stichting
mathematisch
centrum

MC

AFDELING ZUIVERE WISKUNDE
(DEPARTMENT OF PURE MATHEMATICS)

ZW 74/77 APRIL

T.M.V. JANSSEN

SIMULATION OF A MONTAGUE GRAMMAR

Preprint

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

AMS(MOS) subject classification scheme (1970): 68A30, 02C99, 02C10

Simulation of a Montague Grammar*

by

T.M.V. Janssen

ABSTRACT

In his article "The Proper Treatment of Quantification in Ordinary English", the logician R. Montague deals with the syntax and semantics of a certain fragment of English. The present paper is concerned with a computer simulation that follows the proposals of Montague. In order to explain some of the problems and solutions which arose during the design of the program, a -partial- introduction to Montague grammar is presented. Examples of generated sentences are discussed, furthermore several inaccurracies and errors in Montague's article are pointed out.

KEY WORDS & PHRASES: *Montague Grammar, Computational linguistics*

*This report will be submitted for publication elsewhere.

1. INTRODUCTION

In order to give an example of the problems treated by Montague grammar we consider the following sentences:

(1) John seeks a unicorn.

(2) John finds a unicorn.

These sentences are very much alike; only their verbs differ. One is tempted to expect that their meanings are related in the same way: only John's activities are different. The difference in meaning, however, goes further. One sentence gives information about the existence of unicorns which is not implied by the other.

We may describe the meaning of (2) as follows. There are two individuals which stand in the find-relation to each other. The first individual (the finder) is John. The second individual is some -further unspecified- member of the set of unicorns. So the meaning of (2) can be expressed by the following formula:

(3) $\exists x[\mathit{unicorn}(x)\ \&\ \mathit{find}(\mathit{John},x)]$

With a simple logical deduction rule we derive from (3) that, just as desired, (4) holds:

(4) $\exists \mathrm{x}[\mathit{unicorn}(x)]$

Consequently we cannot express the meaning of (1) by a formula analogous to (3), since we may not conclude from (1) that there exist unicorns. So we need another kind of formula in order to express the meaning of (1). It should express that John stands in the seek-relation with "something"; but this "something" is not necessarily an existing individual.

Montague grammar aims to provide a systematic relation between the syntax and semantics of sentences like (1) and (2). The main difference with transformational grammars (grammars in the tradition of Chomsky) is that in transformational grammar one primarily treats the syntactic part of grammar, while in Montague grammar one also deals explicitly with the semantic component.

In "The Proper Treatment of Quantification in Ordinary English", (MONTAGUE 1973, from now on referred to as "PTQ"), the logician Richard Montague presented a treatment of the syntax and semantics of a fragment of English. In this fragment semantically interesting referential expressions occur frequently.

The present paper deals with a computer simulation of the proposals of PTQ. Some of the difficulties which arose during the development of the computer program and their solutions will be discussed, as well as other results. In order to be able to do so, we will present an introduction to Montague's proposals. This introduction will not cover all aspects of them, but will provide the information needed for the discussion. A more elaborated introduction is given in "Montague Grammar and Transformational Grammar", (PARTEE 1976).

2. REFERENTIAL EXPRESSIONS

Consider the following sentences:

(5) John walks.

(6) A man walks.

(7) Every man walks.

These sentences are syntactically much alike; they can all be split up in the verb phrase "walks" and a singular noun phrase. In Montague grammar one wants to relate the semantic interpretation of an expression in a systematic way with the syntactic structure. Therefore we wish to have for the subjects in all three sentences the same kind of semantic interpretation.

The verb phrase "walks" will semantically be considered as a property. In (5) we may consider "John" as the indication of an individual which has this property. In (6) we might consider "a man" as denoting a rather unspecified individual which has the property of walking. But (7) causes problems. There is no individual such as "the universal man". (What would for instance, be his age? Since not every man is 35, the universal man cannot be 35, nor can he have any other age, nor can he be ageless.) So we cannot interprete "every man" semantically as an individual. These considerations force us to try another approach for "every man" and our desire for a systematic relation between syntax and semantics leads us to follow this approach also for "John" and "a man".

In Montague grammar a noun phrase is semantically interpreted as a set of properties. An individual, say John, is characterized by the set of all his properties. When we consider two individuals, there is certainly a property which holds for one of them and not for the other (e.g. the property of being at a certain moment on a certain place; two different individuals are not always at the same moment at the same position). More information on the philosophical arguments for treating noun phrases semantically as sets of properties can be found in "General Semantics" (LEWIS 1970).

We denote the meaning of sentences by means of logical formulas. The sets of properties mentioned above are denoted in a way which might require some explication. Consider

$$P(john)$$

This formula expresses that John has the property P. Consider next

$$\lambda P[P(john)]. \tag{8}$$

By means of the symbols λP is indicated that we have to abstract from the property P in the expression between the square brackets. Formula (8) denotes a function which for each property says whether that property holds for John or not. Let us write χ_j for the expression $\lambda P[P(john)]$. Then χ_j is the function such that for any predicate P:

$$\chi_j(P) = \begin{cases} \underline{true} & \text{if } P \text{ holds for } john \\ \underline{false} & \text{otherwise.} \end{cases}$$

This function is called the characteristic function of the set of properties of John. Instead of speaking about sets of properties, we speak about their characteristic functions. As usual in logic we shall sometimes identify these two concepts and speak about a set where we ought to speak about the characteristic function of a set.

Let us calculate the value of χ_j for the argument *man*.

$$\chi_j(man) = \begin{cases} \underline{true} & \text{if } man \text{ holds for the argument } john, \\ & \text{thus if } man(john) = \underline{true} \\ \underline{false} & \text{otherwise, thus if } man(john) = \underline{false} \end{cases}$$

From this we may conclude that

$$\lambda P[P(john)](man) = \chi_j(man) = man(john).$$

So the value for argument α of a function expressed as $\lambda P[\phi]$ equals ϕ', where ϕ' is obtained from ϕ by substituting α for each occurence of P. (see section 5 for a restriction.)

The semantical interpretation of the sentence "John walks" runs as follows. The expresssion " John " is taken as a function (the characteristic function of the set of properties of John) and the expression "walks" is taken as the argument of this function. This is expressed by the formula

$$[\lambda P[P(john)]](walk)$$

As we observed above, this reduces to

$$walk(john).$$

The noun phrase "a man" is treated analogously . The formula

$$\exists x[man(x) \ \& \ P(x)]$$

expresses that there is an individual x which is a man and which has property P . The formula

$$\lambda P[\exists x[man(x) \ \& \ P(x)]]$$

denotes the characteristic function of the set of properties such that for each property in the set there is a man which has the property. The sentence "A man walks" is semantically interpreted as

$$[\lambda P[\exists x \ man(x) \ \& \ P(x)]](walk)$$

which reduces to

$$\exists x[man(x) \ \& \ walk(x)]$$

Now the sentence "Every man walks" can be treated in the same way. The noun phrase "every man" is semantically taken as the characteristic function

$$\lambda P \forall x[man(x) \rightarrow P(x)]$$

So the sentence as a whole is analysed as

$$\lambda P \forall x[man(x) \rightarrow P(x)](walk).$$

This formula reduces to

$$\forall x[man(x) \rightarrow walk(x)].$$

Notice that the formulas which we obtained and which express the interpretation of the sentences (5), (6) and (7) are exactly the formulas usually associated with them in elementary logic courses. There, however, the formula's are found by pure intuitive considerations, while in Montague grammar they are the result of the formal system which relates syntax and semantics in an explicit way. In the sequel we will consider the system in more detail.

3. SYNTAX

In the preceding section we observed that sentences can be split into parts. In Montague's approach syntactic rules tell us in what ways compound expressions are constructed from smaller ones. The words are the basic units of the syntactic constructions. A category is a set of expressions which behave the same in greater units: replacing an expression by another expression of the same category does not change the syntactic wellformedness of a sentence.

Below we will give, along with some comments, the rules from PTQ which are needed to generate the sentences mentioned in sections 1 and 2. Other rules are presented in the examples.

S4: S $\rightarrow$ T + IV & replace first verb in IV.

The name of this rule is S4. The rule states that a sentence (S) may consist of a term (T) followed (+) by an intransitive verb phrase (IV). In this IV phrase the first verb should be replaced by its third person present singular form. The term is called the first argument of the rule, the IV phrase the second argument. For other rules we use the same terminology.

S1: $IV \rightarrow$ walk

An IV phrase may consist of the word "walk". We use "S1" as the name of several rules; it indicates that the rule consists in selecting a word.

S1: $T \rightarrow$ John | Mary

The symbol | separates alternatives. So "John" is a term and "Mary" is a term.

S2a: $T \rightarrow$ a + CN

A term may consist of the word "a" followed by a common-noun phrase.

S2b: $T \rightarrow$ every + CN

S1: $CN \rightarrow$ man | unicorn | centaur

S5: $IV \rightarrow TV + T$

A TV indicates a transitive verb phrase.

S1: $TV \rightarrow$ find | seek

The rules mentioned above define a set of sentences and some implementation of them is used in the program for generating sentences. The essential features of this part of the program will be described below. For convenience of formulation a rather antropomorphic terminology will be used.

The computer wishes to generate a sentence. Syntactic rule S4 tells him that he has to make first a T phrase, next a IV phrase and finally combine them in a certain way. So first he makes a term. There are several instructions determining how this could be done: S1, S2a and S2b. He chooses at random one of them, say S1, and chooses the word "John". Next he makes a IV phrase. Suppose rule S5 is chosen. Then he has to choose a TV, say "seek", and he has to make a term again. This might be the phrase "a unicorn". Thus the sentence "John seeks a unicorn" is generated.

The structural aspects of the derivational history of this sentence are reflected in the graphical representation given below. Notice that the two arguments of a rule start on the same position on a line. The first argument is the upper one, the second the lower one.

```
S1:  T:           John
S1: TV:               seek
S1: CN:                   unicorn
S2a: T:               a unicorn
S5: IV:           seek a unicorn
S4:  S:       John seeks a unicorn
```

We notice that making a phrase involves making phrases of other categories: the arguments of the chosen rule. For each category this process runs along the same lines. So the natural way to describe this process is by means of recursive procedure. The kernel of this procedure is as follows:

```
procedure make(category)
begin rule := choose rules for (category);
    if rule is not S1
    then begin make(argument1 of (rule));
            if has two arguments(rule)
            then make(argument2 of (rule))
        end
    else choose lexical element of(category)
end
```

4. SEMANTICS

In section 2 we noticed that in Montague grammar we wish to obtain a semantic interpretation for each syntactic unit. This interpretation is represented by a logical expression. Just as we combined smaller syntactic phrases into larger ones, we combine the expressions corresponding to the smaller phrases into a compound expression which represents the interpretation of the larger phrase. This parallelism between the syntax and semantics is obtained as follows. For each syntactic rule there is a semantic rule which describes how the formulas corresponding to the arguments of the syntactic rule have to be combined to form a compound expression. These semantic rules are called translation rules since they constitute a translation of syntactic structures into logical expressions.

The translation rules corresponding to the syntactic rules of section 3 are as follows:

T4: $[\mathrm{T} + \mathrm{IV}]' = \mathrm{T}'(\mathrm{IV}')$

The name of this translation rule is T4. If a sentence is constructed according to rule S4, then its translation is a construct consisting of a function and a argument. The translation of the term (written as T') is taken as the function and the translation of the IV phrase(IV') as the argument. Notice that the translation of ϕ is denoted as ϕ'.

T1: walk' = *walk*

If an IV consists of the word "walk", then its translation is the logical symbol *walk*.

T1: John' = $\lambda P[P(john)]$

The translation of "Mary" is analogous,

T2a: $[\mathrm{a} + \mathrm{CN}]' = \lambda P \exists x[\mathrm{CN}'(x) \ \& \ P(x)]$

T2b: $[\mathrm{every} + \mathrm{CN}]' = \lambda P \forall x[\mathrm{CN}'(x) \rightarrow P(x)]$

T5: [TV + T]' = TV'(T')

T1: man' = *man*

Analogously for "unicorn", "centaur", "find" and "seek".

Let us see how the computer translates "John seeks a unicorn" (see section 3 for the structure of this sentence). According to rule T4 this means that first the term "John" and the IV phrase "seek a unicorn" are translated. The translation of the term is $\lambda P[P(john)]$. In order to translate the IV phrase first the TV "seek" and then the term "a unicorn" have to be translated. Again, this process is implemented by a recursive procedure. The intermediate stages of the translation process are presented in the same way as those of the generation process.

T1: $\lambda P[P(john)]$

T1: $seek$

T1: $unicorn$

T2a: $\lambda P \exists x[unicorn(x)\ \&\ P(x)]$

T5: $seek\ (\lambda P \exists x[unicorn(x)\ \&\ P(x)])$

T4: $\lambda P[P(john)]\ (seek(\lambda P \exists x(unicorn)(x)\ \&\ P(x)))$

As was noticed in section 2, this formula can be reduced to

$$seek(\lambda P \exists x[unicorn(x)\ \&\ P(x)])(john)$$

Usually one treats "seek" as a two-place relation. We therefore accept the convention to write a formula of the form $(\gamma(\beta))(\alpha)$ as $\gamma(\alpha,\beta)$. Applying this convention to the formula given above we obtain the following result:

$$seek(john,\ \lambda P \exists x\ [unicorn(x)\ \&\ P(x)]).$$

This formula expresses that the seek-relation holds between the individual John and (the characteristic function of) a certian set of properties. The formula does not imply that there are individuals with any of these properties. So, as required, the formula does not imply that there exist unicorns.

If in the course of the generation process, the computer had chosen the transitive verb "find" instead of "seek", then sentence (9) would have been generated:

(9) John finds a unicorn.

The translation process would have been the same and the obtained formula would have been:

(10) $find(john, \lambda P \exists x[unicorn(x) \ \& \ P(x)])$.

But in this case we wish to express that it is allowed to conclude that there exist unicorns. Therefore (10) is not satisfactory. In order to remedy this we introduce a logical law in our system. We postulate that for all expresssions α and β, the following formulas are equivalent:

$$find\ (\alpha,\beta) \quad \text{and} \quad \beta(\lambda y[find(\alpha,y)])$$

This law is called the meaning postulate for *find*. Applying it to (10) gives

$$\lambda P \exists x[unicorn(x) \ \& \ P(x)] \ (\lambda y[find(john,y)])$$

this reduces to

$$\exists x[unicorn(x) \ \& \ \lambda y[find(john,y)] \ (x)]$$

which reduces to

$$\exists x[unicorn(x) \ \& \ find(john,x)]$$

This formula implies, as desired, that there exist unicorns.

One of the aspects of Montague grammar that will not be discussed extensively in this paper is the use of the concepts extension and intension. Since these concepts play such an important role in Montague's work, however, I feel obliged to indicate where the system as presented so far is unsatisfacory, and what intensions have to do with it.

Suppose that in the present state of the world there exist no unicorns. Then for no property P it is true that $\exists x[\mathit{unicorn}(x)\ \&\ P(x)]$. Thus in these circumstances $\lambda P\exists x[\mathit{unicorn}(x)\ \&\ P(x)]$ is the characteristic function of the empty set of properties. The semantic interpretation of "John seeks a unicorn" then states that the seek-relation holds between John and this empty set. Suppose moreover that in the present state of the world also no centaurs exist. Then the semantic interpretation of

(11) John seeks a centaur

also expresses that the seek-relation holds between John and the empty set of properties. But this contradicts our intuition that (1) and (11) have a different meaning.

When we wish to describe the difference between centaurs and unicorns we cannot restrict our attention to the present state of this world. We should also consider other worlds (or other states of this world) for example those where unicorns or centaurs do exist. In other worlds the set $\lambda P\exists x[\mathit{unicorn}(x)\ \&\ P(x)]$ might be different from $\lambda P\exists x[\mathit{centaur}(x)\ \&\ P(x)]$. The intension of $\lambda P\exists x[\mathit{unicorn}(x)\ \&\ P(x)]$ is the function which for each world indicates what is the set of properties such that there is a unicorn in that world with such a property. The intension of $\lambda P\exists x[\mathit{unicorn}(x)\ \&\ P(x)]$ is different from the intension of $\lambda P\exists x[\mathit{centaur}(x)\ \&\ P(x)]$ since in some of these worlds these sets are different. The seek-relation will be considered as a relation between individuals and intensions of sets. Since the intensions are different, "seeking a unicorn" will get an interpretation different from the one for "seeking a centaur".

Since, with respect to the computer program, intensions are not such a central aspect, and since they would complicate the exposition, we shall neglect intensions in the sequel. Therefore the translation rules and examples presented in this paper, differ slightly from those in the computer program, respectively output.

5. SIMULATION

5.1 Program Design

The components of a Montague grammar and their relation to each other are presented in the following diagram:

$$\begin{tikzcd}
\boxed{\begin{array}{l}\text{readable}\\ \text{sentence}\end{array}} & \boxed{\begin{array}{l}\text{syntactic}\\ \text{structure}\end{array}} \arrow[l, "\uparrow\ \text{formation}"'] \arrow[r, "\uparrow\ \text{translation}"'] & \boxed{\begin{array}{l}\text{logical}\\ \text{formula}\end{array}} \arrow[r, "\uparrow\ \text{reduction}"'] & \boxed{\begin{array}{l}\text{reduced}\\ \text{formula}\end{array}}
\end{tikzcd}$$

This scheme suggests how the program, according to the principle of modularity, should be designed. Four separately designed procedures, each performing its task completely and then delivering the resulting structure or formula as input for the next procedure. Such a design, however, conflicts with the way in which everybody works by hand. While making the translation one already starts to reduce the intermediate stages. A complete translation (without reductions) of a slightly complex sentence would be quite unreadable because of its length. If we obtain as a final result a formula which is not satisfactory, then we have no intermediate stages to find out where the trouble came in. These considerations show that a modular simulation would be less desirable. In the final version of the program, therefore, after each step of the translation it is tried to reduce the formula obtained in that stage.

5.2 Language

In the sections 3 and 4 we have considered the generation and translation process and we have seen that the natural way to deal with them was by means of a recursive procedure. With respect to this the choice of ALGOL 60 as programming language is a good one: opposed to FORTRAN, it allows for writing recursive procedures. But in another respect this choice was not so good: ALGOL 60 is weak in manipulation of strings of characters. The PTQ fragment has words as basic units and we can work with encodings of them. There is one exception: the rule S4 which changes the main verb (and related rules for the other tenses). So the program must change "love" in "loves", "try to" in "tries to" and "be" in "was". At the moment these changes are effected in a rather ad-hoc way. If one wishes to deal with a Montague grammar for a language in which word changes occur frequently (for instance German with its case endings), then the choice of another language is to be preferred (a good choice would be ALGOL 68).

5.3 Reduction

The formulas obtained in the translation process have to be reduced to simpler ones. The rules needed for this can be distinguished into three types

I	Notational conventions	e.g. writing $\gamma(\alpha,\beta)$ for $(\gamma(\beta))(\alpha)$
II	Meaning postulates	e.g. the meaning postulate for *find*
III	Logical Laws	e.g. the rule for eliminating λ-expressions

There is no list of these rules mentioned in PTQ. When working "by hand" this causes no great trouble since on most cases one can see intuitively what would be a correct and a succesful step for further simplification. A computer however needs a list of universally applicable reduction rules. For the rules of type I and type II the construction of this list required a painstaking examination of PTQ. The formulations in PTQ are sometimes unclear, and sometimes difficult to interpret as general applicable rules (see the examples of reductions in PARTEE 1976).

Rules of type III are not mentioned in PTQ at all. Practical experience in working with PTQ gives an indication which kind of rules were needed. It required investigations in the special kind of logic used in PTQ (intensional logic) in order to prove that the rules would yield a correct result in all circumstances. Even the traditional laws of logic have to be treated carefully. Consider

$$[\lambda x\ \underline{Fut}(powerful(x))]\ (the\ queen\ of\ Holland)$$

In this formula <u>*Fut*</u> indicates the Future tense operator. The first part of the formula denotes the characteristic function of the set of individuals x such that in the future they will be powerful. The formula as a whole expresses that the queen of Holland (thus Juliana) belongs to this set. On the other hand

$\underline{Fut}$ *(powerful (the queen of Holland))*

indicates that on a certain moment in the future the queen of Holland on that moment (e.g. Beatrix) will be powerful. This is not equivalent with the first formulation! The list of rules which are needed for reduction can be found in JANSSEN 1976. This list is the result of an interaction of practical experience, discussion, computer output and theoretical investigations. Thus simulation gave rise to theoretical deepening.

6. INTERESTING EXAMPLES

6.1 Strange Sentences

The grammar of PTQ has no tools for arranging the selection of the right combinations of e.g. verb, subject and object. Because the computer generates at random, strange sentences occur frequently. Examples are:

The park runs;
Mary walks about the pen in the park;
Mary wishes to be the park in ninety .

Presumably, such selection restrictions will be formulated in the semantic component of a Montague grammar, rather than in the syntactic component.

6.2 And

Beside the already mentioned rules for generating IV phrases, we also have:

IV → try to + IV

IV → IV + and + IV.

By means of these rules the following sentence is generated:

John tries to walk and talk.

Since "walk and talk" is an IV phrase, also the following incorrect sentence is generated:

John walks and talk.

This demonstrates that the PTQ formulation of S4 ("replacing the first verb") is too simple.

6.3 Adverbs

One of the categories of PTQ we did not mention yet is IAV, the category of IV-modifying adverbs. Semantically they are considered as operators on the translation of IV-phrases. Some of the rules involving IAV's are:

S10:	IV → IAV + IV
S1:	IAV → slowly
T10:	(IAV + IV)' = IAV'(IV')
T1:	slowly' = *slowly*

These rules generate the following structure

S1: T:	John
S1:IAV:	slowly
S1: IV:	walk
S10:IV:	slowly walk
S4: S:	John slowly walks

The process of translation is indicated by

T1:	$\lambda P[P(john)]$
T1:	$slowly$
T1:	$walk$
T10:	$slowly(walk)$
T4:	$[\lambda P[P(john)]](slowly(walk))$

This reduces to

(slowly (walk))(john)

The formulation in PTQ of the notational convention for $(\gamma(\beta))(\alpha)$ would allow us to rewrite this as a relation:

slowly (john,walk).

Applying the same convention allows us to consider the phrase "about a unicorn" in the sentence "John talks about a unicorn" as a relation. It is clear from the examples mentioned in PTQ, however, that IAV phrases are not meant to be considered as relations. The computer used the rule in all its consequences and thus made us aware of the incorrect formulation of it.

6.4 Such that : Syntax

The PTQ fragment includes relative clauses. The following rules are needed to generate them. Since for each integer n there is an instance of these rules, they constitute an infinite series.

S1: $T \rightarrow he_n$

S3,n: $CN \rightarrow CN$ + such that + S & replace he_n by he/she/it according to the gender of the first term or common-noun in the CN phrase

By means of these rules the following structure is generated

S1:	CN:	price
S1:	T:	he_1
S1:	IV:	rise
S4:	S:	he_1 rises
S3,1:	CN:	price such that it rises
S2b:	T:	every price such that it rises
S1:	IV:	change
S4:	S:	Every price such that it rises changes

This is quite a nice sentence, but the computer also generated many strange sentences by using these rules. In the structure above he chose "he_2" instead of "he_1", but still used rule S3,1. Then this rule changed nothing in the sentence "he_2 rises". Thus the final result is "Every price such that he_2 rises changes". Since he_2 is not an English word, the sentence as a whole is incorrect. Also the choice of a correct word would produce a incorrect sentence: the choice of "John" instead of "he_1" results in "Every price such that John rises changes". Again the computer applied a rule in circumstances where a human being would have recognized nonsense. The computer program now contains instructions which guarantee a correct correspondence between occurrences of he_n and rules eliminating them.

6.5 Such that : Semantics

The translation rules which correspond to the syntactic rules of 6.4 are:

T1 :	$[he_n]' = \lambda P[P(x_n)]$
S3,n:	$[CN + \text{such that} + S]' = \lambda x_n[CN'(x_n)\ \&\ S']$

The translation of the syntactic structure given in 6.4 is described as follows.

T1:	$price$
T1:	$\lambda P[P(x_1)]$
T1:	$rise$
T4:	$\lambda P[P(x_1)](rise)$
reducing to:	$rise(x_1)$
T3,1:	$\lambda x_1[price(x_1)\ \&\ rise(x_1)]$
T2b:	$\lambda P\forall x[\lambda x_1[price(x_1)\ \&\ rise(x_1)](x) \rightarrow P(x)]$
reducing to:	$\lambda P\forall x[price(x)\ \&\ rise(x) \rightarrow P(x)]$
T1:	$change$
T4:	$\lambda P\forall x[price(x)\ \&\ rise(x) \rightarrow P(x)]\ (change)$

reducing to:

$$\forall x[price(x) \ \& \ rise(x) \rightarrow change(x)]$$

Rule S3,1 can be applied more than one time in succession. A part of a structure in which this occurs is indicated by:

```
                        woman
                        he₁ loves a woman
S3,1:               women such that she loves a woman
                    he₁ runs
S3,1:           woman such that she loves a woman such that she runs
```

This noun phrase might occur in the sentence "Mary is a woman such that she loves a woman such that she runs". This is a correct sentence. From this sentence we may conclude that Mary loves a running woman. The semantics, as presented above, considers "he$_1$ runs" and "he$_1$ loves a woman" as specifications of the same woman. The resulting formula for the common-noun phrase is

$$\lambda x_1[woman(x_1) \ \& \ \exists y[woman(y) \ \& \ love(x_1,y)] \ \& \ run(x_1)].$$

This formula expresses that one in the same women is loving and running. So here we obtained a formula which does not express the meaning of the sentence.

7. CONCLUSION

Simulation goes hand in hand with practical experience and theoretical investigations. Decisions made during the design of the program were based upon practical experience. Computer simulation appeared to be an excellent way to test the consequences of the proposed rules. Most of the errors found by the simulation were the result of an unintended, but not forbidden application of a rule. This showed that some of the rules in PTQ were incorrectly formulated. Last but not least: the simulation was a source of theoretical questions; general applicable rules ahd to be found and their correctness had to be proven.

REFERENCES

[1] Janssen, T.M.V., (1976), "A computer program for Montague grammar: theoretical aspects and proofs for the reduction rules", in J. Groenendijk & M. Stokhof (eds.), *Amsterdam Papers in formal grammar 1, Proceedings of the Amsterdam colloqium on Montague Grammar and related topics,* pp. 154-176. Centrale Interfaculteit, University of Amsterdam.

[2] Lewis, D., (1970), "General Semantics", *Synthese* 22, pp. 18-67, Also in B. Partee (1976), *Montague Grammar,* Academic Press, New York.

[3] Montague, R., (1973), "The proper treatment of quantification in ordinary English", in R.H. Thomason (1974): *Formal Philosophy, Selected papers of Richard Montague,* Yale University Press, New Haven and London, pp. 247-270.

[4] Partee, B., (1975), "Montague Grammmar and Transformational Grammar", *Linguistic Inquiry* 6, pp. 203-300.