Automatic Extraction of Efficient Axiom Sets from Large Knowledge Bases

Abhishek Sharma[1] Kenneth D. Forbus[2]

[1]Cycorp, Inc. 7718 Wood Hollow Drive, Suite 250, Austin, TX 78731

[2]Northwestern University, 2133 Sheridan Road, Evanston, IL 60208

abhishek@cyc.com, forbus@northwestern.edu

Abstract

Efficient reasoning in large knowledge bases is an important problem for AI systems. Hand-optimization of reasoning becomes impractical as KBs grow, and impossible as knowledge is automatically added via knowledge capture or machine learning. This paper describes a method for automatic extraction of axioms for efficient inference over large knowledge bases, given a set of query types and information about the types of facts in the KB currently as well as what might be learned. We use the highly right skewed distribution of predicate connectivity in large knowledge bases to prune intractable regions of the search space. We show the efficacy of these techniques via experiments using queries from a learning by reading system. Results show that these methods lead to an order of magnitude improvement in time with minimal loss in coverage.

Introduction and Motivation

Deductive reasoning is an important component of many AI systems. Efficient reasoning systems can be built today only for fixed, small-to-medium sized knowledge bases and by careful hand-tuning. There are two reasons to seek more general solutions. First, hand-tuning does not scale as the size of the knowledge base grows. For example, queries that fail in large knowledge bases frequently take hours to fail.[1] There is still no evidence that general-purpose reasoning in such knowledge bases can regularly be performed in order of a few minutes per query. The second problem is that knowledge bases are no longer being built entirely by hand. Advances in machine reading (cf. [Etzioni *et al* 2005]) provide the opportunity to automatically construct large knowledge bases, but this is useless if we cannot reason with them effectively.

[1]cf. www.projecthalo.com/content/docs/

It is well-known that knowledge representation choices play a crucial role in determining the hardness of problems. Work in SAT solving is moving towards understanding the structure of problem and using it for the design of better heuristics. Logical knowledge bases, though structured, are highly complex networks of concepts. Concepts are connected through different types of relationships, defining intricate networks. Understanding KB structure is fundamental to many important knowledge representation and reasoning problems. These include evaluating inference engines and assessing the effectiveness of heuristics and algorithms.

Here we exploit properties of the structure of a large knowledge base to improve inference. We describe the *ExtractAxioms* algorithm which identifies useful set of axioms by pruning knowledge-poor regions of the KB. We propose that for understanding reasoning performance, the search space represented by the axioms in the knowledge bases should be seen as a graph where the nodes are predicates mentioned in axioms, and the edges connect the predicates mentioned in the antecedents to the predicate of the consequent. In such a network, most nodes have a few neighbors, whereas a small number of nodes have very high degree. We improve the performance of the *ExtractAxioms* algorithm by using these topological properties of predicate connectivity. We use the distribution of ground facts and what might be learned to identify difficult and less useful regions of the KB, and use that information to prune inefficient axioms, which improves performance. The maximum out-degree of nodes in the search graph is a key parameter for controlling the navigability of search spaces. Networks which have high degree nodes, called *hubs*, are inherently unsuitable for focused search. We study the family of search spaces, from disconnected to scale-free, by varying this parameter and show that it helps in identifying efficient sets of axioms.

This paper is organized as follows. We start by discussing relevant previous work. Then we describe our

Abhishek Sharma and Kenneth D. Forbus. (2013). Automatic Extraction of Efficient Axiom Sets from Large Knowledge Bases. *Proceedings of AAAI: Twenty-Seventh Conference on Artificial Intelligence,* Bellevue, WA

motivating tasks and overall approach. Our extraction algorithm, KB analysis, and heuristics for improving efficiency are discussed next. We conclude by discussing our experimental results, and plans for future work.

Related Work

Research in computational complexity and knowledge compilation [Selman *et al* 1996] has shown that Horn clauses provide a good trade-off between expressiveness and tractability. In SAT solving, fixed clause models of hardness, where the ratio of clauses to variables is considered to determine the hardness of the problem [Mitchell *et al* 1992] have received attention. Recently, heavy tails in randomized search costs have been acknowledged as a serious problem [Gomes *et al* 2004]. Non-uniform distribution of search-costs points towards the fact that not all variables are equally important. Such behavior has frequently been explained in terms of *backdoors* and *backbones.* The idea is that different groups of variables in a problem encoding often play quite distinct roles. We identify similar structure for logical KBs. Our work is complementary to work on variable ordering strategies, removal of redundant constraints and identifying backtrack free graphs in CSP because we propose heuristics for simplifying the problem structure and quickly identifying where answers could be. Any inference engine should be able to benefit from them. Our work is closer to [Walsh 1999] who showed that graphs of real world problems aren't uniform but have a 'small-world' structure. To the best of our knowledge, there hasn't been any work in the AI community which has studied the correlation between network structure and time/performance tradeoffs in deductive reasoning.

Motivating Tasks and Approach

Our work is motivated by performing reasoning within large-scale learning systems. In our particular case, the system is designed to learn by reading simplified texts [Forbus *et al* 2007]. The starting endowment for the system is drawn from the ResearchCyc KB, with new material added by a natural language system, which uses a Direct Memory Access Parser [Martin *et al* 1986]. The Question Answering module of this system uses the background KB plus knowledge gained by reading to answer questions, as a means of checking the accuracy of what has been read. The overall system is designed to be domain-independent. The texts so far have been about world history, particularly the Middle East, including its geography, history, and information about current events. The current corpus consists of 62 stories (956 sentences). Given this initial focus, we developed a set of *parameterized question templates* [Cohen *et al*, 1998] for testing the system's knowledge. These templates are: (1) Who is <*Person*>?, (2) Where did <*Event*> occur?, (3) Where might <*Person*> be?, (4) What are the goals of <*Person*>?, (5) What are the consequences of <*Event*>?, (6) When did <*Event*> occur?, (7) Who is involved in <*Event*>?, (8) Who is acquainted with (or knows) <*Person*>? (9) Why did <*Event*> occur?, (10) Where is <*SpatialThing*>? In each template, the parameter (e.g., <*Person*>) indicates the kind of thing for which the question makes sense. For example, one of the queries for question 8 where <*Person*> was given as BillClinton, would be (acquaintedWith BillClinton ?x). Each template expands into a disjunction of formal queries.

For many AI systems, reasoning directly with quantified knowledge (i.e., at least first order) is essential. Brute-force techniques like pre-computing all ground facts (propositionalization) are infeasible. First, they lead to combinatorial explosions and hence do not scale for large knowledge bases. Second, the propositionalization of axioms involving logical terms can lead to infinite sets of statements. Third, propositionalization assumes that the set of facts is static, i.e. that a set of facts is identified once, in advance, and never changes. This does not match the needs of many tasks, including systems that learn.

Our approach to tractable inference is to restrict backchaining to small sets of axioms, automatically extracted from the knowledge base, that are optimized for particular tasks. Each such axiom set, called a *chainer*, corresponds to a single partition in the sense of [Amir *et al* 2005]. Axioms in chainers are restricted to Horn clauses. Using chainers sacrifices completeness for efficiency. In most applications, it is better to fail quickly and try another approach than to pursue a query for hours or days and then fail.

Although large KBs like ResearchCyc contain non-Horn axioms, we limit our attention to Horn axioms here. This problem is still difficult: the problem of determining whether a set of first-order Horn clauses entails an atom is undecidable. As usual, finding reasonable coverage/efficiency tradeoffs as the KB grows is the issue. In the next section, we show how to construct chainers that provide efficiency with little loss of coverage.

Abhishek Sharma and Kenneth D. Forbus. (2013). Automatic Extraction of Efficient Axiom Sets from Large Knowledge Bases.
Proceedings of AAAI: Twenty-Seventh Conference on Artificial Intelligence, Bellevue, WA

Extracting Efficient Sets of Axioms

We have observed that ground facts are not uniformly distributed across predicates. Therefore while searching, we should focus on regions of the search space that (i) are rich in ground facts or (ii) involve facts that can be produced by external systems (e.g., machine learning, learning by reading, or knowledge capture systems). In this algorithm, we represent the predicates used in statements that can be produced by external systems by the set *LearnablePredicates.* If a predicate P belongs to the set *LearnablePredicates* then *ExtractAxioms* would include axioms with P in the antecedent even if it is not currently very frequent in the KB. We assume that the set of *LearnablePredicates* can be constructed by examining the structure of the system that is producing facts. We include all predicates produced by learning systems in *LearnablePredicates* because estimates of the distribution of statements produced may not be known in advance. We focus on a single query predicate, since constructing a chainer for a set of queries can be done by taking the union of axioms generated for each predicate. The essence of the algorithm is a backward sweep from the query predicate through Horn clauses extracted from KB axioms involving that predicate. A depth cutoff is used as a parameter for adjusting coverage versus efficiency for a chainer. Here, *KnownFacts*(p) represents the number of ground facts about the predicate *p*. We define *InferredFacts*(p) as the sum of ground facts of all nodes below *p* in the genlPreds[2] hierarchy. Formally it is $\sum_X$ *KnownFacts*(x), where X is the set of predicates reachable from p via genlPreds links pointed downward. We define *AllFacts(p)* as *KnownFacts(p) + InferredFacts(p)*, i.e., the total number of times a predicate *p* could be proved via ground facts and genlPreds inference. In step 2 of Figure 1, we create an AND/OR backward search graph from Horn clauses that can conclude P. We introduce new nodes to transform the graph into an equivalent structure such that (a) each node is either an AND or an OR node, and (b) Each node has at most one parent[3]. For each node *p*, *Children(p)* represents the children of *p* in the search graph. We would like to include all paths from knowledge-rich regions of the KB to the root. To achieve this, we perform a topological sort of the nodes (step 4) and begin from the leaves. A predicate is chosen if it is in *LearnablePredicates* or if the number of ground statements using it is higher than a given threshold. Since this is a backward search graph, we include the parents of selected nodes (steps 6.b.ii and 6.b.iii). In steps 6.a.ii and 6.b.iv, we prefer those regions of KB which are rich in ground facts. The set *SelectedNodes* represents the predicates which can be frequently proved. In step 7, a Horn clause is included in *SelectedRules* if all predicates in its antecedents are in *SelectedNodes.* The complexity of ExtractAxioms is quite reasonable. Let K and N be the set of axioms and predicates respectively. Moreover, let E be the set of edges of the graph. Then the complexity of computing *InferredFacts(p)* is $O(|N|^2)$. Step 2 requires $O(|N|.|K|)$ time. Topological sort requires $O(|N|+|E|)$ which is $O(|N|^2)$. Step 6 and 7 are $O(|N|^2)$ and $O(|N|.|K|)$ respectively. Therefore, the complexity of the pre-processing step is $O(|N|^2)+O(|N|.|K|)$. Since the axiom extraction process occurs off-line, and infrequently compared to the number of times that the axioms are used, this is a very reasonable cost. Next we use the structure of the KB to further improve the chainer's performance.

Algorithm **ExtractAxioms**:

Input: (a) *pred*: A predicate (b) *depth*: depth cutoff, typically 5 (c) *LearnedPredicates*: A set of predicates which can be generated by the input to the reasoning system. (d) A constant α set to 0.001

Output: A set of axioms, *SelectedRules*, for proving the predicate *pred*.

1. *SelectedPredicates* ←Ø, *SelectedRules*←Ø
2. Make a backward search graph, T, until depth = *depth*, by converting axioms mentioning *pred* into Horn clauses, and recursing on their antecedents.
3. *Threshold*← $\alpha^* \sum_X$ *KnownFacts*(x) where X is the set of all predicates in the graph.
4. Convert T to a queue, by performing a topological sort to order the nodes.
5. Repeat step 6 until T is empty.
6. Pop an element y, from the ordered list T
   a. if Children(y) is empty then include y in *SelectedNodes* if:
      i. y ε *LearnablePredicates* or,
      ii. *AllFacts*(y) >*Threshold*
   b. else if Children(y) is non-empty then include y in *SelectedNodes* if:
      i. y ε *LearnablePredicates* or,
      ii. y is an OR node and at least one element of Children(y) is in *SelectedNodes* or,
      iii. y is an AND node and Children(y) is a subset of *SelecteNodes* or,
      iv. *AllFacts*(y) >*Threshold*
7. *SelectedRules*← {r | r is a Horn clause in KB and all the predicates in its antecedents are in *SelectedNodes*.}
8. Return *SelectedRules*

Figure 1: Algorithm used to extract axioms from KB

Knowledge Bases as Networks

Next we identify some heuristics based on topology and distributions for detecting rules that are likely to fail. It is useful to think about large knowledge bases as networks, where the nodes are predicates and links exist from predicates in the consequent of an axiom to those in the antecedent. For concreteness, this analysis uses the

[2]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.

[3]These changes simplify the description of algorithm *ExtractAxioms.*

Abhishek Sharma and Kenneth D. Forbus. (2013). Automatic Extraction of Efficient Axiom Sets from Large Knowledge Bases. *Proceedings of AAAI: Twenty-Seventh Conference on Artificial Intelligence*, Bellevue, WA

ResearchCyc KB, but we suspect that similar properties will hold for any large, general-purpose KB. For tractability, we focus on 4,864 concepts that are involved in the kinds of queries described earlier. Given our notion of connectivity, the degree distribution of nodes is shown in Figure 2. It is clear that most nodes have very few neighbors[4], whereas a small number of nodes have significantly high degree[5]. This distribution is highly right skewed and resembles a power law distribution[6]. Numerous studies have shown that such networks are ubiquitous in natural phenomenon. In fact, it has been found that many networks, including the world-wide web, a cell's metabolic systems, and Hollywood actors are dominated by a small number of nodes that are connected to many others [Mitchell 2006]. This distribution suggests that the knowledge in ResearchCyc uses a small set of predicates heavily. Our ability to infer useful facts hinges on them. If these predicates are known, inference is surprisingly easy. On the other hand, inferring them can be difficult. Such non-uniform distributions are amenable to targeted intervention or perturbation, which we can exploit to improve inference.

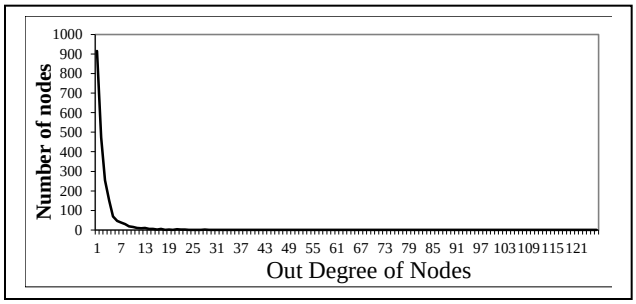

Figure 2: Degree distribution of nodes in search graph.

Consider the connectivity of the network as we remove the nodes. Clearly, the more nodes we remove, the more likely it would be to fragment the network. Random networks fall apart after a critical number of nodes have been removed. However, since the connectivity of scale-free networks depends on a small number of hubs, random failures cannot disconnect them. This extreme robustness is accompanied by fragility to attacks: the systematic removal of a few hubs would disintegrate these networks, breaking them into non-communicating islands. [Jeong *et al* 2001].

These properties have significant implications for making reasoning more efficient. Our heuristics are based on the fact that nodes with high degree are queried for repeatedly and can be proved in many ways. This ensures that these predicates take lot of time to fail. One option is to remove these nodes i.e. stop reasoning about them. However, removing high-degree nodes disconnects the network and coverage drops significantly. Therefore, we need a subtle balance to keep a minimum number of such nodes to ensure that we answer a reasonable number of questions. To do this, we use the distribution of known and inferred facts. If a predicate is frequently known, then the search space below it is less relevant[7]. Moreover genlPreds inference is typically easier than inference with normal axioms. Therefore the function *AllFacts(p)*, defined above, provides a good measure for identifying predicates which should be included. Our results below show that this heuristic can help us to get an efficient set of axioms.

However, we can take another approach for solving the same problem. Networks which have hubs have a diameter which is bounded by a polynomial in log(N), where N is the number of nodes. In other words, there is always a very short path between any two nodes [Kleinberg 2000]. Once the reasoning process reaches one of these hubs, most of the remaining network is easily reachable. In our experiments, we have observed that most of the predicates are accessible during search through predicates like isa, genls and holdsIn. This makes the search intractable and queries are timed out. Therefore, one possible solution is to prevent the network from having hubs[8]. We can do this by limiting the maximum out degree of each node in the search space. Let m be the maximum allowed out-degree of each node. If V is the set of vertices in the resulting graph, then by definition $m = \max_v \deg^-(v)$. In other words, we do not allow any node to have an out-degree greater than m. When m is 0, the graph is a set of disconnected nodes. On the other hand, when m is ∞, the out-degree is not limited and we have the original graph. Between these two extremes, we can study the ease of navigability of a family of search spaces. When m is low, short paths between nodes do not exist. As we increase m, the connectivity of graph improves. After a particular threshold, there are too many potential paths in the graph and relevant/short paths are difficult to find. This threshold determines the optimal complexity of the search space. In the next section, we use *AllFacts(p)* to restrict m, and to determine its optimal value. This is similar to Kleinberg's notion that efficient navigability is a fundamental property of only some networks and after a critical threshold individuals (or algorithms) cannot easily find paths in social networks [Kleinberg 2000].

Experiments

[4] 50% nodes have less than 8 neighbors.

[5] For example, the out-degrees of isa, temporallyIntersects and genls are 854, 358 and 149 respectively.

[6] In its most general form, a power law distribution has the form $p(x) \alpha x^{-a}$, where $2 < a < 3$ although there are occasional exceptions [Clauset et al 2009].

[7] If the query can be answered by using ground facts then the need to explore the search space below a given node does not arise.

[8] In this analysis, we consider a hub to be a node which has more than 50 children. Less than 1% of the nodes satisfy this condition.

Abhishek Sharma and Kenneth D. Forbus. (2013). Automatic Extraction of Efficient Axiom Sets from Large Knowledge Bases. *Proceedings of AAAI: Twenty-Seventh Conference on Artificial Intelligence*, Bellevue, WA

To illustrate the utility of these ideas, we describe a series of experiments using three sets of questions. The first corpus of questions (Q_1 henceforth) was generated by randomly selecting entities from the KB satisfying the question templates discussed above, creating 100 questions of each type, for 1000 questions total. The second corpus of 970 questions (Q_2 henceforth) was generated by automatically generating all legal instantiations of the parameterized questions for the entities constructed by the overall system's reading. We also wanted to check that our methods worked for other predicates. To do this we sorted all predicates by *AllFacts(p)*. We then selected 61 top predicates and replaced their first argument by 100 randomly sampled entities satisfying their argument constraints. This led to a set of 6100 queries. Intuitively, this set has most ground facts in the genlPreds tree below it. This set of question is referred as Q_3 in this section. Each query was timed out after ninety seconds. We used a simple backward-chainer working on a LTMS based inference engine. We sought one instantiation of the answer and searched until depth 3. All experiments were done on a 3.2 GHz Pentium Xeon processor with 3GB of RAM.

Notation	Description
Baseline	All horn rules for all predicates in the search tree.
E_0	Output of algorithm *ExtractAxioms*
P_x	Sort the predicates on the basis of their out degree and keep x% with the highest out degree
Q_x	Sort the predicates on the basis of AllFacts(x) function and keep x% with the highest value.
D_x	Begin with E_0. Find the top x children of each node by sorting the children via AllFacts(p), and keeping top x children. Exclude all axioms in E_0 whose antecedents are not in the top x children of the predicate in the consequent.
E_i	E_0\ All Rules which mention predicates in P_i, i>0
EQ_i	E_0\All Rules which mention predicates in (P_i \Q_i), i>0

Table 1: Notation

For the sets Q_1 and Q_2, we begin with all predicates in the question templates discussed above. For Q_3, we begin with a set of 61 predicates discussed earlier. We make a search tree for all these predicates and use the algorithm *ExtractAxioms* to get the set E_0. The set E_0 for Q_1 and Q_2 had 6,417 axioms while E_0 for Q_3 had 7,894 axioms. For a baseline comparison, we use the set of all Horn clauses for all predicates in the search tree. The description of other rule sets is shown in Table 1. For example, to get P_2, we sort the predicates on the basis of their out degree and keep the top 2%. Then E_2 is obtained by removing all axioms from E_0 which mention predicates in P_2. We see that as we remove high degree predicates (see the performance of E_x in Table 2), the network falls apart and virtually no inference is possible. All questions answered at this stage are obtained by database lookup and minimal unification/chaining takes place. Moreover, the number of questions answered remains roughly same for E_6, E_8 and E_{10} which suggests that the search space had been fragmented to a set of non-communicating islands for E_6 and removing more predicates did not cause any change. Table 2 shows that time required for E_6, E_8 and E_{10} is very close to zero which provides additional evidence that the search space is disconnected. Next we use the predicates in Q_x to keep some of high degree predicates which have many ground facts in the genlPreds tree below them (see the definition of AllFacts(x) above). By including these predicates we get the set EQ_x, which helps us in recovering the coverage lost by removing high-degree nodes (see Table 2). This heuristic leads to a factor of 81, 4.90 and 2.2 improvement for the Q_1, Q_2 and Q_3 sets respectively. The maximum loss in coverage is 8.5%. These heuristics are referred to as **Heuristic 1** in Table 3.

	Q/A (Q_1)	Time (Q_1)	Q/A (Q_2)	Time (Q_2)	Q/A (Q_3)	Time (Q_3)
Baseline	60.0	51.83	43.05	52.22	25.49	64.44
E_0	60.5	40.05	41.86	42.40	30.50	51.50
E_6	27.3	0.03	17.02	0.02	6.70	0.02
E_8	27.1	0.06	16.48	0.03	6.70	0.02
E_{10}	26.8	0.06	16.16	0.02	6.70	0.02
EQ_2	58.0	0.64	36.65	0.45	28.49	6.72
EQ_4	57.4	8.97	39.37	10.64	43.78	30.06
EQ_6	46.0	10.48	35.79	11.08	48.16	28.74
E_0	60.5	40.05	41.86	42.40	30.50	51.50
D_3	64.6	0.07	40.34	0.05	22.52	0.19
D_5	66.2	0.76	45.11	0.68	34.18	2.61
D_9	68.2	3.38	47.50	2.73	38.55	5.25
D_{18}	67.4	14.45	48.04	15.01	50.63	15.68

Table 2: Effect of removal of nodes on Q/A performance and time requirements. Q/A numbers are in %, whereas time requirements are in minutes.

Next we show that maximum out degree is a reasonable parameter for modeling efficient navigability of the network. To remove bottlenecks, we would like to reduce the variance in out-degree of nodes. When *m* is ∞, we have the original rule set or E_0. In Table 2, we report the performance of D_x for different values of x. For example, in the set D_3, all nodes have at most 3 children. The definition in Table 1 reflects our preference for those regions of search space which are rich in ground facts. We see that the performance improves in all cases. This heuristic is referred to as **Heuristic 2** in Table 3.

The final results are shown in Table 3. We note that *ExtractAxiom*'s output, E_0, significantly improves the performance compared to the baseline. It improves the performance by 22%, 18% and 20% for Q_1, Q_2 and Q_3 set respectively. The maximum loss of coverage is 2.7%. Heuristic 1 and 2 further improve the performance of this set. The set of axioms which led to best performance are

shown in parentheses. We see that in most cases we have been able to improve the performance significantly. Heuristic 1 might lead to some loss in coverage but in our experience, the overall performance improvement is worth it. Heuristic 2 always leads to improved performance without any loss of coverage. The results shown in Table 3 are statistically significant ($p < 0.01$).

Query Sets	Rule sets	% Answered	+/- %	Time (min)	Speedup
Q_1	Baseline	60.00	0%	5183.70	1
	E_0	60.50	0.83%	4005.39	1.3
	Heuristic 1 (EQ_2)	58.00	-3.33%	64.00	81
	Heuristic 2(D_3)	64.60	7.67%	7.78	666
Q_2	Baseline	43.05	0%	5222.96	1
	E_0	41.86	-2.76%	4240.55	1.2
	Heuristic 1(EQ_4)	39.37	-8.55%	1064.76	4.9
	Heuristic 2 (D_9)	47.50	10.34%	273.54	19.1
Q_3	Baseline	25.49	0%	6444.21	1
	E_0	30.50	19.65%	5150.39	1.2
	Heuristic 1 (EQ_6)	48.16	88.94%	2874.02	2.2
	Heuristic 2 (D_{18})	50.63	98.63%	1568.48	4.1

Table 3: Summary Comparison of performance

Rule Set E_0	Q_3(%)	Q_3(minutes)
Timeout:30 sec.	23.98	2076.65
Timeout: 60 sec.	27.91	3769.80
Timeout: 90 sec.	30.50	6444.21
Timeout: 120 sec	32.77	6496.51

Table 4: Performance tradeoff with timeout for Q_3

It might seem counterintuitive that by removing some rules, we can answer more questions in less time. This is because as we increase the number of rules, the inference engine is "lost" in less useful regions of search space and queries are timed out. We verified this hypothesis by increasing the timeout for the E_0 set for the Q_3 set of questions. The results are shown in table 4. We see that the improvement is not encouraging. In fact, our heuristics outperform it even though their queries are allowed less than 120 seconds. In other words, allowing more time simply increases failure time and doesn't help in getting answers. We show the scaling of *ExtractAxioms* with depth in Table 5. We conclude that performance tapers off after a threshold and providing more resources isn't useful. We also evaluated our techniques on several sets of problems from the Thousands of Problems for Theorem Provers (TPTP) data set version 3.5.0 [Sutcliffe 2009]. These experiments were divided in two categories. Type 1 problems included 300 problems from the CSR (commonsense reasoning) domain. The Type 2 problems consisted of 50 satisfiable problems from the AGT, NLP and MGT domains. We were able to solve 81% and 88% problems from these sets successfully. The heuristics discussed here led to speedups of 11.2 and 4.0 respectively.

Depth	Q/A Q_1	Time Q_1	Q/A Q_2	Time Q_2	Q/A Q_3	Time Q_3
0	26.7	0.02	16.16	0.02	16.1	0.5
1	54.2	5.11	40.02	3.52	27.7	6.1
2	67.0	24.36	48.04	22.49	33.7	20.2
3	60.5	40.05	41.86	42.40	30.5	51.5
4	58.6	48.36	36.98	54.48	21.9	67.6
5	55.5	59.55	32.86	64.60	16.3	73.7

Table 5: Scaling of ExtractAxioms with depth.

Conclusions

As knowledge bases grow, especially via machine learning and knowledge capture, better ways to automatically use such knowledge efficiently must be found. This paper describes two techniques for this important problem. The first is to automatically extract subsets of the KB targeted at particular tasks; exploiting knowledge of what kinds of statements are available already and what might be supplied via other systems, such as learning systems. The second uses an analysis of the connectivity of knowledge bases to automatically identify nodes to prune which, while losing a small amount of coverage, can yield over an order of magnitude performance improvement. The average speedup is a factor of 129. The worst case is a factor of 4 improvements in time with only 8.5% loss in completeness. These results suggest three lines of future work. First, we need to test these algorithms over a broader range of problems, to ensure their generality. Second, we think coupling a network-based analysis like ours with other factors such as constraint patterns [Walsh 2003] could yield a more complete theoretical picture as to what makes inference hard. Finally, the ability to identify what makes inference hard potentially provides us with the information as to what might make inference easier – in other words, analyze the structure of knowledge bases to ascertain what kinds of knowledge could be added to improve inference, and thus help create cognitive systems that generate and prioritize their own learning goals, so that what they learn improves, instead of degrades, their operation.

Acknowledgements

This paper benefited from discussions with Johan de Kleer, Chris Riesbeck and Matt Klenk. This research was supported by DARPA IPTO and the Office of Naval Research.

References

Amir, E. and McIlraith, S. 2005. Partition-Based Logical Reasoning for First-Order and Propositional Theories, *Artificial Intelligence, 162*(1-2), pages 49-98.

Clauset, A., Shalizi, C. R. and Newman, M. 2009 Power-law Distributions in Empirical Data, *SIAM Review*, 51, pp. 661-703.

Cohen, P. Schrag, R., Jones, E., Pease, et al. 1998. The DARPA High-Performance Knowledge Bases Project. *AI Magazine*, 19(4), Winter, pp. 25-49.

Etzioni, O., Cafarella, M., Downey, D., Popescu, A., Shaked, T., Soderland, S., Weld, D., and Yates, A. 2005. Unsupervised Named-Entity Extraction from the Web: An Experimental Study. *Artificial Intelligence*, 165(1): pp. 91-134.

Forbus, K. D., Riesbeck, C., Birnbaum, L., Livingston, K., Sharma, A. Ureel II, L. 2007. Integrating Natural Language, Knowledge Representation and Reasoning, and Analogical Processing to Learn by Reading. *Proceedings of AAAI*, pp. 1542-1547

Gomes, C., Fernandez, C., Selman, B. and Bessiere, C. 2004. Statistical Regimes Across Constrainedness Regions, *Proceedings of CP*, pp. 32-46.

Jeong, H., Mason, S. P., Barabasi, A. L. and Oltavi, Z. N. 2001. Lethality and Centrality in Protein Networks, *Nature*, Vol. 411, pp. 41-42.

Kilby, P., Slaney, J., Thiebaux, S. and Walsh, T. 2005. Backbones and Backdoors in Satisfiability, *Proceedings of AAAI*, pp. 1368-1373.

Kleinberg, J. 2000. Navigation in a small world. *Nature*, 406, page 845.

Martin, C. E. and Riesbeck, C. K. 1986. Uniform Parsing and Inferencing for Learning. *Proceedings of AAAI*, pp. 257-261.

Mitchell, D., Selman, B. and Levesque, H. 1992. Hard and easy distributions of SAT problems, *Proceedings of AAAI*, pp. 459-465.

Mitchell, M. 2006. Complex systems: Network thinking, *Artificial Intelligence*, 170(1-2), pp. 1194-1212.

Selman, B. and Kautz, H. 1996. Knowledge Compilation and Theory Approximation, *Journal of ACM*, pages 193-224.

Sutcliffe, S. 2009. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0, *Journal of Automated Reasoning*, 43(4), pp. 337-362.

Walsh, T. 1999. Search in a small world. *Proceedings of IJCAI*, pp. 1172-1177.

Walsh, T. 2003. Constraint Patterns, *Proceedings of CP*, pp. 53-64.

Abhishek Sharma and Kenneth D. Forbus. (2013). Automatic Extraction of Efficient Axiom Sets from Large Knowledge Bases. *Proceedings of AAAI: Twenty-Seventh Conference on Artificial Intelligence*, Bellevue, WA