Baseline</td>
<td>200</td>
<td>61</td>
<td>1.00</td>
<td>0.42</td>
</tr>
<tr>
<td></td>
<td>FPEQ</td>
<td>200</td>
<td>61</td>
<td>0.92</td>
<td>0.77</td>
</tr>
<tr>
<td>3</td>
<td>Baseline</td>
<td>1834</td>
<td>433</td>
<td>1.00</td>
<td>0.32</td>
</tr>
<tr>
<td></td>
<td>FPEQ</td>
<td>1834</td>
<td>433</td>
<td>0.92</td>
<td>0.88</td>
</tr>
<tr>
<td>4</td>
<td>Baseline</td>
<td>953</td>
<td>226</td>
<td>1.00</td>
<td>0.34</td>
</tr>
<tr>
<td></td>
<td>FPEQ</td>
<td>953</td>
<td>226</td>
<td>0.93</td>
<td>0.93</td>
</tr>
<tr>
<td>5</td>
<td>Baseline</td>
<td>1309</td>
<td>724</td>
<td>1.00</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>FPEQ</td>
<td>1309</td>
<td>724</td>
<td>0.97</td>
<td>0.94</td>
</tr>
</tbody>
</table>

Table 1: Summary of Inference Results. Experiment
numbers are the same as query numbers.
Conclusion

Learning to reason, while minimizing user intervention, is an important problem. We have shown how plausible inference patterns, expressed in terms of higher-order knowledge and learned via reinforcement learning, can be used to reason with reasonable accuracy. The use of predicate types for representing PIPs leads to a succinct, easily learnable and tractable representation. The FPEQ algorithm mainly reduces false negatives whereas reinforcement learning reduces false positives. By integrating them, we get a 120% improvement over the baseline with an average accuracy of 94%.

While these experiments used the contents of ResearchCyc, we believe they would be applicable to any large-scale KB whose predicate types were classified sensibly. Our technique is especially suitable for knowledge capture because it exploits ground facts, which are much easier to gather than logically quantified facts. We believe that this technique can be used to help bootstrap intelligent systems and reduce the dependence on hand-crafted axioms.

Our results suggest three lines of future work. First, we found that a randomly chosen training set does not represent all regions of the KB adequately. Finding these gaps in coverage could be used to suggest new learning goals for learning by reading and other forms of knowledge capture. Second, being able to refine plausible inference patterns to use more specific predicate types would improve accuracy and coverage. Finally, plausible inference patterns could be used as an intermediate stage for postulating new logically quantified statements, perhaps by using a technique like relational reinforcement learning approach [Dzeroski et al. 2001] to carry out the refinements.

Acknowledgements

This research was funded by the Intelligent Systems program of the Office of Naval Research.

References:


