A Terminological Simplification Transformation for Natural Language Question-Answering Systems

David G. Stallard
BBN Laboratories Inc.
10 Moulton St.
Cambridge, MA.
02238

Abstract

A new method is presented for simplifying the logical expressions used to represent utterance meaning in a natural language system.[1] This simplification method utilizes the encoded knowledge and the limited inference-making capability of a taxonomic knowledge representation system to reduce the constituent structure of logical expressions. The specific application is to the problem of mapping expressions of the meaning representation language to a database language capable of retrieving actual responses. Particular account is taken of the model-theoretic aspects of this problem.

1. Introduction

A common and useful strategy for constructing natural language interface systems is to divide the processing of an utterance into two major stages: the first mapping the utterance to a logical expression representing its "meaning" and the second producing from this logical expression the appropriate response. The second stage is not neccesarily trivial: the difficulty of its design is signifigantly affected by the complexity and generalness of the logical expressions it has to deal with. If this issue is not faced squarely, it may affect choices made elsewhere in the system. Indeed, a need to restrict the form of the meaning representation can be at odds with particular approaches towards producing it – as for example the "compositional" approach, which does not seek to control expression complexity by giving interpretations for whole phrasal patterns, but simply combines together the meaning of individual words in a manner appropriate to the syntax of the utterance. Such a conflict is certainly not desirable: we want to have freedom of linguistic action as well as to be able to obtain correct responses to utterances.

This paper treats in detail the particular manifestation of these issues for natural-language systems which serve as interfaces to a database: the problems that arise in a module which maps the meaning representation to a second logical language for expressing actual database queries. A module performing such a mapping is a component of such question-answering systems as TEAM [4], PHLIQA1 [7] and IRUS [1]. As an example of difficulties which may be encountered, consider the question "Was the patient's mother a diabetic?" whose logical representation must be mapped onto a particular boolean field which encodes for each patient whether or not this complex property is true. Any variation on this question which a compositional semantics might also handle, such as "Was diabetes a disease the patient's mother suffered from?", would result in a semantically equivalent but very different-looking logical expression; this different expression would also have to be mapped to this field. How to deal with these and many other possible variants, without making the mapping process excessively complex, is clearly a problem.

The solution which this paper presents is a new level of processing, intermediate between the other two: a novel simplification transformation which is performed on the result of semantic interpretation before the attempt is made to map it to the database. This simplification method relies on knowledge which is stored in a taxonomic knowledge representation system such as NIKL [5]. The principle behind the method is that an expression may be simplified by translating its subexpressions, where possible, into the language of NIKL, and classifying the result into the taxonomy to obtain a simpler equivalent for them. The result is to produce an equivalent but syntactically simpler expression in which fewer, but more specific, properties and relations appear. The benefit is that deductions from the expression may be more easily "read off"; in particular, the mapping becomes easier because the properties and relations appearing are more likely to be either those of the database or composable from them.

The body of the paper is divided into four sections. In the first, I will summarize some past treatments of the mapping between the meaning representation and the query language, and show the problems they fail to solve. The second section prepares the way by showing how to connect the taxonomic knowledge representation system to a logical language used for meaning representation. The third section presents the "recursive terminological simplification" algorithm itself. The last section describes the implementation status and suggests directions for interesting future work.

2. A Formal Treatment of the Mapping Problem

This section discusses some previous work on the problem of mapping between the logical language used for meaning representation and the logical language in which actual database queries are expressed. The

[1]The work presented here was supported under DARPA contract #N00014-85-C-0016. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the United States Government.

difficulties which remain for these approaches will be pointed out.

A common organization for a database is in terms of tables with rows and columns. The standard formulation of these ideas is found in the relational model of Codd [3], in which the tables are characterized as relations over sets of atomic data values. The elements (rows) of a relation are called "tuples", while its individual argument places (columns) are termed its "attributes". Logical languages for the construction of queries, such as Codd's relational algebra, must make reference to the relations and attributes of the database.

The first issue to be faced in consideration of the mapping problem is what elements of the database to identify with the objects of discourse in the utterance – that is, with the non-logical constants[2] in the meaning representation. In previous work [9] I have argued that these should not be the rows of the tables, as one might first think, but rather certain sets of the atomic attribute-values themselves. I presented an algorithm which converted expressions of a predicate calculus-based meaning representation language to the query language ERL, a relational algebra [3] extended with second-order operations. The translations of non-logical constants in the meaning representation were provided by fixed and local translation rules that were simply ERL expressions for computing the total extension of the constant in the database. The expressions so derived were then combined together in an appropriate way to yield an expression for computing the response for the entire meaning representation expression. If the algorithm encountered a non-logical constant for which no translation rule existed, the translation failed and the user was informed as to why the system could not answer his question.

By way of illustration, consider the following relational database, consisting of clinical history information about patients at a given hospital and of information about doctors working there:

```
PATIENTS(PATID,SEX,AGE,DISEASE,PHYS,DIAMOTHER)
DOCTORS(DOCID,NAME,SEX,SPECIALTY)
```

where "PHYS" is the ID of the treating physician, and "DIAMOTHER" is a boolean field indicating whether or not the patient's mother is diabetic. Here are the rules for the one-place predicate PATIENTS, the one-place predicate SPECIALTIES, and the two-place predicate TREATING-PHYSICIAN:

```
PATIENTS   => (PROJECT PATIENTS OVER PATID)

SPECIALTIES   => (PROJECT DOCTORS OVER SPECIALTY)

TREATING-PHYSICIAN   => (PROJECT (JOIN PATIENTS
                                       TO DOCTORS
                                       OVER PHYS DOCID)
                                  OVER PATID DOCID)
```

Note that while no table exists for physician SPECIALTIES, we can nonetheless give a rule for this predicate in way that is uniform with the rule given for the predicate PATIENTS.

[2]This term, while a standard one in formal logic, may be confused with other uses of the word "constant". It simply refers to the function, predicate and ordinary constant symbols, such as "MOTHER" or "JOHN", whose denotations depend on the interpretation of the language, as opposed to fixed symbols like "FORALL","AND", "TRUE".

One advantage of such local translation rules is their simplicity. Another advantage is that they enable us to treat database question-answering model-theoretically. The set-theoretic structure of the model is that which would be obtained by generating from the relations of the database the much larger set of "virtual" relations that are expressible as formulas of ERL. The interpretation function of the model is just the translation function itself. Note that it is a *partial* function because of the fact that some non-logical constants may not have translations. We speak therefore of the database constituting a "partially specified model" for the meaning representation language. Computation of a response to a user's request, instead of being characterizable only as a procedural operation, becomes interpretation in such a model.

A similar model-theoretic approach is advocated in the work on PHLIQA1 [8], in which a number of difficulties in writing local rules are identified and overcome. One class of techniques presented there allows for quite complex and general expressions to result from local rule application, to which a post-translation simplification process is applied. Other special-purpose techniques are also presented, such as the creation of "proxies" to stand in for elements of a set for which only the cardinality is known.

A more difficult problem, for which these techniques do not provide a general treatment, arises when we want to get at information corresponding to a complex property whose component properties and relations are not themselves stored. For example, suppose the query "List patients whose mother was a diabetic", is represented by the meaning representation:

```
(display ↑(setof X:PATIENT
              (forall Y:PERSON (->(MOTHER X Y)
                                  (DIABETIC Y)))))
```

The information to compute the answer may be found in the field DIAMOTHER above. It is very hard to see how we will use local rules to get to it, however, since nothing constructable from the database corresponds to the non-logical constants MOTHER and DIABETIC. The problem is that the database chooses to highlight the complex property DIAMOTHER while avoiding the cost of storing its constituent predicates MOTHER and DIABETIC – the conceptual units corresponding to the words of the utterance.

One way to get around these difficulties is of course to allow for a more general kind of transformation: a "global rule" which would match against a whole syntactic pattern like the univerally quantified sub-expression above. The disadvantage of this, as is pointed out in [8], is that the richness of both natural language and logic allows the same meaning to be expressed in many different ways, which a complete "global rule" would have to match. Strictly syntactic variation is possible: pieces of the pattern may be spread out over the expression, from which the pattern match would have to grab them. Equivalent formulations of the query may also use completely different terms. For example, the user might have employed the equivalent phrase "female parent" in place of the word "mother", presumably causing the semantic interpretation to yield a logical form with the different predicates "PARENT" and "FEMALE". This would not match the pattern. It becomes clear that the "pattern-matching" to be performed here is not the literal kind, and that it involves unspecified and arbitrary amounts of inference.

The alternative approach presented by this paper takes explicit account of the fact that certain properties and relations, like "DIAMOTHER", can be regarded as built up from others. In the next section we will show how the properties and relations whose extensions the database stores can be axiomatized in terms of the ones that are more basic in the application domain. This prepares the way for the simplification transformation itself, which will rely on a limited and sound form of inference to reverse the axiomatization and transform the meaning representation, where possible, to an expression that uses only these database properties and relations. In this way, the local rule paradigm can be substantially restored.

3. Knowledge Representation and Question-Answering

The purpose of this section of the paper is to present a way of connecting the meaning representation language to a taxonomic knowledge representation system in such a way that the inference-making capability of the latter is available and useful for the problems this paper addresses. Our approach may be constrasted with that of others, e.g. TEAM in which such a taxonomy is used mainly for simple inheritance and attachment duties.

The knowledge representation system used in this work is NIKL [5]. Since NIKL has been described rather fully in the references, I will give only a brief summary here.

NIKL is a taxonomic frame-like system with two basic data structures: concepts and roles. Concepts are just classes of entities, for which roles function somewhat as attributes. At any given concept we can restrict a role to be filled by some other concept, or place a restriction on the number of individual "fillers" of the role there. A role has one concept as its "domain" and another as its "range": the role is a relation between the sets these two concepts denote. Concepts are arranged in a hierarchy of sub-concepts and superconcepts; roles are similarly arranged. Both concepts and roles may associated with names. In logical terms, a concept may be identified as the one-place predicate with its name, and a role as the two-place predicates with its name.

I will now give the meaning postulates for a term-forming algebra, similar to the one described in [2] in which one can write down the sort of NIKL expressions I will need. Expressions in this language are combinable to yield a complex concept or role as their value.

```
(CONJ C1 — CN) ≡ (lambda (X) (and (C1 X) — (Cn X)))

(VALUERESTRICT R C) ≡ (lambda (X) (forall Y (-> (R X Y)
                                                 (C Y)))

(NUMBERRESTRICT R 1 NIL) ≡ (lambda (X) (exists Y (R X Y)))

(VRDIFF R C) ≡ (lambda (X Y) (and (R X Y) (C Y)))

(DOMAINDIFF R C) ≡ (lambda (X Y) (and (R X Y) (C X)))
```

The key feature of NIKL which we will make use of is its classifier, which computes subsumption and equivalence relations between concepts, and a limited form of this among roles. Subsumption is sound, and thus indicates entailment between terms:

```
(SUBSUMES C1 C2)  ->  (forall X (-> (C2 X) (C1 X)))
```

If the classifier algorithm is complete, the reverse is also true, and entailment indicates subsumption. Intuitively, this means that classified concepts are pushed down as far in the hierarchy as they can go.

Also associated with the NIKL system, though not a part of the core language definition, is a symbol table which associates atomic names with the roles or concepts they denote, and concepts and roles with the names denoting them. If a concept or role does not have a name, the symbol table is able to create and install one for it when demanded.

The domain model

In order to be able to use NIKL in the analysis of expressions in the meaning representation language, we make the following stipulations for any use of the language in a given domain. First, any one-place predicate must name a concept, and any two-place predicate name a role. Second, any constant, unless a number or a string, must name an "individual" concept – a particular kind of NIKL concept that is defined to have at most one member. N-ary functions are treated as a N+1 – ary predicates. A predicate of N arguments, where N is greater than 2, is reified as a concept with N roles. This set of concepts and roles, together with the logical relationships between them, we call the "domain model".

Note that all we have done is to stipulate an one-to-one correspondence between two sets of things – the concepts and roles in the domain model and the non-logical constants of the meaning representation language. If we wish to include a new non-logical constant in the language we must enter the corresponding concept or role in the domain model. Similarly, the NIKL system's creating a new concept or role, and creation of a name in the symbol table to stand for it, furnishes us with a new non-logical constant.

Axiomatization of the database in terms of the domain model

The translation rules presented earlier effectively seek to axiomatize the properties and relations of the domain model in terms of those of the database. This is not the only way to bridge the gap. One might also try the reverse: to axiomatize the properties and relations of the database in terms of those of the domain model. Consider the DIAMOTHER field of our sample database. We can write this in NIKL as the concept PATIENT-WITH-DIABETIC-MOTHER using terms already present in the domain model:

```
(CONJ PATIENT
      (VALUERESTRICT MOTHER
                     DIABETIC))
```

If we wanted to axiomatize the relation implied by the SEX attribute of the PATIENTS table in our database, we could readily do so by defining the role PATIENT-SEX in terms of the domain model relation SEX:

```
(DOMAINDIFF SEX
        PATIENT)
```

These two defined terms can actually be entered into the model, and be treated just like any others there. For example, they can now appear as predicate letters in meaning representations. Moreover, to the associated data structure we can attach a translation rule, just as we have been doing with the original domain model elements. Thus, will attach to the concept PATIENT-WITH-DIABETIC-MOTHER the rule:

```
(PROJECT (SELECT FROM PATIENTS WHERE (EQ DIAMOTHER "YES"))
      OVER PATID)
```

The next section will illustrate how we map from expressions using "original" domain model elements to the ones we create for axiomatizing the database, using the NIKL system and its classifier.

4. Recursive Terminological Simplification

We now present the actual simplification method. It is composed of two separate transformations which are applied one after the other. The first, the "contraction phase", seeks to contract complicated subexpressions (particularly nested quantifications) to simpler one-place predications, and to further restrict the "sorts" of remaining bound variables on the basis of the one-place predicates so found. The second part of the transformation, the "role-tightening" phase, replaces general relations in the expression with more specific relations which are lower in the NIKL hierarchy. These more specific relations are obtained from the more general by considering the sorts of the variables upon which a given relational predication is made.

The contraction phase

The contraction phase is an algorithm with three steps, which occur sequentially upon application to any expression of the meaning representation. First, the contraction phase applies itself recursively to each non-constant subexpression of the expression. Second, depending upon the syntactic category of the expression, one of the "pre-simplification" transformations is applied to place it in a normalized form. Third and finally, one of the actual simplification transformations is used to convert the expression to one of a simpler syntactic category.

Before working through the example, I will lay out the transformations in detail. In what follows, X and X1,X2 -- Xn are variables in the meaning representation language. The symbol "<rest>" denotes a (possibly empty) sequence of formulae. The expression "(FORMULA X)" denotes a formula of the meaning representation language in which the variable X (and perhaps others) appears freely. The symbol "<quant>" is to be understood as being replaced by either the operator SETOF or the quantifier EXISTS.

First, the normalization transformations, which simply re-arrange the constituents of the expressions to a more convienent form without changing its syntactic category:

```
(1)  (and (P1 X1) (P2 X1) — (PN X1)
          (Q1 X2) (Q2 X2) — (QN X2)
          <rest>)

        ==>  (and (P' X1) (Q' X2) <rest>)

        where P' := (CONJ P1 P2 — PN)
        and   Q' := (CONJ Q1 Q2 — QN)

(2)  (<quant> X:S (and (P X) <rest>)  ==>

        (<quant> X:S' (and <rest>))
        where S' := (CONJ S P)

(3) (<quant> X:S (P X))  ==>

        (<quant> X:S')
        where S' := (CONJ S P)

(4) (forall X:S (-> (and (P X) <rest>)
                    (FORMULA X))  ==>

        (forall X:S' (-> (and <rest>)
                         (FORMULA X)))
```

In (2) and (4) above, the conjunction or implication, respectively, are collapsed out if the sequence <rest> is empty.

Now the actual simplification transformations, which seek to reduce a complex sub-expression to a one-place predication.

```
(5) (forall X2:S (-> (R X1 X2) (P X2)))
        ==>   (P' X1)
      where P' := (VALUERESTRICT (VRDIFF R S) P)

(6) (exists X2:S (R X1 X2))  ==>  (P' X1)

      where P' := (VALUERESTRICT R S)
      and R must be a functional role

(7) (exists X2:S (R X1 X2))   ==> (P' X1)

     where P' := (NUMBERRESTRICT (VRDIFF R S) 1 NIL)

(8) (and (P X))  ==> (P X)

(9) (R X C) ==> (P X)

      where P := (VALUERESTRICT R C)
      and R is functional, C an individual concept
```

Now, let us suppose that the exercise at the end of the last section has been carried out, and that the concept PATIENT-WITH-DIABETIC-MOTHER has been created and given the appropriate translation rule. To return to the query "List patients whose mother was a diabetic", we recall that it has the meaning representation:

```
(DISPLAY ↑(SETOF X:PATIENTS
                 (FORALL Y:PERSON
                         (-> (MOTHER X Y)
                             (DIABETIC Y)))))
```

Upon application to the SETOF expression, the algorithm first applies itself to the inner FORALL. The syntactic patterns of none of the pre-simplification transformations (2) - (4) are satisfied, so transformation (5) is applied right way to produce the NIKL concept:

```
(VALUERESTRICT (VRDIFF MOTHER PERSON)
               DIABETIC)
```

This is given to the NIKL classifier, which compares it to other concepts already in the hierarchy. Since MOTHER has PERSON as its range already, (VRDIFF MOTHER PERSON) is just MOTHER again. The classifier thus computes that the concept specified above is a subconcept of PERSON - a PERSON such that his MOTHER was a DIABETIC. If this is not found to be equivalent to any pre-existing concept, the system assigns the concept a new name which no other concept has, say PERSON-1. The outcome of the simplification of the whole FORALL is then just the much simpler expression:

```
(PERSON-1 X)
```

The recursive simplification of the arguments to the SETOF is now completed, and the resulting expression is:

```
(DISPLAY ↑(SETOF X:PATIENT
                 (PERSON-1 X)))
```

Transformations can now be applied to the SETOF expression itself. The pre-simplification transformation (3) is found to apply, and a concept expressed by:

```
(CONJ PATIENT PERSON-1)
```

is given to the classifier, which recognizes it as equivalent to the already existing concept PATIENT-WITH-DIABETIC-MOTHER. Since any concept can serve as a sort, the final simplification is:

```
(DISPLAY ↑(SETOF X:PATIENT-WITH-DIABETIC-MOTHER))
```

This is the very concept for which we have a rule, so the ERL translation is:

```
(PRINT FROM (SELECT FROM PATIENT
                    WHERE (EQ DIAMOTHER "YES"))
       PATID)
```

Suppose now that the semantic interpretation system assigned a different logical expression to represent the query "List patients whose mother was a diabetic", in which the embedded quantification is existential instead of universal. This might actually be more in line with the number of the embedded noun. The meaning representation would now be:

```
(display ↑(setof X:PATIENT
                 (exists Y:PERSON (and (MOTHER X Y)
                                       (DIABETIC Y)))
```

The recursive application of the algorithm proceeds as before. Now, however, the pre-simplification transformation (2) may be applied to yield:

```
(exists Y:DIABETIC (MOTHER X Y))
```

since a DIABETIC is already a PERSON. Transformation (6) can be applied if MOTHER is a "functional" role - mapping each and every person to exactly one mother. This can be checked by asking the NIKL system if a number restriction has been attached at the domain of the role, PERSON, specifying that it have both a minimum and a maximum of one. If the author of the domain model has provided this reasonable and perfectly true fact about motherhood, (6) can proceed to yield:

```
(PATIENT-WITH-DIABETIC-MOTHER X)
```

as in the preceding example.

The role tightening phase

This phase is quite simple. After the contraction phase has been run on the whole expression, a number of variables have had their sorts changed to tighter ones. This transformation sweeps through an expression and changes the roles in the expression on that basis. Thus:

```
(10) (R X Y)  ==>  (R' X Y)

     where S1 is the sort of X
     and S2 is the sort of Y
     and R' := (DOMAINDIFF (VRDIFF R S2)
                           S1)
```

One can see that a use of the relation SEX, where the sort of the first argument is known to be DOCTOR, can readily be converted to a use the relation DOCTOR-SEX.

Back conversion: going in the reverse direction

There will be times when the simplification transformation will "overshoot", creating and using new predicate letters which have not been seen before by classifying new data structures into the model to correspond to them. The use of such a new predicate letter can then be treated exactly as would its equivalent lambda-definition, which we can readily obtain by consulting the NIKL model. For example, a query about the sexes of leukemia victims may after simplification result in a rather strange role being created and entered into the hierarchy:

```
PATIENT-SEX-1 := (DOMAINDIFF PATIENT-SEX LEUKEMIA-PATIENT)
```

This role is a direct descendant of PATIENT-SEX; its name is system generated. By the meaning-postulate of DOMAINDIFF given in section 3 above, it can be rewritten as the following lambda-abstract:

```
(lambda (X Y) (and (PATIENT-SEX X Y)
                   (LEUKEMIA-PATIENT X)))
```

For PATIENT-SEX we of course have a translation rule as discussed in section 2. A rule for LEUKEMIA-PATIENT can be imagined as involving the DISEASE field of the PATIENTS table. At this point we can simply call the translation algorithm recursively, and it will come up with a translation:

```
(PROJECT (SELECT FROM PATIENTS
                 WHERE (EQ DISEASE "LEUK"))
         OVER PATID SEX)
```

This supplies us with the needed rule. As a bonus, we can avoid having to recompute it later by simply attaching it to the role in the normal way. The similar computation of rules for complex concepts and roles which are already in the domain comes for free.

5. Conclusions, Implementation Status and Further Work

As of this writing, we have incorporated NIKL into the implementation of our natural language question-answering system, IRUS. NIKL is used to represent the knowledge in a Navy battle-management domain. The simplification transformation described in this paper has been implemented in this combined system, and the axiomatization of the database as described above is being added to the domain model. At that point, the methodology will be tested as a solution to the difficulties now being experienced by those trying to write the translation rules for the complex database and domain of the Fleet Command Center Battle Management Program of DARPA's Strategic Computing Program.

I have presented a limited inference method on predicate calculus expressions, whose intent is to place them in a canonical form that makes other inferences easier to make. Metaphorically, it can be regarded as "sinking" the expression lower in a certain logical space. The goal is to push it down to the "level" of the database predicates, or below. We cannot guarantee that we will always place the expression as low as it could possibly go - that problem is undecidable. But we can go a good distance, and this by itself is very useful for restoring the tractability of the mapping transformation and other sorts of deductive operations [10].

Somewhat similar simplifications are performed in the work on ARGON [6], but for a different purpose. There the database is assumed to be a full, rather than a partially specified, model and simplifications are performed only to gain an increase in efficiency. The distinguishing feature of the present work is its operation on an expression in a logical language for English meaning representation, rather than for restricted queries. A database, given the purposes for which it is designed, cannot constitute a full model for such a language. Thus, the terminological simplification is needed to reduce the logical expression, when possible, to an expression in a "sub-language" of the first for which the database *is* a full model.

An important outcome of this work is the perspective it gives on knowledge representation systems like NIKL. It shows how workers in other fields, while maintaining other logical systems as their primary mode of representation, can use these systems in practical ways. Certainly NIKL and NIKL-like systems could never be used as full meaning representations – they don't have enough expressive power, and were never meant to. This does not mean we have to disregard them, however. The right perspective is to view them as attached inference engines to perform limited tasks having to do with their specialty – the relationships between the various properties and relations that make up a subject domain in the real world.

Acknowledgements

First and foremost, I must thank Remko Scha, both for valuable and stimulating technical discussions as well as for patient editorial criticism. This paper has also benefited from the comments of Ralph Weischedel and Jos De Bruin. Beth Groundwater of SAIC was patient enough to use the software this work produced. I would like to thank them, and thank as well the other members of the IRUS project – Damaris Ayuso, Lance Ramshaw and Varda Shaked – for the many pleasant and productive interactions I have had with them.

References

[1] Bates, Madeleine and Bobrow, Robert J.
A Transportable Natural Language Interface for Information Retrieval.
In *Proceedings of the 6th Annual International ACM SIGIR Conference.* ACM Special Interest Group on Information Retrieval and American Society for Information Science, Washington, D.C., June, 1983.

[2] Brachman, R.J., Fikes, R.E., and Levesque, H.J.
Krypton: A Functional Approach to Knowledge Representation.
IEEE Computer, Special Issue on Knowledge Representation , October, 1983.

[3] Codd,E.F.
A Relational Model of Data for Large Shared Data Banks.
CACM 13(6), June, 1970.

[4] Barbara Grosz, Douglas E. Appelt, Paul Martin, and Fernando Pereira.
TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces.
Technical Report 356, SRI International, Menlo Park, CA, August, 1985.

[5] Moser, Margaret.
An Overview of NIKL.
Technical Report Section of BBN Report No. 5421, Bolt Beranek and Newman Inc., 1983.

[6] Patel-Schneider, P.F., H.J. Levesque, and R.J. Brachman.
ARGON: Knowledge Representation meets Information Retrieveal.
In *Proceedings of The First Conference on Artificial Intelligence Applications.* IEEE Computer Society, Denver, Colorado, December, 1984.

[7] W.J.H.J. Bronnenberg, H.C. Bunt, S.P.J. Landsbergen, R.J.H. Scha, W.J. Schoenmakers and E.P.C. van Utteren.
The Question Answering System PHLIQA1.
In L. Bolc (editor), *Natural Language Question Answering Systems.* Macmillan, 1980.

[8] Scha, Remko J.H.
English Words and Data Bases: How to Bridge the Gap.
In *20th Annual Meeting of the Association for Computational Linguistics, Toronto.* Association for Computational Linguistics, June, 1982.

[9] Stallard, David G.
Data Modeling for Natural Language Access.
In *Proceedings of the First IEEE Conference on Applied Artificial Intelligence, Denver, Colorado.* IEEE, December, 1984.

[10] Stallard, David G.
Taxonomic Inference on Predicate Calculus Expressions.
Submitted to AAAI April 1, 1986.