OUTLINE OF AN INTEGRATED THEORY OF NATURAL LANGUAGE UNDERSTANDING

By

Frank M. Brown

C B Schwind

TR-84 October, 1978

Department of Computer Sciences, The University of Texas at Austin

[illegible]

[illegible]

[illegible]

[illegible]

OUTLINE OF AN INTEGRATED THEORY OF NATURAL LANGUAGE UNDERSTANDING

by
Frank M. Brown
Department of Computer Sciences
University of Texas at Austin

C B Schwind
Technische Universität München

Abstract

We describe and exemplify the underlying structure of a new kind of theory of natural language understanding which would allow for a rich and smooth interaction between syntactic and semantic concepts.

The basic idea is to let the representation language in which the meaning of natural language text is expressed be the same language as the language in which the parsing and translation laws are expressed. This is done by supplementing a modal quantificational state logic with names of both natural language and logical expressions. Natural language expressions are then parsed and translated into logical expressions which in turn are translated into their logical meanings by use of a meaning function. Since both the laws of parsing, translation and meaning are themselves expressed in logic one can easily write such laws so as to interact with the meaning of the natural language text.

Key Words and Phrases: Natural Language Parsing and Understanding, Preference Semantics, Theory of Meaning, Modal Quantificational Logic, Quantified State Logic.

March 1978

Contents

1. Introduction

Our research is aimed towards creating a theory of natural language understanding such that the representation language in which the meaning of natural language text is represented is iteelf the language in which that theory is described. This is, laws for the description of the syntax of natural language, laws for the translation of natural language into meanings, and the representation of those meanings are all written in a single representation language. We believe that a theory of this nature has important technical and methodological advantages over other types of theories of natural language understanding in that we will be able to precisely and concisely state laws involving all aspects of the precess of natural language understanding, Because both syntactic and meaning concepts are represented in a single language we will call such a theory an integrated theory of natural language understanding.

Since not only must laws of natural language understanding be representable in our representation language, but the meaning of natural language text must also be representable, it is clear that this language must be a language of great representational ability. Partially, for this reason we will assume -that the representation language is eome logic at least as strong as modal quantificational logic.

2. Basic Structure of an Integrated Theory

As it is in the case of other theories of natural language as well as with practical natural language understanding systems, an integrated theory involves at least three basic processes: (see Figure 1).

1. The parsing of natural language expressions and their translation into meanings, which are expressed in the representation language.
2. The inference of meanings expressed in a representation langauge from other meanings expressed in a representation language.
3. The generation of natural language expressions from meanings which are expressed in a representation language.

In our case the representational language will be a modal quantificational logic supplemented by a theory of action. The parsing system as well as the translation system are described by axioms of our theory which are intended to be executed by an automatic theorem prover.

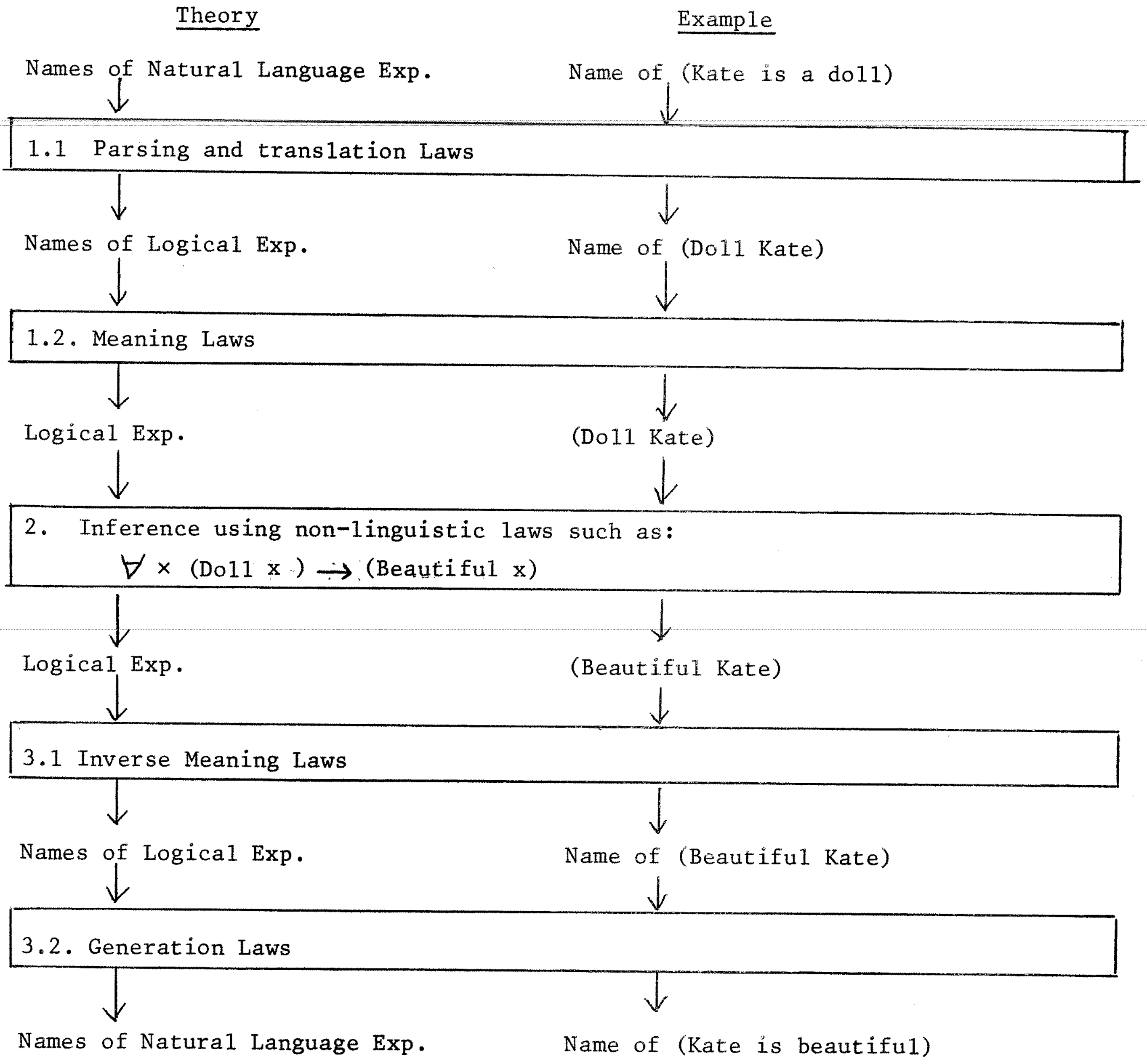

Fig. 1. Basic Structure of an Integrated Theory

IT is important to understand that natural language expressions are syntactic objects which are expressed in our systems by names of such expressions whereas the meaning of such expressions are represented in our system, not by names of logical expressions, but rather by logical expressions themselves. For this reason we can divide each of the parsing and generation steps into two steps: (see Figure 1).

(1.1) A parsing and translation step which translates natural language expressions, which are represented by their names, into logical expressions which are represented by their names.

(1.2) A meaning step which translates logical expressions, which are represented by their names, into their meanings which are represented by logical expressions.

Likewise, the generation step can be divided into two steps: (see Figure 1).

(3.1) An inverse meaning step, which translates meanings represented by logical expressions, into those logical expressions which are represented by names of logical expressions.

(3.2) A generation step which translates logical expressions, which are represented by their names, into natural language expressions, which are represented by their names.

The inference step (2) allows the derivation of logical expressions by means of logical inference rules, axioms of logic, and non-logical laws pertaining to both linguistic subjects such as: parsing, generation and meaning; and non-linguistic subjects such as a model of various physical and social facts about the real world.

There are two basic reasons for using a representation language, and in particular for the use of logic as the representation language rather than using natural language itself as the representation language. The first reason is to simplify the description of both the linguistic and non-linguistic axioms, in that such laws will not have to be stated in terms of the complex syntax of natural language which is often very ambigious, but will be expressed in the much simpler syntax of logic. The second reason is -that the inference laws for making logical derivations have been profoundly studied whereas there is no comparable body of knowledge about how inference in natural language might work. In particular efficient automatic theorem proving techniques have been developed for quantificational logic, and even modal quantificational logic,[1] whereas no such comparable automatic inference techniques have been developed for natural language. Thus by using logic as the representation language we have the option of actually testing our theory by executing it on an automatic theorem prover.

2.1. Formal Orientation

We have indicated that an integrated theory is to be represented in a logic at least as strong as modal quantificational logic. The syntax of such a language includes:

(1) Logical connectives

$p \wedge q$	p and q
$p \vee q$	p or q
$p \rightarrow q$	if p then q
$p \leftrightarrow q$	p iff q
$\sim p$	not p
■	true
¤	false

(2) Logical Quantifiers

$\forall x\, \varphi x$	For all objects x, φ of x holds
$\exists x\, \varphi x$	For some object x, φ of x holds
$\forall p\, \varphi p$	For all propositions p, φ of p holds
$\exists p\, \varphi p$	For some proposition p, φ of p holds

(3) Intentional and Extentional Equality

$x = y$	x intentionally equals y
$x \bumpeq y$	x extentionally equals y

Intentional equality: = possesses substitution properties over all expressions including those containing modal symbols. Extentional equality does not possess substitution properties over modal symbols. Thus:

$$x = y \rightarrow (\varphi x \leftrightarrow \varphi y)$$

$$p = q \rightarrow (\vdash p \leftrightarrow \vdash q)$$

but $p \bumpeq q \rightarrow (\vdash p \leftrightarrow \vdash q)$ does not hold.

(4) Description Operators

$℩x\, \varphi x$ the x such that φ of x

where: $$\Psi(℩x\, \varphi x) \leftrightarrow \left(\exists y\, (\forall x(\varphi x \leftrightarrow x = y) \wedge \Psi y) \vee \sim\exists y (\forall x(\varphi x \leftrightarrow x = y)) \wedge \Psi \,\text{nil}) \right)$$

the $x\, \varphi x$ the x such that φ of x

where: $(\text{the } x\, \varphi x) = \text{df } ℩x(\varphi x \wedge \mathcal{F} x)$

where $\mathcal{F}$ is a special linguistic function which we will leave undefined.

(5) Modal Connectives

$\vdash p$	p is logically true
$\vdash pq$	p entails y
$\Diamond\ p$	p is possible

The axioms of our modal logic are those of S5 modal logic plus Lerbnitz's law which states that something is logically true if it is true in all possible worlds:

$$S5 \left[\begin{array}{ll} M0: & \text{from } p \text{ infer } \vdash p \\ M1: & \vdash p \rightarrow p \\ M2: & \vdash(p \rightarrow q) \rightarrow (\vdash p \rightarrow \vdash q) \\ M3: & \vdash p \vee \vdash \sim \vdash p \end{array}\right.$$

Lieb: M4: $(\forall w(\Diamond w \wedge \forall p \vdash wp \vee \vdash w(\sim p)) \rightarrow \vdash wq) \rightarrow \vdash q$

(6) Tense operators and other non-logical symbols

(present p)	p holds in the present
(past p)	p holds in the past

(7) Variables which range over various domains

Sinee the parsing and generation law~~s~~ of an integrated theory refer to both natural language and logical expressions, it follows that the theory must also include names of these expressions. This is done first by including in the theory a name for each logical or natural language symbol and then by forming names of expressions by representing them as a list of the names of the subexpressions occurring in that expression.

Names of symbols are formed by simply prefixing to that symbol an accent sign $'$. Thus $'\wedge$ is a name of the $\wedge$ symbol of logic, and 'and is the name of the and symbol of English. The apparent visual similarity between a symbol such as $\wedge$ and its name $'\wedge$ is merely a pneumonic.

Lists are formed as in LISP by use of two symbols Nil and Cons in

Nil
(Cons x y)

where (Cons xy) is an ordered pair and Nil is not an ordered pair. A list $[x_1 \ldots x_n]$ of zero or more elements is then defined in terms of cons and il as follows:

$$[x_1 \ldots x_n] = \text{df } (\text{Cons } x_1 \ldots (\text{Cons } x_n \text{ Nil}) \ldots)$$

Note that [ ] is simply Nil. Sometimes we will also use the abbreviation [x,y] for (Cons xy).

Given these lists, and names of symbols we can now form names of arbitrary expressions. For example a name of the expression:

((all men) are mortal)

is

[[′all ′men] ′are ′mortal]

We will also use various selector functions which select subparts of a list. In particular:

(Car x)	such that (Car(Cons x y)) = x
(Cdr x)	such that (Cdr(Cons x y)) = y
(Cadr x)	such that (Cadrx) = (Car (Cdr x))

The Modal Logic on which this theory is based is developed in more detail in [1,2]. The Tense logic used in this theory is described in [3,4]. The Syntactic devices are described in [5]]

2.2. Parsing and Translation

The laws of parsing and translation in an integrated theory state how to translate expressions of natural language into equivalent expressions of logic. Generally such laws are equivalences between an atomic sentence and a conjunction of atomic sentences:

$$A_1 \leftrightarrow (B_1 \wedge \cdots \wedge B_n)$$

For example a law of parsing might say that s is a sentence (Sent_1) iff the first part of s is a noun phrase (NP), the second part of s is a form of the verb "Be" (VBE), and the last part of s is an adjective (adj).

$$(\mathrm{Sent1}\ s) \leftrightarrow \left(\begin{array}{l} (\mathrm{NP}\ (\text{first part of } s)) \\ \wedge (\mathrm{VBE}\ (\text{second part of } s)) \\ \wedge (\mathrm{adj}\ (\text{last part of } s)) \end{array} \right)$$

The "part of" idioms are handled in logic by representing each part of a sentence as the difference of two lists x_i, x_{i+1}. For example if

x0 is [′The ′Tree ′is ′pretty]

x1 is [′Is ′Pretty]

then the difference x0 − x1 intuitively represents [′The ′tree].

Using these difference lists we can write the above parsing law as:

$$(\mathrm{Sent}_1\ x0{-}x3) \leftrightarrow ((\mathrm{NP}\ x0{-}x1) \wedge (\mathrm{VBE}\ x1{-}x2) \wedge (\mathrm{adj}\ x2{-}x3))$$

Thus for example if we wish to parse the sentence

[′the ′tree ′is ′pretty]

we would try to prove

(Sent1 [′the ′tree ′is ′pretty]-[])

Then by using the above parsing law we could deduce

(NP [′the ′tree ′is ′pretty] - x1) ∧ (VBE x1-x2) ∧ (adj x2-[])

By using further parsing laws pertaining to noun phrases we would instantiate x1 to [′is ′pretty] thus finding that [′the ′tree] was a noun phrase. After this only two subgoals remain:

(VBE [′is ′pretty] - x2) ∧ (adj x2- [])

which are easily proven by using parsing laws pertaining to the verb BE and to adjectives, in the course of which x2 will be instantiated to [′pretty].

As we parse a sentence we will also want to translate it into an equivalent logical expression. This is done by including in each atomic sentence an extra argument place for the logical expression which is roughly equivalent to the natural language expression contained in the difference list. For example the above parsing law might be modified to state that x0-x2 is a sentence translated as the logical expression [β α] iff x0-x1 is a noun phrase translated as α, x1-x2 is a form of the verb BE, and x2-x3 is an adjective translated as β:

(Sent1 x0-x3 [β α]) ↔ ((NP x0-x1 α) ∧ (VBE x1-x2) ∧ (adj x2-x3 β))

Thus if the noun phrase [′the ′tree] is translated as the logical constant ′TREE21 and if the adjective [′pretty] is translated as the unary predicate ′PRETTY then the sentence [′phe ′treee′is ′pretty] will be translated as [′PRETTY ′TREE21].

We have just given a parsing law which states that something is a sentence iff it is a noun phrase followed by an intransitive verb followed by an adjective. But is is clear that this is not the only possible immediate constituent structure for a sentence of English. For example something could be a sentence if it begins with a noun phrase followed by a transitive verb. Thus what is needed is a distinct atomic name: Sent1, Sent2 ... for each parsing law about sentences, and then to say that something is a sentence iff it is a sentence of type 1 (Sent1), or a sentence of type 2 (Sent2) etc. Thus in general it is clear that we also need parsing laws in which the right side is a disjunction of atomic sentences:

$$A \leftrightarrow (A_1 \vee \ldots \vee A_n)$$

such as:

(Sent x0-x1 α) ↔ ((Sent1 x0-x1 α) ∨ (Sent2 x0-x1 α))

Disjunctive parsing laws are also used to express the lexicon. For example

$$(\text{adj } [x,y] -y\ z) \leftrightarrow \begin{pmatrix} (x = {}'\text{pretty} \wedge z = {}'\text{PRETTY}) \\ \vee (x = {}'\text{red} \wedge z = {}'\text{RED}) \end{pmatrix}$$

is used to express the fast that ′pretty and ′red are the only adjectives in the lexicon and that ′pretty and ′red are translated into logic as respectively the unary predicates

′PRETTY and ′RED. Note that [x,y] -y intuitively represents [x].

So far we have specified that atomic sentences are to have three arguments: two for the difference list representing a part of the sentence being parsed and one in which to specify the equivalent logical expression. Usually the atomic sentences will have several more arguments. For example often they will have two more arguments usually written v_i v_{i+1} which are essentially a difference list of variables of the logical object languages. It is clear that such variables are needed since a sentence like [′all ′trees ′are ′pretty] would be translated into logic as

$$[{}'\forall\ {}'x\ [{}'\text{tree}\ {}'x] \rightarrow [{}'\text{pretty}\ {}'x]]$$

where ′x is the first variable in the difference list for variables. Also it should be noted that some atomic sentences do not contain all the arguments specified here. For example the intransitive verb sentence does not contain an argument position for the equivalent logical expression, since there is none.

We say that z is a translation of the natural language sentence s iff z is a translation of the sentence s-[] using a sufficiently long difference list of variables V-[]:

$$(\text{Trans } s\ z) \leftrightarrow (\text{Sent } s\text{-}[]\ \ v\text{-}[]\ z)$$

Then to find a translation of a sentence s we merely try to prove that there is an α which is its translation

$$\exists z\ (\text{Trans } s\ z)$$

We now give in Figure 4 some parsing and translation laws for a very small subset of English in order to illustrate how both syntactic and meaning concepts interact in our theory. Figure 2 contains a description of the categories of expressions used in these laws. Figure 3 contains a summary of these parsing laws obtained by deleting all the arguments of the atomic sentences. Figure 3 is of no help in understanding the translation process but it will at least give one a general idea of the grammar used by these laws.

The example parsing and translation laws given in Figure 2 are derived from a larger parsing and translation algorithm, for both English and German, described in [6,7].

We also define a function tran which chooses any one translation z of the sentence S.

$$(\mathrm{trans}\ S = (\mathrm{choice}\ z\ (\mathrm{trans}\ s\ z))$$

This function will be used in section 2.5. The choice function obeys the axiom:

$$\mathrm{Ax:}\quad (\exists x \phi x \rightarrow (\phi(\mathrm{choice}\ \underline{z}\ \phi\ z))$$

for any property ϕ .

Basic Categories	
DETD	definite determiner
DETQ	quantifier
N	noun
ADJ	adjective
PREP	preposition
VBE	the verb "to be"
VDO	the verb "to do"
VT	transitive verb
PRON	pronoun

Derived Categories	
SENT, SENT1, SENT2, SENT3	sentence
CLAUSE	embedded sentence
NP, NP1, NP2, NP3, NP4	noun phrase
NG, NG1, NG2, NG3	noun group
pp	prepositional phrase

Figure 2. Grammar.

Grammar:

Sent ↔ Sent1 ∨ Sent2 ∨ Sent3
Sent1 ↔ NP ∧ VBE ∧ Adj
Sent2 ↔ NP ∧ VT ∧ NP
Sent3 ↔ VDO ∧ NP ∧ VT ∧ NP

clause ↔ relpron ∧ VT ∧ NP

NP ↔ NP1 ∨ NP2 ∨ NP3 ∨ NP4
NP1 ↔ NG
NP2 ↔ Detq ∧ NG
NP3 ↔ Detd ∧ NG
NP4 ↔ Pron

NG ↔ NG1 ∨ NG2 ∨ NG3
NG1 ↔ N
NG2 ↔ NG ∧ PP
NG3 ↔ NG ∧ Clause

PP ↔ Prep ∧ NP

Lexicon:

Detd ↔ the
Detq ↔ (a ∨ some ∨ all ∨ every)
N ↔ (tree ∨ trees ∨ garden ∨ gardens
∨ rose ∨ roses ∨ cone ∨ cones)
adj ↔ (pretty ∨ red)
Prep ↔ in
VBE ↔ is ∨ was
VDO ↔ do ∨ does ∨ did
VT ↔ (owns ∨ owned ∨ grow ∨ grows
∨ have ∨ has ∨ had ∨ grew)
Relpron ↔ which ∨ that ∨ who
Pron ↔ somebody ∨ everybody

Figure 3. Summary of Parsing Laws.

```
(SENT xo-x1 vo-v1 z) ↔ (SENT1 xo-x1 vo-v1 z) ∨ (SENT2 xo-x1 vo-v1 z)
                       ∨ (SENT3 xo-x1 vo-v1 z)
(SENT1 xo-x3 vo-v1 [zo(subst [z2 y] for * in z1)]) ↔
∃ x1 ∃ x2((NP xo-x1 vo-v1 y z1) ∧ (VBE x1-x2 zo) ∧ (ADJ x2-x3 z2))
(SENT2 xo-x3 vo-v2
[(cadr zo) (subst(subst[(car zo)y1 y2] for * in z2) for * in z1)]) ↔
∃x1∃x2 v1((NP xo-x1 vo-v1 y1 y1 z1) ∧ (VT x1-x2 zo) ∧ (NP x2-x3 v1-v2 y2 z2))
(SENT3 xo-x4 vo-v2
[?[zo(subst(subst (car z3)y1-y2] for * in z2) for * in z1)]]L↔
∃x1∃x2 x3 v1((VDO xo-x1 zo) ∧ (NP x1-x2 vo-v1 z1) ∧
            (VT x2-x3 z3) ∧ (NP x3-x4 v1-v2 y2 z2))
(CLAUSE xo-x3 vo-v1 y1 [(cadr zo) (subst[(car zo) y1 y2] for * in z1)])↔
∃x1∃x2((RELPRON xo-x1) ∧ (VT x1-x2 zo) ∧ (NP x2-x3 vo-v1 y2 z1))
(NP xo-x1 vo-v1 y z) ↔ ((NP1 xo-x1 vo-v1 y z) ∨ (NP2 xo-x1 vo-v1 y z) ∨
                       (NP3 xo-x1 vo-v1 y z) ∨ (NP4 xo1x1 vo-v1 y r))
(NP1 xo-x1 vo-v1 y [' y[z '∧ *]]) ↔ (NG xo-x1 vo-v1 y z)
(NP2 xo-x2 vo-v1 y [(car z1) y [z2 (cdr z1)*]]) ↔
∃x1((DETQ xo-x1 z1) ∧ (NG x1-x2 vo-v1 y z))
(NP3 xo-x2 vo-v1 ['the y z]*) ↔
∃x1((DETD xo-x1) ∧ (NG x1-x2 vo-v1 y z))
(NP4 xo-x1 [y.v1]-v1 y [z y *]) ↔ (PRON xo-x1 z)
(NG xo-x1 vo-v1 y z) ↔
((NG1 xo-x1 vo-v1 y z) ∨ (NG2 xo-x1 vo-v1 y z) ∨ (NG3 xo-x1 vo-v1 y z))
(NG1 xo-x1  [y.v1] -v1 y [z y]) ↔ (N xo-x1 z)
(NG2 xo-x2 vo-v2 y1 [z1 '∧ z2]) ↔
∃x1∃v1 ((NG xo-x1 vo-v1 y1 z1) ∧ (PP x1-x2 v1-v2 y1 y2 z2))
(NG3 xo-x2 vo-v2 y1 [z1 '∧ z2]) ↔
∃ x1∃v1 ((NG xo-x1 vo-v1 y1 z1) ∧ (CLAUSE x1-x2 v1-v2 y1 z2))
(PP xo-x2 vo-v1 yo y1(subst [zo yo y1] for * in z1)) ↔
∃x1((PREP xo-x1 zo)∧(NP x1-x2 vo-v1 y1 z1))
```

Lexicon

```
(DETD [x.xo]-xo) ↔ x='the
(DETQ [x.xo]-xo z) ↔ (((x='a ∨ x='some)∧ z=['∃ '∧ ]) ∨
                      ((x='all∨ x='every)∧ z=['∀ '→]))
```

Figure 4. The example Grammar

cont'd...

Fig. 4 continued

(N[x.xo]-xo z) ↔

(((x='tree ∨ x='trees) ∧ z='TREE) ∨ ((x='garden ∨ x='gardens) ∧ z='GARDEN)

∨((x='rose ∨ x='roses) ∧ z='ROSE) ∨((x='cone ∨ x='cones) ∧ z='CONE))

(ADJ[x.xo]-xo z) ↔ ((x='pretty ∧ z='PRETTY) ∨ (x='red ∧ z='RED))

(PREP[x.xo]-xo z) ↔ (x='in ∧ z='IN)

(VBE[x.xo]-xo z) ↔ ((x='is ∧ z='PRESENT) ∨ (x='was ∧ z='PAST))

((VDO x.xo -xo z) ↔ (((x='do ∨ x='does) ∧ z='PRESENT) ∨ (x='did ∧ z='PAST))

(VT[x.xo]-xo z) ↔

(((x='own ∨ x='owns) ∧ z=['OWN 'PRESENT]) ∨ (x='owned ∧ z=['OWN 'PAST]) ∨

((x='grow ∨ x='grows) ∧ z=['GROW 'PRESENT]) ∨ [(x='grew ∧ z=['GROW 'PAST]) ∨

((x='had ∨ x='has) ∧ z=['HAS 'PRESENT]) ∨ (x='had ∧ z=['HAS 'PAST]))

(RELPRON [x.xo]-xo) ↔ (x='which ∨ x='that ∨ x='who)

(PRON [x.xo]-xo z) ↔ ((x='somebody ∧ z='∃) ∨ (x='everybody ∧ z='∀))

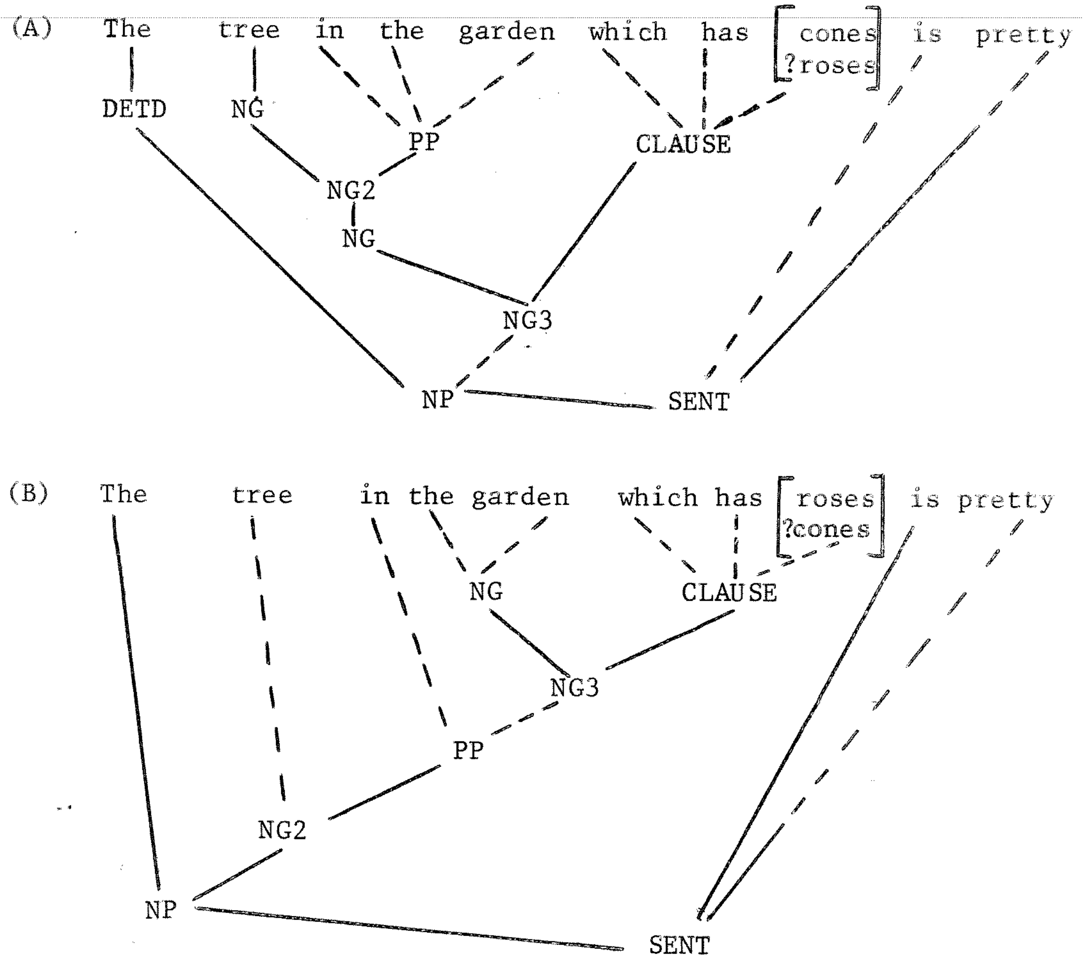

Figure 5. Syntactically ambiguous sentences.

2.3. Meaning

In an integrated theory, the laws of meaning state how to translate logical expressions into their meanings. Such laws are equations which recurse over the syntactic structure of the expressions of the logical object language. These laws are given in Figure 6. In that figure (M S) is interpreted as "the meaning of S" and (m S A) is interpreted as "the meaning of S in the association list A." An association list is simply a list of pairs:

```
[[v1.x1]...[vn.xn]]
```

The association list is used to keep track of which object language variables vi are bound to which metalanguage variables xi.

An example of a meaning law is m∧: which says that the meaning of [S '∧ T] equals the meaning of S and the meaning of T.

Using these laws the meaning of the expressions of the logical object language can be determined. For example, the meaning of the sentence:

```
['∀ "z [['Man 'z] '→ ['Mortal 'z]]]
```

may be obtained as follows:

```
(M['∀ 'z [['Man 'z] '→ ['Mortal 'z]]])   :M
(m(closure['∀ 'z [['Man 'z] '→ ['Mortal 'z]]])   :closure
(m['∀ 'z [['Man 'z] '→['Mortal 'z]]])   :m ∀
∀x (m [['Man 'z] '→ ['Mortal 'z]] [['z.x]])   :m→
∀x (m ['Man 'z] [['z.x]]) → (m['Mortal 'z] [['z.x]])   :m Φ twice
∀x (Man(m 'z [['z.x]])) → (Mortal(m 'z [['z.x]]))   :mV twice
∀x (Man x) → (Mortal x)
```

The Meaning Function given here is described in more detail in [5].

```
                (M S) = (m(closure S) [ ])
m ∧:            (m[S ′∧ T]A)=  (m S A) ∧ (m T A)
m ∨:            (m[S ′∨ T]A)=  (m S A) ∨ (m T A)
m →:            (m[S ′→ T]A)=  (m S A) → (m T A)
m ↔:            (m[S ′↔ T]A) =(m S A) ↔ (m T A)
m ∼:            (m[′∼ S]A) = ∼(m S A)
m ◼:            (m ′◼ A)  =  ◼
m ¤:            (m ′¤ A)  =  ¤
m ∀:            (m[ ′∀ v S]A) = ∀x (m S[[v.x].A])
m ∃:            (m[ ′∃ v S]A) = ∃x (m S[[v.x].A])
m= :            (m[x ′= y ]A) = (m S A) = (m T A)
m ⊢:            (m[′⊢ S]A) =  ⊢(m S A)
m pr:           (M[′present S]A) = present (m S A)
m pa:           (m[′past S]A) = past (m S A)
m v :           (m vA) = (val v A)
m φ :           (m[′φ  x1 ... xn]A) = (φ(m x1 A) ... (m xn A))
                          for every non logical symbol ′φ of the logical object
                          language
m ℩             (m[′℩ v S]A) = (℩x  (m S [[v.x].A]))
m The           (m[′The  v S]A) = (the x (m S[[v.x].A]))
v1:             (val v[[v.x].A]) = x
v2:             u ≠ v → (val v [[u.x].A]) = (val v A)
```

Figure 6. Meaning Laws

Notes for Figure 6:

- x,y,x1 ...-xn   range over object language expressions
- S,T   range over object language sentences
- u,v,v1 ... vn   range over object language variables
- The closure of an object language sentence S with free variables v1 ... vn is:

  [′∀ v1 ... [′∀ vn S]...]

- A rangesover association list where an association list is a list of pairs each whose first argument is an object language variable.

  [[v1,x1]...[vn.xn]]

- The value (val v A) of a variable v in an association list

  A = [[v1.x1]... [vn.xn]]

  is defined to be the first xi whose vi equals v.

2.4. Interaction between Parsing and Meaning

In section 2.2 we described a syntactic parser which made no use what-so-ever of the meaning of the expressions it was parsing. One problem with such syntactic parsers is that there are generally many possible syntactically correct parsings of any given Natural Language sentence. For an example, there are two different syntactically correct parsings of each of the following sentences:

1. The tree in the garden which has cones is pretty.
2. The tree in the garden which has roses is pretty.

The possible parsingsof each of these sentences is given in Figure 5. It should be clear that for semantic reasons we would like to parse the second sentence as parsing (B) and never as parsing (A) because it is false to claim that trees have roses. It should also be clear that for semantic reasons we would like to parse the first sentence as parsing (A) and not parsing (B), not because it is false to say that a garden has cones, but because it is more preferable to say that a tree or a plant has cones rather than to say that a garden or place has cones.

How can this semantic information about which possible parsings to reject be represented in our parser? If we look back at the parsings in Figure 5 and compare them we see that they differ in the parsing they give to the constituent:

"tree in the garden which has {cones/roses}"

for whereas the first parsing states that this is an NG3, the second states that it is an NG2. Clearly then what we need to do is to modify the NG axiom of Figure 4 so as to state that if both NG3 and NG2 parsings are possible then we want to reject one of them on semantic grounds. That is we want to say that the parsing accepted by the NG law should be the parsing produced by the NG2 law only if either there is no parsing produced by the NG3 law or if there is one and the meaning of the logical expression which is the translation of the natural language expression parsed by the NG2 law is more likely than the meaning of the logical expression which is the translation of the natural language expression which was parsed by the NG3 law. Likewise a similar rule should hold for the NG3 law. We modify the NG law as follows:

```
 (NG xo-x1 vo-v1 y z) ↔
((NG1 xo-x1 vo-v1 y z) ∨
((NG2 xo-x1 vo-v1 y z) ∧ (∀z1(∼∃y(NG3 xo-x1 vo-v1 y z1)) ∨
                         (LIKELIER(∼ (M['∼ z])) ('∼(M['∼z1])))))
((NG3 xo-x1 vo-v1 y z) ∧ (∀z1(∼∃y(NG2 xo-x1 vo-v1 y z1)) ∨
                         (LIKELIER( ∼(M['∼ z])) ('∼(M['∼ z1])))))
```

It is important to notice that the arguments to the propositional function Likelier are meanings of sentences of logic rather than the sentences themselves. Thus for example we are not saying that the NG2 parsing is to be used if a particular sentence is more likely than another but rather we are saying that it is to be used if a meaning of a sentence is more likely than another.

When would we like to say that one meaning is more likely than another? In accordance with the example parsings of our two sentences it appears to be reasonable to say that one meaning is more likely than another if the first meaning is consistent with our current knowledge of the world and the second meaning is not. This would allow us to reject the parsing which claimed that a tree has roses since we would expect $N\ \exists x\ \exists y\ [TREEx \wedge ROSEy \wedge HASxy]$to be deducable from general knowledge. Also in accordance with the example parsings producing the statements that a tree has cones, and a garden has cones, it appears to be reasonable to say that one meaning is more likely than another if the first meaning is consistent with the current world knowledge and entails a meaning which is more preferable than a meaning which is entailed by the second meaning. This would allow us to reject the parsing which claimed that a garden has cones, since trees are plants, gardens are places, cones are fruit and it is more preferable for plants to have fruit than it is for places to have fruit.

A definition of Likelier satisfying these intuitions is given below:

$$(\vdash_w (LIKELIERxy)) \leftrightarrow_{df}$$

$$(\Diamond (w \wedge x) \wedge \sim \Diamond (w \wedge y)) \vee$$

$$(\exists P\ \exists Q(\Diamond (w \wedge x) \wedge \vdash (w \wedge x \rightarrow P) \wedge \vdash (w \wedge y \rightarrow Q) \wedge \vdash_w (PREFER\ P\ Q))$$

The symbol w represents a conjunction of the nonlogical axioms expressing facts of the current state of "real world". The PREFER symbol is either to be defined in terms of more elementary concepts or may be axiomatized by the inclusion of axioms like

$$S \vdash_w (PREFER(\exists x \exists y(PLANTx \wedge FRUITy \wedge HASxy))\ (\exists x \exists y(PLACEx \wedge FRUITy \wedge HASxy)))$$

expressing that plants have fruits rather than places do. Since a tree is a plant, a garden is a place, and a cone is a fruit we can prove that it is more likely for trees to have cones than for gardens and hence we can reject the parsing which claims the latter.

2.5. Dialogue Control

The purpose of the dialogue laws are to state the basic social behaviour of a natural language understanding system. That is, such laws determine when and what the system is to communicate with the outside world, and how the world knowledge of system is changed by such communications.

We represent such communication and world state by the use of three time predicates::

(In t) Input communication at time t
(Out t) Output communication at time t

The Input at any time t is defined by the person communicating with the system in the following manner by asserting:

$$(\mathrm{In}\ t) = \Sigma$$

where Σ is a name of the input sentence. For example the input at time 0 is asserted to be: "all trees are pretty" as follows:

(In 0) = ['All 'Trees 'Are 'Pretty]

Once a sentence is given to the system the system will react in various ways depending on whether the input sentence is a declarative sentence or a Yes/No question sentence.

Since our parsing and translation laws return a list consisting of the symbol ? followed by a logical sentence in the case of a question sentence, and only a logical sentence in the case of a declarative sentence. We use the atomic sentence (Isdel α) to determine the type of sentence. (Isdel α) is defined as follows:

$$(\mathrm{Isdel}\ \alpha) \leftrightarrow \mathrm{df}\ \sim(\mathrm{Car}\ \alpha) \hat{=} ?$$

The system is also defined to react differently depending on whether the meaning of a declarative input sentence is consistent or contradictory with the system's state of knowledge. In all there are four social laws which govern the systems basic behaviour.

The first law states hhat if the meaning of a declarative input sentence is consistent with the systems beliefs then the meaning of that sentence is added to the systems beliefs and the next output is that the system believes that sentence:

C1: $(\vdash(\mathrm{Bel}\ t)\ x \hat{=} (\mathrm{trans}(\mathrm{Int}))) \wedge (\mathrm{Isdel}\ x) \wedge \Diamond ((\mathrm{Bel}\ t) \wedge (\mathrm{M}\ x))$
$\rightarrow (\mathrm{Bel}\ t+1) = ((\mathrm{Bel}\ t) \wedge (\mathrm{M}\ x)) \wedge (\mathrm{Out}\ t+1) = [\text{'I 'Believe 'You}]$

The second law states that if the meaning of a declarative input sentence is inconsistent with the systems beliefs then the system remains the same and the next output is that the system disbelieves that sentence:

C2: $(\vdash(\mathrm{Bel}\ t)\ x \overset{\frown}{=} (\mathrm{trans}(\mathrm{In}\ t))) \wedge (\mathrm{Isdel}\ x) \wedge \sim \Diamond((\mathrm{Bel}\ t) \wedge (\mathrm{M}\ x))$

$\rightarrow (\mathrm{Bel}\ t+1) = (\mathrm{Bel}\ t) \wedge (\mathrm{Out}\ t+1) = [\text{'I 'disbelieve 'you}]$

The third law states that if the meaning of a Yes/No question is true according to the systems beliefs then yes is returned:

C3: $(\vdash(\mathrm{Bel}\ t)\ x \overset{\frown}{=} (\mathrm{trans}(\mathrm{In}\ t))) \wedge \sim (\mathrm{Isdel}\ x) \wedge (\vdash(\mathrm{Bel}\ t)\ (\mathrm{M}(\mathrm{Cadr}\ x)))$

$\rightarrow (\mathrm{Bel}\ t+1) = (\mathrm{Bel}\ t) \wedge (\mathrm{Out}\ t+1) = [\text{'Yes}]$

Finally the fourth law states that if the meaning of a Yes/No question is not true according to the systems knowledge then No is returned:

C4: $\vdash(\mathrm{Bel}\ t)\ x \overset{\frown}{=} (\mathrm{trans}(\mathrm{In}\ t)) \wedge \sim (\mathrm{Isdel}\ x) \wedge \sim(\vdash(\mathrm{Bel}\ t)\ (\mathrm{M}(\mathrm{Cadr}\ x)))$

$\rightarrow (\mathrm{Bel}\ t+1) = (\mathrm{Bel}\ t) \wedge (\mathrm{Out}\ t+1) = [\text{'No}]$

It will be noted that the simple control laws described here make no use of any inverse meaning laws or generation laws as the outputs by the system are merely pre-stored canned phrases. In general this of course is not adequate and complex laws for generation would be needed. Laws for inverse meaning are essentially the same as the meaning laws but are intended to be used in reverse.

The control system is initialized by two axioms:

$(\mathrm{Bel}\ 0) = \emptyset$

$(\mathrm{Out}\ 0) = [\text{'Hello}]$

3. Examples

We will give two examples of the use of the laws given in section 2. The first example which is given in section 3.1. exemplifies how syntactic and semantic concepts can smoothly interact using the parsing and translation laws, and the meaning laws so as to handle syntactically ambiguous sentences. The second example which is given in section 3.2. exemplifies the dialogue theory, showing how the natural language system based on this theory might interact with its environment.

3.1. Example

In the following we shall show by a detailed example how the laws of parsing, translation and meaning work. We shall not show the entire search space for the whole sentence because this would be much too complex. The detailed parsings for the ambiguity branches are shown in Figures 7, 8 and 9. The "world" is given by the following nonlogical axioms:

A1 $\text{PLACE}x \leftrightarrow \text{GARDEN}x \vee \text{CITY}x$

A2 $\text{PLANT}x \leftrightarrow \text{TREE}x \vee \text{FLOWER}x$

A3 $\text{FRUIT}x \leftrightarrow \text{CONE}x \vee \text{APPLE}x$

A4 $\sim\exists x\ \exists y\ (\text{TREE}x \wedge \text{ROSE}y \wedge \text{HAS}xy)$

$w = A1 \wedge A2 \wedge A3 \wedge A4 \wedge L \wedge S$

From the parsings as far as executed in Figures 7 and 8 we get:

(1) $\text{ⓩ} \wedge (((\text{NG2...z3}) \wedge (\text{LIKELIER}('\sim(M['\sim z3]))('\sim(M['\sim z33])))) \vee$
$((\text{NG3...z33}) \wedge (\text{LIKELIER}('\sim(M['\sim z33]))('\sim(M['\sim z3])))))$

where

$z3=[\text{'TREEy1 '}\wedge\text{ ['INy1 ['they2 ['GARDENy2 '}\wedge\text{ '}\exists\text{ y3 ['CONEy3 '}\wedge\text{ 'HASy2y3]]]]]}$

$z33=[\text{'TREEy1 '}\wedge\text{ ['INy1 ['the y2 ['GARDENy2]]] '}\wedge\text{ '}\exists\text{ y3 ['CONEy3 '}\wedge\text{ 'HASy1y3]]}$

It is easy to see that

$\vdash (w \wedge \sim M['\sim z3]) \rightarrow \exists y2\,\exists y3\ (\text{GARDEN}y2 \wedge \text{CONE}y3 \wedge \text{HAS}y2y3)$ and

$\vdash (w \wedge \sim M['\sim z3]) \rightarrow \exists y1\,\exists y3\ (\text{TREE}y1 \wedge \text{cone}y3 \wedge \text{HAS}y1y3)$ and since

$\vdash (w \rightarrow A2)$ and $\vdash (w \rightarrow A1)$ we get

(2) $\vdash (w \wedge \sim (M['\sim z3])) \leftrightarrow \exists y2\,\exists y3(\text{PLACE}y2 \wedge \text{FRUIT}y3 \wedge \text{HAS}y2y3)$ and

(3) $\vdash (w \wedge \sim (M['\sim z33])) \leftrightarrow \exists y1\,\exists y3(\text{PLANT}y \wedge \text{FRUIT}y3 \wedge \text{HAS}y1y3)$

$\Diamond (w \wedge \sim (M['\sim z3]))$ is derivable as well as $\Diamond (w \wedge \sim(M['\sim z33]))$
The idea of that proof is to use axioms of the form $\Diamond (px \wedge qy)$ for all the nonlogical predicates p,q of the world. The reader may imagine that we cannot present that proof here since we would need such axioms of all combinations of our 11 nonlogical symbols. So we use that w is consistent with $\sim M['\sim z3]$ and with $\sim M['\sim z33]$. If we instantiate the existential expressions of (2) and (3) to P and Q resp. we get $\text{LIKELIER}(\sim M['\sim z33])\ (\sim M['\sim z3])$ by S and L. So, the NG2-branch of (1) evaluates to ¤. Now we are able to finish the parsing and translation derivation and we get:

$z=[\text{zo z1 ['THE y1 ['TREEy1 '}\wedge$
$\text{['INy1['THEy2['GARDENy2 '}\wedge\text{ '}\exists\text{ y3 ['CONEy3 '}\wedge\text{ 'HASy1y3]]]]]]]}$
$\wedge\ \exists x1\,\exists x2(x1=\text{' (is pretty))} \wedge (\text{VBE x1-x2 zo}) \wedge (\text{ADJ x2- [ ]z2})$

Laws VBE, ADJ:

$z=\text{'PRESENT['PRETTY['THEy1['TREEy1 '}\wedge$
$\text{['INy1['THEy2['GARDENy2 '}\wedge$
$\text{'}\exists\text{ y3['CONEy3 '}\wedge\text{ 'HASy1y3]]]]]]}$

$$\underbrace{xo='(\text{the tree in the garden which has cones is pretty})}_{\textcircled{o}} \wedge (\text{SENT}\ xo\text{-}[\ ]\ z)$$

$$\textcircled{o} \wedge \underbrace{z=[(zo(\text{subst}[z\ 2y]\ \text{for}\ *\ \text{in}\ z1)]}_{\textcircled{z}} \wedge \exists x1 \exists x2((\text{NP}\ xo\text{-}x1\ y\ z1) \wedge \underbrace{(\text{VBE}\ x1\text{-}x2\ zo) \wedge (\text{ADJ}\ x2\text{-}[\ ]z2))}_{..}$$

$$\textcircled{o} \wedge \textcircled{z} \wedge \exists x1 \exists x2(y=['\text{THE}y1\ z3] \wedge z1=* \wedge \exists x4(\underbrace{x4='(\text{tree in the garden which has cones is pretty})}_{\textcircled{4}} \wedge$$

$$((\text{NG2}\ x4\text{-}x1\ y1\ z3) \wedge \underbrace{(\forall z33(\neg\exists b(\text{NG3}\ x4\text{-}x1\ b\ z33) \vee (\text{LIKELIER}\sim(M['\sim z3])('\sim M['\sim z33]\ )))))}_{S1} \vee$$

$$\underbrace{((\text{NG3}x4\text{-}x1\ y1\ z3)}_{A2} \wedge (\underbrace{\forall z33(\neg\exists b(\text{NG3}\ x4\text{-}x1\ b\ z33) \vee (\text{LIKELIER}'\sim M['\sim z3])))))..)}_{S2}$$

Figure 7

A1 S1:NG2

$\underbrace{\text{z=[zo z2 'THE y1 z3]}}_{\textcircled{z}}$ ∧∃x1 ∃x2 ∃x4(④ ∧((∃x5((NG x4 -x5 y1 z4)∧ (PP x5-x1 y1 y2 z5)∧z3=[z4 '∧ z5]∧S1))∨(A2 S2))..)

Laws NG,NG1, N:

ⓩ∧∃x1 ∃x2((∃x5(x5='(in the garden which has cones is pretty)∧z4=['TREEy]∧z3=[z4 '∧ z5] ∧
(PP x5-x1 y1 y2 z5)∧S1)) ∨(A2∧S2))..)

Law PP:

ⓩ ∧∃ x1 ∃x2(((∃x6(x6='(the garden which has cones is pretty)∧z3=['TREEy1 '∧ (subst['INy1y2] for * in z6)∧
(NP x6-x1 y2 z6) ∧ S1)) ∨ (A2 ∧ S2))..)

Laws NP, NP3:

ⓩ∧ ∃x1 ∃x2(((∃x7(x7='(garden which has cones is pretty)∧y2=['THEy3 z7] ∧ z6=* ∧ (NG x7-x1 y3 z7)∧
z3=['TREEy1 ∧ (subst['INy1y2] in z6 for *) ∧S1)) ∨ (A2 ∧ S2))..)

Laws NG, NG3, NG, NG1, N:

ⓩ∧∃x1∃x2(((∃x8(x8='(which has cones is pretty) ∧ z7=['GARDENy2 ∧ z8] ∧ (CLAUSE x8-x1 y2 z8) ∧
z3=['TREEy1 '∧ ['INy1['THEy2 z7]]] ∧ S1)) ∨ (A1 ∧ S2))..)

Laws CLAUSE, RELPRON, VT, NP, NP;, NG, NG1, N:

ⓩ∧ ∃x1∃ x2(((x1='(is pretty) ∧ S1 ∧ z3=['TREEy1 '∧ ['INy1['THEy2['GARDENy2]∧'PRESENT[∃ y3 ['CONEy3 ∧ HASy2y3]]]]]) ∨
(A2 S2))..)

Figure 8

A2∧ S2: Law NG3:

Law NG3:

$\underbrace{\text{z =[zo z1 'THEy1z3]}}_{\textcircled{z}}$ ∧∃x1 ∃x2 ∃x4(④ ∧ ((A1 ∧ S1)∨∃ x5((NG4x4-x5 y1 z4) ∧ (CLAUSE x5-x1 y1 z5) ∧ z3=[z4 '∧ z5]∧S2))..)

Laws NG, NG2:

ⓩ∧∃x1∃x2∃x4(④ ∧ ((A1∧ S1) ∨∃ x5 ∃x6((NG x4-x6 y1 z6) ∧ (PP x6-x5 y1 y2 z7)∧ z4=[z6 '∧ z7]
z3= x4 '∧ z5 ∧ S2 ∧ (CLAUSE x5-x1 y1 z5)))..)

Laws NG, NG1, N:

ⓩ∧ ∃x1∃x2(((A1∧ S1)∨ ∃ x5∃ x6(x6='(in the garden which has cones is pretty)∧ z3=['TREEy1 '∧ z7 '∧ z5] ∧
(PP x6-x5 y1 y2 z7) ∧ (CLAUSE x5-x1 y1 z5)∧ S2)))..)

Law PP:

ⓩ∧∃ x1 ∃x2(((A1 ∧ S1)∨ ∃ x5 ∃x7(x7='(in the garden which has cones is pretty) ∧ S2 ∧
z3=['TREEy1 '∧ (subst['INy1y2] for * in z8) ∧ (NP x7-x5 y2 z8) ∧ (CLAUSE x5-x1 y1 z5))))..)

Laws NP, NP3, NG, NG1, N:

ⓩ∧∃x1∃x2(((A1 ∧ S1)∨ ∃ x5(x5='(which has cones is pretty)∧ (CLAUSE x5-x1 y1 z5) ∧
z3=['TREEy1 '∧ ['INy1['THEy2['GARDENy2]]] '∧ z5] ∧S2))..)

Laws CLAUSE, RELPRON, VT, NP, NP1, NG, NG1, N:

ⓩ∧∃x1∃x2((A1 ∧ S1) ∨ (x1='(is pretty) ∧ S2 ∧
z3=['TREEy1 '∧ 'INy1['THEy2['GARDENy2]]] ∧ 'PRESENT ∃ y3['CONEy3∧ HASy1y3])..)

Figure 9

3.2. An Example Dialogue

We give below an example dialogue illustrating the four laws of social behaviour. It should be noted that this example uses an extended version of the grammar and lexicon that was described in Section 2.2.

$$0 \left\{ \begin{array}{l} O = \text{'Hello} \\ B = \emptyset \\ I = \text{'(All Men are Mortal)} \end{array} \right.$$

At time 0 the beliefs of the system is simply $\emptyset$. The system begins by saying "Hello" and inputs the first input which in this case is "All Men are Mortal". Since the meaning of the translation of this declarative sentence namely: $\forall x(\text{Men } x \rightarrow \text{Mortal } x)$ is possible with respect to the current beliefs, this proposition is now assumed by the system using the social law C1

$$1 \left\{ \begin{array}{ll} O & \text{'(I believe you)} \\ B & \forall x(\text{Man } x) \rightarrow (\text{Mortal } x)) \\ I & \text{'(John is a Man)} \end{array} \right.$$

Again using the social law C1 we get:

$$2 \left\{ \begin{array}{ll} O & \text{'(I believe you)} \\ B & (\text{Man John}) \wedge \forall x((\text{Man } x) \rightarrow (\text{Mortal } x)) \\ I & \text{'(Is John Mortal ?)} \end{array} \right.$$

The input at time 2 is an interrogative sentence whose translation is

[? ['Mortal 'John]]

Since (Mortal John) is deducible from the beliefs at time 2 using social law C3 we get:

$$3 \left\{ \begin{array}{ll} O & \text{Yes} \\ B & (\text{Man John}) \wedge \forall x((\text{Man } x) \rightarrow (\text{Mortal } x)) \\ I & \text{'(Is John Green?)} \end{array} \right.$$

The input at time 3 is an interrogative sentence, but since (Green John) is not true according to the systems it answers NO using social law C4.

$$4 \left\{ \begin{array}{ll} O & \text{No} \\ B & (\text{Man John}) \wedge \forall x((\text{Man } x) \rightarrow (\text{Mortal } x)) \\ I & \text{'(No Man is Mortal)} \end{array} \right.$$

Since the imput at time 4 is a declarative sentence which contradicts the systems beliefs at that time, the system replies using social axiom C2

$$5 \left\{ \begin{array}{ll} O & \text{'(I disbelieve you)} \\ B & (\text{Man John}) \wedge \forall x((\text{Man } x) \rightarrow (\text{Mortal } x)) \end{array} \right.$$

4. Theoretical Claims

We now compare various features of our theory of natural language understanding to related work in a number of subject areas.

We shall try to summarise our work by listing a number of theoretical claims about language understanding.

4.1. Relationship to Formal Grammar

A formal grammar is defined as a finite set σ_N of nonterminal symbols, a set σ_T of terminal symbols, a start symbol S such that $S \epsilon \sigma_N$, and a set of production rules which have the form p => q, where p and q are strings over $\sigma_N \cup \sigma_T$ and p contains at least one element of σ_N, that is p = X N Y for $X,Y \epsilon (\sigma_N \cup \sigma_T)^*$ and $N \epsilon \sigma_N$. It is worth remembering that transformational grammars are equivalent to formal grammars.

As we stated in Section 2.2. our parsing laws have the form:

$$(A\ xo\text{-}xn) \leftrightarrow \exists\ x1 \ldots \exists xn\text{-}1((B1\ xo\text{-}x1) \wedge \ldots \wedge (Bn\ xn\text{-}1\text{-}xn))$$

This law roughly corresponds to the production rule:

A => B1 ... Bn

However it should be noted that this law does not exactly correspond to the production rule, and that the meaning of the law is quite different from the meaning of the production rule. Production rules form a device or algorithm to generate sentences and hence the meaning of such a rule is something like: Whenever A occurs within a sentence then replacing A by B1 ... Bn generates another sentence. On the other hand, the meaning of our parsing laws is like: a string xo-xn is an A iff xo-x1 is a B1 and xn-1-xn is a B.

This difference of meaning implies that a formal grammar cannot always be rewritten as a set of "corresponding" parsing laws as described above, but sometimes must be changed in order to describe the same set of sentences which the formal grammar generates. An example of such a case is given below:

(a) Formal Grammar:

R1: S => NP + VP + PP
R2: VP => VP + PP
R3: VP => V

(b) "Corresponding" Parsing Laws

L1: (S xo-x3) ↔ ((NP xo-x1) ∧ (VP x1-x2) ∧ (PP x2-x3))
L2: (VP xo-x2) ↔ ((VP xo-x1) ∧ (PP x1-x2))
L3: (VP xo-x1) ↔ (V xo-x1)

The sentences which can be generated by this formal grammar are:

NP+V+PP, NP+V+PP+PP, NP+V+PP+PP+PP ...

However the parsing laws describe all these sentences plus the sentence: NP+V which is obtained by applying the law L2 backwards to L1, and applying L3 in the normal manner.

We can of course rewrite a formal grammar as parsing laws. For example the above formal grammar is correctly described by the following parsing laws:

L1': (S xo-x2) ↔ ((NP xo-x1) ∧ (VP x1-x2))
L2': (VP xo-x2) ↔ (((VP xo-x1) ∧ (PP x1-x2)) ∨ ((V xo-x1) ∧ (PP x1-x2)))

This difference between parsing laws and generation rules can explain some apparent problems in formal grammars such as the deadlock problem. Consider for example the following grammar for arithmetic expressions:

P1: E => T
P2: E => E+T
P3: T => F
P4: T => T*F
P5: F => (E)
P6: F => V

where the syntactic categories are:

E	Expressions
T	Term
F	Factor
V	Variable

This grammar has what is called a deadlock: namely a sentence which can be generated by it can be analysed, by applying the rules in reverse only as in bottoms up parsing xo produce a sentence which cannot be generated by this grammar:

E	: P2
E+T	: P4
E+T+F	:PP1
T+T*F	: P3
T+F*F	: P3

F+F*F : P6
V+F*F : P6
V+V*F : P6
V+V*V

Thus V+V*V can be generated by these production rules. Analysing V+V*V we obtain:

V+V*V
V+V*F : P6
V+F*F : P6
F+F*F : P6
T+F*F : P3
T+T*F : P3
E+T*F : P1
E+T*T : P3
E*T : P2

E*T however cannot be generated from these production rules.

We point out that there is nothing really strange about the existence of such deadlocks, but rather that this is merely the consequence of writing grammars which don't really say what one thought they meant. This point is easily seen once we rewrite the production rules as the roughly "corresponding" parsing laws. In this case the parsing law for P2 states:

$$(E\ xo\text{-}x2) \leftrightarrow ((E\ xo\text{-}x1) \wedge (T\ x1\text{-}x2))$$

that something is an expression iff it consists of an expression followed by a term. But this parsing law is clearly false because an expression followed by a term is not always an expression, but is an expression only if subexpressions is not followed by *. This merely reflects the fact that times * binds more closely than plus +.

A correct version of the parsing law corresponding to P2 would be:

$$((E\ xo\text{-}x2) \wedge (x2{=}[\ ]\ \vee\ (Car\ x2) \neq *))$$
$$\leftrightarrow (E\ xo\text{-}x1) \wedge (Car\ x1) = {}'{+} \wedge (T (Cdr\ x1) - x2)$$

Since we require a grammar,(i e parsing laws) to specify what a sentence is and not merely how it might be generated we are constrained to write grammars which are not wrong.

Chomsky [8,9] claims that a grammar for a natural language should be a generating device, or rather a formal grammar, for producing the sentences of that language. Our purpose however is to write laws which tell us what a sentence is rather than merely to design an abstract machine which generates sentences. Although in a theoretical sense it could be argued that such a generating device does define what are the sentences of a language, we point out that it does so only by virtue of the overall interaction of all the production rules in the system. In a larger grammar, particularly for natural language these interactions will be so complex that it will in a practical sense be impossible for anyone to understand, correct or modify the grammar. Parsing laws by contrast have a clear and directly understood meaning in which each law can be understood purely in terms of the syntactic categories appearing within it.

A further problem with the use of grammars based on production rather than on parsing laws is that it is practically impossible to tell what aspects of the production rules make substantial claims about a natural language, and what are mere artifacts forced by the use of production rules. Indeed in a large grammar this situation is so bad that for example in the UCLA English Syntax Project grammar [10] the authors felt constrained to write (P-37):

> "In the development of the analysis we shall take pains to distinguish between complexity in the formulation that seems to have a substantive basis, and complexity that is attributable rather to some artifact in the general theory or in this particular implementation of it".

In other words the use of production rules forces the authors formal system to make more claims than they actually wished to make, and they are therefore going to try to tell us in English what they really did not wish to claim!

We can summarize the relationship of our system to formal grammars by the following claim:

Claim 1: The laws of parsing bear a logical relationship between syntactic categories. They do not bear a generative production rule relationship.

4.2. Relationship to transformational grammar

Transformational grammar has three basic faults when applied to natural language understanding. Since these three faults do not occur in an integrated theory we shall simply list these faults in the form of theoretical claims made by our theory. We will not however argue why these claims are correct, as we believe this will be apparent to anyone with the least familiarity with this subject.

Claim 2: Recognition grammars are different from Generation grammars. Transformational rules such as those used in Transformational grammars are not really acceptable in a recognition grammar.

Claim 3: Parsing laws must include translation into some general meaning representation (such as a logic) in which inference may be able to be performed.

Claim 4: The laws of parsing must be able to refer to the meanings of the expressions being parsed, so that those meanings may interact with general world knowledge during the parsing process. That is, the theory must be capable of meta theoretic reasoning.

4.3. Relationship to Artificial Intelligence

Contemporary research in Artificial Intelligence on implementing Natural Language Understanding systems such as [11, 12, 13, 14] shares with our theory, at least to some extent claims 2, 3 and 4. However, most of this research differs from our theory in two important ways:

Claim 5: The Meaning Representation must be rich enough to allow every inference that the system might need to make. That is, we claim that the system must be a logic, and in view of claim 4, this logic must be capable of meta theoretic reasoning.

Claim 6: (Methodological). The theory must be capable of being easily modified and communicated to other researchers. This implies that the basic theory must be a logic, for logic due for example to the localness of its variables is easily modifiable and communicatable. Other languages such as programming languages like Algol are comparatively speaking difficult to modify or communicate.

4.4. Relationship to the Philosophy of Language

Our theory makes two substantial philosophical claims:

Claim 7: Meaning, not truth, is the fundamental meta theoretic concept.

It should be noted that our notion of meaning is an entirely new concept in the Philosophy of Language. In particular our concept of meaning cannot be interpreted as the concept of empirical truth $\mathcal{E}$ which was defined by A Tarski [15].

We can see this by simply showing that at least one of our meaning laws is false when meaning M is interpreted as being empirical truth E as follows. Choosing the law $M\vdash$:

$$(M['\vdash\ S]\ A) = \vdash(m\ S\ A)$$

we let S be the sentence "The morning star is the evening star" and replace m by E.

$$(E['\vdash\ S]\ A) = \vdash(E\ S\ A)$$

Since the sentence S is empirically true but not logically true we deduce that:

$$(E\ \ '\square\ A) = \vdash\blacksquare.$$

by replacing the false sentence $['\vdash\ S]$ by an equivalent false sentence $'\square$, and by replacing (E S A) by true: $\blacksquare$. Then, since $\square$ is not empirically true and since $\blacksquare$ is logically true we deduce that:

$$\square = \blacksquare$$

which is clearly a contradiction.

In conclusion we can say that our concept of meaning is an entirely different concept from concepts such as empirical truth , (i.e. truth in our world), and satisfaction (i.e. truth in an arbitrary world) which have been studied by logicians and philosophers of language.

Claim 8: The Modal Logic axiomitized in Section 2.1 captures the modal notion of logical truth and hence is the correct Modal Logic. All other intensional concepts can be easily defined in terms of this one intentional concept and extensional concepts. Set Theoretic Semantics (i.e. Model Theory) is not needed.

5. Conclusion

We think that our natural language understanding system is an entirely new approach to the natural language understanding problem because it is an integrated theory. Most natural language understanding systems use several languages for their description: English, Programmar, Planner, Lisp, semantic networks. Approaches like [11], or [12] are wonderful programs but do not contribute much to our understanding of the processes underlying natural language understanding. Programs have built in more or less ad hoc devices which simulate special situations. They do not apply or know general laws underlying the natural language understanding processes. They do not allow to derive general statements about natural language understanding. Moreover we think that a theory which is based on logic has immense advantages of expressional power and deductive capacity over theories using needleworked knowledge structures [13] the properties of which have not been investigated.

Problems of preference semantics [14] are very frequent in natural language understanding. Our theory suggests a very elegant and adequate way to solve them. Since our theory is formulated in the same language as the meaning of the natural language ambiguous sentences can be disambiguated during the syntactic analysis in a very natural way.

References

1. Brown, F. "A Sequent Calculus for Modal Quantificational Logic", (1978). To appear in *Proceedings of 3rd A.I.S.B./G.I. Conference,* Hamburg, (1978).

2. Brown, F. "A Theory of Meaning", (November 1976). *D.A.I. Working Paper No. 17,* Department of Artificial Intelligence, University of Edinburgh.

3. Schwind, Camilla B. "A State Logic for the Representation of Natural Lanugage Based Intelligent Systems", (1977). *Proceedings of 5th International Joint Conference on Artificial Intelligence.* M.I.T.

4. Schwind, Camilla B. "Representing Actions by a Modal Tense Logic". To appear in *Proceedings of 3rd A.I.S.B./G.I. Conference,* Hamburg, (1978).

5. Brown, F.M. "The Theory of Meaning", (June 1977). *D.A.I. Research Report No. 35,* Department of Artificial Intelligence, University of Edinburgh.

6. Schwind, Camilla B. *"Ein Formalismus zur Beschrerbung der Syntax und Bedentung von Frago- Antwart - Systemen".* (1977). Thesis, Technischen Universitat, Munchen.

7. Schwind, Camilla B. "A Formalism for the Description of Question Answering Systems. To be published in *Natural Language Communication with the Computer,* Springer Verlag.

8. Chomsky, Noam. *"Syntactic Structures"* (1957). (Janua Linguarum, 4). Mouton & Co., The Hague.

9. Chomsky, Noam. *"Aspects of the Theory of Syntax".* (1965). M.I.T. Press, Cambridge, Mass.

10. Stockwell, Schachter, Partee. *"The Major Syntactic Structures of English"* (1973). Holt, Rinehart & Winston.

11. Winograd, Terry. *"Understanding Natural Language"* (1972). Academic Press.

12. Charniak, E. "Ms Malaprop, A Language Comprehension Program" (1977). *Proceedings of 5th International Joint Conference on Artificial Intelligence,* M.I.T.

13. Schank, Roger C. "The primitive acts of conceptual dependency". *Theoretical Issues in Natural Language Processing,* (1975), Cambridge, Mass.

14. Wilks, Yorick. "Knowledge Structures and Language Boundaries". (1977). *Proceedings of 5th International Joint Conference on Artificial Intelligence,* M.I.T.

15. Tarski, A. "The Concept of Truth in Formalized Languages" (1931). *Logic, Semantics, Metamathematics.* Trans. by J.H. Woodger, Oxford, Clarendon Press, 1956.