A RISC Approach to Reasoning with Natural Language

Robert M. MacGregor

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
macgregor@isi.edu

Introduction

Many developers of natural language (NL) processing systems believe that a single internal representation (of sentences) should support all levels of reasoning (linguistic, semantic, and pragmatic). Such a representation is necessarily quite expressive, containing a large variety of syntactic constructs necessary to model the nuances of natural language. Others (including ourselves) take the view that reasoning can be more efficient if two different representations are used—a richer representation is used for linguistic reasoning, while an ontologically much sparser representation is used for reasoning about semantics and pragmatics. We argue for a representation that eliminates as many syntactic constructs as possible while preserving full expressivity. We are currently implementing a reasoning system that performs efficient logical deductions using a fully expressive (but syntactically sparse) graph-based representation.

Why RISC It?

Many NL systems use semantic network (graph-based) representations. Graphs are commonly employed both for their flexibility of representation (as compared, e.g., with a linear syntax) and with the aim of supporting efficient forms of inference. However, the available technology for efficiently computing logical deductions using graph-based representations is still young, and considerable progress will be needed to enable its routine use in large-scale NL applications.

The field of knowledge representation (KR) has evolved a technology for efficiently computing logical deductions over a highly restricted class of languages called description logics [MacGregor 91]. In [MacGregor 94], we show how this technology can be applied to a fully-expressive graph-based language. The heart of our system is a reasoner called a "classifier" that computes and caches subsumption relationships between descriptions (between graphs). The completeness and/or efficiently of a description classifier depends heavily on its ability to derive canonical representations of logical expressions. Although we have developed techniques that reduce our classifier's reliance on canonical representations, a key feature in our choice of representation is that it contains relatively few syntactic constructs. Thus, we endorse a minimalist, i.e., a RISC[1] approach to designing graph languages.

A forte of description logic systems is their ability to reason about (relationships between) sets, especially, intensional sets. Commonly, this is a weak point of theorem-prover -based systems, and of most other deduction architectures. Our system continues to emphasize intensional sets as a fundamental syntactic construct. In this paper we illustrate how the use of intensional sets can supplant (eliminate the need for) certain other syntactic constructs, and we show how our classifier aggressively reduces many logical problems to problems of computing set relationships.

The first half of this paper introduces our graph representation, and describes the algorithm that computes subsumption relationships between graphs. The second half compares and contrasts our technology with that of three other representations (EPILOG, SNePS, and Conceptual Graphs). In each case we illustrate some fundamental architectural differences. Our purpose is not to refute the validity of the other approaches, but instead to point out alternative choices that might ultimately yield more efficient strategies for logical deduction.

Background

Beginning with KL-ONE[Brachman&Schmolze 85], description logic systems have a history of use in natural language applications. A description classifier is relatively efficient at computing subsumption, instantiation, and disjointness relationships, and at detecting inconsistencies in a set of sentences; these computations are of central importance in many NL applications. An unfortunate drawback to the KL-ONE family of systems is their lack of expressive power—they can only reason with

[1] Reduced Instruction Set Computer.

severely restricted subsets of first order logic. The Loom KR system [MacGregor&Burstein 91] is the most expressive of the currently implemented classifier-based systems; not coincidentally, many more NL applications use Loom than use all other classifier-based systems combined.

Loom can be employed to efficiently solve non-trivial problems. For example, Loom is able solve Schubert's Steamroller problem in about 25 seconds (on a Macintosh Quadra). While this doesn't set a speed record, it represents a fairly respectable time. None of the other existing classifier-based systems can represent the Steamroller problem (let alone solve it). A couple of years ago, we initiated an effort to discover how much farther we could push classifier technology, with the goal of making the language recognized by our classifier as expressive as possible. We have developed a classifier [MacGregor 94] that reasons with the full predicate calculus (PC), extended to include sets, cardinality, equality, scalar inequalities, and predicate variables. The syntax and semantics of the language recognized by our PC classifier strongly resembles that of KIF. Internally, the classifier manipulates graph structures analogous to a semantic network representation. The PC classifier is relatively efficient—benchmarks have shown it to run about half as fast as the Loom classifier. We expect that a mature version of the PC classifier will find extensive use in NL applications.

Descriptions, Graphs, and Structural Subsumption

The primary task of a classifier is to construct a taxonomy of "descriptions", ordered by a subsumption relationship. A *relation description* specifies an intensional definition for a relation. It has the following components:

- a name (optional);
- a list of domain variables < dv1, ... ,dvk >, where k is the arity of the relation;
- a definition—an open sentence in the prefix predicate calculus whose free variables are a subset of the domain variables.
- a partial indicator (true or false)— if true, it indicates that the predicate represented by the relation definition is a necessary but not sufficient test for membership in the relation.

Relation descriptions are introduced by the **defrelation** operator, with the syntax

```
(defrelation <name>
   (<domain variables>)
   [:def | :iff-def] <definition>)
```

The keyword **:def** indicates that the definition is partial, while the keyword **:iff-def** indicates that it is not. Terms of the form

```
(setof (<variables>) <definition>)
```

represent unnamed *set expressions* .

The fundamental match operation within a classifier is the computation of a subsumption relationship. A description C *subsumes* a description D if the set of instances that satisfy the (definition of) C contains the set of instances that satisfy D. Consider the descriptions, "persons that have at least one son":

```
(defrelation At-Least-One-Son (?p)
  :iff-def (and
  (Person ?p)
  (>= (cardinality
        (setof (?c) (son ?p ?c)))
      1))
```

and "persons that have more sons than daughters":

```
(defrelation
      More-Sons-Than-Daughters (?pp)
  :iff-def (and
  (Person ?pp)
  (> (cardinality
       (setof (?b) (son ?pp ?b)))
     (cardinality
       (setof (?g) (daughter ?pp ?g)
       )))))
```

Below, we briefly illustrate how the subsumption test implemented in the PC classifier is able to prove that **At-Least-One-Son** subsumes **More-Sons-Than- Daughters.**

The **At-Least-One-Son** relation is associated with the following set expression:

```
(setof (?p) (and
   (Person ?p)
   (>= (cardinality
        (setof (?c) (son ?p ?c)))
       1).
```

The addition of variables to represent the nested **setof** expression and the Skolemized cardinality function produces the following equivalent set expression

```
(setof (?p) (and                          [1]
  (Person ?p)
  (exists (?s1 ?card1)
    (= ?s1 (setof (?c) (son ?p ?c)))
    (cardinality ?s1 ?card1)
    (>= ?card1 1)))).
```

Figure 1 below illustrates the graph of the set expression [1]

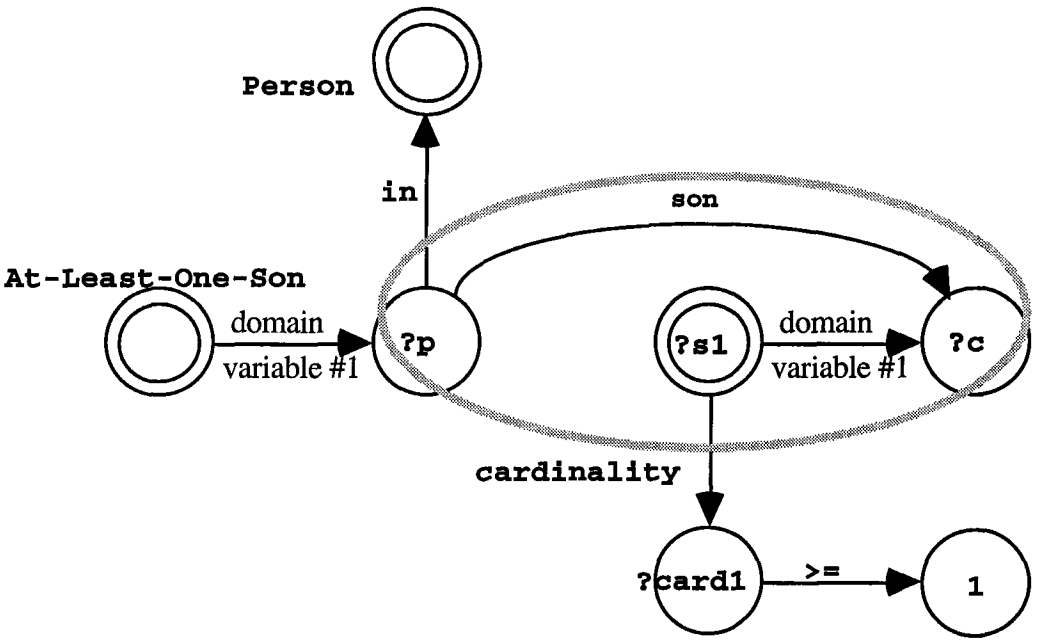

Figure 1.

Most description classifiers, including the PC classifier, use a "structural" subsumption test to prove the existence of a subsumption relationship between descriptions. To prove that C subsumes D, a structural prover tries to demonstrate that for each component in the representation of C there is a corresponding component in the representation of D. Internally, the PC classifier represents descriptions as graphs. Thus, its subsumption test is equivalent to a test for subgraph isomorphism.

By itself, structural subsumption is a relatively weak test. The classifier employs two procedures, *canonicalization* and *elaboration*, to increase the power of the subsumption test. Canonicalization transforms a description to a semantically equivalent canonical form. Consider the following descriptions, each of which defines the set of "things having at least one friend":

```
(setof (?a)                                  [2]
  (exists (?y1) (friend ?a ?y1)))

(setof (?b)                                  [3]
  (exists (?s) (and
    (= ?s (setof (?y2)
                 (friend ?b ?y2)))
    (>= (cardinality ?s) 1))))
```

During canonicalization, the (graph of the) second description will be converted into a graph that is isomorphic to the (graph of the) first description. The principle underlying this transformation is that, all other things being equal, the classifier prefers to reason about properties of individuals (and individual variables) rather than about properties of sets of individuals. After canonicalization, a structural test suffices to determine that [2] and [3] are equivalent.

Elaboration is a forward chaining operation that adds structure to a graph—it creates new nodes and (hyper) edges representing variables and relationships that are implicitly defined by the semantics of the existing graph structures. Here is a description of the relation **More-Sons-Than-Daughters** after canonicalization and elaboration:

```
(setof (?pp) (and                          [4]
   (Person ?pp)
   (exists (?s2 ?s3 ?card2 ?card3)
      (= ?s2 (setof (?b)
                 (son ?pp ?b)))
      (= ?s3 (setof (?g)
                 (daughter ?pp ?g)))
      (cardinality ?s2 ?card2)
      (cardinality ?s3 ?card3)
      (> ?card2 ?card3)
      (Integer ?card2)        ; new
      (>= ?card2 0)           ; new
      (Integer ?card3)        ; new
      (>= ?card3 0)           ; new
      (>= ?card2 1))))        ; new
```

The five clauses marked "new" represent elaborations. Adding the last new clause enables the subgraph isomorphism test to determine that the description [1] subsumes the description [4].

Table 1 lists many of the elaboration rules that are built into the PC classifier. These rules can be thought of as built-in common sense knowledge about things such as set relationships, cardinalities, inequalities, and simple arithmetic. This knowledge is considered sufficiently important that the system derives it using forward chaining rather than goal-driven inference. One of our future goals is to augment these rules with resolution-based inference, resulting in a constraint propagation capability analogous to that implemented in the SCREAMER system [McAllester&Siskind 93]. A key feature of these elaboration rules is that (individually) they do not significantly increase the size of our graphs. A rule that, for example, expands a description into disjunctive normal form would not be appropriate because the cost in terms of space (and time) could be prohibitive. For the same reason, the system omits rules that compute transitive closure for relationships such as greater than, less than, etc. In general, we expect to use backward chaining and/or resolution to solve problems that are too expensive to solve using forward chaining inference.

Recall that we opened this discussion by remarking that our system uses a relatively minimal syntax for representing knowledge. The classifier functions best when it can derive a unique canonical representation. Thus, although the syntax of our input language is unrestricted, the graphs structures manipulated internally are much sparser. The most striking example of this is in our treatment of universal quantifiers—internally, there aren't any! Instead, when the parser encounters an expression of the form

```
(forall (?v1 ... ?vk)
   (implies <antecedent>
            <consequent>))
```

it transforms this expression into the equivalent expression

```
(contained-in
 (setof (?v1 ... ?vk) <antecedent>)
 (setof (?v1 ... ?vk) <consequent>))
```

where **contained-in** is the subset/superset relation. In effect, reasoning about universally quantified variables is transformed into reasoning about set relationships. There is a great deal of parallelism between reasoning about set containment relationships, reasoning about superclass relationships between concepts and relations, and reasoning about implication relationships. In our architecture, all of these are collapsed into a single representation. This exemplifies what we mean when we refer to it as a RISC architecture.

In the next sections we will briefly compare and contrast the PC classifier representation with that of some other prominent KR languages/systems that support reasoning with NL. We note that our language contains no built-in capabilities for representing or reasoning with such things as episodes, nested beliefs, causality, etc. Representations for these constructs could and should be added to our language if we are to more fully support NL processing, but they will not be discussed here.

Comparison with Three NL Systems

EPILOG

The EPILOG system [Schubert et al 93a, 93b] enhances the standard predicate calculus with a variety of constructs that enable it to more closely represent natural language sentences. EPILOG is intended to represent linguistic as well as semantic and pragmatic knowledge. The internal representation of sentences apparently closely resembles the linear syntax manipulated by users (except that it is augmented by various indexing mechanisms).

EPILOG supports a variety of quantifiers. Quantifiers such as "most", "few", etc. are converted into implications having probabilities attached to them. Our system would favor representing this class of quantifiers as statements about sets rather than using probabilities. For example,

Representative Selection of Elaboration Rules

Inequality rules:

I1 >= MIN and Number(MIN) and I2 >= I1 fi I2 >= MIN	; propagate lower bound
I1 > MIN and Number(MIN) and I2 >= I1 fi I2 > MIN	; propagate strict lower bound
I1 >= MIN and Number(MIN) and I2 > I1 fi I2 > MIN	; propagate strict lower bound
I1 <= MAX and Number(MAX) and I2 <= I1 fi I2 <= MAX	; propagate upper bound
I1 < MAX and Number(MAX) and I2 <= I1 fi I2 < MAX	; propagate strict upper bound
I1 <= MAX and Number(MAX) and I2 < I1 ⇒ I2 < MAX	; propagate strict upper bound
I > MIN and Integer(I) ⇒ I >= floor(MIN) + 1	; round lower bound up
I >= MIN and Integer(I) and Number(MIN) and not(Integer(MIN)) ⇒ I >= floor(MIN) + 1	
I < MAX and Integer(I) ⇒ I <= ceiling(MAX) - 1	; round upper bound down
I <= MAX and Integer(I) and Number(MAX) and not(Integer(MAX)) ⇒ I <= ceiling(MAX) - 1	
I1 >= I2 and I2 >= I1 ⇒ I1 = I2	; equate two-way greater or equal
I1 >= I2 ⇒ I2 <= I1	; inverse greater or equal
I1 <= I2 ⇒ I2 >= I1	; inverse lesser or equal
I1 > I2 ⇒ I2 < I1	; inverse greater
I1 < I2 ⇒ I2 > I1	; inverse lesser

Cardinality rules:

set(S) ⇒ exists(I) cardinality(S,I)	; sets have cardinalities
cardinality(S,I) ⇒ Integer(I)	; integer cardinality
cardinality(S,I) ⇒ I >= 0	; non-negative cardinality
contained-in(S1,S2) fi cardinality(S1) <= cardinality(S2))	; greater cardinality superset
I >= MIN and I <= MAX and Integer(I) and Integer(MIN) and Integer(MAX) and domain-variable(S,I) and arity(S) = 1 ⇒ cardinality(S) <= MAX - MIN	
in(I,S) ⇒ cardinality(S) >= 1	; non-empty set
cardinality(S) = 1 and in(I,S) and in(J,S) ⇒ I = J	; equate members of singleton set

<u>Contained-in rules</u>:

contained-in(S1,S2) and in(I,S1) ⇒ in(I,S2)	; propagate members up
contained-in(S1,S2) and cardinality(S1) = cardinality(S2) ⇒ S1 = S2	;equate equal cardinality superset
contained-in(S1,S2) and contained-in(S2,S3) ⇒ contained-in(S1,S3)	; transitivity of contained-in
contained-in(S1,S2) and contained-in(S2,S1) ⇒ S1 = S2	; equate two-way containment
S1 = S2 ⇒ contained-in(S1,S2)	; reflexivity of contained-in
contained-in(S1,S2) and contained-in(S1,S3) and intersection(S2,S3,S4) ⇒ contained-in(S1,S4)	; contained-in intersection set
contained-in(S1,S2) ⇒ contains(S2,S1)	; inverse contained-in
contains(S1,S2) ⇒ contained-in(S2,S1)	; inverse contains

<u>Other rules</u>:

in(I,S1) and in(I,S2) and intersection(S1,S2,S3) ⇒ in(I,S3)	; member of intersection set
domain-variable(S1,I1) and arity(S1) = 1 and in(I1,S2) ⇒ contained-in(S1,S2)	

Table 1

consider the sentence "Most wolves are fierce." A conventional definition of "most" translates as "more than half". We can define **most** as a relation over a pair of sets:

```
(defrelation most (?s1 ?s2)
   :iff-def
   (>= (/ (cardinality
              (intersection ?s1 ?s2))
          (cardinality ?s1))
       .50)
```

Then, assuming that **Wolves** and **fierce** are names for the corresponding (unary) relations, our sentence about wolves can be represented as:

```
(most Wolves fierce)
```

Our definition of "most" maps nicely onto situations involving finite sets. For example, if we assert that 10 people are at party P5 and 6 of them are drunk

```
(assert
  (exists (?s) (and
    (= ?s (setof (?p) (and
                 (Person ?p)
                 (at-party ?p P5))))
    (= (cardinality ?s) 10)
    (= (cardinality
         (setof (?p) (and
                 (?s ?p) (Drunk ?p))))
       6)))
```

then our definition of **most** directly supports an affirmative response in answer to the question "Are most of the people at party P5 drunk?"

```
(ask (most (setof (?p) (and
              (Person ?p)
              (at-party ?p P5)))
           Drunk))
```

Instead of representing "most" as a statement about (relative) cardinalities, EPILOG associates "most" with a probability (of around .85). Given that one of EPILOG's strongpoints is reasoning about probabilities, its choice of translation is understandable—if more than half of the members of a set satisfy a property P, than the *mean* number of individuals having property P will lie somewhere between 50% and 100%. However, since only .6 of the above partygoers are drunk, EPILOG might conclude that the proposition "Most of the people at party P5 are drunk" is false (its hard for us as outsiders to guess exactly where EPILOG would choose to draw the line). For the case of the quantifier "many", we see that the probabilistic representation has lost a great deal of fidelity, while fidelity is preserved by a representation based on intensional sets. Intensional sets can also be employed to accurately represent the meaning of statements like"Between 60 and 80 percent of the partygoers were male." While we agree that an ability to manipulate probabilities is essential component of a commonsense reasoning system, we claim that an ability to accurately reason with statements about relative cardinalities (of intensional sets) is also necessary. In the above example, we have illustrated how this capability will sometimes be superior to a probabilistic approach.

Continuing, we note that the availability of "setof" negates the need for an equivalent of EPILOG's lambda construct. EPILOG also provides an explicit syntax to represent predicate modifiers (e.g., to represent "very" in the phrase "very big"). We prefer that a modified predicate be translated at the linguistic level into an ordinary relation (e.g., "very-big", or "huge"). This would eliminate the need for representing predicate modifiers at the semantic and pragmatic levels. Summarizing, we have shown that our RISC representation suffices to represent some problems that EPILOG solves by the introduction of new syntax.

SNePS

Our comparison here with SNePS [Shapiro 91] is particularly directed at SNePS as extended by Syed Ali [Ali 93], resulting in the ANALOG system. The ANALOG extensions add representation and reasoning capabilities to SNePS that to some extent mimic those of a classifier-based system. However, ANALOG consciously seeks to mimic the syntax of natural language (more so than SNePS), and in that regard it diverges from our choice of representation.

Ali argues against universal quantifiers, claiming that they are not "conceptually complete". He considers the sentence "Dogs bite", represented in first order logic as

```
(forall (?x)
   (implies (Dog ?x) (bites ?x))
```

and asks "What is the meaning of 'x'?". He proposes a graphical notation in which a node D represents "any dog", part of the graph defines the meaning for the node D, and another part of the graph represents the assertion that D bites.

Our scheme has something in common and something different with his. Instead of inventing a special kind of node to represent "any Foo", where Foo is the name of a type, we use a set, e.g., in the above case we represent "Dogs bite" as

```
(contained-in Dog bites).
```

Suppose we add more structure to the definition of the "thing that bites", e.g., "Every big dog that is owned by a bad-tempered person bites:

```
(contained-in
   (setof (?x) (and
      (Dog ?x) (big ?x)
      (exists (?p) (and
         (owned-by ?x ?p)
         (Person ?p)
         (bad-tempered ?p))))
   bites)
```

In this statement, part of the graph defines the meaning of the set (of things that bite), and part of the graph defines an assertion about that set (the "contained-in" proposition). So, from a certain viewpoint we are partitioning or nesting our graph structures in a fashion very similar to the approach used in ANALOG. However, we accomplish this result without introducing new syntax (e.g., an "any" node).

There is a long history in semantic networks of using similar syntax to treat both individuals and set-like entities. The use of "any" node in ANALOG follows this tradition, e.g., ANALOG graphs representing the sentences "Fido bites" and "Dogs bite" are very similar. Our own experience suggests that most (although not all) of the operations that apply to sets and set-like entities are quite distinct from those that apply to individuals. Hence, unlike ANALOG, the PC classifier makes a very sharp syntactic distinction between these two notions (corresponding to the syntactic difference between descriptions [2] and [3] above).

We attribute part of the success of classifier-based systems to the fact that they are relatively good at reasoning about sets (both intensional and extensional). For example, they are proficient at computing subsumption relations between classes, reasoning about cardinalities of role sets, computing disjointness relationships between sets, and reasoning about various properties of enumerated sets. This reasoning is built-in at the lowest levels of the system. SNePS, on the other hand, evolved for a considerable length of time without including specialized support for reasoning about sets [Cho 92].

Conceptual Graphs

Conceptual graphs (CGs) [Sowa 91]have been deliberately designed to represent natural language sentences. The assumption that CGs are an appropriate data structure to support reasoning seems to be implicit in the papers about CGs, but thus far there have been few tools developed that provide such support. So, the question of whether or not one can reason efficiently with CGs is still open, and opinions on this question must take the form of conjectures.

The descriptions of such operations as graph matching and graph subsumption for CGs bear a strong resemblance to operations implemented in a description classifier. Attempts to formalize algebraic operations on CGs [Ellis&Willems 93] can be viewed as (more complex) variations on operations commonly applied to descriptions [Borgida 92]. We conjecture that the match technology being developed for reasoning with descriptions might also apply (after suitable modifications) to CGs. We also conjecture that the experience we have acquired while implementing a reasoner for PC graph representations may have predictive power when applied to the problem of reasoning with CGs.

In the PC architecture, a match operation to determine if G1 is a subgraph of G2 can be abstracted as follows: First, transformations are applied to graphs G1 and G2 to convert both of them to canonical form. Then, elaboration axioms (as illustrated in Table 1) are applied to G2 (only). Finally, a subgraph isomorphism test is applied to determine if the canonical version of G1 is a subgraph of the canonicalized and elaborated version of G2. Analysis of the results of performance tests reveals that most of the work performed during description classification occurs within the canonicalization and elaboration procedures. The PC classifier caches the results of canonicalization and elaboration in an attempt to minimize the overhead of these operations. Nevertheless, the basic lesson is that when measuring the cost of a match operation, the speed of the subgraph isomorphism test is *not* the determining factor.

The completeness of a classifier directly depends on how successful the canonicalization and elaboration operations are in uncovering similarities in semantics between syntactically dissimilar representations. We would expect that more complex graph representations would incur increased overhead during the canonicalization and elaboration phases, i.e., we predict an inverse relationship between complexity of representation structures and efficiency of a match operation that includes (analogs of) canonical transformations and elaboration. We expect that reasoning with CGs may prove to be more difficult (i.e., slower) than reasoning with PC graphs.

Conclusions

Classifier-based knowledge representation systems have traditionally sacrificed expressive power in favor of efficiency (and completeness). Our PC classifier eliminates all restrictions on expressivity while preserving most of the efficiency of the classifier based approach (we have, necessarily, sacrificed completess). Within the context of an NL application, our system is not oriented towards the representation of linguistic knowledge. We envision

that it would be used within an NL system only for semantic and pragmatic reasoning.

The intrinsic merit of the PC architecture would be evidenced if we could demonstrate that reasoning with more complex representations is inherently more difficult. We draw an analogy with the experience of the hardware folks, where RISC architectures are currently outperforming the CISC (Complex Instruction Set Computer) architectures, both in terms of performance and manufacturability. Another analogy can be drawn with PROLOG, where a very sparse representation has been heavily exploited, down to the level of silicon implementations of the PROLOG unification operation. However, current KR technology for NL has not matured to the point where empirical testing can determine the relative performance of various deductive architectures.

While maintaining full expressive power, we have sought to minimize the complexity of the graph structures used within the PC classifier, under the assumption that this approach maximizes both completeness and performance. Following the description logic tradition, our representation treats intensional sets as first class entities. Where possible, we have chosen to map other representational constructs (e.g., material implications and generalized quantifiers) into set-based representations. We view this approach as running counter to many of the semantics network based approaches to KR.

Our use of a graph-based internal representation brings us closer architecturally to several other NL technologies, and creates a new point of reference in the spectrum of semantic network based reasoning systems.

References

[Ali 93] Syed S. Ali, "A 'Natural Logic' for Natural Language Processing and Knowledge Representation, Ph.D. Dissertation, Dept. of Computer Science, SUNY Buffalo, New York, Nov. 1993.

[Brachman&Schmolze 85] Brachman, R.J. and Schmolze, J.G.", "An Overview of the KL-ONE Knowledge Representation System", Cognitive Science, August, 1985, pp. 171-216.

[Borgida 92] Alex Borgida, "From Types to Knowledge Representation: Natural Semantics Specifications for Description Logics", *International Journal on Cooperative and Intelligent Information Systems* [1,1] 1992.

[Cho 92] Sung-Hye Cho, "Collections as Intensional Entities and their Representations in a Semantic Network", *Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence*, pp. 388-394, 1992.

[Ellis&Willems 93] Gerard Ellis and Mark Willems, "Analysis of Semantic Networks using Graph Grammars", Journal of Experimental and Theoretical Artificial Intelligence, vol. 5, 1993.

[MacGregor 91] Robert MacGregor, "The Evolving Technology of Classification-based Knowledge Representation Systems", *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Chapter 13, John Sowa, Ed., Morgan-Kaufman, 1991.

[MacGregor&Burstein 91] Robert MacGregor and Mark H. Burstein, "Using a Description Classifier to Enhance Knowledge Representation", IEEE Expert, 6:3, June, 1991, pp.41-46.

[MacGregor 94] Robert MacGregor, "A Description Classifier for the Predicate Calculus", in AAAI-94, The National Conference on Artificial Intelligence, Seattle, Washington, August, 1994.

[McAllester&Siskind 93] Jeffrey M. Siskind and David A. McAllester, "Nondeterministic Lisp as a Substrate for Constraint Logic Programming", *AAAI-93 Proc. of the Eleventh Nat'l Conf. on Artificial Intelligence*, Washington, DC, pp.133-138.

[Schubert et al 93a] Stephanie Schaeffer, Chung Hee Hwand, John de Haan, and Lenhard Schubert, EPILOG New User's Tutorial, Dept. of Computer Science, U. of Alberta, Edmonton, Canada, August, 1993.

[Schubert et al 93b] Stephanie Schaeffer, Chung Hee Hwand, John de Haan, and Lenhard Schubert, EPILOG User's Guide, Dept. of Computer Science, U. of Alberta, Edmonton, Canada, August, 1993.

[Shapiro 91] Stuart Shapiro, "Cables, Paths, and 'Subconscious' Reasoning in Propositional Semantic Networks", *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Chapter 4, John Sowa, Ed., Morgan-Kaufman, 1991.

[Sowa 91] John Sowa, "Toward the Expressive Power of Natural Language", *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Chapter 5, John Sowa, Ed., Morgan-Kaufman, 1991.