THALES: a Software Package for Plane Geometry Constructions with a Natural Language Interface

K. FÁBRICZ
JATE University
Szeged, Egyetem u. 2., Hungary, H-6722

Z. ALEXIN, T. GYIMÓTHY, T. HORVÁTH
Research Group on the Theory of Automata at the
Hungarian Academy of Sciences,
Szeged, Somogyi u. 7., Hungary, H-6720
e-mail: sztaki!ella!h42gyi

Abstract

THALES is a software package for plane geometry constructions, supplied with a natural language interface. Using THALES requires no knowledge of a programming language. The interface is capable of processing practically all kinds of instructions within the subset of plane geometry English. The "static semantic" module has been generated on the basis of a high-level attribute specification. Transportability, modifiability and generality -- the key issues of natural language interface design -- are investigated in the project note. The notion of **specifiability** is introduced to replace the three features mentioned above.

1. Introduction*

Natural language interfaces represent one of the most common applications of natural language processing. In the eighties, not only a considerable increase in natural language interface refinement has been achieved, but also methods for design and evaluation have been worked out [Mart 83, Schr 88]. One might think that a natural language interface of the nineties would be properly described in terms of three parameters, viz.,

(i) transportability
(ii) modifiability by the user
(iii) generality

Our experience with THALES shows that none of these features is attainable in the near future. Rather, natural language interfaces based on well-defined subsets of languages and supplemented with possibly full semantics appear to be real candidates for applications in the following years. Requirements for transportability and generality run counter to the need to supply the interface with as full semantics as possible. Modifiability by the user can be maintained at a price that is hardly worth paying. If one lets the user tailor the interface to his/her needs, that natural language interface should provide this possibility by relaxing syntax or semantics. We do not consider this a real alternative. Neither does modifiability in the form of substitution of one expression by another on the basis of their semantic identity seem to be a realization of what the term "modifiability by the user" suggests (Cf. [JAKE 88]).

However, there appear to be different parts of natural language interface design that may be reused in other applications. We shall refer to this possibility by the term **"specifiability"** -- a feature that merits thorough consideration.

2. An Overview of THALES

THALES is a program package for plane geometry constructions. The constructions are carried out as series of instructions formulated in English. Through a menu system, it is possible to save constructions and parts of constructions for some later application, the constructions may be inserted into other applications. The instructions can be stored and edited as normal ASCII files. A number of examples covering the majority of tasks related to plane geometry are supplied to help the teacher or student in teaching or learning plane geometry. Manipulating objects during constructions is also supported without the risk of running into ambiguities in the course of reference resolution.

3. The natural language interface of THALES

The natural language interface of THALES is briefly characterized as follows.

Linguistic processing is based on an attribute grammar description. From this specification, the PROF-LP language processor [Gyim 88] generates the parser and the evaluator. The lexicon consists of a fixed number of items covering the sublanguage of plane geometry. The parser contains a

lexical and a syntactic analyzer. The former does morphological analysis as well in order to keep the size of the vocabulary at a minimum. The syntactic module represents a top-down, basically LL(1) algorithm augmented by attributes for initiating semantic processing. Semantic interpretation is based on a meta-level object description that provides a clue to the implementation of relations in the form of different steps in producing geometrical constructions [Ale 89].

Intersentential reference is resolved by building a symbol table. The range of syntactic and semantic coverage can be illustrated by the examples below.

```
Pick a point.
Label the point by ~A.
Draw a circle around point ~A.
Pick a point ~B on the circle.
Draw radius ~A~B.
Pick a point ~C outside the circle.
Construct triangle ~A~B~C.
Inscribe a circle ~q in the triangle.
Find a point ~P that is inside the
triangle and outside circle ~q.
From point ~P, draw a circle with
radius ~A~B.
Label the intersections of the
circles by ~D and ~E.
Connect the points of intersection.
Create the hexagon ~A~B~C~P~D~E.
```

4. Generating natural language interfaces

The experience gained from designing and implementing THALES has led to the following conclusions.

4.1. Evaluation of natural language interfaces

A natural language interface establishes connection between the user and some other language or set of user-defined procedures. In constructing a natural language interface, the primary aim is to provide a means for the translation of natural language sentences into sentences of that other language or into procedure calls. The latter define the semantics of execution for the input sentence(s).

When the value of a natural language interface is being assessed, transportability, modifiability and generality seem to be false starting points. These features would make a natural language interface truly valuable, were it not for the fact that they cannot be taken *prima facie*. A transportable natural language interface means some kind of over-generalization. For a natural language interface to be transportable, it is necessary that we have a sublanguage with too wide a coverage to carry specific information encoded in its semantics. That is why transportability is achieved only in highly identical domains (characteristically, interfaces to databases). A natural language interface can be modified by the user according to his/her specific purposes exceptionally in the lexicon. Hence, the semantics is prewired, the syntax is fixed, and what you gain is hardly more than a replacement of one or two mnemonics with some other character strings that you might prefer. The notion of generality of a natural language interface is too vague to deserve special attention. It might refer to syntactic and/or semantic coverage, it might imply the existence of a fairly large word stock, but, we should think, it would hardly ever be taken to mean "general" in the sense that it could be applied in different domains.

4.2. Specifiability of natural language interface architecture

A basic question in designing natural language interfaces of the THALES type is that of assessing which parts can and which cannot be generated. An answer to this question would throw light on the problem of transportability, modifiability and generality. We argue that those parts of a natural language interface which can be **generated** from some high level specification appear to closely satisfy the demand for transportability, modifiability and generality. It seems to be the case that the lexical, syntactic and static semantic analyzers are suitable for generation on the basis of an attribute grammar description.

It should be noted that the lexical analyzer also contains the morphological module. The syntactic parser processes a CF grammar. Dynamic semantics is also suitable for analysis by an attribute grammar description, but there are too many application-specific elements whose generation does not seem to be a realistic goal for the time being.

Of crucial importance is the notion of static (compile time) semantics in the case of natural language interfaces like THALES. In a conventional programming language, we define its static semantics as the set of context-sensitive properties that can be evaluated, checked or calculated during compile time. The most important properties of this kind are

a) the differentiation between defining and applied occurrences of variables
b) the assessment of the scope of defining occurrences
c) the problem of identification: finding the defining occurrences that match the applied occurrences

d) the question of type compatibility

Conventional compilers contain a symbol table that will serve as a basis for the investigation of the questions raised above. Elements of the symbol table are built from objects of the types recognized by the language, supplemented with auxiliary information. During compilation the symbol table undergoes continuous modification, problems related to static semantics are solved on the basis of the actual state of the symbol table.

If one compares static semantic analysis in the case of natural language interfaces like THALES, it appears that basic types correspond to objects with a hierarchic structure. E.g. a triangle is represented as having vertices, bisectors, altitudes etc. Again, e.g. altitudes have endpoints, a midpoint, length etc. Endpoints, in turn, have coordinates expressed as numbers. Such hierarchic structure can be ascribed to other natural language interfaces as well. A specification language [Ale 89] is used for type definition. In the course of specification, other data can also be incorporated (adjectival, prepositional co-occurrence restrictions etc). On the basis of this specification it is possible to generate procedures for symbol table manipulation. The symbol table can then serve as a tool for deciding type compatibility. For example, one can refer to the midpoint of an altitude due to the fact that an altitude is of segment type, and segments do have a midpoint.

In the differentiation of defining and applied occurrences, determiners play a crucial role. While in conventional programming languages identification is resolved by a relatively simple algorithm, this problem in THALES-like natural language interfaces is much more complicated. Below we give a sample of the identification methodology used in THALES. This methodology can be extended on the specificational level or, vice versa, the application of some of the methods can be prohibited. The abbreviations ID, OBJ, and ADJ stand for **identifier** (a unique name of an object, denoted by '~' in THALES), **object** (the name of an object, possibly a part of another object), and **adjective**, resp.

OBJ ID - an object with a unique name to be found unambiguously, cf.

Draw the radius of **circle ~C**.

OBJ - search for the first occurrence of the object with the given type by tracking the symbol table backward, cf.

Pick a point in **the circle**.

ADJ OBJ - a search for the first occurrence of that type of object specified by the adjective, cf.

Draw a circle inside **the acute triangle**.

4.3. Conclusions

From the experience with THALES we can conclude that the notion of static semantics can be successfully applied to the group of natural language interfaces represented by THALES. The static analyzer in THALES appears to satisfy the requirement for transportability, modifiability and generality, although in a modified and more realistic sense. That is, we do not think that there is a one-to-one correspondence between a generated static analyzer and the requirements. Rather, the static analyzer can be specified in a high-level language, and that specification can meet the transportability, modifiability and generality requirements.

---

* The authors gratefully acknowledge the financial support of Cogito Ltd., Philadelphia.

References

[Ale 89] Alexin, Z., Fábricz, K., Gyimóthy, T., Horváth, T. CONSTRUCTOR: A Natural Language Interface Based on Attribute Grammar. In: T. Gyimóthy (ed.) *Proceedings of the First Finnish-Hungarian Workshop on Programming Languages and Software Tools*, Szeged, 1989, pp. 135-145.

[Cliff 88] Clifford, J. Natural Language Querying of Historical Databases. *Computational Linguistics* 14 (4), 10-34.

[Gyim 88] Gyimóthy, T., Horváth, T., Kocsis, F., Toczki, J. Incremental Algorithms in PROF-LP. *Lecture Notes in Computer Science*, 371. pp. 93-103.

[JAKE 88] JAKE. The Application-Independent Natural Language User Interface. English Knowledge Systems, Inc., Scotts Valley, California, 1988.

[Mart 83] Martin, P., Appelt, P., Pereira, F. Transportability and Generality in a Natural Language Interface System. *Proceedings of IJCAI-83*, 1, pp. 573-581.

[Schr 88] M. Schröder. Evaluating User Utterances in Natural Language Interfaces to Databases. *Computers and Artificial Intelligence*, 7 (4), pp. 317-337.