

Object-based semantics for natural language engineering in higher education

J. Johnson

*Department of Computer Science, University of Regina, Regina,
Saskatchewan, S4S 0A2, Canada*

Abstract

We present an object-based semantics for informative question answering in the University domain. Semantic rules specify how the internal representation is built from the parse tree for a sentence. A collection of rules is given for generating internal representations for the natural language requests that arise in the University domain. The action of the semantic interpreter is also illustrated by showing how it applies the rules to a particular sentence to produce an internal representation.

1 Introduction

In software engineering education, one of the most challenging problems is to teach students how to go from an informalized description of a problem domain to a precise conceptualization. The informalized description is usually, in whole or in part, expressed in natural language (e.g., English). To assist training in the area of software specification (see also Johnson & Johnson 1992 [7]; 1994 [8]) we wish to capture some aspects of the role that natural language plays in the process of explicating a conceptual or mental model. The student (or the domain expert) must distinguish between concepts and referents and between concepts and words that appear either explicitly or implicitly in the informalized description of the problem domain. A referent is a socially created and in that sense, real world entity “that exists out there”, whereas a concept exists in the informant’s mental model of the phenomenon about which knowledge is being acquired (Ford & Agnew, 1992[2]; Regoczei & Hirst, 1990[14]).

A natural language interface is proposed as a useful system for training in the area of software specification. In such a system, a referent is explicitly represented as a *surrogate* or an internal system identifier. All lexical entries and all concepts in an expert's mental model must also be associated with surrogates. The natural language interface is designed to prompt the domain expert for lexical entries and information about concepts that would permit referents to be associated with surrogates. In this sense part of the job of the system analyst, that of forcing the domain expert to distinguish between concepts, referents and words, is being automated. The operations result in information about the referent domain (the facts) being recorded in the database. It is this process of identification of referents (by associating them with surrogates) that drives the production of a precise description of an informalized reality.

A typical architecture for a natural language interface (NLI) consists of three main components as illustrated by Figure 1 from [12]:

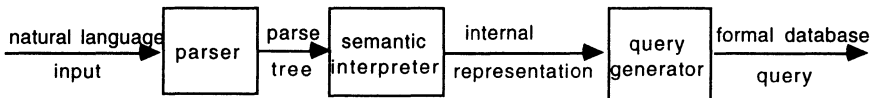


Figure 1: A Typical Natural Language Interface from Kao et. al 1988

As linguistic coverage increases the computational complexity of answering a natural language request is expected to increase. If a natural language interface is designed for a variety of domains then the efficiency of processing will be reduced compared with a design for a single domain. We have argued in [9] that an approach to customization based on designing databases for transportability should avoid the coverage/efficiency tradeoff.

(Ford & Agnew, 1992[2]; Regoczei & Hirst, 1990[14]) suggest that two domains of discourse are needed for knowledge acquisition: the domain expert's "private view" and the "facts". In this paper, we attempt to become more concrete about these proposals. The domain experts private view is conceptualized by means of the SET model and the natural language mediated interaction between expert and analyst by means of an object-based semantics for question answering in natural language.

The object-based approach serves as an integrating formalism for semantic and procedural aspects of natural language processing. Johnson and Rosenberg[10] describe an object-based semantics for translating a natural language request to a formal database query. A semantic modeling capability is provided by a conceptual model (Gilmore's SET model [4]) and a procedural modeling capability by a deductive relational database.

Here we adopt Wood's method [15], for translating the parse tree for a natural language database request to its internal representation. Wood's method is adapted to a new target language (the language DEFINE for the SET model rather than the functional calculus). The advantage of DEFINE

over the functional calculus is a greater economy in the length of the semantic rules used for translating the parse tree to its internal representation. A facility of the SET model to define a set in terms of previously declared sets permits the name of a set to be used in the semantic rules rather than its full definition, simplifying the process of rule definition. The language DEFINE and the SET model are useful for conceptual modeling and a theoretical foundation exists for them [3, 6].

A graphical representation of a University domain appears in [11]. It will be used to facilitate our presentation of an object-based semantics. A sample rule follows:

1 - (G8: (1) = Student) and
2 - (G10: (1) = in and Dept((2)))
 $\Rightarrow$ [For some $\langle 2-2 \rangle$: Dept] $\langle X, 2-2 \rangle$:SD

The rules refer to partial tree structures which serve as templates for language structures. The syntactic component produces a parse tree for the natural language input request. Partial tree structures are matched to subtrees of the parse tree. The right hand side of a rule gives a partial query expressed in the context of a particular SET schema.

Our partial tree structures differ from those originally proposed by Woods by the addition of only one partial tree. The ease with which Woods' partial trees could be adapted to a new domain suggests that they are domain independent and hence useful for transportable natural language interfaces. Since the rules provide an integration of procedural and semantic aspects of language processing, we refer to them as *object - based* semantic rules.

2 Object-Based Semantics for the University Domain

The rules are based on Woods' procedural semantics approach [15]. The action of the semantic interpreter is also illustrated by showing how it applies the rules to a particular sentence to produce an internal representation.

2.1 Semantic Rules for the University Domain

The rules refer to the partial tree structures. The reader is referred to Wood's original paper for diagrams of the partial tree structures. An example of a rule follows:

1 - (G8: (1) = Student) and
2 - (G10: (1) = in and Dept((2)))
 $\Rightarrow$ [For some $\langle 2-2 \rangle$: Dept] $\langle X, 2-2 \rangle$:SD

The syntactic component produces a parse tree for the natural language input request. Partial tree structures are matched to subtrees of the parse tree. The numbers in parentheses in the semantic rules match the numbers in parentheses in the partial tree structures. Arguments for the predicates are specified using a pair of numbers $\langle n1-n2 \rangle$ where $n1$ refers to a numbered

part of the left side of the rule and n_2 to one of the parenthesized numbers in that part.

Conditions in the semantic rules are expressed in terms of the type checking predicates and the relation '='. The condition " $(n)=W$ " is true if string W is identical to the terminal string of the subtree of the parse tree rooted at node n . This is to say that node n dominates the string W . Type checking predicate $P((n))$ is true if n denotes an object of type P . Type checking predicates $Student(x)$, $Course(x)$, $Dept(x)$, $Dname(x)$ for arguments are assumed to be available.

The subconditions on the left-hand side of a rule are numbered, and each refers to exactly one partial tree structure. At any given moment the semantic interpreter is working on a given node in the parse tree which is called the root. The partial tree structures identified by subconditions on the left-hand side of a rule must match a subtree of the parse tree starting at the root. Each subcondition must be true of its matched subtree for the rule to be applicable.

Subcondition 1 of the given rule refers to the partial tree structure labeled G_8 . It is satisfied if G_8 matches the parse tree starting at the root, and the node in the parse that matches the node labeled (1) in G_8 dominates the string "student".

The type of arguments is determined by a top-down analysis of the structure of the object definition. When the given rule is applied to the natural language fragment "students in the Computer Science department", for example, variable 2 in subcondition 2 refers to the noun phrase (NP) "the Computer Science department". It remains to be determined if that NP denotes a department. Further semantic analysis yields a definition the structural properties of which are sufficient to determine if the object defined is a department.

The semantic rules used here differ from Woods' in the right hand side which contains a query fragment expressed in the language DEFINE rather than a functional calculus. The left-hand side is still expressed in a functional calculus. For example, $Student((1))$ on the left and $<1>$: *Student* on the right have the same meaning but are expressed in different languages and serve different purposes.

S-Rules for the University Domain

S1

1- (G_1 : $Student((1))$ and $((2)=enroll$ or $(2) = register)$) and

2- (G_3 : $(1) = be$ and $((2)=enroll$ or $(2)=register)$) and

3- (G_4 : $(1) = in$ and $Course((2))$)

$\Rightarrow <1-1, 3-2>$: SC

e.g.s.

"Jones is enrolled in CPSC 101"

"Jones is registered in CPSC 101"

S2

1- (G_1 : $Prof((1))$ and $((2) = teach$ or $(2) = instruct)$ and

2- (G_2 : $((1) = teach$ or $(1) = instruct)$ and $Course((2))$)

$\Rightarrow <2-2, 1-1>$: CP

e.g. "Dr. Lee teaches CPSC 101"

S3

1- (G_1 : $Course((1))$ and $((2)) = teach$ or $(2) = instruct$ and

2-(G3: ((2) = teach or (2) = instruct) and (1) = be) and
3-(G4: (1) = by and Prof((2)))
 $\Rightarrow <1-1, 3-2>$: CP

e.g.s. "CPSC 101 is taught by Dr. Lee"

S4

1-(G1: Student((1)) and ((2) = receive or (2) = earn or (2) = obtain or
(2) = get)) and
2-(G2: (4) = receive or (1) = earn or (1) = obtain or (1) = get) and
Grade ((2))) and
3-(G4: (1) = in and Course((2)))
 $\Rightarrow <2-2, <1-1, 3-2>>$: SCG

e.g.s.

"Smith received A in CPSC 101"

"Smith in CPSC 101 received A"

Note: An order on the prepositional and verb phrases is not specified by rule S4. To impose an order a different tree structure from G4 would be needed which includes a PP for each of the direct and indirect objects.

S5

1-(G1: Student((1)) and ((2) = take or (2) = study) and
2-(G2: ((1) = take or (1) = study) and Course((2)))
 $\Rightarrow <1-1, 2-2>$: SC

e.g. "Jones takes CPSC 101"

Determiner Rules (D-Rules)

D1

1-(G7: ((1) = some or (1) = a or (1) = any) and (2) = SG)
 $\Rightarrow ([\textbf{For some X: } \Delta] \nabla)$

e.g.s. "any student enrolled in CPSC 101"

"some professor in the computer science department"

D2

1-(G7: ((1) = each or (1) = every or (1) = all) and (2) = SG)
 $\Rightarrow ([\textbf{For all X: } \nabla] \Delta)$

e.g.s.

"all students enrolled in CPSC 101"

"every professor in computer science"

D3

1-(G7: (1) = no and (2) = SG)
 $\Rightarrow (\textbf{not } [\textbf{For some X: } \nabla] \Delta)$

e.g. "no student in CPSC 101"

D4

1-(G7: (1) = **not** every and (2) = SG)
 $\Rightarrow (\textbf{not } [\textbf{For all X: } \nabla] \Delta)$

e.g. "Not every student taught by Dr. Lee"

The symbol ∇ in the D-rules will be replaced by a set name resulting from an N-rule. The symbol Δ will be replaced by an assertion resulting from R-rules (and applications of D-rules, N-rules, and R-rules to NPs lower down in the parse tree).

Noun Rules (N-Rules)

N1

1-(G8: (1) = student or (1) = pupil)

⇒ Student

e.g. "a student"

N2

1-(G8: (1) = graduate)

⇒ Graduate

e.g. "a graduate"

N3

1-(G8:(1) = professor or (1) = instructor)

⇒ Prof

e.g. "a professor"

N4

1-(G8:(1) = course)

⇒ Course

e.g. "a course"

N5

1-(G8: (1) = department)

⇒ Dept

e.g. "a department"

2.2 Additions to Wood's Framework

The N-rules are applied differently depending on whether the head noun is universally of existentially quantified. For existential quantification, the symbol ∇ is replaced by the right-hand side of the applicable N-rule. For universal quantification the NP-processor generates the name of an embedded set which replaces ∇ in the working string, and the initial part of an embedded set declaration replaces the symbol Δ .

Example:

If the working string generated by the D-rules is [**For some X:** ∇] Δ where X is a variable that denotes the NP, then application of an N-rule will replace ∇ with the name of a set appearing on the right-hand side of the N-rule. If the working string is

[**For all X:** ∇] Δ

then the N-rules will produce a string such as

[**For all X:**S]
(S def
select X:T
where Δ)

where X is a variable generated by the NP-processor to represent the NP, S is the name of an embedded set declaration also generated by the NP-processor, and T is the name of a set which appears on the right-hand side of the applicable N-rule.

Restriction Rules (R-rules)

R1

1-(G8: (1) = student) and

2-(G9: (1) = average) and

3-(G10:(1) = in and Course((2)))

$\Rightarrow \langle \langle \langle 3-2 \rangle : \text{CourseAvg}; \rangle, \langle X, 3-2 \rangle \rangle : \text{SCG and } \Delta$

e.g. "an average student in CPSC 101"

R2

1-(G8: (1) = student) and

2-(G9: (1) = average)

$\Rightarrow X:\text{Avg_Student}$

e.g. "an average student"

R3

1-(G8: (1) = student) and

2-(G9: (1) = graduate)

$\Rightarrow X:\text{Graduate}$

e.g. "a graduate student"

R4

1-(G8: (1) = student) and

2-(G10: (1) = in and Dept((2)))

$\Rightarrow [\text{For some } \langle 2-2 \rangle : \text{Dept}] \langle X, 2-2 \rangle : \text{SD and } \Delta$

e.g. "a student in the computer science department"

R5

1-(G8: (1) = department) and

2-(G9: DName ((1)))

$\Rightarrow \langle X, 2-1 \rangle : \text{DN}$

e.g. "the computer science department"

R6

1-(G8: (1) = professor) and

2-(G10: (1) = in and Dept((2)))

$\Rightarrow [\text{For some } \langle 2-2 \rangle : \text{Dept}] \langle X, 2-2 \rangle : \text{PD and } \Delta$

e.g. "a professor in the computer science department"

R7

1-(G8: (1) = computer science course

2-(G9: CName ((1)))

$\Rightarrow X:\text{CPSC_course}$

e.g. "a computer science course"

The symbol X in the R-rules stands for the variable that the NP-processor has created for the noun phrase which it is looking at when it applies the R-rule.

2.2.1 Example: A constraint in the University domain

This example describes the processing of the sentence "Every student in the Computer Science department takes a Computer Science Course". A parse tree for the sentence is given in Figure 2.

The S-processor begins with a working string Δ . The NP-processor is called to process the quantified noun phrase "every student in the computer science department". The NP-processor applies the D-rules, N-rules and R-rules in succession. Δ in the working string is replaced by rule D2 resulting

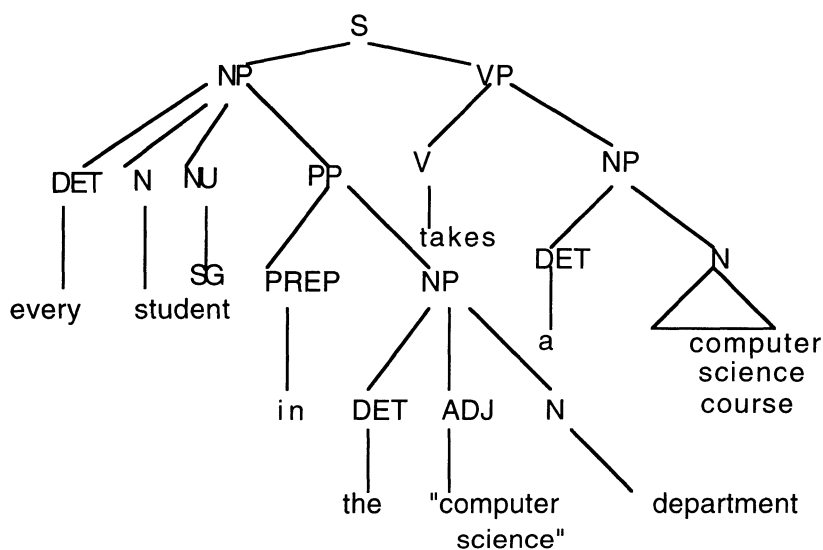


Figure 2: Parse Tree for the Sentence "Every student in the computer science department takes a computer science course."

in the new working string ([**For all** X1: ∇] Δ). X1 is a new variable created by the NP-processor for the noun phrase. Next the N-rules are applied. The result of applying rule N1 is the string

```
[For all X:S]
(S def
select X: Student
where  $\Delta$ )
```

Finally, the R-rules are applied. Rule R4 substitutes Δ in the working string producing

```
[For all X:S]
(S def
select X:Student
where [For some Y:Dept] <X,Y>: SD and  $\Delta$ )
```

X is a variable generated by the NP-processor to denote the embedded NP. The NP-processor is called again to process the embedded NP. None of the D-rules or N-rules apply. The R-rules (R5) substitute Δ in the working string producing

```
[For all X:S]
(S def
select X: Student
where [For some Y:Dept] <X,Y>: SD and <Y, 'Computer
Science'>:DN)
```

The NP-processor is called on the NP "a computer science course" producing

```
[For some Z: Course] Z:CPSC-Course
```

by applying D1, N4. and R7.

The S-processor applies S5 substituting Δ in the working string to produce

```
[For all X:S] [For some Z:Course]
(<X,Z> SC and Z:CPSC_Course)
(S def
select X: Student
where [For some Y: Dept] <X,Y>:SD and <Y,'Computer
Science'>:DN)
```

This concludes our example of object-based semantic processing. Our rationale for selecting the object-based approach is founded on the belief that it provides the power to link declarative and procedural aspects of natural language processing.

This belief is supported by the results of investigations by Beringer and Wandmacher into object-based action planning[1]. In that paper, it is argued that the object-based approach can serve as an integrating formalism for semantic and procedural aspects of human-computer interaction in general.

We have provided an integration of semantic and procedural aspects of modeling for semantic interpretation of sentences and noun phrases. The semantic modeling capability has been provided by the SET conceptual

model and procedural modeling by Wood's procedural semantics. Semantically structured object representations provide meaningful arguments for predicates in production rules which model procedural aspects of language competence.

The ability to change the schema[5, 13] (e.g., the University domain graph) permits an NLI to be customized to the particular mental model of a given expert. The interface for updating production rules is easily expanded to accept new lexical entries and referent concepts by permitting them to be associated with surrogates. There is a need for this form of customization because our notion of expertise evolves based on societal factors[2].

References

- [1] Jörg Beringer and Jens Wandmacher. Object-based action planning. In M.J. Tauber and D. Ackermann, editors, *Mental Models and Human-Computer Interaction 2*, pages 135–155. North-Holland, 1991.
- [2] K.M. Ford and N.M. Agnew. Expertise: Socially situated, personally constructed, and “reality” relevant. In *Proceedings of the 5th Florida Artificial Intelligence Research Symposium*, pages 228–234, 1992.
- [3] Paul C. Gilmore. Concepts and methods for database design. Technical Report TR87-31, Department of Computer Science, University of British Columbia, 1987.
- [4] Paul C. Gilmore. A foundation for the entity relationship approach: How and why? In *Proceedings of the 7th International Conference on Entity-Relationship Approach*, pages 95–113, North-Holland, Amsterdam, 1988.
- [5] C.S. Jensen and L. Mark. Queries on change in an extended relational model. *IEEE Transactions on Knowledge and Data Engineering*, 4(2):192–200, 1992.
- [6] J.A. Johnson. Semantic relatedness. *Computers & Mathematics with Applications*, 29(5):51–64, 1995.
- [7] J.A. Johnson and G.M. Johnson. Student characteristics and computer programming competency: A case study. *Journal of Studies in Technical Careers*, 14(1):33–46, 1992.
- [8] J.A. Johnson and G.M. Johnson. Automating the professional journal technique for computer science education. *Journal of Research on Computing in Education*, 26(4):547–557, 1994.
- [9] J.A. Johnson and R.S. Rosenberg. A data management strategy for transportable natural language interfaces. In *Proceedings of the 5th Florida Artificial Intelligence Research Symposium*, pages 146–150, 1992.

- [10] J.A. Johnson and R.S. Rosenberg. Knowledge sharing for natural language understanding and conceptual modeling for database design. In *Proc. of the 2nd Pacific Rim International Conference on Artificial Intelligence PRICAI'92*, pages 980–988, Seoul, Korea, 1992.
- [11] J.A. Johnson and R.S. Rosenberg. A measure of semantic relatedness for resolving ambiguities in natural language database requests. *Data & Knowledge Engineering*, 7(3):201–225, 1992.
- [12] M. Kao, N. Cercone, and W. Luk. Providing quality responses with natural language interfaces: the null value problem. *IEEE Transactions on Software Engineering*, 14(7):959–984, 1988.
- [13] M.M. Morsi, S.B. Navathe, and H.-J. Kim. An extensible object-oriented database testbed. In *8th International Conference on Data Engineering*, pages 150–157, 1992.
- [14] Stephen Regoczei and Graeme Hirst. The meaning triangle as a tool for the acquisition of abstract, conceptual knowledge. *Int. J. Man-Machine Studies*, 33:505–520, 1990.
- [15] W. Woods. Procedural semantics for a question-answering machine. In *Fall Joint Computer Conference*, pages 458–471, 1968.