ANALOG: A Logical Language for Natural Language Processing*

Syed S. Ali
Department of Computer Science
Southwest Missouri State University
901 South National Avenue
Springfield, MO 65804
ssa231f@cnas.smsu.edu

Abstract

We present a formal description of a logical language that is based on a propositional semantic network. Variables in this language are not atomic and have potentially complex structure. We start from the individual components of a semantic network system, atomic nodes and relations that connect nodes, and provide a complete specification for the structure of nodes and a subsumption procedure between nodes. We differ from other work in subsumption in that the representation language is uniform and based on an extended first-order predicate logic. The language is particularly suitable for addressing some problems associated with natural language processing, namely the representation of complex natural language descriptions and inference associated with description subsumption.

1 Introduction

We present a formal description of a propositional semantic-network-based knowledge representation system. Variables in this representation are not atomic and have potentially complex structure. We start from the individual components of a semantic network system, atomic nodes and relations that connect nodes, and provide a complete specification for the structure of nodes and a subsumption procedure between nodes. We differ from other work in subsumption in that the representation language is uniform and based on a first-order predicate logic. The language is particularly suitable for addressing some problems associated with natural language processing, namely the representation of complex natural language descriptions and inference associated with description subsumption.

*This is a extended version of a paper that appears in the proceedings of AI'94, the Canadian Artificial Intelligence Conference, Banff Park Lodge, Banff, Alberta, Canada, May 16-20, 1994

Subsumption is a partial ordering on related concepts (nodes in a semantic network) that relates more general concepts to more specific instances of those concepts. The manner in which "more general" is determined characterizes the type of subsumption. Woods has classified subsumption into *extensional, structural, recorded, axiomatic,* and *deduced* subsumption [Woods, 1991]. In general, the more complex the structure of the concepts (and their associated semantics) the more difficult subsumption becomes.

Our formalism embeds a procedure for determining *structural* and *deduced* subsumption between concept nodes in a propositional semantic network representation (one in which propositions are represented by nodes and not arcs of the network). We specify the structure of the nodes of the propositional semantic network system in terms of its components, nodes and relations between nodes, and specify the subsumption procedure in terms of these components. This subsumption procedure can do the type of subsumption inference associated with KL-ONE classification [Brachman and Schmolze, 1985] and its successors (most notablely KRYPTON [Brachman *et al.*, 1985]). This subsumption procedure does this without a distinction between an assertional and terminological component (and their associated difficulties [Beierle *et al.*, 1992]).

Sections 2 and 3 summarize the formal specification of the logical language. Section 4 describes the subsumption procedure in detail. Sections 5 and 6 describe the concept matcher and inference mechanism. These sections provide a framework in which the natural language processing examples of Section 7 can be best understood.

2 Syntax and Semantics of the Logic

We specify the syntax and semantics of a logic whose variables are not atomic and have structure. We call these variables *structured variables*. The syntax of the logic is specified by a complete definition of a propositional semantic network representation formalism (an augmentation of [Morgado, 1986, Shapiro, 1991]). By a

propositional semantic network, we mean that all information, including propositions, "facts", etc., is represented by nodes. The implemented system, ANALOG, is used here, for convenience, to refer to the logical system.

2.1 Semantics

As a propositional semantic network formalism, any theory of semantics that ascribes propositional meaning to nodes can be the semantics used in ANALOG. In this paper, examples and representations are used that follow the case frame semantics of [Shapiro and Rapaport, 1987] which provide a collection of propositional case frames and their associated semantics based on an extended first-order predicate logic. We augment that logic further with arbitrary individuals (for the semantics of structured variables) in a manner similar to the semantic theory of [Fine, 1985]. For a more complete specification of the syntax and semantics of ANALOG and its suitability for NLP, see [Ali, 1994, Ali and Shapiro, 1993, Ali, 1993b, Ali, 1993a, Ali, 1993c].

2.2 The Domain of Interpretation

ANALOG nodes are terms of a formal language. The interpretation of a node is an object in the domain of interpretation, called an entity. Every ANALOG node denotes an entity, and if n is an ANALOG node, then $[\![n]\!]$ denotes the entity represented by n. Nodes are atomic or structured. In the latter case, they are connected to other nodes by labelled arcs. The labels on arcs are called "relations". It is useful, for discussing the semantics of ANALOG networks, to present them in terms of an "agent". Said agent has beliefs and performs actions, and is actually a model of a cognitive agent. In the rest of this explication, the term "node" will be used for ANALOG node, and "relation" for ANALOG relation.

2.3 Metapredicates

To help formalize this description we introduce the metapredicates *Conceive*, *Believe*, and $=$. If n, n_1, n_2 are metavariables ranging over nodes, and p is a metavariable ranging over proposition nodes, the semantics of the metapredicates listed above are:

- *Conceive*(n) Means that the node is actually constructed in the network. *Conceive*(n) may be true without $[\![n]\!]$ being known to be true or false. Also note that n need not be a proposition.
- *Believe*(p) Means that the agent believes the proposition $[\![p]\!]$.
- $n_1 = n_2$ Means that n_1 and n_2 are the same, identical, node.

Belief implies conception, as specified in Axiom 1.

Axiom 1: $Believe(p) \Rightarrow Conceive(p)$

In practical terms, this means that for a proposition to have a belief status it must be a node in the semantic network. This is a reasonable assumption since for an agent to believe something the agent must have some conceptualization for it.

2.4 Definition of Nodes

Informally, a node consists of a set of labeled (by relations) directed arcs to one or more nodes. Additionally, a node may be labeled by a "name" (e.g., `BILL`, `M1`, `V1`) as a useful (but extra-theoretic) way to refer to the node. This naming of a proposition node is of the form `M`n, where n is some integer. A "!" is appended to the name to show that the proposition represented by the node is believed to be true (*Believe*(`M1`)). However, the "!" does not affect the identity of the node or the proposition it represents. Similarly, variable nodes are labeled `V`n, where n is some integer, and base nodes are named `B`n, where n is some integer (additionally, base nodes may be named for the concept they represent, e. g., `man`). More formally a node is defined as follows:

Definition 1: There is a non-empty collection of labelled atomic nodes called **base nodes**. Typically, base nodes are mnemonically labelled to indicate the entity they denote. *Example:* `bill` is a base node.

Definition 2: A **wire** is an ordered pair $\langle r, n \rangle$—, where r is a relation, and n is a node. Metavariables $w, w_1, w_2, \ldots$ range over wires. *Example:* <`member`, `john`> is a wire.

Definition 3: A **nodeset** is a set of nodes, $\{n_1, \ldots, n_k\}$. Meta-variables $ns, ns_1, ns_2, \ldots$ range over nodesets. *Example:* {`john, bill`} is a nodeset if `john` and `bill` are nodes.

Definition 4: A **cable** is an ordered pair $\langle r, ns \rangle$, where r is a relation, and ns is a non-empty nodeset. Meta-variables $c, c_1, c_2, \ldots$ range over cables. *Example:* <`member`, {`john, bill`}> is a cable.

Definition 5: A **cableset** is a non-empty set of cables, $\{\langle r_1, ns_1 \rangle, \ldots, \langle r_k, ns_k \rangle\}$, such that $r_i = r_j \iff i = j$. Meta-variables $cs, cs_1, cs_2, \ldots$ range over cablesets. *Example:* {<`member`, {`john, bill`}>, <`class`, {`man`}>} is a cableset.

Definition 6: Every node is either a base node or a cableset. *Example:* `bill` is a base node, {<`member`, {`john, bill`}>, <`class`, {`man`}>} is a cableset.

Definition 7: We overload the **membership** relation "$\in$" so that $x \in s$ holds just under the following conditions:

1. If x is a node and s is a nodeset, $x \in s \iff \exists y\ [y \in s \wedge Subsume(y, x).]$
   Example: `M1` $\in$ {`M1, M2, M3`}

2. If x is a wire such that $x =$ <r_1, n>—, and s is a cable such that $s =$ <r_2, ns>, then
   $x \in s \iff r_1 = r_2 \wedge n \in ns$.
   Example:
   <`member, john`> $\in$ <`member, {john, bill}`>

3. If x is a wire and s is a cableset, then $x \in s \iff \exists c[c \in s \wedge x \in c]$.
   Example: <`member, john`> $\in$ {<`member, {john, bill}`>, <`class, {man}`>}

Because we need more definitions before *Subsume* can be defined, we defer its definition to Figure 2.

Definition 8: An **nrn-path** from the node n_1 to the node n_{k+1} is a sequence, $n_1, r_1, \ldots, n_k, r_k, n_{k+1}$, for $k \geq 1$ where the n_i are nodes, the r_i are relations, and for each i, <r_i, n_{i+1}> $\in n_i$. *Example:* If `M1` = {<`member, {john, bill}`>, <`class, {man}`>}, then `M1, member, john` and `M1, class, man` are some nrn-paths.

Definition 9: A node n_1 **dominates** a node n_2 just in case there is an nrn-path from n_1 to n_2. The predicate *dominate*(n_1, n_2) is true if and only if n_1 dominates n_2. *Example:* If `M1` = {<`member, {john, bill}`>, <`class, {man}`>}, then `M1` dominates `john`, `bill`, and `man`.

Definition 10: A **variable** node is a cableset of the form {<*any, ns*>} (**universal variable node**) or {<*some*, ns_1>, <*depends*, ns_2>} (**existential variable node**). Additionally, a node is a variable node only if:

1. If it has the form {<*any, ns*>}, then every $n \in ns$ *must* dominate it.

2. If it has the form {<*some*, ns_1>, <*depends*, ns_2>}, then every $n \in ns_1$ *must* dominate it *and* every $n \in ns_2$ *must* be a universal variable node.

Example: `V1` = {<`any`, {{<`member, {V1}`>, <`class, {man}`>}}>} is the variable node corresponding to *every man*. The variable label `V1` is just a convenient extra-theoretic method of referring to the variable.

We define two selectors for variable nodes:

$$rest(v) = \begin{cases} ns & \text{if } v = \{<any, ns>\} \\ ns_1 & \text{if } v = \{<some, ns_1>, <depends, ns_2>\} \end{cases}$$

$$depends(v) = ns_2 \quad \text{if } v = \{<some, ns_1>, <depends, ns_2>\}$$

Informally, *rest*(v) is the set of restriction propositions on the types of things that may be bound to the variable node v. *Depend*(v) is the set of universal variable nodes on which the existential variable node, v, is scope-dependent.

Definition 11: A **molecular** node is a cableset that is *not* a variable node. *Example:* {<`member`, {`john, bill`}>, <`class`, {`man`}>} is a molecular node, since it is a cableset but not a variable node.

Definition 12: A **rule** node is a molecular node that dominates a variable node that does not, in turn, dominate it. *Example:*
Given the labelled nodes below:
`V1` = {<`any`, {`M1`}>}
`M1` = {<`member`, {`V1`}>, <`class`, {`man`}>}
`M2` = {<`member`, {`V1`}>, <`class`, {`mortal`}>}
`M2` is a rule node since `M2` dominates `V1`, which does not, in turn, dominate `M2`. `M1` is *not* a rule node because, while it dominates `V1`, it is also dominated by `V1`. The non-rule nodes that dominate variable nodes correspond to restrictions on binders of those same variable nodes.

Definition 13: A **wireset** of a node n is defined as $\{w | w \in n\}$. *Example:* If `M2` = {<`member`, {`john, bill`}>, <`class`, {`man`}>} then: *wireset*(`M2`) = {<`member, john`>, <`member, bill`>, <`class, man`>}.

2.5 The ANALOG model

Definition 14: An **ANALOG model** is a tuple (A, B, M, R, U, E, ,) where A is a set of relations, B is a set of base nodes, M is a set of non-rule molecular nodes, R is a set of rule nodes, U is a set of universal variable nodes, and E is a set of existential variable nodes, and , $\subseteq M \cup R$. B, M, R, U, and, E are disjoint. , consists of the set of asserted nodes.

2.6 Reduction

We follow [Shapiro, 1986, Shapiro, 1991] in arguing for a form of reduction inference (defined in Axioms 2 and 3 below) as being useful. This is a form of structural subsumption [Woods, 1991], peculiar to semantic network formalisms, which allows a proposition to "reduce" to (logically imply) propositions whose wires are a subset of the wires of the original proposition. Figure 1 gives an example of a proposition expressing a brotherhood relation among a group of men. Node `M1` represents the proposition that `bill, john, ted,` and `joe` are brothers. By reduction subsumption, all proposition nodes (such as `M2` and `M3`) involving fewer brothers follow.

However, we must restrict the use of reduction inference to precisely those propositions and rules which are

In the following model (A, B, M, R, U, E, ,):

A ={relation, arg}, B ={bill, john, ted}, M ={M1, M2, M3}, , ={M1, M2, M3}

where:

```
M1 = {<relation, {brothers}>, <arg, {bill, john, ted}>}
M2 = {<relation, {brothers}>, <arg, {john, ted}>}
M3 = {<relation, {brothers}>, <arg, {bill, john}>}
```

Some reductions: *Reduce*(M2, M1), and *Reduce*(M3, M1)

Figure 1: Example of Subsumption by Reduction for a Particular Model

reducible through the use of the *IsReducible* metapredicate.

Axiom 2: $Reduce(cs_1, cs_2) \iff (\forall w[w \in cs_2 \Rightarrow w \in cs_1] \wedge IsReducible(cs_1, cs_2))$.

Note that the semantics of the metapredicate *IsReducible* will be specified in terms of the particular case frames used in a representation language. Propositions like M1 are clearly reducible, but not all propositional case frames are reducible. For example,

$$\forall x((\text{man}(x) \wedge \text{rich}(x)) \Rightarrow \text{happy}(x)) \quad (1)$$

should not allow the reduced proposition:

$$\forall x(\text{man}(x) \Rightarrow \text{happy}(x)) \quad (2)$$

which involves fewer constraints on x than than proposition (1), as the latter does not follow from the former. New propositions derived by reduction should be implied by the propositions from which they are derived. For example note that reduction to proposition (2) is appropriate when the constraints in the antecedent of the rule are disjunctive as in:

$$\forall x((\text{man}(x) \vee \text{rich}(x)) \Rightarrow \text{happy}(x)) \quad (3)$$

IsReducible should be define appropriately for the particular representation language to allow (or disallow) these reductions. In Section 3.3 we specify some of the reducible case frames we use in this paper.

A proposition that is a reducible reduction of a believed proposition is also a believed proposition. Since nodes are, by definition, cablesets, we state this as in Axiom 3.

Axiom 3: $(Reduce(n_1, n_2) \wedge Believe(n_1)) \Rightarrow Believe(n_2)$

2.7 Types of Nodes

We have defined four types of nodes: base, molecular, rule, and variable nodes. Informally, base nodes correspond to individual constants in a standard predicate logic, molecular nodes to sentences and functional terms, rule nodes to closed sentences with variables, and variable nodes to variables. Note that syntactically all are terms in ANALOG, however.

3 Semantic Issues

ANALOG is specified in terms of a propositional semantic network. By this is meant that all information, including propositions, "facts", etc., is represented by nodes. Labelled arcs are purely structural and carry no assertional import. The benefit of representing propositions by nodes is that propositions about other propositions may be represented. This means that ANALOG is not strictly first-order as variables may quantify over nodes that correspond to propositions or predicates rather than individuals. This has useful consequences when processing natural language, particularly questions whose answers are propositions, in that *any* node (including non-base nodes) can be bound to the variable associated with a question.

3.1 Intensional Representation

ANALOG nodes correspond to intensional entities. What is being represented, in ANALOG, is an agent's mind. Objects in that mind need not represent any extensional object in the world. To connect objects of mind to extensional objects in the world, ANALOG uses a case frame with a lex arc to the object of thought denoting its extension. Additionally, sensory nodes can also make this connection to the external world.

3.2 The Uniqueness Principle

No two nodes in the network represent the same individual, proposition, or rule.

Axiom 4: $n_1 = n_2 \iff [\![n_1]\!] = [\![n_2]\!]$

This is a consequence of the intensional semantics, since nodes correspond to objects of thought in an agent's mind. The objects of thought are intensional: a mind can have two or more objects of thought that correspond to only one extensional object, or no extensional object. The classic example of this is the Morning Star and the Evening Star which might be distinct objects of thought, but have a single extensional referent. Thus the nodes representing the Morning Star and the Evening Star are distinct, and any node can denote only *one* intensional object which no other node can denote [Maida and Shapiro, 1982].

A benefit of this is a high degree of structure-sharing in large networks. Additionally, the network representation of some types of sentences can reflect the re-use of natural language terms expressed by pronouns and other reduced forms.

3.3 Case Frame Semantics

ANALOG can support any propositional representations that have a consistent syntax and semantics. In this paper, examples of representations used will follow the syntax and semantics of [Shapiro and Rapaport, 1987]. Here we describe only two case frames (those used in the brothers example of Figure 1), due to space limitations. We specify their syntax, semantics, and status for reduction.

1. $\{$<*member*, ns_1>, <*class*, ns_2>$\}$: For any $n_1 \in ns_1$, and any $n_2 \in ns_2$, $[\![n_1]\!]$ is a member of class $[\![n_2]\!]$. Valid reductions are specified by:

   IsReducible($\{$<*member*, $ns_1 \cup ns_3$>, <*class*, $ns_2 \cup ns_4$>$\}$, $\{$<*member*, ns_1>, <*class*, ns_2>$\}$)

   where $ns_1 \neq \{\}$, and $ns_2 \neq \{\}$.

2. $\{$<*relation*, ns_1>, <*arg*, ns_2>$\}$: For every $n_1 \in ns_1$, and every $n_2, n_3 \in ns_2, n_2 \neq n_3$, $[\![n_2]\!]$ is in relation $[\![n_1]\!]$ to $[\![n_3]\!]$, and $[\![n_3]\!]$ is in relation $[\![n_1]\!]$ to $[\![n_2]\!]$. Valid reductions are specified by:

   IsReducible($\{$<*relation*, $ns_1 \cup ns_3$>, <*arg*, $ns_2 \cup ns_4$>$\}$, $\{$<*relation*, ns_1>, <*arg*, ns_2>$\}$)

   where $ns_1 \neq \{\}$, and $|ns_2| \geq 2$

Note that most of the valid reductions of these propositional case frames follow directly from their semantics. There are numerous other case frames that are useful and necessary (for a complete dictionary of them, see [Shapiro *et al.*, 1993]).

4 Subsumption

Semantic network formalisms provide "links" that relate more general concepts to more specific concepts; this is called a *taxonomy*. It allows information about concepts to be associated with their most general concept, and it allows information to filter down to more specific concepts in the taxonomy via inheritance. More general concepts in such a taxonomy *subsume* more specific concepts, the subsumee inheriting information from its subsumers. For atomic concepts, subsumption relations between concepts are specified by the links of the taxonomy. We specify subsumption for non-atomic concepts, below.

Definition 15: A **binding** is a pair v/u, where either u is a universal SV and v is any node or u and v are both existential nodes. *Examples:* `V1/V2`, `JOHN/V1`.

Definition 16: A **substitution** is a (possibly empty) set of bindings, $\{t_1/v_1, \ldots, t_n/v_n\}$. *Examples:* {`V1/V2, JOHN/V3`}, {`B1/V1, M1/V2`}.

Definition 17: The result of **applying** a substitution, $\theta = \{t_1/v_1, \ldots, t_m/v_m\}$, to a node n is the instance $n\theta$ of n obtained by simultaneously replacing each of the v_i dominated by n with t_i. If $\theta = \{\}$, then $n\theta = n$. *Example:* If `M1 = {<member, {V1}>, <class, {MAN}>}` then: `M1{JOHN/V1} = {<member, {JOHN}>, <class, {MAN}>}`

Definition 18: Let $\theta = \{s_1/u_1, \ldots, s_n/u_n\}$ and $\rho = \{t_1/v_1, \ldots, t_m/v_m\}$ be substitutions. Then the **composition** $\theta \cdot \rho$ of θ and ρ is the substitution obtained from the set: $\{s_1\rho/u_1, \ldots, s_n\rho/u_n, t_1/v_1, \ldots, t_m/v_m\}$ by deleting any binding $s_i\rho/u_i$ for which $u_i = s_i\rho$. *Example:* If $\theta =$ {`V1/V2, V4/V3`}, and $\rho =$ {`V2/V1, JOHN/V4`} then $\theta \cdot \rho$ = {`JOHN/V3, JOHN/V4`}.

Definition 19: A substitution, θ, is **consistent** iff neither of the following hold:

$$\exists u, t, s[t/u \in \theta \wedge s/u \in \theta \wedge s \neq t]$$
$$\exists u, v, t[t/u \in \theta \wedge t/v \in \theta \wedge u \neq v]$$

We note that this is different from the standard definition of consistency for substitutions. A substitution that is not consistent is termed **inconsistent**. The motivation for the second constraint (called the unique variable binding rule, UVBR) is that in natural language, users seldom want different variables in the same sentence to bind identical objects [Shapiro, 1986]. For example, *Every elephant hates every elephant* has a different interpretation from *Every elephant hates himself.* Typically, the most acceptable interpretation of the former sentence requires that it not be interpreted as the latter. UVBR requires that within an individual sentence that is a rule

Subsume(x, y) in a model $(A,\ B,\ M,\ R,\ U,\ E,\ ,\ )$ if and only if, one of:

1. $x = y$.
2. *Reduce*(x, y)
3. For $x \in U$ and $y \in B \cup M \cup R$, if not *occurs-in*$(x,\ y)$ and there exists a substitution S such that

$$\forall r[r \in rest(x),\ ,\ \vdash r\{y/x\} \cdot S].$$

Logical derivation is here denoted by "$\vdash$."

4. For $x \in U$ and $y \in U \cup E$, if

$$\forall r[r \in rest(x) \Rightarrow \exists s[s \in rest(y) \wedge Subsume(r, s)]]$$

5. For $x, y \in E$, if all of the following hold:

$$\forall s[s \in rest(y) \Rightarrow \exists r[r \in rest(x) \wedge Subsume(r, s)]]$$
$$\forall r[r \in rest(x) \Rightarrow \exists s[s \in rest(y) \wedge Subsume(r, s)]]$$
$$\forall d[d \in depends(y) \Rightarrow \exists c[c \in depends(x) \wedge Subsume(c, d)]]$$

Figure 2: Subsumption Procedure

In the following model $(A,\ B,\ M,\ R,\ U,\ E,\ ,\ )$:

A ={`member, class, any`}, B ={`man, mortal, Socrates`}, M ={`M1, M3, M4`}, R ={`M2`}, U ={`V1`}

where:

```
M1 = {<member, {V1}>,<class, {man}>},          M2 = {<member, {V1}>,<class, {mortal}>}
M3 = {<member, {Socrates}>,<class, {man}>},    M4 = {<member, {Socrates}>,<class, {mortal}>}
V1 = {<any, {M1}>}
```

The resulting subsumption: *Subsume*(`M2, M4`).

Figure 3: Example of Subsumption for a Particular Model

(has bound variables), any rule use (binding of variables) must involve different terms for each variable in the rule to be acceptable. *Examples:*

{`JOHN/V2, BILL/V2`} is inconsistent.
{`JOHN/V1, JOHN/V2`} is inconsistent.
{`JOHN/V1, BILL/V2`} is consistent.

Definition 20: The predicate *occurs-in*(x, y) where x is a variable is defined: *occurs-in*$(x, y) \iff$ *dominate*(y, x). *Occurs-in* is used for the standard occurs check of the unification algorithm (and is just a more perspicacious naming of *dominate*).

In ANALOG, we specify subsumption as a binary relation between arbitrary nodes in the network. We define subsumption between two nodes x and y in Figure 2. This definition of subsumption includes subsumption mechanisms that Woods classifies as *structural, recorded, axiomatic,* and *deduced* subsumption [Woods, 1991]. In Figure 2, case (1) corresponds to identical nodes (a node, obviously, subsumes itself). Case (2) is the reduction inference case discussed in section 2.6. Case (3) applies when a universal structured variable node subsumes another node. This corresponds to a description like *any rich man* subsuming *John* if *John* is known to be a man and rich. Such a variable will subsume another node if and only if every restriction on the variable can be derived (in the current model) for the node being subsumed. Subsumption, consequently, requires derivation. For the examples shown here, standard first-order logical derivation may be assumed. Case (4) allows a more general universal variable node to subsume a less general existential variable node. For this to happen, for every restriction

in the universal variable node there must be a restriction in the existential variable node, and the former restriction must subsume the latter restriction. For example, the variable node corresponding to *every rich girl* would subsume *some rich happy girl* (but not *some girl*). Case (5) allows one existential variable node to subsume another. The requirement for this case is, essentially, that the variables be notational variants of each other, except for those universal structured variables they are scope dependent upon. This is because it is not, in general, possible for any existential variable to subsume another except when they are structurally identical. The reason this case is needed (rather than just excluding it entirely) is that for a rule node corresponding to *every boy loves some girl* to subsume *every rich boy loves some girl*, the existential variable node corresponding to the *some girl* in the first rule node must subsume the existential variable node corresponding to the *some girl* in the second rule node.

In Figure 3, a more detailed example for a particular model is given. Node `M2` represents the proposition that *all men are mortal*, `M3` the proposition that `Socrates is a man`, and `M4` the proposition that `Socrates is mortal`. `V1` is the structured variable representing any man. It then follows that `M4` is a special case of `M2` directly by subsumption, since `V1` subsumes `Socrates`. Note that the restrictions on subsumption involving variables is stricter than *Reduce*, which only requires that the wires of one node be a subset of the other.

As with reduction (Axiom 3), a proposition that is subsumed by a believed proposition is also a believed proposition. This can be stated as a more general form of Axiom 3.

Axiom 5: $(Subsume(n_1, n_2) \wedge Believe(n_1)) \Rightarrow Believe(n_2)$

Axiom 5 allows the sorts of commonsense description subsumption evident in natural language.

5 Node Matching

Matching in ANALOG is the process of determining the most general common instance (MGI) of two nodes such that one instance subsumes the other instance. Matching is specified in terms of nodes (and the arcs connecting them) and sets of substitutions (*substitutionsets*) for variables in the nodes.

The process looks like Figure 4. When two nodes are being considered to determine a subsumption relationship (n_1 and n_2), the matcher attempts to find two substitutions s and t (called the source and target substitutions, respectively), such that if s is applied to n_1 to produce n_3, and if t is applied to n_2 to produce n_4, then n_3 will subsume n_4. Given two nodes, the matcher returns a set of (s_i, s_j) pairs that are possible substitutions

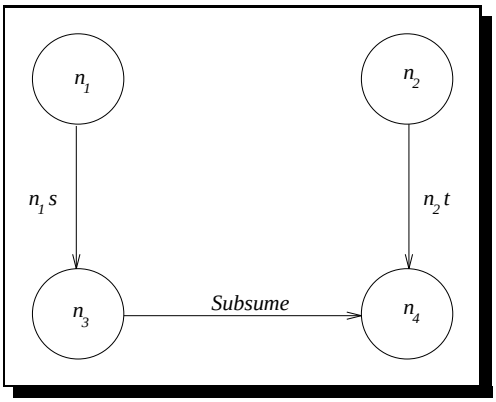

Figure 4: Pictorial View of Node Matching Process

for the two nodes that result in the subsumption relationship. If none is possible, the empty set is returned.

5.1 Match Algorithm

The two-way matching procedure takes a pair of nodes i, j and returns a substitution set consisting of source-target substitutions (s_i, t_i) such that $Subsume(is_i,\ jt_i)$. To account for the possibility of multiple source-target substitutions, match also takes a substitution set which constrains the possible matches (based on previous recursive matchings). *Match* is defined in Algorithm 1 of Figure 5 and works by recursively generating all possible consistent source-target substitutions on a case by case basis. Each case of the match procedure potentially adds a new binding to the source or target substitutions in the substitution set that is the result of the matching.

In Algorithm 1, case (1) will succeed only if the two nodes are identically the same node. This follows from the uniqueness principle. Since this match involves no variables, no new bindings are added in the event of a successful match. This case occurs when matching two terms that share a subterm. This is particularly likely to occur for base nodes, since there will only be one base node for any intensional individual in the network. Cases (2) and (3) deal with attempted matches where one node is a universal variable. If the universal variable node subsumes the other node (subject to the current source and target substitutions), a new binding is added to all the source-target substitution sets and the consistent source-target substitutions become the resulting substitution set. Cases (4) and (5) deal with attempted matchings where both nodes are existential variable nodes. Depending on whether the source subsumes the target or the target subsumes the source (subject to the current source and target substitutions), a new binding is added to all the source or target substitutions and the consistent source-target substitutions become the resulting substitution set. Case (6) is the general case and deals with attempts to match two molecular or rule nodes. Both the source and target nodes are converted into their equivalent wiresets,

and all possible matchings of wires are attempted to determine if the source node can match the target node. This is done using the *matchwires* function in Algorithm 2. *Matchwires* works by attempting to find consistent source-target substitutions such that all wires of a source instance's wireset are in the target instance's wireset.

6 Node Inference

The primary form of inference in ANALOG is instantiation, that is, replacing more statements with more specific statements by applying substitutions generated by the matcher. The metarule for this is:

$$\frac{\begin{array}{l} Believe(n_1) \\ Conceive(n_2) \\ (s,t) \in match(n_1, n_2, \{\}) \end{array}}{Believe(n_2 t)}$$

In this metarule, n_1 is a node that represents a statement that is believed, and n_2 is a node that is merely conceived, typically corresponding to a question. If match returns a substitution pair that allows the belief of an instance of n_2 this rule allows the system to believe that instance. Because syntactically similar natural language statements are mapped into structurely similar representations (including questions) this rule allows general inference, as illustrated in the next section.

7 ANALOG for NLP

So far, we have motivated some aspects of the logic underlying the ANALOG knowledge representation and reasoning system and formalized some important concepts, such as subsumption, associated with the logical system. At this point, we will attempt to illustrate the utility of the system for natural language processing with specific examples the system modelling natural language use.

ANALOG includes a generalized augmented transition network (GATN) natural language parser and generation component linked up to the knowledge base (based on [Shapiro, 1982a]). A GATN grammar specifies the translation/generation of sentences involving complex noun phrases into/from ANALOG structured variable representations.

The representation of structured variables suggested here can represent first-order quantifying expressions directly. In general, there is a direct mapping from natural language quantifying expressions into structured variable representations, since structured variables correspond directly to noun phrases with restrictive relative clause complements. The natural language processing subsystem of ANALOG is based on an interpreter for generalized augmented transition network (GATN) grammars [Woods, 1970, Shapiro, 1982b]. These are networks of nodes and arcs, similar to finite-state machines, in which arcs are labeled by parts of speech or other GATN network names. Transitions from state to state occur when the correct parts of speech are matched to words in the text to be parsed. Each transition consumes the matched portion of the input. Because these networks are non-deterministic all possible correct parses are found.

Figure 8 shows a fragment of the GATN grammar that processes noun phrases corresponding to structured variables. So in Figure 8, if the text to be processed were: `the happy man` the sequence of transitions would be NP, NP1, NP1, NP2. The arc labeled "pop" out of state NP2 corresponds to a successful parse. The NP subnetwork processes surface noun phrases and builds the appropriate representation by processing articles, adjectives, and restrictive relative clauses.

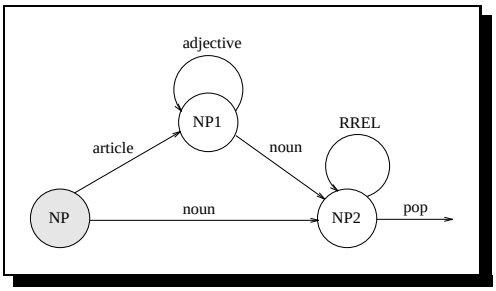

Figure 8: GATN NP subnetwork for noun phrases corresponding to structured variables.

We present two demonstrations of the NLP component of ANALOG. The first illustrates the representation and use of complex noun phrases, the second is a demonstration that illustrates the use of description subsumption and inference in providing useful answers to questions.

7.1 Representation of Complex Noun Phrases

An apparent advantage of the use of structured variables lies in the representation and generation of complex noun phrases that involve restrictive relative clause complements. The restriction set of a structured variable typically consists of a type constraint along with property constraints (adjectives) and other more complex constraints (restrictive relative clause complements). So, when parsing a noun phrase all processing is localized and associated with building its structured variable representation. When generating a surface noun phrase corresponding to the structured variable, all constraints associated with the variable are part of its structure and can be collected and processed easily. This is in contrast to non-structured variable representations (such as first-order predicate logic) where the restrictions on variables are disassociated from the variables themselves, in the antecedents of rules.

In Figure 6, user input is italicized, the text at the beginning is a standard message and will be omitted

from the remaining figure. Figure 6 shows example sentences with progressively more complex noun phrases being used. These noun phrases are uniformly represented using structured variables. Parsing and generation of these noun phrases is simplified because structured variables collect all relevant restrictions on a variable into one unit, a structured variable. The parser parses the user's sentence and builds an ANALOG representation for the user input. The resulting representation is then passed to the generation component, which generates the output response (sometimes prefixed by the canned phrase `I understand that`). Notice how, in Figure 6, the descriptions in the sentences get progressively more complex. Because structured variables collect all constraints on a variable into one term, it is relatively simple to parse complex natural language noun phrases, as in the examples. Similarly, for generation of language from the representation, it is simple to generate complex natural language descriptions from structured variables. If constraints on variables corresponding to the complex noun phrases were represented using first-order predicate logic, then it would be difficult to generate natural language noun phrases corresponding to these variables. This is because the constraints on variables would, likely, be well-separated from the variables in the antecedents of rules involving these variables. This is not the case in a structured variable representation.

7.2 Description Subsumption and Question Answering

Because the structure of the representation of rules is "flat", that is, there is not the artificial antecedent-consequent structure associated with first-order logic-based representations, it is possible to frame questions whose answers are rules and not just ground formulas. Since the structure of the question will mirror the structure of the rule, any rule that is subsumed by a question is an answer to that question. Figure 7 gives a sample dialog involving questions whose answers are ground propositions (e.g., *Is John mortal*) as well as questions whose answers are rules (e.g., *Who is mortal*). This dialog also illustrates the uses of subsumption. Since we told the system *Every man is mortal*, it follows that any more specifically constrained man (e.g., *Every rich young man that owns some car*) must also be mortal. Note that this answer (a rule) follows directly by subsumption from a rule previously told to the system. This is another way in which rules may be answers to questions.

We examine the subsumption inference in Figure 7 in detail. All references to sentences will be to the numbered sentences in Figure 7. After sentence (1) is processed, sentence (3) then asks who is mortal. In a standard first-order-predicate-logic-based system, no answer could be given because there are, as yet, no instances of men in the knowledge base. This is contrary to the commonsense answer of sentence (4), which reiterates the rule of sentence (1). This is possible in ANALOG because the structure of the representation of the question (*Who is mortal*) is similar to that of any of its answers. Thus, any asserted proposition that is subsumed by the question is a valid answer (including rules).

Sentence (5) is an example of a question about a rule. Since *every man is mortal* is believed (the system was told this in sentence (1)) it follows that any more restricted sort of man is also mortal. The subsumption procedure specifies this explicitly. The representation of sentence (5) in is a less general form of sentence (1), since *any man* subsumes *any rich man*. Since the rule of sentence (5) is subsumed by a believed node (that of sentence (1)), it follows by Axiom 5 that sentence (5) is believed (thus, the representation of the question itself is a believed proposition) and the system answers yes to the question. Sentence (7) asserts that *John is a man*. At this point, the system knows *all men are mortal* and *John is a man*. When the question of sentence (9) is asked, the system finds the rule of sentence (1) and determines that it subsumes sentence (9) because John is a man and, again by axiom 5, the result follows. However, note that in this derivation the result is a ground formula rather than a rule. Sentence (11) illustrates the retrieval of the system's information about who is mortal; note the additional believed propositions. Sentence (13) is an example of a more complex noun phrase in a rule. The representation of (13) is subsumed by that of sentence (1) or (5) leading to the yes answer. In sentence (15), a new rule about rich young car-owning men being happy is introduced. The question of sentence (17) (*is John happy*) cannot be answered, because the system cannot determine that any rich young car-owning man subsumes `John`. This is because, while `John` is a man, he is not known to be young, rich, and owning a car (requirements for this subsumption). Sentence (19) informs the system of these requirements the systems understanding is verified by question (21). Note that the structure of the question involving two variables (*Who* and *a car*) is identical to that of the structure of its answer, which would not be the case if constraints were separated from variables in the antecedents of rules (as is done in typical logics). The question is asked again and, because the subsumption relationship can be determined, is answered in the affirmative.

8 Summary

We have described a logical language designed for natural language processing. We have shown how the primary means of automated deduction is a form of subsumption, that models reasoning methods used in natural language. We have illustrated the suitability of our work for natural language processing.

References

[Ali and Shapiro, 1993] Syed S. Ali and Stuart C. Shapiro. Natural Language Processing Using a Propositional Semantic Network with Structured Variables. *Minds and Machines*, 3(4):421–451, November 1993. Special Issue on Knowledge Representation for Natural Language Processing.

[Ali, 1993a] Syed S. Ali. A Propositional Semantic Network with Structured Variables for Natural Language Processing. In *Proceedings of the Sixth Australian Joint Conference on Artificial Intelligence.*, November 17-19 1993.

[Ali, 1993b] Syed S. Ali. A Structured Representation for Noun Phrases and Anaphora. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 197–202, Hillsdale, NJ, June 1993. Lawrence Erlbaum.

[Ali, 1993c] Syed S. Ali. Node Subsumption in a Propositional Semantic Network with Structured Variables. In *Proceedings of the Sixth Australian Joint Conference on Artificial Intelligence.*, November 17-19 1993.

[Ali, 1994] Syed S. Ali. *A "Natural Logic" for Natural Language Processing and Knowledge Representation.* PhD thesis, State University of New York at Buffalo, Computer Science, January 1994.

[Beierle *et al.*, 1992] C. Beierle, U. Hedtstuck, U. Pletat, P. H. Schmitt, and J. Siekmann. An Order-sorted Logic for Knowledge Representation Systems. *Artificial Intelligence*, 55(2-3):149–191, June 1992.

[Brachman and Schmolze, 1985] Ronald J. Brachman and J. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, 9(2):171–216, 1985.

[Brachman *et al.*, 1985] Ronald J. Brachman, Victoria Pigman Gilbert, and Hector J. Levesque. An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON. *Proceedings IJCAI-85*, 1:532–539, 1985.

[Fine, 1985] Kit Fine. *Reasoning with Arbitrary Objects.* Basil Blackwell, Oxford, 1985.

[Maida and Shapiro, 1982] A. S. Maida and S. C. Shapiro. Intensional Concepts in Propositional Semantic Networks. *Cognitive Science*, 6(4):291–330, 1982. Reprinted in *Readings in Knowledge Representation*, R. J. Brachman and H. J. Levesque (eds.), Morgan Kaufmann, San Mateo, CA, 1985, 170–189.

[Morgado, 1986] E. J. M. Morgado. Semantic Networks as Abstract Data Types. Technical Report 86–19, Department of Computer Science, SUNY at Buffalo, 1986.

[Shapiro and Rapaport, 1987] S. C. Shapiro and W. J. Rapaport. SNePS Considered as a Fully Intensional Propositional Semantic Network. In N. Cercone and G. McCalla, editors, *The Knowledge Frontier*, pages 263–315. Springer–Verlag, New York, 1987.

[Shapiro *et al.*, 1993] S. C. Shapiro, W. J. Rapaport, and the SNePS Research Group. A Dictionary of Case Frames, 1993. In preparation.

[Shapiro, 1982a] S. C. Shapiro. Generalized augmented transition network grammars for generation from semantic networks. *The American Journal of Computational Linguistics*, 8(1):12–25, 1982.

[Shapiro, 1982b] S. C. Shapiro. Generalized Augmented Transition Network Grammars for Generation from Semantic Networks. *The American Journal of Computational Linguistics*, 8(1):12–25, 1982.

[Shapiro, 1986] S. C. Shapiro. Symmetric relations, intensional individuals, and variable binding. *Proceedings of the IEEE*, 74(10):1354–1363, 1986.

[Shapiro, 1991] S. C. Shapiro. Cables, Paths, and "Subconscious" Reasoning in Propositional Semantic Networks. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 137–156. Morgan Kaufmann, 1991.

[Woods, 1970] W. A. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606, 1970.

[Woods, 1991] William A. Woods. Understanding Subsumption and Taxonomy: A Framework for Progress. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 45–94. Morgan Kaufmann, 1991.

Algorithm 1 *Match(i, j, S)* in a model $(A,\ B,\ M,\ R,\ U,\ E,\ ,\ )$

IF $i = j$ THEN (1)
  RETURN S
ELSEIF $i \in U$ and $Subsume(ia_k, jb_k)$ THEN (2)
  RETURN $\{(a_k \cdot \{j/i\}, b_k) \mid (a_k, b_k) \in S \wedge a_k \cdot \{j/i\}$ is consistent.$\}$
ELSEIF $j \in U$ and $Subsume(jb_k, ia_k)$ THEN (3)
  RETURN $\{(a_k, b_k \cdot \{i/j\}) \mid (a_k, b_k) \in S \wedge b_k \cdot \{j/i\}$ is consistent.$\}$
ELSEIF $i, j \in E$ THEN
  IF $Subsume(ia_k, jb_k)$ THEN (4)
    RETURN $\{(a_k \cdot \{j/i\}, b_k) \mid (a_k, b_k) \in S \wedge a_k \cdot \{j/i\}$ is consistent.$\}$
  ELSEIF $Subsume(jb_k, ia_k)$ THEN (5)
    RETURN $\{(a_k, b_k \cdot \{i/j\}) \mid (a_k, b_k) \in S \wedge b_k \cdot \{j/i\}$ is consistent.$\}$
  END
ELSEIF $i, j \in M \cup R$ THEN (6)
  RETURN *matchwires(wireset(i), wireset(j), S)*
ELSE
  RETURN `fail`
END

Algorithm 2 *Matchwires(ws_1, ws_2, S)*

IF $S =$ fail or $ws_1 = \{\}$ THEN
  RETURN `fail`
ELSEIF $ws_2 = \{\}$ THEN
  RETURN S
ELSE

$$S \leftarrow \bigcup_{\substack{<r,n> \in ws_1 \\ <r,m> \in ws_2}} \left[\begin{array}{ll} matchwires(& ws_1 - \{<r,n>\}, \\ & ws_2 - \{<r,m>\}, \\ & match(n,m,s)) \end{array} \right] - \{\texttt{fail}\}$$

  IF $S = \{\}$ THEN
    RETURN `fail`
  ELSE
    RETURN S
  END
END

Figure 5: Match and Matchwires Algorithms

```
(parse -1)
ATN parser initialization...
Input sentences in normal English orthographic convention.
Sentences may go beyond a line by having a space followed by a <CR>
To exit the parser, write ^end.
: Every man owns a car
I understand that every man owns some car.
: Every young man owns a car
I understand that every young man owns some car.
: Every young man that loves a girl owns a car that is sporty
I understand that every young man that loves any girl owns some sporty car.
: Every young man that loves a girl that owns a dog owns a red car that is sporty
I understand that every young man that loves any girl that owns any dog owns some red
sporty car.
: Every young man that loves a girl and that is happy owns a red sporty car that wastes gas
I understand that every young happy man that loves any girl owns some sporty red car that
wastes gas.
: ^end
ATN Parser exits...
```

Figure 6: Examples of complex noun phrase use that correspond to structured variables

```
: Every man is mortal                                                    (1)
I understand that every man is mortal.                                   (2)
: Who is mortal                                                          (3)
Every man is mortal.                                                     (4)
: Is any rich man mortal                                                 (5)
Yes, every rich man is mortal.                                           (6)
: John is a man                                                          (7)
I understand that John is a man.                                         (8)
: Is John mortal                                                         (9)
Yes, John is mortal.                                                     (10)
: Who is mortal                                                          (11)
John is mortal and every rich man is mortal and every man is mortal.     (12)
: Are all rich young men that own some car mortal                        (13)
Yes, every young rich man that owns some car is mortal.                  (14)
: Any rich young man that owns any car is happy                          (15)
I understand that every young rich man that owns any car is happy.       (16)
: Is John happy                                                          (17)
I don't know.                                                            (18)
: Young rich John owns a car                                             (19)
I understand that mortal rich young John owns some car.                  (20)
: Who owns a car                                                         (21)
Mortal rich young John owns some car.                                    (22)
: Is John happy                                                          (23)
Yes, mortal rich young John is happy.                                    (24)
```

Figure 7: Deduced Subsumption with Complex Descriptions.