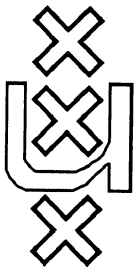
Institute for Language, Logic and Information

A BLISSYMBOLICS TRANSLATION PROGRAM

Anneke Kleppe

ITLI Prepublication Series
for Logic, Semantics and Philosophy of Language LP-88-10



University of Amsterdam

Abstract

This report describes the Blissymbolics language and a translation program that translates simple Dutch sentences into Blissymbolics. Blissymbolics is a very simple language based on graphical symbols. It is currently being used by handicapted people. Its origin, its structure and the way it is used will be discussed in part one. The second part of this report is a description of the development of a translation program. The translation method used and its consequences will be described. The two parts can be read almost independently. This report is a comprised version of 'A Blissymbolics Translation Program' which is my thesis to obtain the Masters degree. In part two some references are made to appendices. They are part of the above mentioned thesis. The appendices are not needed for reading this report, but anyone who is really interested can obtain a copy of the original thesis from the Universiteit van Amsterdam.

Contents:

BLISSYMBOLICS - THE LANGUAGE

1. Introduction

Blissymbolics is an artificial language which has found a useful application in communicating with handicapted people. Blissymbolics was developped during and after the second world war by Charles K. Bliss, who was born in Austria but fled from the Nazi-regime. After living in and traveling through England and China Charles Bliss took residence in Australia. There he began to develop a 'universal language', just as the makers of Volapuk and Esperanto had wanted to do. Like these, Bliss had the ideal that his universal language would promote world peace. And like these, Blissymbolics has much resemblance to european languages, especially English.

The written forms of Volapuk and Esperanto however are based on phonetics, i.e. a word symbol represents the sounds that are produced when it is spoken. While most languages use some kind of letters to represent a word, in Blissymbolics a word symbol is a drawing. Blissymbolics is based on the semantics of the symbols and uses graphics to represent their meaning, hence its original name: Semantography.

From the moment that Bliss published his first work in 1949 until 1971 Blissymbolics was not used and hardly studied. In the autemn of 1971 the Ontario Crippled Children's Center initiated a program with the object to develop "a system of communication which would serve as a complement to or substitute to speech for the pre-reading child." (McDonald, p.16) In this program Blissymbolics was used very succesfully. In 1975 the Blissymbolics Communication Institute (BCI) was founded in Toronto, which guards the standards for Blissymbols. And for this purpose it holds the worldwide copyright on the symbols. Nowadays the use of Blissymbolics by handicapted people has spread over the world. It is used in Canada, England, Sweden, the Netherlands and a number of other country's.

Blissymbolics users are mostly physically handicapted people, mainly spastics, but also people who are handicapted differently, like aphatics, use it. (see section 6 for an elaborate view on the use of Blissymbolics) The succes of the use of Blissymbolics by these groups of people has two reasons. In the first place physically handicapted people have often trouble with the spoken language because it is difficult for them to articulate. But not only it is hard for them to speak, this also makes the understanding of spoken word more difficult. The meaning based Blissymbols are therefore easier to remember and more appealing then phonetically based symbols like written words. In the second place Blissymbolics can be used very freely and in an associative way new symbols with new meanings can be made by combining the existing ones. Thus with a small set of basic symbols one can obtain a large power of expression. The meaning of these new symbols is said to be derived from the meanings of the composing symbols. In the opinion of the author this is only partly true. At best we can say that in general this rule applies but a lot of human fantasy is needed. (see section 5)

This part of the paper will describe the Blissymbolics language. The aim was to give a description which had to be as strict and complete as possible so that it could serve as basis for the computer application which will be described in part two of this paper. The following subjects will be discussed. In section 2 the configuration of the Blissymbols, or what Blissymbols look like, will be explained. In section 3 an alphanumerical representation of the graphical symbols wil be introduced. In section 4 the syntax rules or grammar of Blissymbolics will be illustrated. In section 5 we will discuss the question whether or not Blissymbolics answers to Frege's compositionality principle. The paper will be concluded with some reflections on the use of Blissymbolics by handicapted people and the place of the computer therein.

2. Construction of the Symbols

Every Blissymbol has a simple graphical form. Circles, arrows, rectangles, straight lines and the like are combined to make word-symbols. A difference in the size, positioning, spacing, direction and configuration of simple graphical symbol parts distinguishes between one symbol or another. Additional letters, numbers and positional referents can also be decisive. But the graphical forms of Blissymbols are developped according to the meaning the symbols are ment to represent. Thus three kinds of basic Blissymbols have arised (Bliss):

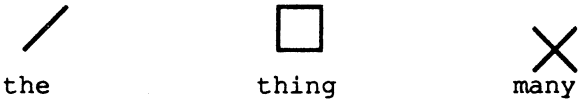
1. pictographs, symbols which resemble what they represent,
2. ideographs, symbols which have a graphic association with the concept they represent, and
3. arbitrary symbols, symbols that have no relation between the concept they represent and their form.

Example 1:

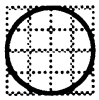
Pictographs:

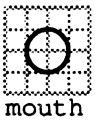
Ideographs:

Arbitrary symbols:

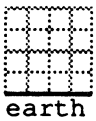


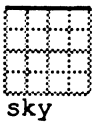
Difference in size:


sun

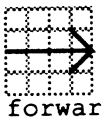

mouth

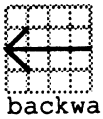
Difference in position:


earth

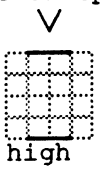

sky

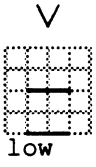
Difference in direction:


forward

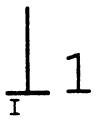

backward

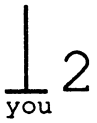
Difference in spacing:


high


low

Numbers or letters are decisive in:


I


you


country


Germany


man


Frits

(Notice the fact that the use of letters makes Blissymbolics language dependent.)

The number of symbols Bliss initially made is not very large (about 100) and they were not always suited for the use by handicapted children. But Bliss did give several ways to make new symbols from the existing ones. First, symbols can be grouped together. Second, indicators can be placed above symbols. And third, stategies can be used which place additional symbolic material somewhere in the symbols. In the following each of these generative methods will be illustrated.

2.1 Grouping:

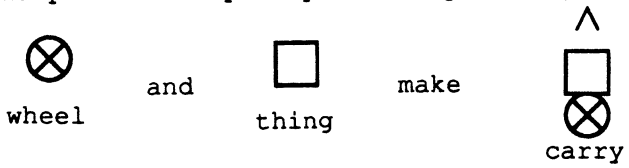
There are several ways to make groupings of two or more symbols. The symbols which have been grouped together by Charles Bliss or the BCI are called *compounds*. When the combining symbols are placed in or onto each other they are called *superimposed,* when they are placed next to each other they are called *sequenced.*

Grouping can also be used by persons communicating in Blissymbolics. The symbols which are combined by a user to arrive at a new meaning must always be sequenced. Such newly made symbols must be surrounded by

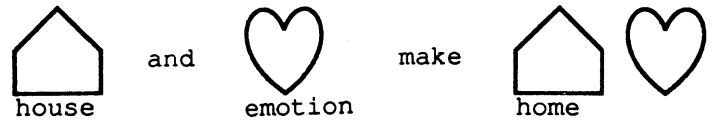
combine indicators, and are not called compound but *combined* symbols. As it is very clearly put in (McDonald, p. 31): "Placing the combine indicator before AND after the sequence of symbol units not only delimits the symbols that are being grouped but also indicates that the combination is not to be regarded as an approved Blissymbol but rather as a symbol created by the user for a specific communicative situation."

Example 2:

An example of a superimposed compound symbol:

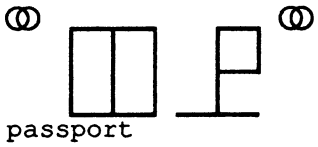


An example of a sequenced compound symbol:



This symbol is the combine indicator: 

This is the way it should be used:



When drawn alone the symbols mean:

2.2 Indicators:

All Blissymbols can be classified semantically into six groups (only the first four are given by Bliss, the other two are augmented by the author):

Class	example
1. ideas	emotion
2. things	a hart
3. actions	to feel
4. evaluations	beautiful
5. persons	you, me, Mr. Bliss
6. relations	which, yours

The first group, the ideas, is the most important. From a symbol which expresses an idea, symbols of almost all other categories can be made by placing an indicator above the symbol. For instance, the verb '(to) feel' can be made from the word 'emotion' by placing an action indicator above it.

In the dictionary (Hehner) nine singular indicators occur. Studies of the author show that they work in the following manner:

indicator	semantics	syntactics
thing	idea -> thing	noun -> noun
plural	thing -> thing	noun -> noun
	person -> person	pronoun -> pronoun
description	idea -> evaluation	noun -> adjective
action	idea -> action	noun -> verb
past action	idea -> action	noun -> verb
future action	idea -> action	noun -> verb
passive	idea -> action	noun -> verb
active	idea -> action	noun -> verb
conditional	action -> action	verb -> verb

The missing categories in the above list are persons and relations. The way to make a symbol which indicates a person from a symbol of the category of ideas is grouping it with the symbol for 'person'. For instance, a nurse is a grouping of the symbols for person, protection and medical, a policeman is a grouping of the symbols for person and protection, and a student is a person plus school. To make a symbol from the class of relations one must use a strategy. For examples of this see the next section.

All actions indicators point to the fact that the symbol is to be taken as an action, something which takes place in a certain time. The passive indicator produces a verb in the passive tense. The active indicator is its counterpart and stresses the fact that the subject of the sentence is being active, further it has the same meaning as the action indicator. The plural indicator indicates that the thing or person represented by the symbol is present in a larger number. Syntactically more important is the description indicator which indicates that the word is an adverb or adjective. The one indicator which is semantically more important is the thing indicator. The thing indicator indicates that the symbol to which it is added must be seen as a thing, something real that you can touch. Furthermore there is the conditional indicator. This indicator functions as an auxiliary verb which indicates uncertainty about whether or not the action has or will take place.

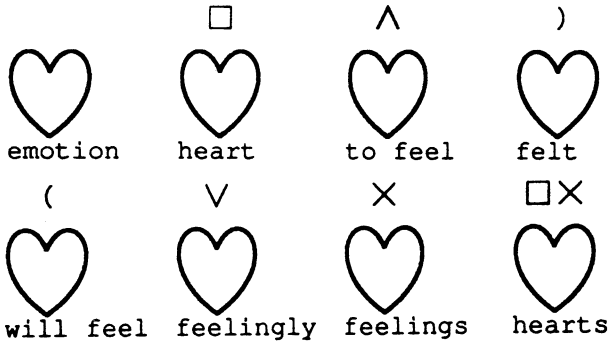
All these indicators can be combined: the passive indicator plus the past action indicator gives a passive verb in past tense. There are no limits of the number of indicators but the request to keep it simple. So if you can understand a certain combination of indicators you can use it.

A special combination is that of the description indicator with a dot (a positional referent, see section 2.3). In this case the time of

evaluation of the symbol is important. Is the dot in front of the description indicator then the evaluation must take place before the action or fact which is represented by the symbol. Is the dot placed after the description indicator then the evaluation must take place after the action or fact which is represented by the symbol, as in the examples below. This last way of modifying an indicator is already some kind of strategy, but a special one that can also be applied to indicators.

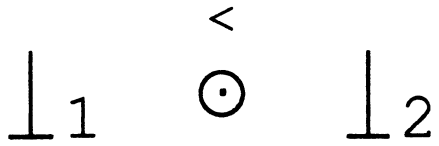
Example 3:

Examples of the use of indicators:

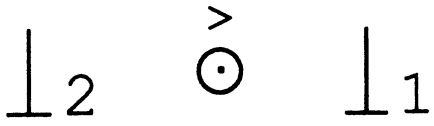


The influence of the passive, active and conditional indicator are most clear when they are used in a sentence.

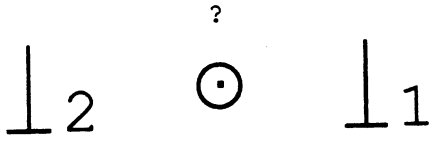
"I am being seen by you":



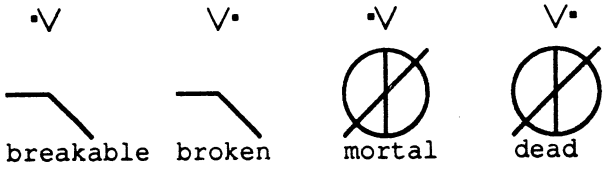
"You see me":



"You may see me.":



The evaluation indicator combined with a dot:



2.3 Strategies:

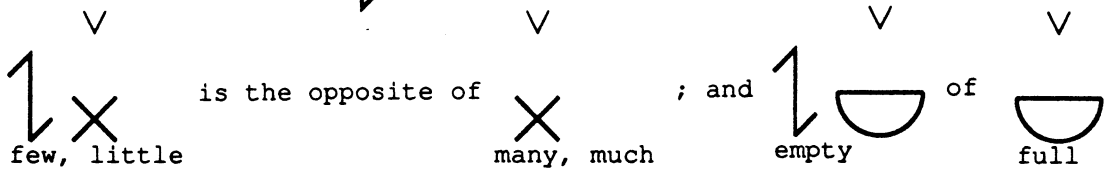
Strategies are ways to change the meaning of a symbol by placing additional material in, before or after the graphical form of the symbol. Strategies have their influence on the semantic level. There is, for instance, the strategy *'opposite meaning'* which makes the word 'dangerous' out of the word 'safe'. But there is also the strategy *'symbol part of'* which actually works an a morphological level, by indicating that only a part of the old symbol is used to represent the meaning of the new symbol. The symbol for 'egg' is made out of the symbol for 'seed' followed by the symbol for 'life'; the new word 'symbol part of egg' means either 'seed' or 'life'. How a choice between these meanings is made is not regulated.

The strategies can be divided into the ones in which the additional material in itself can be viewed as a symbol and the ones in which this can not be done. When the additional material can be viewed as a symbol this symbol cam occur independently in a sentence. All kind of word forming operations, like the adding of an indicator, can be applied to them. Most strategies occur in (McDonald, p. 49 and further) but no division is being made there. All strategies will be illustrated in the example below.

Example 4:

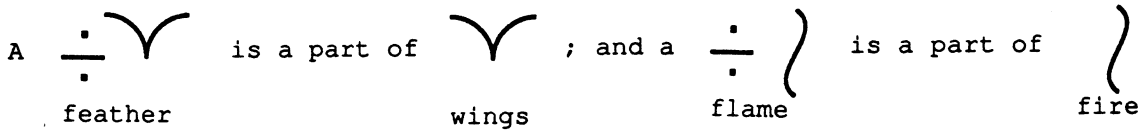
A. Strategies which add a symbol:

1. opposite meaning: ; the symbol is changed to mean the opposite:



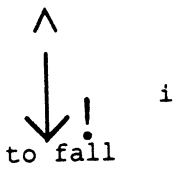
The 'opposite meaning' symbol can also be used independently in a sentence like: "I mean just the opposite (of what you just said)."

2. part (of):  ; only a part of what the symbol represent is meant:

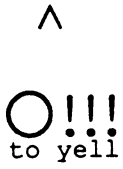


The 'part of' symbol can also be used in a sentence like: "Please, give me a small part (of the cake)."

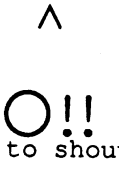
3. intensity: ! ; the meaning of the symbol is intensified:



is more intens than



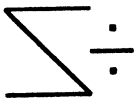
is stronger than

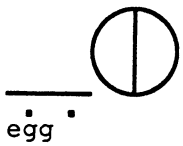


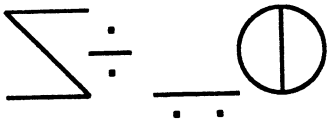
, which is stronger than

does not exist.

The 'intensity' symbol can also be used in a command statement. (see section 4)

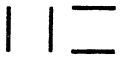
4. symbol part:  ; only one symbol part of the symbol is to be taken into account:

The symbol part of 

:  means either

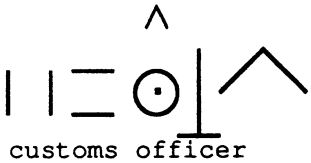
or

The 'symbol part' symbol can also be used in a sentence like: "This symbol consists of three symbol parts." (This strategy is probably used when only a small number of symbols are available, for instance on a Blissymbols chart, see section 6.)

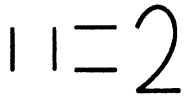
5. similar to:  ; the meaning of the resulting symbol is very much alike the meaning of the old symbol:

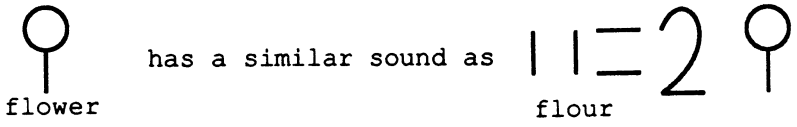
A 

is similar in appearance (

) to a 

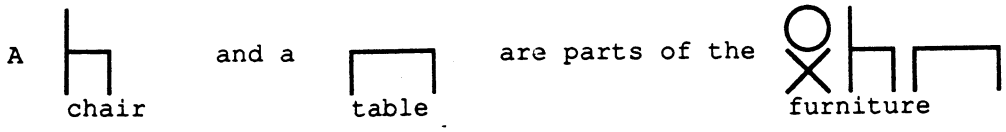
This symbol can also be used in a sentence like: "They look very similar." (of twins)

6. similar sound:  ; the English word which represents the meaning of the old symbol sounds like the word which represent the new symbol:



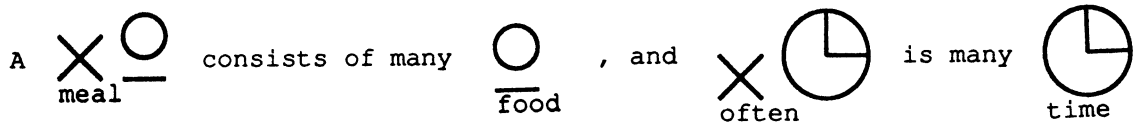
The 'similar sound symbol can also be used in a sentence like: "This bell has a similar sound." (Notice the fact that through this strategy Blissymbolics becomes dependent on phonetics.)

7. classifier:  ; the meaning of the new symbol is the class in which all symbols following the classifier are contained:

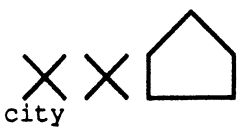


This symbol can also be used in a sentence like: "The symbol parts are used to classify the symbols."

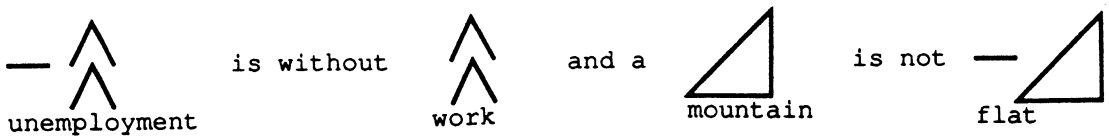
8. many, much:  ; the meaning of the new symbol is the group that is formed by many or much of the old symbol.



Can also be used in a sentence like: "I like you very much." The symbol for city is irregular. It consists of twice the strategy for 'many, much' and the symbol for house. But the symbol which is build up from the house symbol and one occurence of this strategy does not exist.



9. minus: — ; the meaning of the new symbol is 'not' the old symbol:



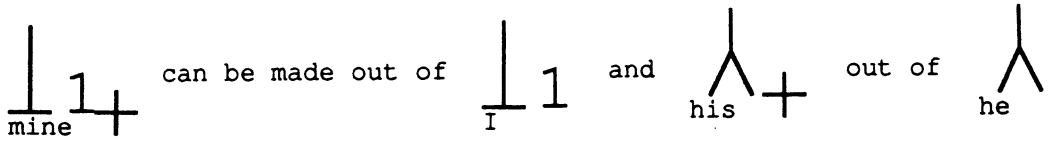
The 'minus' symbol can also be used in a sentence like: "Ten minus two is eigth."

10. metaphor:  ; the meaning of the old symbol(s) is not to be taken litterally:

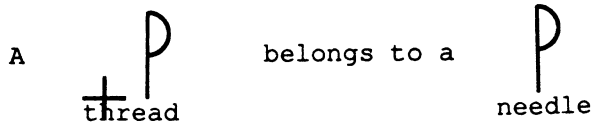
This symbol can also be used to change the meaning of a whole sentence or a sentence part, for instance in proverbial phrases.

B. Strategies which do not add a symbol, only some additional symbol parts:

1. possessive:  ; indicates the possessive form of a noun:

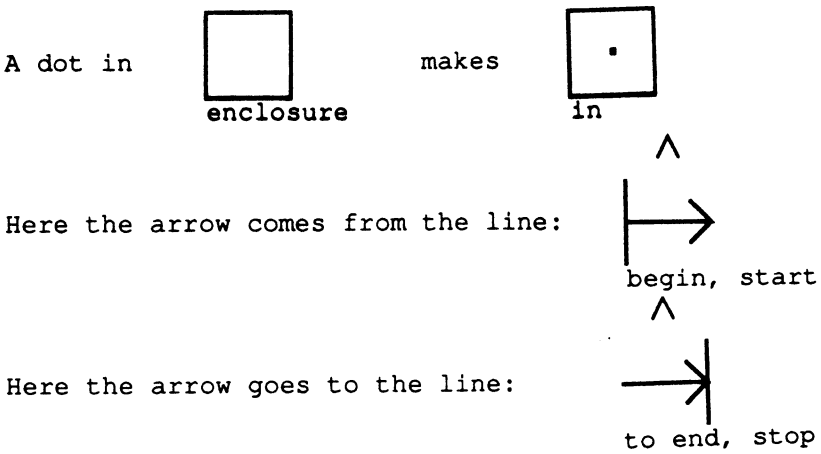


2. belongs to, belongs with:  ; the meaning of the new symbol is something which belongs to the old symbol:

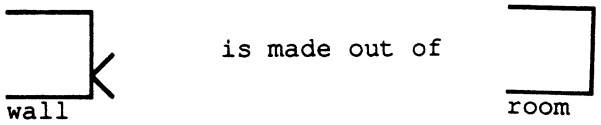


(The additional symbol part is the same as being used in the possessive strategy but here it is placed in front of the old symbol instead of behind it.)

3. the positional referent: • or → ; the place of the dot or arrow in the symbol indicates in which sence the old symbol must be understood:



4. the pointer: > ; the detail to which the pointer attracts the attention is the new meaning of the symbol:



So there are three methods of making new symbols from old ones. When grouping is used combine indicators have to be placed around the new symbol. When indicators and/or strategies are used combine indicators are not neccesary. The only restrictions on the repetitive use of these generative methods are of a semantic nature. What you can understand, you can use. Because Blissymbolics users are limited by their handicaps these methods have not yet been used to their full extent.

All these ways to make new symbols with new meanings were used by Charles Bliss himself and by the BCI. Despite these efforts the dictionary which was published in 1980 comprised only about 1400 words (Hehner). While the number of word-symbols is still growing, under careful supervision of the BCI, it is clear that not everything can be 'said' with Blissymbolics. To communicate in Blissymbolics human fantasy and associative strength are very much needed.

3. Graphical Construction and Representation of the Symbols

Each Blissymbol is composed of a small number of simple graphical forms, which are called the *symbol elements*. Symbol elements are for instance a

circle, a square, a heart, a dot. At the BCI a method is established to sort the symbols according to their symbol elements for the making of a dictionary. Every symbol element is associated with a letter from the alphabet with possibly some augmentations.

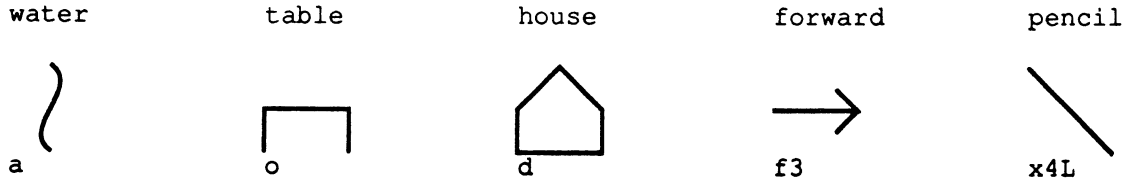
The same symbol element can occur in many different forms. Its size, vertical position and orientation can vary. The starting point for measuring these variations are three lines: the earth or ground line, the mid line and the sky line. With these lines one can indicate the place of a symbol and the size (from ground to sky or from mid to sky).

To indicate variations in size the letter associated with the symbol element, which are all in small case, is augmented by L (for large) or S (for small). Variations in vertical position are indicated by B (for above the symbol space), H (high), M (mid), G (ground) and U (under the symbol space). Variations in orientation are indicated by the numbers of the clock. The direction north is indicated by zero and can be left out, east is indicated by three, south by six and west by nine.

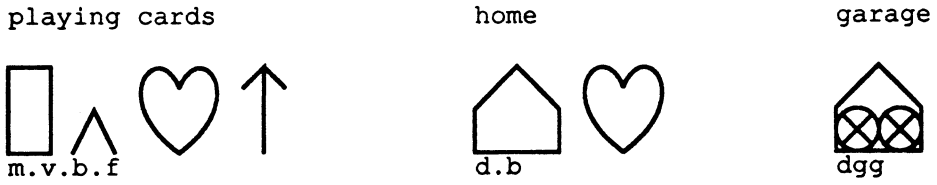
If you can draw an imaginary line between elements in a symbol then that symbol is made up of more than one unit. All symbol elements which form a unit are enumerated in alphabetical order and a dot marks the border between the units. Thus an alpha-numerical representation of the graphical form is made.

Example 5:

Some symbol elements:



Some compound symbols with their codings:



Because this representation was made in order to form a dictionary of Blissymbols (Hehner), the augmentations are used only when two symbols can not be distinguished otherwise. Especially the following codings are left out whenever possible. First, if symbols differ only in their indicators, further letters are added to their codings to distinguish them:

T: thing indicator
X: plural indicator
A: action indicator
V: evaluation indicator
DV: evaluation indicator + dot
VD: dot + evaluation indicator

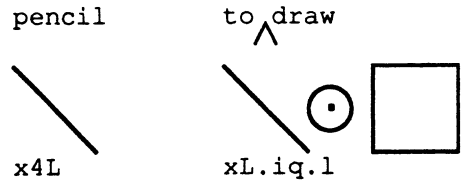
The other actions indicators are never used in the dictionary and therefore they can not be coded this way. That they are never used in the dictionary is understandable, they indicate the tense of a verb.

The second coding is used when the only difference between two symbols is that in the first symbol two symbol elements touch and in the second the same two symbol elements intersect. Then the intersecting symbol elements are augmented with N. The third coding is used when two symbols are identical but in the first the symbol element is to the left of the center of the symbol and in the second the same symbol element is to the right of the center. Then F will denote left and R will denote right.

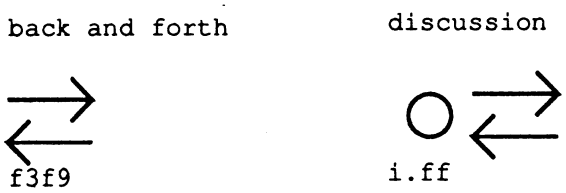
Here are some examples to stress that these codings are only used when two symbols can not be otherwise distinguished (Hehner, p. 205):

Example 6:

'pencil' is coded: x4L, but '(to) draw' is not coded: x4L.iq.l but xL.iq.l



'back and forth' is coded: f3f9, but 'discussion, conversation' is not coded: i.f3f9, but i.ff



The prime directive in making this coding was that it should be simple. "Although a 'full' coding of a symbol could use all the appropriate additional letters and numbers, an attempt has been made to keep the codings as short and simple as possible so that a larger number of people can learn the coding system." (Hehner, p. 205) But by keeping it simple two problems were overlooked.

The first problem in the use of this coding is that the number of Blissymbols is growing rather rapidly. At the time the dictionary was

published the coding used for the symbols existing at that moment was effective but what will happen with the codings of all the new symbols. These codings will be larger and more complicated than the old ones because difference with the old symbols will have to be indicated by one or more augmentations. Or, even worse, the codings of the old symbols will have to be changed.

A further problem is that the coding works only one way. If you have the symbol you can produce the coding if you know the coding of the symbol elements. But if you have the coding you can not produce the symbol even if all the additional coding is used. You would have to look up how the symbol looks like in the dictionary.

These problems are both not very relevent for daily use. But when the coding is used for computer applications they introduce large problems. Smaller problems will occur when the dictionary is used. Still it is more than a missed opportunity that these problems were not taken into account when the effort was being made to produce such a coding.

4. Syntax

When discussing the syntax of Blisssymbolics one must make a distinction between the grammar rules established by Charles Bliss and those used in various country's. Bliss wanted to simplify grammar and so he gave very few rules with the recommendation to use simple rather than cumbersome grammatical constructions. To illustrate Bliss' point of view: "The grammar of semantography is not intended for learning a foreign tongue. It is solely constructed as a simple device for the people, who can speak their mother tongue (no mattter how 'incorrect'), but who need some rules to construct internationally valid sentences in semantography." (Bliss, p.315)

Because of this Blissymbolics has different syntax rules in, for instance, England than in the Netherlands. The syntax rules of the country in which it is used fill in the parts that Charles Bliss left out. Verweij who has studied the practical use of Blissymbolics in the Netherlands even puts it likes this: "Blissymbolics does not have its own syntaxis, just a few word order rules. These can be useful for international correspondence but when the symbol system is being used as replacement for oral use of language no attention to them should be made." (Verweij, p.152) And: "Blissymbolics functions for those who know English as a logography, that is, as a manner of writing." (Verweij, p .151)

Blissymbolics is dependent on the language of the country in which it is used not only on the syntactical level but also on the lexical level. The use of letters to indicate a special entity of the kind the symbol represents (see section 2) and the strategy 'similar sound' (see section 2.3) are these sources of lexical dependence.

Here is a list of the rules Bliss did establish (adapted from McDonald):

1. The normal statement form:

[place], [time], thing/person, action, thing/person

This is the normal Agent-Act-Object order of English sentences. The []-brackets indicate that the place and time can be left out. When the time is mentioned it is not necessary to indicate the tense of the verb.

2. Command and polite command:

intensity, action

To make a polite command the intensity symbol can be replaced with the symbol for 'please'.

3. Negative form:

The negative form can be used in any other kind of sentence. It is indicated by the symbol for 'not' placed in front of the action.

4. Question:

A question is indicated by the question symbol in front of the normal sentence. When a question word like 'who' is used, the question symbol can be eliminated.

5. Passive voice:

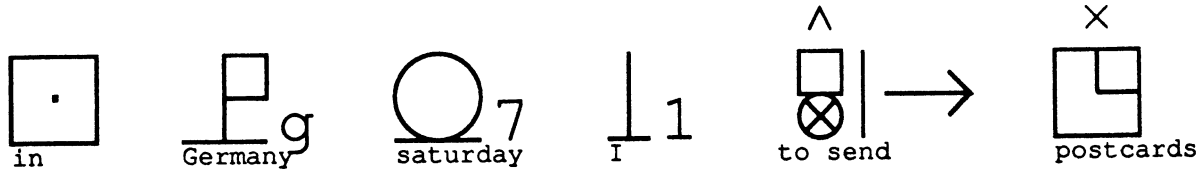
Bliss did not improve of the use of the passive voice but did make it possible in Blissymbolics. The action indicator above the action is replaced by the passive indicator. If an indicator of time is already present above the action the passive indicator will be added to it.

Bliss advises users 'not to use long sentences in semantography', but conjunctions like 'but' and 'because' do exist. Furthermore Bliss invented the 'relativizer', a symbol which indicates the start of a subordened clause.

Blissymbolics is a highly ambiguous language. There is, for instance, no rule which determines the order in which the objects appear in a sentence. So in "I give my sister the dogs." it is not clear whether I give the dogs to my sister or my sister to the dogs. But even the rules introduce ambiguities when used for translation. Take for example the normal statement rule. In the English sentence "I sit in the chair." the phrase "in the chair" indicates a place and therefore it should be placed according to the normal statement rule in front of the equivalent sentence in Blissymbolics. But 'to sit' is also the action which acts upon the chair. So the Blissymbolics equivalent of the above sentence could easily well be: "I sit chair." as "In chair I sit."

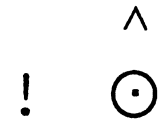
Example 7:

The normal statement form:



English equivalent: 'In Germany Saturday I did send a postcard.'

The command form:

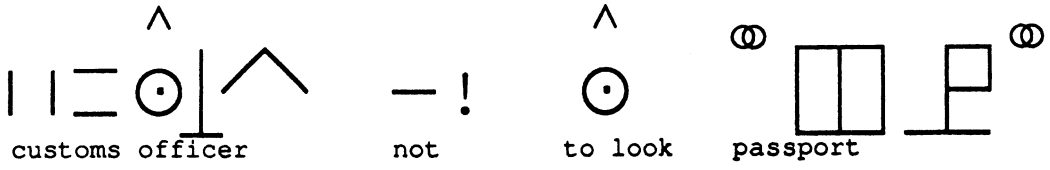


English equivalent: 'Look!'

The polite command form:

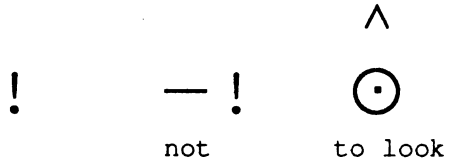
English equivalent: 'Please, help!'

The negative form in a statement:



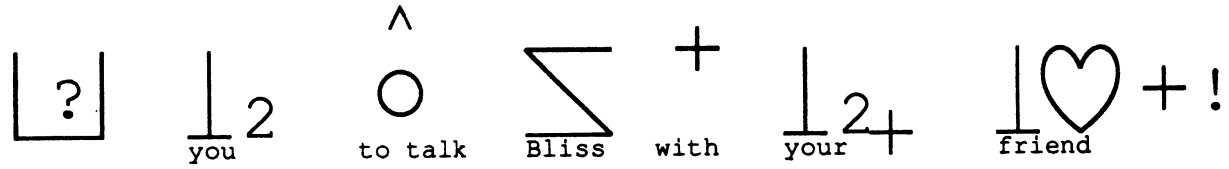
English equivalent: 'The customs officer did not look into the pasport.'

The negative form in a command:



English equivalent: 'Don't look!'

The question form:



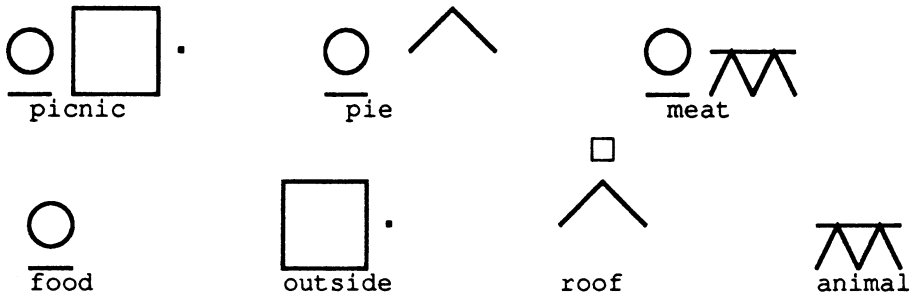
English equivalent: 'Did you talk Blissymbolics with your friend?'

5. Semantics

Blissymbolics is a meaning based language. The meaning of combined and compound symbols is said to be contrived of the meaning of the combining symbols. This is exactly Frege's principle of compositionality: "The semantic interpretation of a combined expression is a function of the semantic interpretation of the combining parts." (Gamut, p.206) But does this hypothese also apply to Blissymbolics?

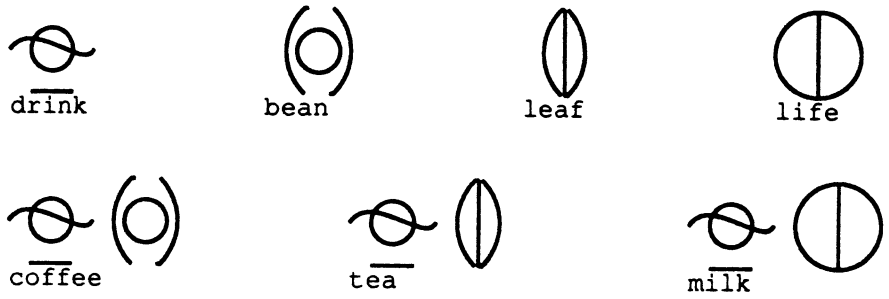
It does apply syntactically; the meaning of a sentence is made up out of the meaning of the words in it. And in a lot of compound symbols it does also apply, as will have been clear from some of the previous examples. But there are symbols were this rule works very freely. Look at the examples below. A picnic is food that is eaten outside, but a pie is not food that is eaten under a roof. It has a 'roof' itself. And so meat is not food that is eaten by an animal but food that is produced by an animal.

Example 8:



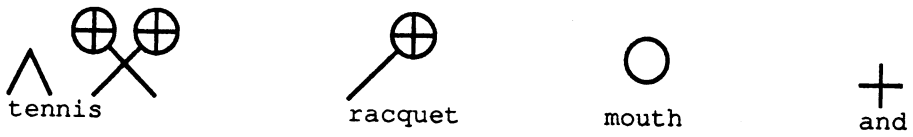
Coffee is made of beans, tea is made of leafs but is milk made of life? No, it is the thing which all baby mammals drink and in this way it gives life. So there is a function which tells you what the meaning of the compound symbol is. But that function is so irregular that the meaning of every compound symbol will have to be learned by Blissymbolic users. There is no other way to compute what a compound symbol will mean.

Example 9:



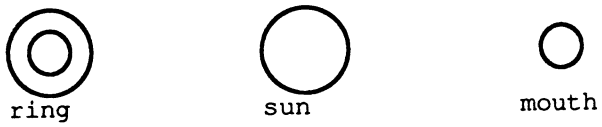
Furthermore, what are the combining elements in Blissymbolics? Lets assume that the symbol parts we discussed in section 3 are the basic semantic elements. But there is no 1-1 relation between graphical symbol elements introduced in section 3 and semantic basic elements. This problem is specially clear in the pictographic symbols. The small cirkel with the plus in it that appears in the symbol 'tennis' has no meaning of it own. It is not a compound from the small circle which means 'mouth' and the plus which means (in this position) 'and' or 'also'. But there is a pictograph that means 'racquet' in which these symbol parts appear.

Example 10:



And what to think of the combination of 'sun' and 'mouth' which appears in the superimposed compound symbol for 'ring'. There is no way the meaning 'ring' can be made out of 'sun' plus 'mouth'. So in this case the symbol for 'ring' itself is a basic semantic element. But when the naive Blissymbolics user who knows only a few symbols, sees this symbol how will he or she know that this symbol is a pictograph and not to be split apart to arrive at its meaning. Here again the combining rules will have to be learned symbol by symbol.

Example 11:



The strategies 'similar sound' and 'symbol part of' also attack the compositionality principle. The strategy 'symbol part of' has no rules to tell which part of the symbol is meant. Furthermore, this strategy is only necessary when the symbol part that is meant is not available as

separate symbol. And this is only the case when the compound symbol does not apply to the compositionality principle.

The strategy 'similar sound' uses the fact that in English homonyms exist. But homonyms mostly have very different meanings because they will have to be told apart by the context in which they are used. So in no way does the compositionality principle apply here. Besides, the strategy 'similar sound' as well as the adding of letters to symbols makes Blissymbolics dependent of the language of the country in which it is used. In English a 'G' is added to the symbol 'country' to indicate Germany, in Dutch this would be a 'D' from the word 'Duitsland'.

Even if Frege principle does not apply fullly Blissymbolics is still for a large part a meaning based language. But it is too much to say that the meaning of compound or combined symbols can be easily read from the combining symbols. There is a way to find the intended meaning but it needs a lot of human fantasy and associative strenghth. So much even that most compound symbols will simply have to be learned by the Blissymbolic users.

6. The Use Made of Blissymbolics by Handicapted People

Blissymbolics is used to support communication of handicapted people which can not or hardly speak. Some of these can control only their eyes purposely, others can use only their heads. More can use (one of) theirs hands but only to push a large button, not to use a typewriter. All the children which use Blissymbolics in the Netherlands at this moment, could not or hardly communicate until they were tought Blissymbolics. As a consequence these children must "not only learn the Blissymbols, but mostly how to communicate" (Verweij, p.120). Besides, the children that did grow up using Blissymbolics are just grown-up at the moment. So one can indeed say that Blissymbolics is barely outgrown its infancy.

Most children learn to communicate in Blissymbolics with the aid of a Blissymbols chart. A small number of symbols is placed on a chart in such a way that the child can easily see them and point at them (possibly with his of her head or eyes). The chart will for instance be placed in front of the child on his or her wheelchair. The number of symbols will be increased when the child learns more symbols. But the number is always limited by the size of the chart.

The Blissymbols chart is coloured to make it more surveyable. The verbs are green, the nouns are orange, the persons are yellow, adverbs are blue and other kinds of words are white. Every chart is unique and made only to serve the communicating purposes of one person. But even with such a uniquely adapted aid the production of a symbol is an inmense strain for most users.

In many country's computer applications for Blissymbolics are being made. In the Netherlands only one application is currently being used. With this application it is possible to type a number on a terminal, to see the matching symbol on a computer screen and to print one or more of them

on a matrix-printer. In other words this application is some sort of typewriter for Blissymbols which can not be used by a handicapted person without aid.

In the mytylschool of Heliomare (Wijk aan Zee) the author was glad to see a prototype of a new application which uses the colours of the Blissymbols chart and uses scanning as a way in which the user can indicate a symbol. This is an approvement on the application being used but it is far from ideal. There are no ways of correcting mistakes, so like an ordinary typewriter, if one makes a mistake in for instance a letter one must write it all over again. Taking in account the efforts of symbolproduction this is a serious discount. Besides this, only a small number of symbols can be made available.

The ideal Blissymbolics computer application has to be very flexible, in order to be usefull for both teachers and pupils with various handicapts. It should be possible to set the speed, the manner of input, as wel as the ways of ordering of the symbols available differently for every user. The manner of input could be scanning of a terminal screen or typing or clicking a mouse. It should be possible to add simple ergonomic controls (for example: one botton by which all functions could be operated) which are adaptable to most handicaps but these should also be removeable so as not to hinder the teacher which has to work with the application.

Because the application will be used very often as an aid which will replace spoken language it should run on a small portable computer which at the same time has enough memory to contain all Blissymbols relevant for one particular user. At best it should use the colours of the Blissymbols chart and for that it would need a coloured screen. Furthermore there should be many ways to make the output available, in symbols or in syntactically right Dutch (or language of the country), on screen, printer, through speach synthesizer or digital (telephone lines).

Especially for the teacher the application should have the possibilities of an ordinary computer editor. And it would be very useful if all kinds of other applications, like training programs, could be added. And of course for the handicapt pupil a Blissymbolics application would be very fault tolerant.

7. Bibliography

- Bliss, Charles K., Semantography, 2nd enlarged edition, Sydney: Semantography Publications, 1965

- Davies, Ena, The development and application of Blissymbolics, Child Language Teaching and Therapy, volume 1, number 3, October 1985

- Gamut, L.T.F., Logica, taal en betekenis, part 2, Utrecht 1982

- Hehner, B. (ed), Blissymbols for Use, Toronto: Blissymbolics Communication Institute, 1980

- Koerselman, Els and Thijssen, Anke, Computers "spreken" Bliss, studie in opdracht van het Centrum voor Onderwijs en Informatietechnologie, December, 1986

- McDonald, E. T., Teaching and Using Blissymbolics, Toronto: Blissymbolics Communication Institute, 1980

- Verweij, J.H., Bliss-symbolen; een taalkundige orientatie, Afstudeerscriptie Universiteit van Amsterdam, Amsterdam September, 1982

BLISSYMBOLICS - THE PROGRAM

1. Introduction

The Blissymbolics program is a computer application that translates Dutch sentences into Blissymbolics. The input of the program consists of simple Dutch sentences, namely those sentences whose equivalents are actually being used by the Blissymbolics users in the Netherlands. The output consists of Blissymbolics sentences in which the graphical symbols are represented by the coding that was made by the Blissymbols Communication Institute (BCI) in Toronto, Canada. A way to produce the graphical output is provided by a PostScript program which takes as input the BCI coding.

In a translation process one can distinguish three components: analysis, transfer and synthesis, the last of which is also called generation. During analysis the structure of the input sentence is established. The subject, the object, etc., of the input sentence are determined. The current ideas in linguistics all put the result of such an analysis in a treelike structure.

During the transfer phase parts of this tree, i.e. certain linguistic structures, are replaced by their corresponding structures in the target language. During synthesis a target language sentence is made out of the tree. The morphology of the target language is always an important issue in this component. In most systems the syntax of the target language is also an issue in this component.

This paper will describe the design of the Blissymbolics program and the choices that were made in order to arrive at a working program. Section 2 contains a survey of the total translation process. In section 3 the development of a grammar of Blissymbolics is described. The differences between this grammar and the Dutch grammar that is being used to analyse the input of the system are discussed in section 4.

The actual form of the grammar and dictionaries used by the Blissymbolics translation program is explained in section 5. Section 6 describes the main lines of the translation program. In section 7 the coded output of the system is discussed and an alteration that had to be made to the BCI coding is explained. The PostScript program which takes care of the graphical output is also briefly described in section 7. And, finally, in section 8 some conclusions are drawn.

2. The Translation process

The translation process as it is performed by the Blissymbolics program is build up out of the three major parts of a translation system that were introduced in the previous section. At first there is the analysis which is performed by a program that was not made by the author. This is

the *'natural language parser'* made at the Vrije Universiteit by Tanno Altena and Thijs Ott de Vries (Altena and Ott de Vries). The advantage of this parser is that it is not language bounded. The user can define the grammar being used to analyse the input sentences. The output of this parser exists of a series of one or more derivation trees for each input sentence, see figure 1 for an example of such a derivation tree.

Figure 1:

The input sentence: "ik geef Els veel boeken." was analysed by the parser according to the grammar in appendix A into the following derivation tree:

```
Zin
   mededeling
      uitgangszin_geen
         Onderwerp_geen_eerste_enkelvoud
            zelfvnwgroep_geen_enkelvoud_eerste
               zelfstandigvoornaamwoord_enkelvoud_geen_eerste
                  persvnw_geen_eerste_enkelvoud_onderwerp: 'ik'
         zelfww_eerste_enkelvoud: 'geef'
         Objecten
            indirectobject_zonder
               NounPhrase_geen_enkelvoud
                  nomen_enkelvoud: 'Els'
            directobject
               NounPhrase_geen_meervoud
                  Determinator_geen_meervoud
                     telwoord_meervoud: 'veel'
                  nomen_meervoud: 'boeken'
```

An example of the output of the natural language parser.

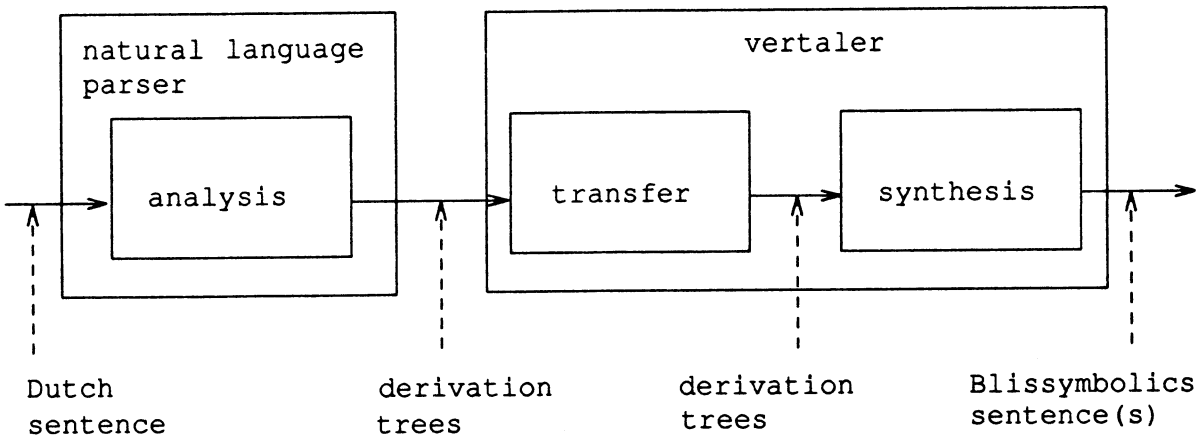
A disadvantage of the natural language parser is that the program code is hardly documented and therefore the program can be changed only at the cost of large efforts. This is the reason it is not an integrated part of the Blissymbolics system. The parser is used as a front-end processor that is like a black box to the rest of the system. Furthermore, it appeared very difficult to integrate knowledge of the source language morphology in the analysis. So all different forms of the same word will have to occur in the dictionary even if these forms are very regularly, like, for instance, work-works-worked.

The transfer and the synthesis component are combined in one program: the *'vertaler'*, which takes as input a series of derivation trees as produced by the parser. In the transfer component every derivation tree is altered until it has the form of a derivation tree in the target language. Various transformations (which will be discussed in section 4) are applied. Then each source language word in this altered tree is replaced by its translation. The transfer component has knowledge of both

the source language grammar and the target language grammar and of course of the differences between them.

From the altered derivation tree a sentence is made by the synthesis component. Essentially this component produces the yield of a derivation tree but at the same time morphological suffixes are added to the words. So this component must have knowlegde of the target language morphology. In the case of translation into Blissymbolics this component adds the indicators to the Blissymbolics words.

Figure 2:



A global view of the Blissymbolics System.

When designing a translation system the most important question is how much of the work is being done by the transfer component. If the input is analysed so deeply that the result of this analysis is independent of the source language and target language, the transfer component only has to put in the right word translations. This kind of translation system is called a *interlingua system.* If the input is analysed less thouroughly the transfer component has to translate bigger parts of the sentence into structures of the target language with the same meaning, i.e. during transfer the result of the analysis will have to be changed to reflect the structure of the target language. This kind of translation system is called a *transfer system.* When there is hardly any analysis in the translation system, it is called a *direct translation system* (see Neijt or Schoorl for an elaborate discussion of this kind of classification).

The Blissymbolics translation system can be classified as a transfer system (Neijt, p.104). It's analysis is a very important part but one can not say that an intermediate language which is independent of the source and target language, a so called *interlingua,* is formed. The intermediate form of the translated sentence in this system is the set of derivation trees some of which reflect most the structures of the target language, some of which reflect most the structures of the source language.

3. Grammars

Because it was developped in Australia by a European who had lived in England, Blissymbolics resembles much of the English language. Therefore the Blissymbolics language and Dutch are very much alike. So it should be possible to make grammars of Dutch and Blissymbolics that look very much alike. This notion is called *'isomorphism'*. "The grammars of different languages are isomorphic when every syntactic rule or basic expression in the one language corresponds qua meaning with at least one syntactic rule or basic expression in the other language." (Neijt, p.109)

But what should the grammar of Blissymbolics look like? Obviously it is not enough to take into account only the few rules Charles Bliss did give about syntax (see Part one, section 4). In that case a very common construction like 'my friend Frits' could not be part of Blissymbolics, because there is no rule that makes one phrase out of two nouns. No other rules are given, so the actual use of Blissymbolics, in particular in the Netherlands, had to be the ground on which to build a grammar. But looking closer at the actual use of the language, especially written Blissymbolics because that would be the input or output of the system, it seemed that the more advanced the users were, the more their use of Blissymbolics resembled Dutch.

In the end the choice was made to use the syntax rules of Charles Bliss and whenever they were not adequate the syntax rules of Dutch should be used. An obvious consequence of this choice is that the grammar of Blissymbolics becomes too complicated. Blissymbolics is a small language and its sentences are very simple. Large compound sentences, for instance, do not occur. So a second choice was made not to include any compound sentences for the time being. When the compound structures that do occur in Blissymbolics are rightly annalysed these will form a third kind of grammar rules that can be added to the system.

The result of these choices is that the overall structure of the sentences generated by this grammar is clearly Blissymbolics but the structure of parts of such a sentence is Dutch (see example 1). A second consequence is that achieving isomorphism between the two grammars becomes easier. But the price one has to pay is that Blissymbolics becomes almost as complicated as Dutch. Appendix A contains the Dutch grammar used to analyse the input by the system. The source of the Dutch syntax rules was (van Bart and Sturm) and the official reference (Geerst).

Example 1:

The Blissymbolics rule for the normal statement:

Normal Statemant =
[place], [time], thing/person, action, thing/person

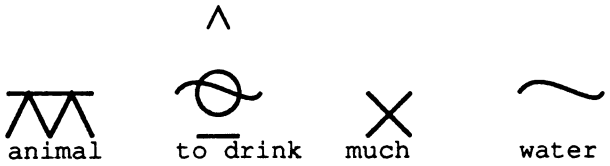
One of the Dutch noun phrase rules:

Noun Phrase =
[article], [adjective], noun

(The [-] brackets indicate that part of the sentence can be left out.)
To obtain a Blissymbolics grammar for the normal statement these rules are combined:

Normal Statement =
[place], [time], Noun Phrase, action, Noun Phrase
Noun Phrase =
[article], [adjective], noun

With this new grammar a Blissymbolics sentence like:



(English equivalent: "The animal drinks much water.") can be analysized.

4. Transformations

Given the above view on the grammar of Blissymbolics the differences in the two languages are not structural but in the order of the constituents. To change the order of the constituents in a derivation tree some transformations are applied. Transformations are operations which take a derivation tree from the Dutch grammar as input and give as output an altered derivation tree which form reflects more of the Blissymbolics grammar.

A few points are important when considering transformations. First, the order in which they are applied. Second, to which (sub)tree(s) they are applied, and third, if a transformation can be applied, should it be applied in any case or are there exceptions? The differences between the two languages are covered by the transformations given below. The order in which they are being discussed is the order in which they should be applied.

Transformation T1: inversion
In the Dutch language the subject of a sentence and the finite verb can occur in two orders: subject-verb and verb-subject. In Blissymbolics only the first order appears. The transformation T1 exchanges subject and the finite verb if they appear in the wrong order. This transformation is always applied if the subtree to which it should be applied (labelled 'inversezin' according to the grammar of appendix A) occurs in the derivation tree.

Transformation T2: action phrase
Different forms of verbs can appear in many places in the Dutch sentence, only the verb with congruence (persoonsvorm) must be next to the subject. The transformation T2 combines all the verbs of a sentence in one subtree and puts this subtree in the place of the verb with congruence. In other words all the verbs are combined into one 'action'-phrase. This transformation is applied to the whole derivation tree when more than one verb occurs.

Transformation T3: passive verbs
A same kind of transformation must be performed on a passive sentence. All verbs must be combined, but the verb which indicates the passive voice in the Dutch sentence must be removed. Because in Blissymbolics the passive voice is not indicated by a special verb but by the passive indicator above the main verb. But the verb to be removed is often the one which carries information about the tense of the Dutch sentence. So this information must be saved and combined with that verb which has to carry that information in the Blissymbolics sentence. The transformation T3 which performes this task is only applied at the subtree which contains a passive sentence. If this subtree is present it is always applied.

Transformation T4: removing 'door'
Another part of the Dutch passive sentence is to be removed too: the word 'door' (Eng: 'by') in the sentence part that indicates the actor of the deed (e.g. 'the cat' in "The fish was eaten by the cat."). In Blissymbolics the passive indicator together with the order of constituents implies this word so it can be left out. This transformation (T4) is applied in the same cases as the previous one.

Transformation T5: not
The occurence of the word 'niet' (Eng: 'not') as a separate part of a sentence in the Dutch language indicates a special kind of negative sentence. It must be translated into Blissymbolics by placing the symbol for 'not' in front of the action-phrase. This case is different from the sentence where the word 'niet' occurs as part of (e.g.) a noun phrase. Compare "Het is niet van mij." ("It does not belong to me.") with "Niet een van hen overleefde." ("Not one of them survided."). The translation T5 which implements this task is applied only when the subtree containing the word 'niet' is a direct child to the subtree which indicates what kind of sentence (passive, question, etc.) the main clause is.

Transformations T6 and T7: time and place adjuncts
In Dutch time and place adjuncts can occur in various parts of the sentence. In Blissymbolics these should be placed in front of the sentence except when the sentence is a yes/no-question. A yes/no-question is a question which is not indicated by a question word as first word in the sentence. The transformation T6 removes the adjunct of time and puts it in front of the sentence. Transformation T7 does the same for the adjunct of place. To obtain the right order of adjuncts T6 must be applied for T7. Both transformations are not applied on the subtree that contains a yes/no-question.

Transformations T8 and T9: question mark and exclamation mark
A yes/no-question is called a 'vraagtekenvraag' in the grammar of appendix A. Such a question in Blissymbolics must have a question symbol in front of it. The same is the matter in case of a command: an intensity symbol must be put in front of it. The transformation T8 en T9 perform these tasks. They are applied in any case when a subtree 'vraagtekenvraag' or 'bevel' is part of the derivation tree.

5. Form of the Grammar and the Dictionaries

The form of the grammar used in the Blissymbolics translation program is defined by the 'natural language parser'. The parser asks for an affix grammar without primitive predicates. An affix grammar is divided into two parts: the meta rules and the hyper rules. The hyper rules have the form of context free grammar rules but affixes can be added to the terminal or nonterminal symbols. If a terminal symbol with an affix can be derived from a nonterminal symbol than this affix, which will have the same value in both cases, can also be added to the nonterminal symbol.

The meta rules also form a context free grammar but they must generate a finite language. The meta rules define the values of the affixes that are being used in the hyper rules. From the meta and hyper rules a context free grammar is made which is used to analyse the input sentences. All terminal symbols in the meta and in the hyper grammar must be defined in the *'codefile'* which contains all the terminal symbols of the resulting context free grammar. (for more information see Altena and Ott de Vries).

The codefile also acts as way to simplify the monolinguistic dictionary. The monolinguistic dictionary is neccesary to determine for each word the place it can possibly take in a sentence. In the codefile the terminal symbols of the grammar are shortened to their abbreviations or any other coding meaningfull for the user of the system. Then in the monolingal dictionary words from the source language are combined with the codings from the codefile. Obviously, some words can be combined with more than one coding. It is, for instance, possible to combine the word 'you' with the quality (for number) 'plural' as well as 'single'.

The bilingual dictionary is not a part of the parser input. It has however a form that is like to the monolingual dictionary because it has almost the same contents. The bilingual dictionary consists of the source language words combined with their target language translations. As in the monolingual dictionary every source language word can be augmented by its coding in the codefile. But this coding can also be left out. So only if more than one coding is possible for a word and the translation of the word is different for each of these codings, the same word (with the different codings and the different translations) will occur more than once in the bilingual dictionary. And if the word is analysed differently it will be translated differently.

The bilingual dictionary must be alphabetically sorted. Comments can be added to the dictionary on separate lines that begin with a $-sign. The

following example will illustrate the use of meta and hyper rules, and clarify the form of the codefile and the dictionaries.

Example 2:

Meta rules that fix the possible values of an affix:

```
Persoon ::
   eerste;
   tweede;
   derde.

Getal ::
   meervoud;
   enkelvoud.
```

A hyper rule:

```
ZelfstandigVoornaamwoord + Getal - Persoon :
   persoonlijkvoornaamwoord + Persoon + Getal.
```

From the above rules the next context free grammar rules are produced:

```
ZelfstandigVoornaamwoord_meervoud :
   persoonlijkvoornaamwoord_eerste_meervoud;
   persoonlijkvoornaamwoord_tweede_meervoud;
   persoonlijkvoornaamwoord_derde_meervoud.

ZelfstandigVoornaamwoord_enkelvoud :
   persoonlijkvoornaamwoord_eerste_enkelvoud;
   persoonlijkvoornaamwoord_tweede_enkelvoud;
   persoonlijkvoornaamwoord_derde_enkelvoud.
```

The codefile for this grammar must contain the following information:

```
% persoonlijkvoornaamwoord_eerste_enkelvoud:  PvEE
% persoonlijkvoornaamwoord_tweede_enkelvoud:  PvTE
% persoonlijkvoornaamwoord_derde_enkelvoud:   PvDE
% persoonlijkvoornaamwoord_eerste_meervoud:   PvEM
% persoonlijkvoornaamwoord_tweede_meervoud:   PvTM
% persoonlijkvoornaamwoord_derde_meervoud:    PvDM
```

The monolingual dictionary:

```
ik     : PvEE.
jij    : PvTE.
hij    : PvDE.
zij    : PvDE, PvDM.
wij    : PvEM.
jullie : PvTM.
```

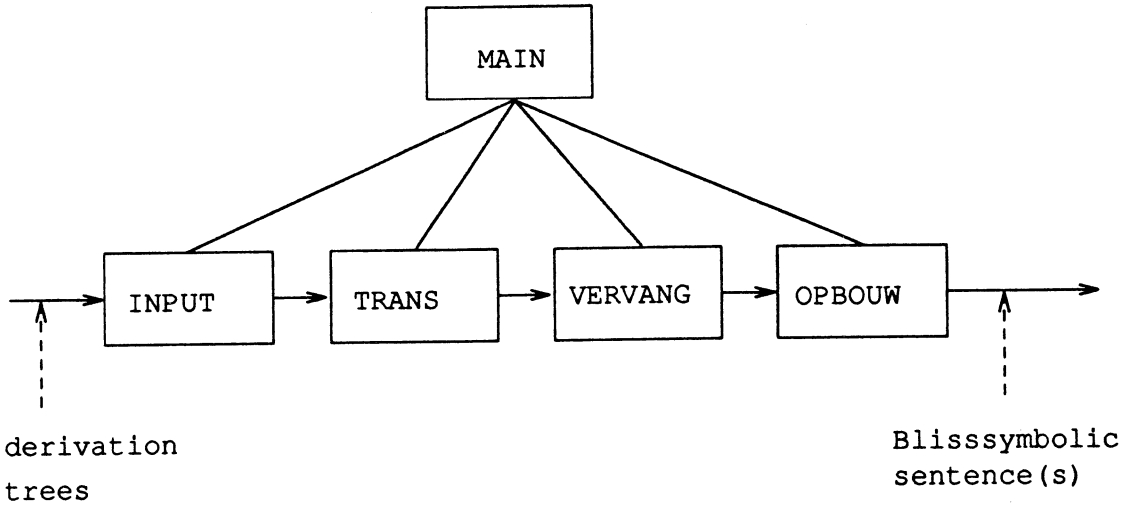
The bilingual dictionary:

```
  hij       -> he
  ik        -> I
  jij       -> you
  jullie    -> you
  wij       -> we
$ the next word has two different translations
  zij PvDE -> she
  zij PvDM -> they
```

6. The Program

In main lines the task performed by the 'vertaler' is this: after the dictionary is read, a series of trees that are derivations of one input sentence are read and processsed. Then the transformations are applied and after that the source language words are replaced. In the next step sentences are made out of the resulting trees. If the same sentence is made from more than one tree then this sentence occurs only once in the output. This process is repeated for following series of derivation trees if the input is not yet exhausted.

Figure 3:



A global view of the program 'vertaler'.

The program 'vertaler' is written in the C programming language (Kernighan and Ritchie) and is divided into eigth modules. The path of control of the program is determined by module MAIN which contains the main procedure and procedures that control the output and the use of memory. All procedures that control the input of derivation trees are contained by module INPUT. This module has knowledge of the exact output format of the natural language parser. When this parser will be replaced only module INPUT has to be changed.

The module TRANS contains all the procedures that implement the transformations on the derivation trees. TRANS is the only module which knows about the differences in the syntax of Blissymbolics and the syntax of the Dutch language. The implementation of all transformations is dependent on the grammar (appendix A) used to analyse the input, so various procedures from TRANS will have to be changed if this grammar is changed.

The module VERVANG takes care of the replacement of the source language words by their translations. This part of the program knows all about the bilingual dictionary but has no knowledge of the grammars. A binairy search is used to find the source language word in the dictionary. If this word has more than one translation then the additional information about the word in the dictionary is compared with the information in the derivation tree. In this way the right translation is found.

The module OPBOUW produces a Blissymbolics sentence from every derivation tree. The output procedure from module main takes care that only one of two or more identical sentences is part of the output. Essentially such a sentence is nothing more than the yield of the derivation tree. But this part of the program also places the indicators above he appropriate symbols. For example, it puts a plural indicator above a noun when the information in the derivation tree says that this noun was analysed as being plural. Therefore it has to know about the morphology of Blissymbolics. When other word forming operations like compounding are added to the program, their implementations should be part of module OPBOUW.

The concept of information hiding is applied to the program code as far as possible. Because of this an independent module DATA which implements the structure of the derivation trees was made. Other supporting modules are GLOB and DEBUG. GLOB contains some global procedures that were not especially written for this program, like all sorts of procedures that compare strings. In DEBUG the debug routines for the program are collected.

7. Coding and PostScript Output

As described in Part one, section 3, the BCI has produced an alpha-numerical coding to represent the graphical Blissymbols. Because this coding is an international standard it is used as output of the Blissymbolics system. The coding has however some defects, especially when used by such a ruthless machine as a computer (see also Part one, section 3).

Some of these defects had to be mended. In the BCI coding there must be a blank between the coding of the symbol and the coding of the indicators above the symbol. In most computer systems a blank in a string indicates that the previous word is ended. To avoid this problem a change was made to the BCI-coding: the blank between the coding of the symbol and the coding of the indicators is replaced by an underline.

Furthermore, some indicators do not appear as part of the symbols in the dictionary of Blissymbolics. these indicators, for example the passive indicator, are only useful in a sentence. A coding for them was made in the same manner as the codings for the other indicators. The passive indicator is represented by a 'P', the active indicator by a 'C', the past indicator by a 'S' and the future indicator by a 'F'.

A way to obtain graphical output is provided by a PostScript program. PostScript subroutines were made for (almost) every symbol element in the Blissymbolics alphabet. If it were possible to produce a symbol from its coding these subroutines would be sufficient. For every letter in the input coding a subroutine would be called and the symbol could be printed. But as explained in Part one this is not the case. So for every Blissymbol a larger routine had to be made which makes the drawing of the symbol. Still these larger routines can for a large part be made up of the small subroutines for the symbol elements. Sometimes they could be used straight away, sometimes the small routine had to be subject of a translation to obtain the right graphical output.

Obviously not all of the approved 1400 Blissymbols could be programmed this way. The PostScript program which is a part of this system describes graphical output for about 160 words. To make a different coding and to produce a more efficient way of obtaining graphical output is an interesting problem which will hopefully be attacked by another graduate student.

8. Conclusions

The study of the Blissymbolics language and the making of the grammars for Blissymbolics and for a small part of the Dutch language have absorbed a great deal of the attention and time available for this project. Actually these problems are not part of computer science but they had to studied because clearly they are important when making a translation program for Blissymbolics.

The available references to the Blissymbolics language or the Dutch language lacked the strictness and completeness required by computer applications. There were, for example, no references to a context free grammar of Dutch. And, as explained before, the syntax rules of Blissymbolics were very incomplete. So a major conclusion of this project is that most disciplines, among which linguistics, have not yet answered to the strict demands of the computer. So there is a lot of work for computer scientists on the edge of their work field.

The development of the translation program has not obtained as much attention as it should have. Several ad hoc methods were used. One of the first alterations should be the adding of the morphology of the Dutch language into the analysis. Another alteration could be the adding of the other word forming operations of Blissymbolics into the synthesis. This would imply the adding of a semantical component in which problems as explained in Part one, section 5 should be solved. These word forming operations of Blissymbolics could also be used by the word-by-word translation, so that user combined symbols or Dutch words without a

Blissymbolics equivalent could be translated properly. Furthermore, a more complete coding of the symbols and a more efficient way of programming the graphical output is an interesting object of study.

9. Bibliography

- Adobe Systems, Inc., Postscript language reference manual, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1986

- Adobe Systems, Inc., Postscript language tutorial and cookbook, Addison- Wesley Publishing Company, Inc., Reading, Massachusetts, 1987

- Altena, T. and Ott de Vries, T., Een parser voor natuurlijke taal: ontwerp en implementatie, Afstudeerscriptie Vrije Universiteit, Amsterdam 1987

- Bart, P. van, and Sturm, A., Zinsanalyse en de termen die daarbij gebruikt worden, Nijhoff, Leiden, 1987

- Dik, S.C. and Kooij, J.G., Algemene Taalwetenschap, Het Spectrum BV, Utrecht, 1981

- Geerst, G. (editor), Algemene Nederlandse Spraakkunst, Wolters-Noordhof, Groningen, 1984

- Hopcroft, J.E. and Ullman, J.D., Introduction to automata theory, languages and computation, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1987

- Kernighan, B.W. and Ritchie, D.M., The C programming language, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1978

- Landsbergen, J., 'Isomorphic grammars and their use in the ROSETTA translation system.' Paper presented at the ISSCO Tutorial on Machine Translation, Lugano, 2-6 April 1984

- Neijt, A., Automatisch vertalen in Nederland, Spektator 14-2, 1984/1985, 101-114

- Schoorl, J.J., De computer als vertaler, Meppel, 1986

Acknowledgements

I would like to thank Els Koerselman of the Revalidatiecentrum "De Trappenberg" for introducing me to Blissymbolics, and Onno Wagemakers of the metylschool of "Heliomare", Wijk aan Zee, for showing me some Blissymbolics computer applications in practice. Leo Zeef I thank for his assistance with analysing the semantics of Blissymbolics. And I am very gratefull to Tanno Altena and Thijs Ott de Vries for the use I could make of the 'natural language parser' which they have created.

Furthermore, I want to thank Jos Warmer for his help with the PostScript language and the debugging of the program code. And last but not least I thank Peter van Emde Boas and Theo Janssen, my guides in this project, for initially putting my attention on Blissymbolics and their overall assistence during the project.

The ITLI Prepublication Series

1986

86-01 The Institute of Language, Logic and Information

86-02 Peter van Emde Boas — A Semantical Model for Integration and Modularization of Rules

86-03 Johan van Benthem — Categorial Grammar and Lambda Calculus

86-04 Reinhard Muskens — A Relational Formulation of the Theory of Types

86-05 Kenneth A. Bowen, Dick de Jongh — Some Complete Logics for Branched Time, Part I Well-founded Time, Forward looking Operators

86-06 Johan van Benthem — Logical Syntax

1987

87-01 Jeroen Groenendijk, Martin Stokhof — Type shifting Rules and the Semantics of Interrogatives

87-02 Renate Bartsch — Frame Representations and Discourse Representations

87-03 Jan Willem Klop, Roel de Vrijer — Unique Normal Forms for Lambda Calculus with Surjective Pairing

87-04 Johan van Benthem — Polyadic quantifiers

87-05 Víctor Sánchez Valencia — Traditional Logicians and de Morgan's Example

87-06 Eleonore Oversteegen — Temporal Adverbials in the Two Track Theory of Time

87-07 Johan van Benthem — Categorial Grammar and Type Theory

87-08 Renate Bartsch — The Construction of Properties under Perspectives

87-09 Herman Hendriks — Type Change in Semantics: The Scope of Quantification and Coordination

1988

Logic, Semantics and Philosophy of Language:

LP-88-01 Michiel van Lambalgen — Algorithmic Information Theory

LP-88-02 Yde Venema — Expressiveness and Completeness of an Interval Tense Logic

LP-88-03 — Year Report 1987

LP-88-04 Reinhard Muskens — Going partial in Montague Grammar

LP-88-05 Johan van Benthem — Logical Constants across Varying Types

LP-88-06 Johan van Benthem — Semantic Parallels in Natural Language and Computation

LP-88-07 Renate Bartsch — Tenses, Aspects, and their Scopes in Discourse

LP-88-08 Jeroen Groenendijk, Martin Stokhof — Context and Information in Dynamic Semantics

LP-88-09 Theo M.V. Janssen — A mathematical model for the CAT framework of Eurotra

LP-88-10 Anneke Kleppe — A Blissymbolics Translation Program

Mathematical Logic and Foundations:

ML-88-01 Jaap van Oosten — Lifschitz' Realizabiility

ML-88-02 M.D.G. Swaen — The Arithmetical Fragment of Martin Löf's Type Theories with weak Σ-elimination

ML-88-03 Dick de Jongh, Frank Veltman — Provability Logics for Relative Interpretability

ML-88-04 A.S. Troelstra — On the Early History of Intuitionistic Logic

ML-88-05 A.S. Troelstra — Remarks on Intuitionism and the Philosophy of Mathematics

Computation and Complexity Theory:

CT-88-01 Ming Li, Paul M.B.Vitanyi — Two Decades of Applied Kolmogorov Complexity

CT-88-02 Michiel H.M. Smid — General Lower Bounds for the Partitioning of Range Trees

CT-88-03 Michiel H.M. Smid, Mark H. Overmars Leen Torenvliet, Peter van Emde Boas — Maintaining Multiple Representations of Dynamic Data Structures

CT-88-04 Dick de Jongh, Lex Hendriks Gerard R. Renardel de Lavalette — Computations in Fragments of Intuitionistic Propositional Logic

CT-88-05 Peter van Emde Boas — Machine Models and Simulations (revised version)

CT-88-06 Michiel H.M. Smid — A Data Structure for the Union-find Problem having good Single-Operation Complexity

CT-88-07 Johan van Benthem — Time, Logic and Computation

CT-88-08 Michiel H.M. Smid, Mark H. Overmars Leen Torenvliet, Peter van Emde Boas — Multiple Representations of Dynamic Data Structures

CT-88-09 Theo M.V. Janssen — Towards a Universal Parsing Algorithm for Functional Grammar

CT-88-10 Edith Spaan, Leen Torenvliet Peter van Emde Boas — Nondeterminism, Fairness and a Fundamental Analogy

CT-88-11 Sieger van Denneheuvel Peter van Emde Boas — Towards implementing RL

Other prepublications:

X-88-01 Marc Jumelet — On Solovay's Completeness Theorem

1989

Logic, Semantics and Philosophy of Language:

LP-89-01 Johan van Benthem — The Fine-Structure of Categorial Semantics

Computation and Complexity Theory:

CT-89-01 Michiel H.M. Smid — Dynamic Deferred Data Structures