A LOGICAL-FORM AND KNOWLEDGE-BASE DESIGN FOR NATURAL LANGUAGE GENERATION*

Norman K. Sondheimer
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292

Bernhard Nebel
Technische Iniversitaet Berlin
Sekr. FR 5-8, Franklinstr. 28/29
D-1000 Berlin 10, West Germany

ABSTRACT

This paper presents a technique for interpreting output demands by a natural language sentence generator in a formally transparent and efficient way. These demands are stated in a logical language. A network knowledge base organizes the concepts of the application domain into categories known to the generator. The logical expressions are interpreted by the generator using the knowledge base and a restricted, but efficient, hybrid knowledge representation system. This design has been used to allow the NIGEL generator to interpret statements in a first-order predicate calculus using the NIKL and KL-TWO knowledge representation systems. The success of this experiment has led to plans for the inclusion of this design in both the evolving Penman natural language generator and the Janus natural language interface.

1. INTRODUCTION

We have as a general goal the development of natural language generation capabilities. Our vehicle for these capabilities will be a reusable module designed to meet all of a system's needs for generated sentences. The generation module must have an input notation in which demands for expression are represented. The notation should be of **general** applicability. For example, a good notation ought to be general useful in a reasoning system. The notation should have a **well-defined** semantics. In addition, the generator has to have some way of **interpreting** the demands. This interpretation has to be **efficient**.

In our research, we have chosen to use **formal logic** as a demand language. **Network knowledge-bases** are used to define the domain of discourse in order to help the generator interpret the logical forms. And a **restricted, hybrid knowledge representation** is utilized to analyze demands for expression using the knowledge base.

Arguments for these decisions include the following: Formal logic is a well established means of expressing information with a well-defined semantics. Furthermore, it is commonly used in other aspects of natural language processing, as well as other AI systems. Network knowledge-base notations have been shown to be effective and efficient in language processing. Work on network representations has shown that they too can be given formal semantics [Schmolze and Lipkis 83]. Finally, recent work on hybrid knowledge representation systems has shown how to combine the reasoning of logic and network systems [Brachman 85]. Restricted-reasoning hybrid systems have shown this reasoning can be done efficiently.

On our project, we have:

1. Developed a demand language based on first order logic,

2. Structured a NIKL (New Implementation of KL-ONE) network [Kaczmarek 86] to reflect conceptual distinctions observed by functional systemic linguists.

3. Developed a method for translation of demands for expression into a propositional logic database,

4. Employed KL-TWO [Vilain 85] to analyze the translated demands, and

5. Used the results of the analyses to provide directions to the Nigel English sentence generation system [Mann & Matthiessen 83].

This paper presents our design and some of our experiences with it.

Others have attempted to design an interface between a linguistic generation engine and an associated software system using an appropriate information representation [Goldman 75, Appelt 83, Hovy 85, Kukich 85, Jacobs 85, McKeown 85]. Still others have depended on information demand representations with similar well-defined semantics and expressive power, e.g., [Shapiro 79]. However, he produces a logician's reading of expressions rather than colloquial English. For example, the popular song "Every man loves a woman.", might be rendered "For all men, there exists a woman that they love.".

The generation component of HAM-ANS [Hoeppner et al. 83] and one effort of McDonald's [McDonald 83] are probably closest to our design. HAM-ANS also uses a logical language (the same one used for representing the analyzed input), has an extensive network domain model, and has a separable linguistic engine (although not as broad in coverage as Nigel). However, the interface language is close to surface linguistic representation, e.g., there are particular expressions for tense and voice. So while it is easier to generate sentences from such structures, it is correspondingly **harder for software systems to produce demands for expressions** without having access to significant amounts of linguistic knowledge.

McDonald accepts statements in the first order predicate

*This research is supported by the Defense Advanced Research Projects Agency under Contract No MDA903 81 C 0335 and by the Air Office of Scientific Research under FQ8671-84-01007. Views and conclusions contained in this report are the author's and should not be interpreted as representing the official opinion or policy of DARPA, AFOSR, the U.S. Government, or any person or agency connected with them.

calculus, processes them with a grammar, and outputs excellent English forms. It is hard to evaluate the coverage of McDonald's grammar, however, the program does depend on extensive procedurally-encoded domain-dependent lexical entries (Actually, the entries are language fragments associated with terms in that can appear in the input language). Our domain dependencies are limited to the correct placement of concepts in the NIKL hierarchy and the association of lexical entries with the concepts. These lexical entries are only characterized by syntactic features.

In Section 2, we present the component technologies we have applied. Section 3 presents the method by which they are combined. Section 4 presents several examples of their use. We conclude with a section describing the open problems identified by our experiences and our plans for future work.

2. BASIC COMPONENTS

The processes and representations we have employed include a unique linguistic component (Nigel), a frame-based network knowledge representation (NIKL), a propositional reasoner that can take advantage of the network knowledge representation (KL-TWO), and our own first order logic meaning representation.

2.1. Nigel

The Nigel grammar and generator realizes the functional systemic framework [Halliday 76] at the level of sentence generation. Within this framework, language is viewed as offering a set of grammatical choices to its speakers. Speakers make their choices based on the information they wish to convey and the discourse context they find themselves in. Nigel captures the first of these notions by organizing minimal sets of choices into **systems**. The grammar is actually just a collection of these systems. The factors the speaker considers in evaluating his communicative goal are shown by questions called **inquiries** inside of the **chooser** that is associated with each system [Mann 83a]. A choice alternative in a system is chosen according to the responses to one or more of these inquiries.

For example, because the sentences describing different types of processes differ grammatically, the grammar has a system distinguishing the four major process types. This is shown graphically in Figure 2-1. The chooser associated with this system is represented as a decision in Figure 2-2. The nodes of the tree show inquiries. The branches are labeled with responses to the inquiries immediately to their left. The leaves show the choices made when that point is reached. For example, there is an inquiry, **VerbalProcessQ**, to test whether the process is one of communication. If the answer is "verbal", then the alternation "Verbal" is chosen in the system.

There are many inquiries like VerbalProcessQ. Elsewhere, as part of deciding on number, Nigel has an inquiry **MultiplicityQ** that determines whether an object being described is unary or multiple. These are examples of **information characterization** inquiries.

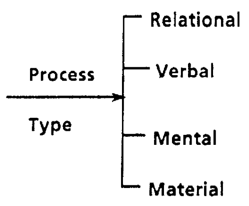

Figure 2-1: Sample Nigel System: Process Type

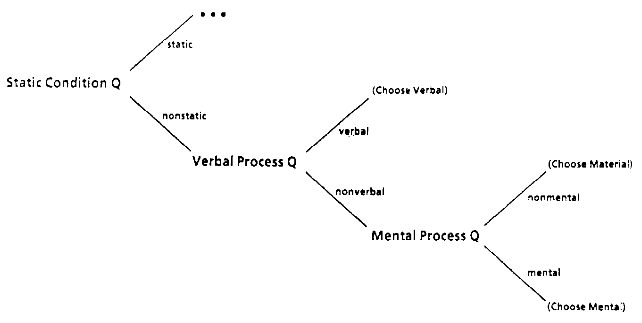

Figure 2-2: Chooser for the Process Type System

Another type of inquiry, called **information decomposition**, picks out of the environment the conceptual entities to be described. For example, at appropriate times, Nigel asks for descriptions of the causers of events, **CauserID**, or the objects affected in them, **AffectedID**.

One very special inquiry, **TermSpecificationID**, establishes the words that will be used. Nigel asks the environment for a set of lexical entries that can be used to describe an entity. So Nigel might find itself being told to describe some event as a "Send" or some object as a "Message".

Nigel currently has over 230 systems and 420 inquiries and covers a large subset of English.

Up until the effort described here, the developers of Nigel had only identified the inquiries of the grammar, but not implemented them.

2.2. NIKL

NIKL is a network knowledge-base system descended from KL-ONE [Brachman and Schmolze 85]. This type of reasoner supports description of the categories of objects, actions, and states of affairs that make up a domain. The central components of the notation are sets of concepts and roles, organized in IS-A hierarchies. The concepts are used to identify the categories of entities. The roles are associated with concepts (as "role restrictions"), and identify the relationships that can hold between actual individuals that belong to the categories. The IS-A hierarchies identify when membership in one category (or the holding of one relationship) entails membership in (or the holding of) another.

We have experimented with a mail and calendar NIKL domain model developed for the Consul project [Kaczmarek, Mark, and Sondheimer 83]. It has a concept **Send** that is meant to identify the activity of sending messages. **Send** IS-A type of **Transmit** (intended to identify the general activity of transmission of information). **Send** is distinguished from **Transmit** by having a role restriction **actee** that relates **Sends** to **Messages.** The concept of **Message** is defined as being a kind of a communication object, through the IS-A relation to a concept **Communication**. In addition, role restrictions connect **Message** to properties of messages which serve to distinguish it from other communication objects. The overall model has over 850 concepts with over 150 roles.

In flavor, NIKL is a frame system, with the concepts equivalent to frames and the role restrictions to slots. However, the NIKL representation can be given a formal semantics.

2.3. KL-TWO
KL-TWO is a hybrid knowledge representation system that uses NIKL's formal semantics. KL-TWO links another reasoner, PENNI, to NIKL. For our purposes, PENNI, which is an enhanced version of RUP [McAllester 82], can be viewed as restricted to reasoning using propositional logic. As such, PENNI is more restricted than those systems that use first order logic and a general purpose theorem prover. But it is also more efficient.

PENNI can be viewed as managing a data base of propositions of the form **(P a)** and **(Q a b)** where the forms are variable free[**]. The first item in each ordered pair is the name of a concept in an associated NIKL network and the first item in each ordered triple is the name of a role in that network. So the assertion of any form **(P a)** is a statement that the individual **a** is a kind of thing described by the concept **P**. Furthermore, the assertion **(Q a b)** states that the individuals **a** and **b** are related by the abstract relation described by **Q**.

NIKL adds to PENNI the ability to do taxonomic reasoning. Assume the NIKL database contained the concepts just described in discussing NIKL. Assume that we assert just the following three facts: **(Transmit x)**, **(actee x y)** and **(Message y)**. Using the knowledge base, PENNI is able to recognize that any **Transmit**, all of whose actees are **Messages**, is a **Send**. So if we ask if **(Send x)** is true, KL-TWO will reply positively.

KL-TWO can also retrieve information from its database. For example, if asked what individuals were the **actees** of **x**, it could respond with **y**.

2.4. THE LOGICAL LANGUAGE
Our logical language is based on first order logic. To it, we have added restricted quantification, i.e., the ability to restrict the set quantified over. In addition, we allow for equality and some related quantifiers and operators, such as the quantifier for "there exists exactly one ..." ($\exists!$), the operator for "the one thing that ..." (ι). We permit the formation and manipulation of sets, including a predicate for set membership (**ELEMENT-OF**). And we have some quantifiers and operators based on Habel's η operator [Habel 82].

Figure 2-3 gives three examples of the forms accepted. Included are a few individuals: two people (RICHARD and SMITH), the computer (COMPUTER), a set of messages (MM33), and the current time (NOW). Later on, we will show how these are turned into English by our system.

We include in our language a theory of the categories of conceptual entities and their relationships. We have taken what is often referred to as a Davidson approach [Davidson 67]. This is marked by quantification over events and state of affairs. We refer to these as **ActionOccurrences** and **RelationshipOccurrences**, respectively. We associate time and place with these entities. We differ from Davidson by identifying a class of abstract **Actions** and **Relationships** that are recorded by **ActionOccurrences** and **RelationshipOccurrences**. This approach is inspired by representations that associate time and place indices with formulas [Montague 74]. With **Actions** and **Relationships**, we associate the participants and circumstances of the actions and states-of-affairs, e.g., the actor and actee.

[**] PENNI actually works with the quantifier-free predicate calculus with equality. It has a demon-like facility capable of some quantificational reasoning as well.

$$\begin{array}{l}
\text{A. } (\exists x \in \mathrm{ActionOccurrence})((\exists p \in \mathrm{Past})(\mathrm{timeofoccurrence}(x,p)) \wedge \\
\quad (\exists y \in \mathrm{Transmit})(\mathrm{records}(x,y) \wedge \mathrm{actor}(y,\mathrm{SMITH}) \wedge \\
\qquad (\exists z \in \mathrm{Message})\mathrm{actee}(y,z)))
\end{array}$$

$$\begin{array}{l}
\text{B. } (\exists x \in \mathrm{ActionOccurrence})((\exists t \in \mathrm{Future})(\mathrm{timeofoccurrence}(x,t)) \wedge \\
\quad (\exists y \in \mathrm{Display})(\mathrm{records}(x,y) \wedge \mathrm{actor}(y,\mathrm{RICHARD}) \wedge \\
\quad\; \mathrm{requestedobject}(y,\iota q((\exists z \in \mathrm{ActionOccurrence}) \\
\qquad ((\exists p \in \mathrm{Past})(\mathrm{timeofoccurrence}(z,p)) \wedge \\
\qquad\; (\exists w \in \mathrm{Send})(\mathrm{records}(z,w) \wedge \mathrm{actor}(w,\mathrm{SMITH}) \wedge \mathrm{actee}(w,q))))) \wedge \\
\quad\; \mathrm{beneficiary}(y,\mathrm{COMPUTER})))
\end{array}$$

$$\begin{array}{l}
\text{C. } (\exists z \in \mathrm{ActionOccurrence}) \\
\quad (\exists d \in \mathrm{Display})(\mathrm{records}(z,d) \wedge \mathrm{actor}(d,\mathrm{RICHARD}) \wedge \\
\quad\; \mathrm{beneficiary}(d,\mathrm{COMPUTER}) \wedge \\
\quad\; (\forall m \in \mathrm{MM33})(\exists!\, r \in \{r | (\exists s \in \mathrm{RelationshipOccurrence}) \\
\qquad (\mathrm{timeofoccurrence}(s,\mathrm{NOW}) \wedge \\
\qquad (\exists t \in \mathrm{InspectionStatus}) \\
\qquad\quad (\mathrm{records}(s,t) \wedge \mathrm{range}(t,r) \wedge \mathrm{domain}(t,m)))\}) \\
\quad\;\; \mathrm{requestedobject}(d,r))
\end{array}$$

Figure 2-3: Example Logical Expressions

In addition to using the logical language for the demands for expression, we use it to maintain a database of factual information. Besides the "facts of the world", we assume the availability of such knowledge as:

- Hearer, speaker, time and place.
- The theme of the ongoing discussion.
- Objects assumed to be identifiable to the hearer.

Work on maintaining this database is proceeding in parallel with the work reported here.

Finally, we have allowed for a speech act operator to be supplied to the generation system along with the logical form. This can be **ASSERT, COMMAND, QUERY, ANSWER** or **REFER. ANSWER** is used for Yes/No answers. The others are given the usual interpretation.

3. CONNECTING LANGUAGE AND LOGIC

Restating the general problem in terms of our basic components, a logical form submitted as a demand for expression must be interpreted by the Nigel inquiries. Nigel must be able to decompose the expressions and characterize their parts.

To achieve this, we have used NIKL to categorize the concepts (or terms) of the domain in terms of Nigel's implicit categorizations. We have written Nigel inquiries which use the structure of the logical language and the NIKL model to analyze the logical forms. To do this efficiently, we have developed a way to translate the logical form into a KL-TWO database and use its reasoning mechanisms.

3.1. Functional Systemic Categorizations in a NIKL Knowledge Base

Our NIKL knowledge base is structured in layers. At the top are concepts and roles that reflect the structure we impose on our logical forms. Here we find concepts like **ActionOccurrence** and **Action**, as well as roles like **records**. At the bottom are the entities of the domain. Here we find concepts like **Transmit** and **Send**, as well as roles like **requestedobject**. All of these concepts and roles must be shown as specializing concepts at a third, intermediate level, which we have introduced to support

Nigel's generation.

Functional systemic linguists take a Whorfian view: that there is a strong connection between the structures of thought and language. We follow them in categorizing domain concepts in a way that reflects the different linguistic structures that describe them. For example, we have distinguished three types of actions (verbal, mental and material) because the clauses that describe these actions differ in structure. For the same reason, we have at least three types of relationships: ascription, circumstantial and generalized possession. Roughly, ascription relates an object to an intrinsic property, such as its color. Circumstantials involve time, place, instrument, etc. In addition to ownership, generalized possession includes such relationships as part/whole and social association.

Some of these categories are shown graphically in Figure 3-1. The double arrows are the IS-A links. The single arrows are role restrictions.

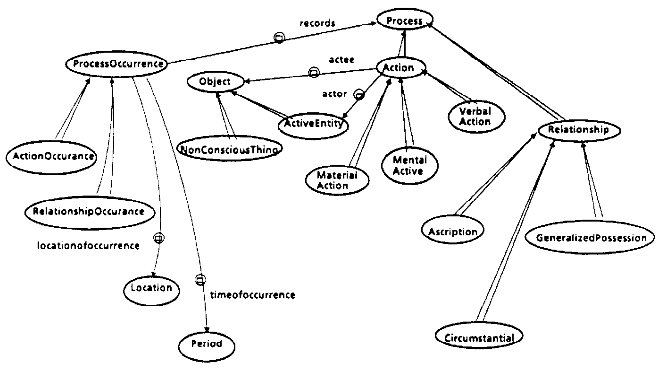

Figure 3-1: Example Upper and Intermediate Layer Categories

Relating these distinctions to our earlier examples, the concepts **Transmit** and **Send** are modeled as subclasses of **MaterialAction**. **Message** is a kind of **NonConsciousThing**.

This modeling extends to the role hierarchy, as well. For example, the role **requestedobject** is modeled as a kind of **actee** role.

The insertion of systemic distinctions does not compromise other factors, since non-linguistic categorizations can **co-exist** in the knowledge base with the systemic categories.

Once the domain model is built we expect the systems using the generator to never have to refer to our middle level concepts. Furthermore, we expect Nigel to never refer to any domain concepts.

Since the domain concepts are organized under our middle level, we can note that **all domain predicates in logical forms are categorized in systemic terms**. To be complete, the domain model must identify each unary predicate with a concept and each binary predicate with a role. The concepts in a logical form must either reflect the highest, most general, concepts in the network or the lowest layer. The domain predicates must therefore relate through domain concepts to systemic categories.

3.2. Logical Forms in KL-TWO

Gary Hendrix [Hendrix 75] developed the notion of **Partitioned Semantic Networks** in order to add the representational power of quantifier scoping, belief spaces, etc., to the semantic network formalism. This does not pay off in terms of faster inferences, but it allows us to separate the two structures inherent in logical formulas, the quantification scopes and the connections of terms. In partitioned networks, these are represented by hierarchically ordered partitions and network arcs, respectively.

This separation of the scope and connection structure is needed. The **connection structure** can be used to evaluate Nigel's inquiries against the model, and the **scope structure** can be used to infer additional information concerning quantification.

We translate a logical form into an equivalent KL-TWO structure. All predications appearing in the logical form are put into the PENNI database as assertions. Figure 3-2 shows the set of assertions entered for the formula in Figure 2-3A. These are shown graphically in Figure 3-3 which includes the partitions. KL-TWO does not support partitions. Instead of creating scope partitions, a tree is created which reflects the variable scoping. Here we diverge from Hendrix because of the demands of our language. Separate scopes are kept for the range restriction of a quantification and its predication. In addition the scope of the term forming operators, ι and η are kept in the scope structure.

During the translation, the variables and constants are given unique names so that these assertions are not confused with true assertional knowledge (this is not shown in our examples.). These new entities may be viewed as a kind of hypothetical object that Nigel will describe, but the original logical meaning may still be derived by inspecting the assertions and the scope structure.

3.3. Implementation of Nigel Inquiries

Our implementation of Nigel's inquiries using the connection and scope structures with the NIKL upper structure is fairly straightforward to describe. Since the logical forms reflecting the world view are in the highest level of the NIKL model, the information decomposition inquiries use these structures to do

(ActionOccurrence x) (Past p) (timeofoccurrence x p)
(Transmit y) (records x y) (actor y SMITH)
(Message z) (actee y z)

Figure 3-2: Sample PENNI Assertions

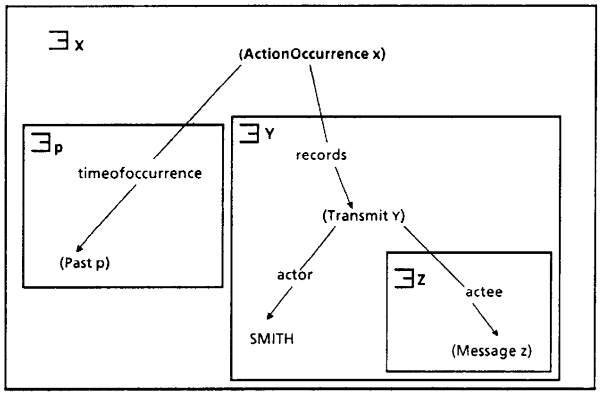

Figure 3-3: Sample Partition Structure

search and retrieval. With all of the predicates in the domain specializing concepts in the functional systemic level of the NIKL model, information characterization inquiries that consider aspects of the connection structure can test for the truth of appropriate PENNI propositions. The inquiries that relate to information presented in the quantification structure of the logical form will search the scope structure. Finally, to supply lexical entries, we associate lexical entries with NIKL concepts as attached data and use the retrieval methods of PENNI and NIKL to retrieve the appropriate terms.

Let's consider some examples. The generation activity begins with a pointer to the major **ProcessOccurrence**. By the time **CauserID** is asked, Nigel has a pointer to what it knows to be a caused **Action**. **CauserID** is realized by a procedure that finds the thing or things that are in **actor** type relationships to the **Action**. **AffectedID** works similarly through the **actee** predicate. When **VerbalProcessQ** is asked, Nigel simply asks PENNI if a proposition with **VerbalAction** and the **Action** is true.

These examples emphasize the use of the connection structure to analyze what functional systemic grammarians call the **ideational** content of an utterance. In addition, utterances are characterized by **interpersonal** content, e.g., the relation between the hearer and the speaker, and **textual** content, e.g., relation to the last utterance. We have been developing methods for storing this information in a PENNI database, so that interpersonal and textual inquiries can also be answered by asking questions of PENNI.

MultiplicityQ is an example of a more involved process. When it is invoked, Nigel has a pointer to an individual to be described. The inquiry identifies all sets as multiple and any non-set individuals as unitary. For non-set variables, it explores their scoping environment. Its most interesting property involves an entity whose quantification suggests an answer of unitary. If the entity is shown in the logical form as a property of or a part of some entity and it is inside the scope of the quantifier that binds that entity and this second entity must be treated as multiple, then both entities are said to be multiple.

TermSpecificationID is unique in that it explores the NIKL network directly. It is given a pointer to a PENNI individual. It accesses the most specific generic concept PENNI has constructed to describe the individual. It looks at this concept and then up through more general categories until it finds a lexical entry associated with a concept.

4. EXAMPLE SENTENCES

Space constraints forbid presentation of a complete example. Let's look at a few points involved in transforming the three example logical forms in Figure 2-3 into English. Assume for Example 2-3A, that, at this moment, the COMPUTER wishes to communicate to RICHARD the information as an assertion, and that SMITH is known by name through the PENNI database. The flow starts with **x** identified as the central **ProcessOccurrence**. From there, **y** is identified as describing the main process.

TermSpecificationID is applied to **y** in one of the first inquiries processed. This is stated to be a **Transmit**. However, we are also told that its **actee** is a **Message**. Assuming the model described in Section 2.2, PENNI concludes that **y** is not just a **Transmit**, but a **Send** as well. This leads **TermSpecificationID** to look first at **Send** for a lexical entry.

Next, Nigel asks for a pointer to the time being referred to and receives back **p**. Later this is evaluated against the speaking time to establish the tense.

Further on, Nigel characterizes the process. The inquiries attempt to prove, in turn, that **y** is a **Relationship**, a **MentalActive**, and a **VerbalAction**. When none of these queries are answered positively, it concludes that **y** is a **MaterialAction**.

After establishing that **y** is a kind of event that is caused, Nigel uses **CauserID** and **AffectedID**. It receives back **SMITH** and **z**, respectively.

The actual decision on how to describe **SMITH** and **z** are arrived at during separate passes through the grammar. During the pass for **SMITH**, **TermSpecificationID** returns his name, "Smith". **MultiplicityQ** is invoked and returns unitary. During the pass for **z**, **TermSpecificationID** returns "message", while **MultiplicityQ** returns unitary.

In the end, the sentence "Smith sent a message." is generated.

Looking at Example 2-3B, one difference on the level of the outermost **ActionOccurrence** is the absence of an **actee** relationship. However, **requestedobject** is shown in the model as a type of **actee** relationship and **AffectedID** returns **q**. In order to describe **q** the grammar forms a relative clause, "which was sent by Smith". There is no overt indication of the type of entity **q** is. However, from the model of **Send**, PENNI infers that **(Message z)** is true. **TermSpecificationID** for **z** returns "message". Treating the sentence as a command and assuming "show" is associated with **Display**, Nigel will produce "Show me the message which was sent by Smith.".

Example 2-3C allows us to consider the use of the scope structure in deciding the multiplicity of **r**. We are required to describe the displaying of the single inspection status (or read status) that is found for each message in a set of messages. As noted, we have modeled **InspectionStatus** as an **Ascription** relationship. The grammar uses "of" to describe this sort of relation in a noun phrase. **MultiplicityQ** evaluates **m** as multiple. Because **r** is in **m's** scope, it too is evaluated as multiple and the noun phrase is rendered as "the read statuses of the messages". If the scopings were reversed, the logical form would indicate that there was only one read status for all the messages. **MultiplicityQ** would evaluate **r** as unitary and the noun phrase would be "the read status of the messages". If both the quantifiers were existential, then each scoping would result in **MultiplicityQ** evaluating both as unitary. The noun phrase would be rendered as "the read status of the message". If **m** were bound by an $\exists!$, an ι or replaced by a unitary constant, and **r** bound by a universal quantifier, the rendering would be "the read statuses of the message".

In Figure 4-1, we display a set of sentences to give the reader some idea of the generator's range as of January 1986. Nigel played the part of both participants in the dialogue, which used hand constructed logical forms and dialogue contexts in the absence of an associated software system.

Figure 4-2 gives the logical form which forced the generation of the next to the last sentence in the example set.

*User: What is the read status of the message which was sent by Jones?
*System: Message 40 is unread.
*User: Are any messages which were sent by Jones in the tape file?
*System: Yes. There are messages which were sent by Jones. However, the tape is in use.
*User: Did Smith send any messages?
*System: Message 45 was sent by Smith.
*User: Did Smith schedule his New York meeting with me?
*System: Yes.
*User: What does the window show?
*System: The window shows a request by Jones.
*User: Did I create the window?
*System: The window was created when you asked me to display Message 50. You deleted the message, then you asked me to display it.

Figure 4-1: A Sample Set of Generated Sentences

```
(∃e ∈ ActionOccurrence)
 (∃p ∈ Past)(timeofoccurrence(e,p)∧
   (∃f ∈ ActionOccurrence)
    (∃q ∈ Past)(timeofoccurrence(f,q)∧
      (∃r ∈ RelationOccurrence)
       (∃t ∈ Simultaneity)
         (records(r,t)∧domain(t,p)∧range(t,q))∧
      (∃n ∈ NaturalLanguageRequestAction)
       (records(f,n)∧actor(n,RICHARD)∧
         beneficiary(n,COMPUTER)∧
         (∃x ∈ NaturalLanguageRecord)(actee(n,x)∧
          (∃d ∈ Command)(forceofnlrecord(x,d)∧
            (∃ee ∈ ActionOccurrence)(focus(x,ee)∧
             (∃dd ∈ Display)
               (records(ee,dd)∧actor(dd,COMPUTER)∧
                 actee(dd,M50))))))))∧
   (∃c ∈ Create)(records(e,c)∧actee(c,W34))
```

Figure 4-2: A More Complex Logical Form

5. CONCLUSION

5.1. Summary

To summarize, we have developed a first-order predicate-calculus language which can be used to make demands for expressions to the Nigel grammar. This works by translating the logical forms into two separate structures that are stored in a PENNI database. Nigel inquiries are evaluated against these structures through the aid of a NIKL knowledge base. Discourse context is also stored in the data base and lexical entries are obtained from the knowledge base.

Adding this facility to Nigel seems to have added only 10 to 20 percent to Nigel's run time. The system is currently implemented in Interlisp and runs on the Xerox family of Lisp machines. A reimplementation in Common Lisp is underway.

5.2. Limitations and Future Plans

For the sake of presentation, we have simplified our description of the working system. Other facilities include an extensive tense, aspect and temporal reference system. There is also a facility for dynamically constructing logical forms for referring expressions. This is used when constants are found in other logical forms that cannot be referred to by name or through pronoun.

There are also certain limitations in our approach. One of which may have occurred to the reader is that the language our system produces is ambiguous in ways formal logic is not. For example, "the read statuses of the messages" has one reading which is different from the logical form we used in our example. While scope ambiguities are deeply ingrained in language, they are not a problem in most communication situations.

Related to this problem is a potentially important mismatch between logic and functional systemic grammars. These grammars do not control directly for quantification scope. They treat it as only one aspect of the decision making process about determiner choice and constituent ordering. Certainly, there is a great deal of evidence that logical scoping is not often a factor in the interpretation of utterances[***]

Another set of problems concern the limits we place on logical connectives in logical forms. One limit is the position of negations: we can only negate **ProcessOccurrences**, e.g., "John didn't send a message.". Negation on other forms, e.g., "John sent no messages.", affects the basic connection with the NIKL model. Furthermore, certain conjunctions have to be shown with a conjunctive **Relationship** as opposed to logical conjunction. This includes conjunctions between **ProcessOccurrences** that lead to compound sentences, as well as all disjunctions.

Furthermore, we impose a condition that a demand for expression must concern a single connected set of structures. In operation the system actually ignores parts of the logical form that are independent of the main **ProcessOccurrences**. Because the underlying grammar can only express one event or state of affair(not counting dependent processes) and its associated circumstances at a time, in order to fit in one sentence all the entities to be mentioned must be somehow connected to one event or state of affair.

We expect that the limitations in the last two paragraphs will be overcome as we develop our text generation system, **Penman** [Mann 83b]. A theory of text structure is being developed at USC/ISI that will take less restrained forms and map them into multi-sentence text structures [Mann 84]. The use of this intermediate facility will mediate for logical connectives and connectivity by presenting the sentence generator with normalized and connected structures.

The word choice decisions the system makes also need to be enhanced. It currently takes as specific a term as possible. Unfortunately, this term could convey only part of the necessary information. Or it could convey more information than that conveyed by the process alone, e.g., in our transmit/send example, "send", unlike "transmit", conveys the existence of a message. We are currently developing a method of dealing with word choice through descriptions in terms of primitive concepts that will support better matching between demands and lexical resources.

[***] For example, Keenan and Faltz state "We feel that the reason for the poor correspondence is that NP scope differences in natural language are not in fact coded or in general reflected in the derivational history of an expression. If so, we have a situation where we need something in LF which really doesn't correspond to anything in SF" [Keenan 85, p. 21].

A related limit is the requirement in the current NIKL that all aspects of a concept be present in the logical form in order for the NIKL classifier to have effect. For example, the logical forms must show all aspects of a **Send** to identify a **Transmit** as one. A complete model of **Send** will certainly have more role restrictions than the **actee**. However, just having an **actee** which is a **Message** should be sufficient to indicate that a particular **Transmit** is a **Send**. We are working with the developers of NIKL to allow for this type of reasoning.

Two other areas of concern relate directly to our most important current activity which is described in the next paragraph. First, it is not clear that first-order logic will be sufficiently expressive for all possible situations. Second, it is not clear the use of hand-built logical forms is sufficient to test our design to its fullest extent.

5.3. JANUS

The success of our work to date has led to plans for the inclusion of this design in the **Janus** natural language interface. Janus is a joint effort between USC/ISI and BBN, Inc., to build the next generation natural language interface within the natural language technology component of the Strategic Computing Initiative [Walker 85]. One feature of the system will be the use of higher-order logics. Plans are underway to test the system in actual use. The future direction of the work presented here will be largely determined by the demands of the Janus effort.

ACKNOWLEDGMENTS

We gratefully acknowledge the assistance of our colleagues: Bill Mann with whom many of the ideas were developed; Richard Whitney who implemented many of the inquiry operator functions; Tom Galloway who suggested many improvements in the paper, as did Robert Albano, Susanna Cumming, Paul Jacobs, Christian Matthiessen, and Lynn Poulton; and Marc Vilain who gave us KL-TWO. We also gratefully acknowledge the Technische Universitaet Berlin and the members of ESPRIT Project 311 who made it possible for the second author to participate in the project described here.

References

[Appelt 83] Douglas E. Appelt, "Telegram: a grammar formalism for language planning," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 595-599, IJCAI, Aug 1983.

[Brachman 85] R. J. Brachman, V. P. Gilbert, H. J. Levesque, "An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 532-539, Los Angeles, CA, August 1985.

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

[Davidson 67] D. Davidson, "The Logical Form of Action Sentences," in N. Rescher (ed.), *The Logic of Decision and Action*, pp. 81-95, The University of Pittsburgh Press, Pittsburgh, 1967.

[Goldman 75] Goldman, N. M., "Conceptual generation," in R. C. Schank (ed.), *Conceptual Information Processing*, North-Holland, Amsterdam, 1975.

[Habel 82] Christopher Habel, "Referential nets with attributes," in Horecky (ed.), *Proc. COLING-82*, North-Holland, Amsterdam, 1982.

[Halliday 76] Halliday, M. A. K., *System and Function in Language*, Oxford University Press, London, 1976.

[Hendrix 75] G. Hendrix, "Expanding the Utility of Semantic Networks through Partitioning," in *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence*, pp. 115-121, Tbilisi, September 1975.

[Hoeppner et al. 83] Wolfgang Hoeppner, Thomas Christaller, Heinz Marburger, Katharina Morik, Bernhard Nebel, Mike O'Leary, Wolfgang Wahlster, "Beyond domain-independence: experience with the development of a German natural language access system to highly diverse background systems," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 588-594, IJCAI, Aug 1983.

[Hovy 85] E. H. Hovy, "Integrating Text Planning and Production in Generation," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 115-121, Los Angeles, CA, August 1985.

[Jacobs 85] Paul Jacobs, *A Knowledge-Based Approach to Language Production*, Ph.D. thesis, University of California, Berkeley, CA, August 1985.

[Kaczmarek 86] T. Kaczmarek, R. Bates, G. Robins, "Recent Developments in NIKL," in *AAAI-86, Proceedings of the National Conference on Artificial Intelligence*, AAAI, Philadelphia, PA, August 1986.

[Kaczmarek, Mark, and Sondheimer 83] T. Kaczmarek, W. Mark, and N. Sondheimer, "The Consul/CUE Interface: An Integrated Interactive Environment," in *Proceedings of CHI '83 Human Factors in Computing Systems*, pp. 98-102, ACM, December 1983.

[Keenan 85] Edward L. Keenan, Leonard M. Faltz, *Boolean Semantics for Natural Language*, Reidel, Boston, 1985.

[Kukich 85] Karen Kukich, "Explanation Structures in XSEL," in *Proceedings of the 23rd Annual Meeting*, ACL, Jul 1985.

[Mann 83a] Mann, W. C., "Inquiry semantics: A functional semantics of natural language grammar," in *Proceedings of the First Annual Conference*, Association for Computational Linguistics, European Chapter, September 1983.

[Mann 83b] Mann, W. C., "An overview of the Penman text generation system," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 261-265, AAAI, August 1983. Also appears as USC/Information Sciences Institute, RR-83-114.

[Mann 84] Mann, W., *Discourse Structures for Text Generation*, USC/Information Sciences Institute, Marina del Rey, CA, Technical Report RR-84-127, February 1984.

[Mann & Matthiessen 83] William C. Mann & Christian M.I.M. Matthiessen, *Nigel: A Systemic Grammar for Text Generation*, USC/Information Sciences Institute, Technical Report ISI/RR-83-105, Feb 1983.

[McAllester 82] D.A. McAllester, *Reasoning Utility Package User's Manual*, Massachusetts Institute Technology , Technical Report, April 1982.

[McDonald 83] David D. McDonald, "Natural language generation as a computational problem: an introduction," in Brady & Berwick (eds.), *Computational Problems in Discourse*, pp. 209-264, MIT Press, Cambridge, 1983.

[McKeown 85] Kathleen R. McKeown, *Text generation: using discourse strategies and focus constraints to generate natural language text*, Cambridge University Press, Cambridge, 1985.

[Montague 74] R. Montague, *Formal Philosophy*, Yale University Press, New Haven, CN, 1974.

[Schmolze and Lipkis 83] James Schmolze and Thomas Lipkis, "Classification in the KL-ONE Knowledge Representation System," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, IJCAI, 1983.

[Shapiro 79] Shapiro, S. C., "Generalized augmented transition network grammars for generation from semantic networks," in *Proceedings of the Seventeenth Meeting of the Association for Computational Linguistics*, pp. 25-29, August 1979.

[Vilain 85] M. Vilain, "The Restricted Language Architecture of a Hybrid Representation System," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 547-551, Los Angeles, CA, August 1985.

[Walker 85] E. Walker, R. Weischedel, N. Sondheimer, "Natural Language Interface Technology," in *Strategic Systems Symposium*, DARPA, Monterey, CA, October 1985.