# SPOKEN LANGUAGES UNIVERSAL NUMERIC TRANSLATION (SLUNT)

(Not for publication)

Walter Goshawke 63
Barrington Road
Bexleyheaih Kent DA7
4UW England Europe

December 1974

## Part 1. What is Slunt?

### The need for language translation by computer

The need for language translation by computer is greater now than it has ever been. This applies especially in the fields of science, technology and business correspondence. Slunt has been designed to deal eventually with all of these, and also with many other fields. It has been decided, however, to make business correspondence the first major objective, for the following reasons.

i The vocabulary is small.

ii The sentence structure is simple.

iii The number of businesses which could use language translation by computer is enormous in a large number of countries,

It is felt that a successful universal system for business correspondence would establish language translation by computer as a recognised branch of technology, would make it a commercially profitable enterprise, and would enable it to develop rapidly to include all 01: nearly all fields of translation.

### Number Language

The basis of Slunt is the translation of all material from whatever source language into Number Language by computer as the first step. At this stage all ambiguities in the original are resolved and the Number Language version becomes the official version, and can be translated accurately into any spoken language.

Number Language consists entirely of the digits 0 to 9 inclusive and can therefore be used on almost any digital computer in the world. Normally it will only be necessary for any user to translate from his own spoken language into Number Language and from Number Language into his own language. All messages will be sent in Number Language.

All Number Language statements are, subdivided into 10-digit units and most of these units represent words or groups of words, all closely defined in every spoken language. Other 10-digit units deal with sentence structure and tense. The type and length of statement is coded in a clearly defined code, and the areas which involve the subject, verb, object and so on are all clearly defined.

The emphasis at all stages is on clarity of definition. This ensures that the Number Language version is completely unambiguous, and can be relied on to express the true intentions of the writer of the material to be translated.

## The writer co-operates

An important feature of Slunt is that the writer of the material to be translated co-operates. If he writes anything ambiguous or uses an idiom which the then current Slunt programs cannot translate, the passage is re-written in simpler language which is acceptable. The work is then translated into Number Language by computer and the Number Language version is translated back into the original language, also by computer. If the author is then satisfied, the Number Language version becomes the official version.

## The Statement Code

The Statement Code is of fundamental importance in Slunt. Every statement in Number Language has a Statement Code, and this code shows clearly the nature of the statement. There are many types of statement. Some are very simple, consisting of one noun and one verb. Others are more complex. Each type has a different Statement Code.

The Statement Coda in Slunt gives the system tremendous flexibility and greatly simplifies its progress, especially in the early period of its development for any spoken language. Programming begins with statements of the simplest type, and accurate translations can be obtained from these at an early stage. Valuable experience is gained in using the early programs, and these can then be used as the basis of more complex programs.

## The Grades of Slunt

The use of Statement Codes enables Number Language messages and other material to be graded according to difficulty. In all grades accurate translations are obtained, but the earlier grades contain simple sentences and the later grades progressively more difficult ones. Users would know which grade they had reached and that they would not be able to translate Number Language messages written in a higher grade of Slunt than the grade they had programmed for.

## The vocabulary of Slunt

No word can be used in a Slunt translation until it has had a Number Language number (also called an NLnumber) allotted to it. To begin with, the NLnumbers are allotted to English words and each is accurately defined in English. Similar accurate definitions are needed for all other spoken languages, A difficulty in all methods of translation is that some words in spoken languages have more than one meaning. In Number Language this is not permitted. If an English word has six different meanings, there must be six different NLnumbers to represent each of the six meanings and each must be accurately defined in the Slunt dictionary.

## The Stages of Vocabulary

The creation of the Slunt dictionary is a very lengthy task and is likely to proceed much more rapidly in some countries than in others. For this and other reasons it is proposed to define vocabulary in clear stages, each stage being numbered, the earliest stages containing the most common vocabulary and the later stages having less common words. Still later, specialised words will be added. At this period in the development of Slunt it is possible to envisage three of the early stages of vocabulary as follows.

Vocabulary for program testing. A vocabulary of about 100 words is sufficient for testing the simple programs which are the first to be written for any spoken language.

Computer pan friend vocabulary. After a minimum number of basic programs have been written and tested it will be possible for Slunt users in different countries to exchange messages in Number Language, provided each has reached the same Grade of Slunt. The vocabulary needed at this stage is predominantly conversational. It is proposed to use the type of vocabulary found in books for students learning a foreign language for the first time.

Business correspondence vocabulary. The basis of the business correspondence vocabulary has been taken as the 700 Common Words used in teaching Pitman Shorthand. This vocabulary is likely to be particularly useful for testing Slunt programs as many hundreds of sentences have been printed using only these 700 words.

Specialised vocabularies

In many branches of science, technology and business and also in many other spheres there are specialised vocabularies. It is proposed at a later stage to cater for these in the Slunt dictionary by allotting a different 3 digit number to each speciality; each specialised word would then have this number as the first 3 digits of its NLnumber.

## Slunt Dictionary Makers

The accuracy of the definitions in the Slunt dictionary is of fundamental importance to the whole system, and at least one Slunt Dictionary Maker must be found for every spoken language. As Slunt progresses and specialised vocabularies are introduced, experts in the various specialised fields will also have to be consulted.

As Slunt originated in England, NLnumbers begin by being allotted English words, and each is closely defined. Thus Slunt Dictionary Makers for other spoken languages need to have a thorough knowledge of English, so that each definition they provide truly represents the meaning allotted to the NLnumber. Slunt Dictionary Makers are the only people involved in Slunt who need a knowledge of a foreign language.

## Classification of Number Language texts

The Grades of Slunt and the Stages of Vocabulary, both referred to earlier, enable a dual classification to be made of all Number Language texts. This will be particularly useful to users for whose spoken language Slunt is in an early stage of development, and when either the compilation of the Slunt Dictionary is well in advance of the programming, or the programming is well in advance of the dictionary. At all periods of development it will be possible to obtain Number Language texts recognisable as suitable for both the level of programming reached and the character of the vocabulary available.

The classification will also help fully established users, as very few users will want to carry all the possible Slunt programs and all the possible Slunt vocabulary.

## The grammar and word order of Number Language

The rules of Number Language perform to a large extent the same function for Number Language as grammar and word order do for spoken languages. They have been devised purely for convenience and differ from those of all spoken languages so far as is known. It is for the programmer to produce the correct grammar and word order from his own knowledge of his own spoken language, and to produce correct Number Language texts from his knowledge of the rules of Number language. Thus a word for word translation through the medium of Number Language from one spoken language into another will only happen as a coincidence.

## Idioms

After the first few elementary Slunt programs have been written, the problem of idioms will have to be faced. It is important to remember that idioms must never be translated literally into Number Language. A common idiom is "We have closed the door on further negotiations". The true meaning is "We have refused further negotiations". The door is imaginary and must not be included in the Number Language version of the sentence. An idiom must always be rewritten in simple words which themselves express the true meaning of the sentence.

Obviously it is possible for the computer to rewrite idioms before translating into Number Language. This would involve an enormous amount of additional programming as there arc so many common idioms. At present there is such a large volume of programming of a basic character to be done that it would be illogical to delay this for the treatment of idioms which can easily be rewritten in simple language. At a later stage the position regarding idioms can be reconsidered.

## Slunt is in its infancy

Only a limited number of the rules of Number Language have so far been formulated, only a small vocabulary is in use and only *a* few Slunt programs have been written. Enough has been done to demonstrate that Slunt is able to translate simple English sentences accurately into German, using very simple programming techniques. There is no reason to suppose that handling other spoken languages and sentences which are less simple will present any insuperable difficulties. All the work is simple.

The difficulties are the enormous amount of the work to be done and the fact that it will be a long time before Slunt becomes commercially profitable. It is hoped that research establishments, schools and college will work on Slunt. This may not happen for a long time, especially in some countries, and in others may never happen at all. It may be that in many countries the early development of Slunt will be in the hands of individuals working alone and unpaid, Slunt is well adapted to this type of development and to all others. The problems of making Slunt a reality are dealt with in Part 2.

# Part 2. Putting Slunt into practice.

## Computer pen friends

The first serious aim of Slunt is to provide business correspondence translation by computer. This objective cannot be achieved without a very great deal of experience in exchanging correspondence of a much more informal nature. In the early stages of every new system there are bound to be practical difficulties which only experience will reveal. In translation particularly, there are many pitfalls and frequently ludicrous translations, are produced. It is at the computer pen friend per that these appear and are eliminated.

After programs have been written for a number of Statement Codes, it will be possible to translate simple messages into and from Number Language. When this level has been reached Slunt users in different countries can become computer pen friends and can correspond with each other in Number Language. This is a very important stage and the aim should be to reach it as soon as possible.

## Standardisation

More rapid progress can be achieved if in all areas of Slunt activities the maximum possible degree of standardisation is introduced among all users working with the same spoken language. Valuable time and effort can be saved and duplication minimised. This should apply to all users, big and small. If you are *a* big organisation, you can make progress faster if you have help from other organisations or from individuals. If, at the other extreme, you are the only Slunt worker for your own spoken language it will be to your advantage to anticipate the arrival of future co-operators and to arrange your work so that it is readily intelligible and also easily transferable to other computers. There are two areas in particular in which standardisation is very desirable.

i Dictionary Design

ii Programming

## Dictionary Design

A basic task before any programming can begin for any spoken language is to create a Dictionary Design, that is to say, a standard form of the dictionary card which will give the NLnumber and all the forms of the word that can have that number. In Number Language the verb always has the same NLnumber whatever the tense, person or number, which are defined in other 10-digit units. Similarly the noun and other variable words always have the same NLnumber. Thus the design of the dictionary card is very important in programming, the position of the word on the dictionary card being of great significance in translation.

Each programmer could of course use his own Dictionary Design, but a design accepted by all programmers using the same spoken language would have great advantages. It would simplify the distribution of additions to the vocabulary and would make the exchange of programs between users much easier. It is worth while, therefore, devoting a considerable amount of thought to the Dictionary Design.

There is no great problem in the Dictionary Design for English. Only verbs have more than two forms. One verb has eight forms and others four or five. Every verb can be accommodated on one card. A design has been adopted for the programs already written, and seems quite adequate.

In some other languages verbs and other words have many more variations. If all are included additional cards are necessary. However, it would sometimes be possible to limit the number of words included in the card by modifying some of the words where necessary by programming. Whether this would be more efficient in the working of the system has to be decided when the dictionary is designed.

It seems preferable therefore that the Dictionary Designer should himself be a Slunt programmer. It is not necessary for him to know any spoken language other than his own.

## Slunt programming

There are many different programming languages which could be used for Slunt programming but it is preferable that for each spoken language all programs should be written in the same programming language, so that all programmers could use not only their own programs, but those of other programmers also. The question is which programming language is to be chosen.

Certain Slunt programs have already been written and Cobol is the programming language used. The principal reason for the choice was that Cobol is very widely used in business establishments in many counties, and it seems natural to use it for business correspondence. Many business firms are sympathetic to Slunt, but cannot justify anything more than trifling expenditure on research. Some are willing to allow ten minutes or so of computer time without charge to be used for this purpose every week or so. This can only be made use of in a programming language for which the computer has a compiler in use. Cobol was in use in the firms who very kindly granted time for the programs already written to be tested.

Unfortunately there are differences between the Cobol of one type of computer and the Cobol of another, and this can mean considerable alterations of programs being required when moving from one computer to another. There are many sophisticated programming techniques which can be used in Cobol. On the whole, the more advanced these are, the more troublesome it becomes to move from one computer to another.

## Slunt Cobol

On the other hand, very elementary Cobol programs can be moved with very little adjustment from any computer to any other. This is one of the reasons why a very simple version of Cobol, known as Slunt Cobol, is recommended for Slunt programming. Another reason is that this version of Cobol can be mastered in a few weeks, and that with its help, a person who has never done any programming before can soon begin to write valuable Slunt programs.

A list of the 17 Cobol statements used in the Procedure Division in Slunt Cobol programs is given below. Programs for all stages of Slunt, however advanced, can be written in Slunt Cobol. Slunt Cobol is easy to learn. The rules of Number Language are easy to learn. No mathematics is involved. No knowledge of any spoken language other than the programmer's own is necessary. Any person with a sound knowledge of the grammar of his own spoken language can quickly learn Slunt programming.

## The 17 statements used in the Procedure Division of Slunt Cobol

```
 1 OPEN INPUT SOMETHING.
 2 OPEN OUTPUT SOMETHING.
 3 READ SOMETHING AT END GO TO SOMETHING-ELSE.
 4 MOVE SOMETHING TO SOMETHING-ELSE.
 5 GO TO SOMETHING.
 6 ADD SOMETHING TO SOMETHING-ELSE.
 7 SUBTRACT SOMETHING FROM SOMETHING-ELSE.
 8 IF SOMETHING IS GREATER THAN SOMETHING-ELSE
 9 IF SOMETHING IS NOT GREATER THAN SOMETHING-ELSE
10 IF SOMETHING IS LESS THAN SOMETHING-ELSE
11 IF SOMETHING IS NOT LESS THAN SOMETHING-ELSE
12 IF SOMETHING IS EQUAL TO SOMETHING-ELSE
13 IF SOMETHING IS NOT EQUAL TO SOMETHING-ELSE
14 WRITE SOMETHING AFTER SOMETHING-ELSE.
15 WRITE SOMETHING BEFORE SOMETHING-ELSE.
16 CLOSE SOMETHING.
17 STOP RUN.
```

In many of the above statements SOMETHING and SOMETHING-ELSE may have COUNT or OTHER-COUNT in brackets, as in the following example.

```
MOVE SOMETHING(COUNT) TO SOMETHING-ELSE(OTHER-COUNT).
```

SOMETHING, SOMETHING-ELSE, COUNT and OTHER-COUNT are data names, paragraph headings, numbers or words.

## The first Slunt programs

The Statement Code in Number Language has already been referred to as giving great flexibility to Slunt, especially in its early period of development. It is possible to concentrate on one Statement Code at a time, writing the necessary programs to obtain satisfactory translations. These programs are of permanent value for future translations, besides being capable of being used as the basis for more advanced sentences with other Statement Codes. Thus the various grammatical features of the spoken language can be tackled in relatively small instalments, and standard procedures established which can be used wherever necessary in future programs.

An important part of speech in all spoken languages is the verb. There is at least one verb in almost every sentence, and as in many spoken languages the grammar of the verb is complicated, it is best to start with simple sentences which have only one noun and one verb. These have Statement Code 0001. Three programs for Statement Code 0001 are suggested.

A   Program to translate Number Language statements with Statement Code 0001 into the spoken language.

B   Program to translate sentences with Statement Code 0001 from the spoken language into Number Language, each sentence being input on one or more punched cards.

C Program to translate sentences with Statement Code 0001 from the spoken language into Number Language, each sentence being input by a succession of dictionary cards.

These three programs can be used to test thoroughly all the common tenses of the verb. Program A can contain a verb variation option. This option, if exercised, will first print a translation of the Number Language statement input, and then will translate the same statement in every tense of the verb, both affirmative and negative. The output from this operation, which will be in the spoken language, can be checked for accuracy by inspection.

The output from program A, when the verb variation option has been exercised, can be used as input for program B to test whether all the tenses are being correctly dealt with by program B.

Program C is only a slight variation of program B. It avoids the need for hand punching the spoken language sentences for input. The output from program C, which will be in Number Language, can be translated by program A into the spoken language and checked.

When these three programs have been satisfactorily tested, they can be used as the basis of other programs for dealing with sentences with other Statement Codes.

## The allotment of Statement Codes

In the early stages of the development of Slunt, the Statement Code for each sentence must be determined by inspection. The appropriate four-digit Statement Code is fed into the computer immediately before the sentence in the spoken language. After a number of programs handling a variety of different Statement Codes have been written, it will be possible to attempt to write a program which will allot Statement Codes by computer. This is a very interesting problem, and one which must be solved before Slunt can be successful for business correspondence. It is felt that it will not be difficult to allot the correct Statement Code by computer in the great majority of cases.

## Slunt organisations

Standardisation and co-operation are so essential at all stages of the development of Slunt, that it is important that as soon as possible a Slunt organisation should be set up in every country and that these should be affiliated to a world Slunt organisation. A Slunt organisation for a spoken language should perform the following functions.

i It should keep a register of all Slunt users and Slunt programmers for its spoken language.

ii It should keep a register of all well-wishers including computer users who are willing to give free computer time for Slunt program running and testing.

iii It should appoint a panel of Slunt Dictionary Makers.

iv It should appoint a panel of translators whose task it would be to translate all new rules and other material provided by the world Slunt organisation.

v It should keep its members informed of all Slunt developments.

vi It should keep the world Slunt organisation informed of its progress.

vii It should set up a panel of program testers whose task it would be to approve a Dictionary Design, to test programs and to authorise new additions to its Slunt program library.

viii It should form a library of Slunt programs for its spoken language.

ix It should widely publicise its library of Slunt programs and encourage its use.

x It should encourage programmers to increase its Slunt program library.

xi When its Slunt program library had reached an appropriate level, it should encourage its members to become computer pen friends.

xii It should form a library of classified Number Language texts, and make it available to its own members and to the world Slunt organisation.

xiii It should do all in its power to increase interest in Slunt in its country and to encourage Slunt research in new areas of translation.

xiv It should do all in its power to ensure that Slunt remains freely available to all users in the area where its language is used.

A world Slunt organisation should perform the following functions.

i It should appoint a panel of experts whose task it would be to codify the rules of Number Language, and to modify and extend the rules whenever necessary in the light of the needs and experience of Slunt users throughout the world.

ii It should appoint a panel of Slunt Dictionary Makers whose task it would be to establish official definitions for all Number Language Numbers in the Slunt Dictionary, and to modify and extend the Slunt Dictionary whenever necessary in the light of the needs and experience of Slunt users throughout the world.

iii It should have affiliated to it all Slunt organisations for the various spoken languages throughout the world.

iv It should keep all its member organisations informed of all developments in Slunt.

v It should form a library of classified Number Language texts and make these available to users throughout the world.

vi It should bring together Slunt users in different countries who wish to correspond in Number Language.

vii It should do all in its power to introduce Slunt to countries where it is not in use, and to increase the use and scope of Slunt in all countries by research and other means.

viii It should do all in its power to ensure that Slunt remains freely available to all users throughout the world.

## Who can help with Slunt?

Slunt has been designed in such a way that all effective work on it, wherever performed and in however small quantities, can be useful to all users of Slunt. Every interested person can help with Slunt. All the work is simple and within the grasp of any educated person. For all but a few workers in each country, a thorough knowledge of the grammar of one's own spoken language is all that is necessary. The rules of Number Language and the programming of Slunt are easily learnt.

Workers are required in every country, even if they are working alone. More rapid progress can of course be made if research establishments, universities, colleges and schools engage in Slunt research, which is very suitable for many of these institutions. The possible roles of individuals and of institutions in Slunt research is considered in detail in the next few paragraphs.

### Universities

Many universities are already engaged in a variety of kinds of literary and linguistic research involving computers. Many have created dictionaries. Such universities are very well equipped to take part in all Slunt activities, and it is hoped that they will do so. It is a very great advantage to have as Slunt Dictionary Makers expert linguists who have had experience of compiling dictionaries, and can undertake to work on the Slunt Dictionary over a long period. Universities could be very valuable in establishing libraries of Slunt programs and of classified Number Language texts. They could take an active part in the Slunt organisation of their country and in the world Slunt organisation. They could write Slunt programs and engage in Computer Pen Friend correspondence. Their continuity of existence would be a very great asset to Slunt. It is also felt that Slunt could become a very great asset to universities.

### Schools and colleges

Slunt is a suitable subject for almost every educational establishment at every level where computer courses are available and in every country. The continuity of existence of schools and colleges makes them particularly suitable to be active members of their local Slunt organisation and of the world Slunt organisation. If the need were there, they could be Slunt Dictionary Makers and librarians. This would depend on whether this work were being undertaken for their spoken language more effectively elsewhere. Above all, they could develop Slunt programs.

Slunt programming has the following advantages for educational establishments.

i Slunt programming is very simple.

ii Mathematics and statistics are not involved, other than the addition and subtraction of very small numbers.

iii At all stages correct translations are obtained.

iv Slunt programming is progressive. Every program written increases the scope and usefulness of the system.

v Slunt programmers can co-operative with other Slunt programmers using the same spoken language and can exchange programs.

vi There is a great incentive to write enough programs to enable Computer Pen Friend letters to be sent in Number Language to Slunt programmers in other countries.

vii Computer Pen Friend letters can be written both by programmers themselves and by other students. This will provide valuable material for testing programs besides arousing a wider interest in computers.

viii Slunt research does not make great demands on computer storage space. The limited dictionary necessary for a translation can be input at the same time as the program.

## Individual workers working alone

Slunt is very well adapted for the use of the individual programmer working alone. It is expected that a very great deal of the early development of Slunt will be in the hands of individuals in a variety of countries. If the individual can obtain free time on a computer for testing his programs, he can make progress. His work will be of value to all Slunt users throughout the world, especially during the early stages, because it is important to establish that Slunt is valid for as many spoken languages as possible.

In many cases an individual may be the only person working on Slunt for his spoken language. In this case he would have to be Dictionary Maker, Dictionary Designer and programmer. He is likely to find the work immensely satisfying. It is worth doing just as a hobby.

Even when Slunt becomes well established, there will still be scope for the individual working alone. It will be many years before all aspects of translation are touched by Slunt. One has only to think of the enormous variety of spare time activities which are available, such as fishing, bird watching, embroidery, stamp collecting and so on. Slunt vocabularies will need to be provided and thoroughly tested for these and many other special subjects.

## Part 3. The rules of Number Language.

### The present state of the rules

Number Language has been devised solely for the purpose of computer translation of spoken languages. Only a limited number of rules have so far been formulated, but the general framework is such that additional rules can be added without invalidating those already in operation. An effort could be made at this stage to compile a complete set of rules for all possible translation requirements. It is not, however, proposed to do this. It is felt that a better plan is to provide sufficient rules for current requirements from time to time, and to examine closely the working of existing rules each time further rules are added, always bearing in mind possible future needs. By this means the maximum benefit can be obtained from experience and from the views and wishes of users. The rules devised so far are given below.

### Examples of NLstatements

A series of examples of English sentences coded in Number Language is given immediately following these rules. These are referred to as NLstatement (1), NLstatement (2), etc., whenever an illustration of the rules is felt to be necessary.

### NLunits and NLstatements

Number Language consists entirely of the 10 digits 0 to 9 inclusive, some of the digits occurring more frequently than others. The digits are always in groups of 10. These 10-digit groups are known as Number Language units or NLunits. An NLunit can represent a word or group of words in a sentence, or can have other functions. A Number Language statement or NLstatement consists of a number of 10-digit NLunits and is the equivalent of a sentence or phrase in a spoken language.

Units in a statement are referred to by number, the first unit in a statement being 01, the second 02 and so on. These NLunit numbers are used for reference only and do not appear in the unit itself. NLstatements are also referred to by numbers which are not part of the statements. A reference to another unit in the same statement is coded 50. Thus the tenth unit of the same statement is 5010 and the twelfth 5012. The previous NLstatement is coded 49 and the statement before that 48 and so on. Later statements are referred to by the codes 51, 52, 53 and so on. Thus the fourth and fifth units of the previous statement are referred to as 4904 and 4905 respectively. The ninth unit of the statement before that is 4809, and the eleventh unit of the statement after the present statement is 5111.

There are three distinct types of NLunit, each having certain specified functions. These types are known as Type 1 Units, Type 2 Units and Type 3 Units respectively, and appear in that order in NLstatements.

### Type 1 Units

Every NLstatement begins with one or more Type 1 Units, and these give general information as to the form of the statement and its relation to other statements. The first Type 1 Unit of a statement always commences with the digits 99 and this is the signal that a new NLstatement has commenced. The third and fourth digits of the first unit give the reference number of the final Type 2 Unit in the statement. The fifth and sixth digits of the first unit give the reference number of the final unit of the statement. The final four digits of the first unit are the Statement Code, and indicate the type of statement and whether there are further Type 1 Units in the statement. Further details of Statement Codes are given later.

A typical initial Type 1 Unit for a simple statement is shown in NLstatement (1). The statement "The man comes" has the Statement Code 0001, which does not require a second Type 1 Unit. The final Type 2 Unit is the third, and there are seven units altogether in the statement. The Type 1 Unit is thus 9903070001. Another simple case is NLstatement (2), "The man read the book". This has Statement Code 0003, which again does not require a second Type 1 Unit. Here the final Type 2 Unit is the fourth and the final unit of the statement is the tenth. Thus the first unit is 9904100003.

## Type 2 Units

Type 2 Units may be said to have a structural function, and to indicate how the NLstatement in which they appear is made up. The number of Type 2 Units in a statement is defined in the first unit of the statement, so there is no confusion between Type 2 Units and Type 3 Units in the statement.

Just as the Statement Code defines the nature of the statement, the Type 2 Units indicate how the statement is constructed. The simplest Statement Code is 0001, and indicates that the statement consists of a subject and a verb. For this Statement Code there are two sets of Type 2 Units, each consisting of one or more units. The first set specifies which Type 3 Units are associated with the subject. The second set of Type 2 Units specifies which Type 3 Units are associated with the verb. For other Statement Codes there are other sets of Type 2 Units, each set defining the Type 3 Units involved in one grammatical section of the statement.

As an example, in a statement where the Type 1 Unit is 9903070001, the first Type 2 Unit might be 5004050000 and the second 5006070000. This would mean that the fourth and fifth units of the statement refer to the subject, and the sixth and seventh to the verb. See NLstatement (1). Alternatively, with the same initial Type 1 Unit, the first Type 2 Unit might be 5004040000 and the second 5005070000. This would mean that only the fourth unit refers to the subject, and the fifth, sixth and seventh to the verb. It will be seen that in each case the first and last units only are recorded, and that where there is one unit only, it is recorded twice, once as first unit and once as last unit. See NLstatement (3). The final 4 digits of Type 2 Units are usually zero.  In most of the simple sentences dealt with in these notes, each set of Type 2 Units has only one Type 2 Unit. An exception is shown in NLstatement (11), where the indirect object includes the father and the young children. There is a Type 2 Unit for the father and a Type 2 Unit for the children.

## Type 3 Units

Type 3 Units may be described *as* the basic material of Number Language. Most of them have a dictionary meaning, and this is indicated by the Number Language number or NLnumber, which occupies the first eight digits of the unit. The final two digits are used for other purposes, but are usually zero. The seventh and eighth digits of the NLnumber indicate which part of speech the corresponding word is. The various codes are as follows.

| Code | Part of Speech |
|---|---|
| 1 | Verb |
| 2 | Preposition |
| 3 | Adjective |
| 4 | Adverb |
| 5 | Noun |
| 17 | Some-word (singular) |
| 18 | Pronoun |
| 27 | Some-word (plural) |
| 94 | Time adverb |

Most 8-digit NLnumbers which have an ordinary dictionary meaning have their first digit as zero. Where the first digit is not zero the fact is significant. Specialised vocabularies have a three digit code for each speciality and these three digits form the first three digits of each specialised NLnumber. There are also Type 3 Units for special purposes with a two-digit code in their first two digits.

## Type 3 Units for special purposes

Certain Type 3 Units with specified first and second digits have special purposes. These codes have the following meanings.

| Code | Nature of Type 3 Unit |
|---|---|
| 85 | Pronouns and possessive some-words |
| 88 | Tenses of verbs |
| 89 | Questions |
| 91 | Proper names |
| 92 | Numerals |
| 93 | Geographical names |

## Linking codes for Type 2 Units and for Type 3 Units

Where the ninth and tenth digits of a Type 2 Unit form a linking code, the unit and the Type 2 Unit immediately following it are linked grammatically. Thus two Type 2 Units can represent the subject of a statement, and two the verb, and so on. If two successive Type 2 Units have a linking code, the subject or verb would be represented by three Type 2 Units, and so on. The linking codes for the ninth and tenth digits of Type 2 Units are as follows.

| Code | 01 | 02 | 03 | 04 | 05 | 06 |
|---|---|---|---|---|---|---|
| Meaning | and | or | but | of | from | in |

Similar arrangements apply for Type 3 Units. In NLstatement (11) Type 2 Units (06) and (07) are linked. In the same NLstatement Type 3 Units (10) and (11) are linked.

## Statement Codes

The final four digits of the first Type 1 Unit of an NLstatement are the Statement Code and indicate the type of statement which follows. Examples of three simple Statement Codes are given below, showing the Type 2 Unit sets, examples in English, and references to the equivalent NLstatements following these rules.

| Statement Code | Type 2 Unit sets | English examples | NLstatement reference |
|---|---|---|---|
| 0001 | i subject | The quick man writes | (4) |
| | ii verb | The small child was quick. | (5) |
| | | Fifteen women will have written. | (6) |
| 0003 | i subject | I drink my milk. | (7) |
| | ii verb | The man saw you. | (8) |
| | iii object | He will be sending my letters. | (9) |
| 0005 | i subject<br>ii verb<br>iii direct object<br>iv preposition<br>v indirect object | The mother gave the books to the father. | (10) |

For these Statement Codes no further Type 1 Units are required other than the first. The statements given above are very simple ones. Adding adjectives and other descriptive matter does not necessarily alter the Statement Code. For example, the sentence quoted for Statement Code 0005 could be expanded to "The small and old mother yesterday quickly gave the books to the father and young children" and the Statement Code would remain 0005. See NLstatement (11).

## Nouns and some-words

The NLnumber for the English word "man" is 03110005, which can also mean "men", and is unchanged whatever the position of the word in the sentence. Whether the man is the subject or object or indirect object of the sentence is defined by the Type 1 Units and the Type 2 Units. The same applies to all nouns. Whether the noun is singular or plural is determined by *a* some-word. Every noun in Number Language must be followed by a numeral or by a some-word, which besides having a meaning indicates whether the noun is singular or plural. Some-words are of two kinds, possessive some-words and others.

Examples of possessive some—words are the following.

His, her, my, your, its, etc.

Examples of other some-words are the following.

A, an, the, some, many, this, that, these, those, etc.

NLnumbers of some-words have 17 as their seventh and eighth digits if they are singular and 27 if they are plural. Some-words like "the" and "some", which can be singular or plural, have two forms, one for singular nouns and one for plural. See NLstatements (8) and (10).

Some sentences in English and other languages include nouns without some-words. Examples are "Milk arrived" and "Men are working". In all such cases a no-some-word-singular (NLnumber 05000117) or a no-some-word-plural (NLnumber 05000127) must be introduced into the NLstatement. See NLstatements (25) and (26).

Possessive some-words have the same NLnumbers as the corresponding pronouns, except that their seventh and eighth digits are 17 if applied to singular nouns or 27 if applied to plural nouns. Thus there are two forms for each. "My" in "My milk" is 35011117 and "my" in "my letters" is 85011127. See NLstatements (7) and (9).

## Pronouns

Pronouns as used in spoken languages are not usually necessary in Number Language. Instead of a pronoun, there is simply a cross reference to the NLunits where the original noun was defined. However, NLnumbers for the pronouns are useful in testing programs and *a* set of such numbers has been devised. The 8-digit NLnumber for a pronoun is made up as follows.

| | |
|---|---|
| Digits 1-2 | Always 85 |
| Digit 3 | 0 Formal |
| | 1 Formal |
| Digit 4 | 1 Male persons |
| | 2 Female persons |
| | 3 Persons of either sex |
| | 4 Other than persons |
| Digit 5 | 1 First person |
| | 2 Second person |
| | 3 Third person |
| Digit 6 | 1 Singular |
| | 2 Plural |
| Digit 7-8 | Always 18 |

Thus the formal version of "he" or "him" is 85013118, and the informal female singular version of "you" (or "du" in German) is 85122118. See NLstatements (9) and (3).

## Verbs

Like the noun, the verb in Number Language is invariable. The tense of a verb is indicated by a Type 3 Unit with 88 as its first two digits and which immediately follows the NLunit containing the NLnumber of the verb. The 10-digit Type 3 Unit which denotes the tenses of the verb is made up as follows

| | English examples | NLstatement reference |
|---|---|---|
| Digits *1-2,* always 88 | | |
| Digit 3 active-passive | | |
| 0 active | I ask, the fish eats | (12),(14) |
| 1 passive | I am asked, the fish is eaten | (13),(15) |
| Digit 4 tense-type | | |
| 0 complete action | I wrote | (16) |
| 1 incomplete action | I was writing | (17) |
| 2 repeated complete action | I usually wrote | (18) |
| 3 repeated incomplete action | I was usually writing | (19) |
| Digit 5 tense-code | | |
| 3 past-past | I had eaten | (20) |
| 4 past | I have eaten | (21) |
| 5 present | I eat | (22) |
| 6 future | I shall eat | (23) |
| 7 future-future | I shall have eaten | (24) |
| Digit 6 affirmative-negative | | |
| 0 affirmative | | (1) etc |
| 1 negative | | (3) |
| Digits 7-10, zeroes. | | |

Adjectival verbs

In such sentences as "The child was quick" and "The man was old", "to be quick" and "to be old" are treated as verbs in Number Language. Thus in NLstatement (5), which is the equivalent of "The small child was quick", "small" is *an* adjective and "to be quick" is a verb. Such verbs are known as adjectival verbs.

NLnumbers of certain adjectives, adjectival verbs and adverbs are related, the first seven digits being identical and the eighth being 3, 1 or 4. An example is given below.

| | Part of speech | NLnumber | NLstatement reference |
|---|---|---|---|
| quick | adjective | 04280003 | (4) |
| to be quick | adjectival verb | 04280001 | (5) |
| quickly | adverb | 04280004 | (11) |

The place of adjectives and adverbs in NLstatements

Adjectives are always associated with nouns and are therefore treated as part of the subject, object or indirect object in the Type 2 Unit sets which define the form of NLstatements.

Adverbs are always associated with the verb in the Type 2 Unit sets which define the form of NLstatements. This applied both to time adverbs and to other adverbs. Time adverbs are words like the following.

Now, yesterday, recently, Tuesday and so on,

The treatment of adjectives and adverbs is illustrated in NLstatement (11).

### Numerals

Numerals in Number Language are coded in Type 3 Units commencing with the digits 92, followed by six digits indicating the numeral. The remaining two digits are a suffix, usually zero.  If the suffix is 92 the numeral continues into the next unit. Examples are as follows.

| Numeral | Type 3 Units |
|---|---|
| 6 | 9200000600 |
| 824,538 | 9282453800 |
| 24,529,034 | 9200002492 9252903400 |

See NLstatement (6).

## An example of coding. NLstatement (1)

(1) The man comes.
A simple example of coding in Number Language is now given. The sentence "The man comes" appears in Number Language as follows.

| Type 1 Unit | Type 2 Units | | . | Type 3 Units | | |
|---|---|---|---|---|---|---|
| | . Noun<br>The man | Verb<br>comes | man | the<br>(singular) | to come | (present<br>tense) |
| 9903070001 | 5004050000 | 5006070000 | 0311000500 | 0537101700 | 0111000100 | 8800500000 |
| 01 | 02 | 03 | 04 | 05 | 06 | 07 |

For the sake of clarity the 10-digit units are numbered (01) to (07) in the above illustration, but this numbering, although used, is not part of the Number Language statement.

The Type 1 Unit indicates (a) by its first *2* digits (99) that a new statement has commenced, (b) by its final 4 digits (0001) that the statement contains one noun and one verb and that there are no further Type 1 Units, (c) by its third and fourth digits (03) that the final Type 2 Unit is *the* third and (d) by its fifth and sixth digits (07) that the final unit of the statement is the seventh.

The Type *2* Units indicate which Type 3 Units are associated with the noun and the verb respectively.  In unit (2) the digits 500405 indicate that the units (04) and (05) are associated with the noun.  In unit (03) the digits 500607 indicate that units (06) ad (07) are associated with the verb.

All the Type 3 Units in the statement are dictionary units except unit (07) which indicates the tense of the verb. In order to alter the tense of the statement it is only necessary to alter unit (07), the rest of the statement remaining unchanged. The NLnumber for the verb "to come" is the first 8 digits of unit (06), (01110001). This NLnumber for the verb never changes whatever the tense, person or number.

Similarly the NLnumber for the noun "man" is the first 8 digits of unit (04), (03110005(. This NLnumber never alters whether the noun, is singular or plural or whether the noun is the subject, the object or the indirect object of the statement. Unit (05) in, this statement includes the NLnumber for "the" in the singular (05371017). The NLnumber for "the" in the plural is 05371027.

Further examples of coding. NLstatements (2) and later

(2) The man read the book.

```
           man      read      book        man      - the      to read
9904100003 5505060000 5007080000 5009100000 0311000500 0537101700 0435000100
01         02         03         04         05         06          07

   (past)     book      the
8800400000 0071000500 0537101700
    08         09        10
```

(3) You had not arrived quickly.

```
               you     arrive     you     to arrive (past-past) quickly
9903070001 5004040000 5005070000 8512211800 0037500100 8800310000 0428000400
   01         02         03         04         05         06          07
```

(4) The quick man writes.

```
           man    writes     man         the      quick   to write
9903080001 5004050000 5006070000 0311000500 0537101700 042800300 0612000100
01         02         03         04         05          06         07

(present)
8800500000
08
```

(5) The small child was quick,

```
          child     quick     child        the    small  to be quick
9903080001 5004060000 5007080000 010300500 0537101700 0498000300 0428000100
01         02         03         04         05         06          07

  (past)
8800400000
    08
```

(6) Fifteen women will have written.

```
                women     written     women     fifteen  to write (future-future)
9903070001 5004050000 5006070000 0605000500 9200001500 0612000100 8800700000
    01         02         03         04         03         06         07
```

(7) I drink my milk.

```
                  I        drink       milk        I       to drink   (present)
9904090003 5005050000 5006070000 5008090000 8501111800 0158000500 8800500000
    01         02         03         04         05         06         07

   milk        my
0332000500 8501111700
    08         09
```

(3) The man saw you.

```
                 man        saw        you        man        the      to see
9904090003 5005060000 5007080000 5009090000 0311000500 0537101700 0477000100
    01         02         03         04         05         06         07

  (past)      you
8800400000 8501211800
    08         09
```

(9) He will be sending my letters.

```
           he      sending    letters      he      to send    (future)
9904090003 5005050000 5005070000 5008090000 8501311800 0480000100 8801600000
01         02         03         04         05         06         07

letters      my
.0294000500 8501112700
08         09
```

(10) The mother gave the books to the father.

```
         mother      gave      books       to       father     mother
9906150005 5007080000 5009100000 5011120000 5013130000 5014150000 0334000500
01         02         03         04         05         06        07

the    to give       (past)     books      the        to         father
0537101700 0217000100 8800400000 0071000500 0537102700 0544000200 0194000500
08         09         10         11         12         13        14

    the
0537101700
    15
```

(11) The small and old mother yesterday quickly gave the books to the father and young children.

```
         mother       gave      books       to   father and children
9907230005 5008110000 5012150000 5016170000 5018180000 5019200001 5021230000
01         02         03         04         05         06         07

mother       the   small and    old        gave    (past)   yesterday
0334000500 0537101700 0498000301 0376000300 0217000100 8800400000 0617009400
08        09         10         11         12         13         14
```

```
   quickly     books      the        to      father     the     children
0428000400 0071000500 0537102700 0544000200 0194000500 0537101700 0103000500
        15         16          17          18        19          20          21

the    young
0537102700 0619000300
22        23
```

(12) I ask.

```
                I         ask        I            to ask   (present)
9903060001 5004040000 5005060000 8501111800      0040000100 8800500000
    01         02        03          04              05         06
```

(13) I am asked.

```
                I      asked         I            to ask   (present)
9903060001 5040400000 5005060000 8501111800      0040000100 8810500000
    01         02        03          04              05         06
```

(14) The fish eats.

```
                         fish      eats       fish      the       to eat  (present)
9903070001 5004050000 5006070000 0205000500 0537101700 0166010100 8800500000
    01         02        03          04         05         06         07
```

(15) The fish is eaten.

```
                         fish      eaten      fish      the      to eat  (present)
9903070001 5004050000 5006070000 0205000500 0537101700 0166010100 8810500000
    01         02         03         04         05         06         07
```

(16) I wrote.

```
                I       wrote        I            to write   (past)
9903060001 5004040000 5005060000 8501111800      0612000100 8800400000
    01         02         03         04              05         06
```

(17) I was writing.

```
                I      writing       I            to write   (past)
9903060001 5004040000 5005060000 8500111.800     0612000100 8801400000
    01         02         03          04             05         06
```

(18) *I* usually wrote.

```
                I       wrote         I           to write    (past)
9903060001 5004040000 5005060000 8501111800      0612000100 8802400000
    01         02         03         04              05         06
```

(19) I was usually writing.

```
                I       writing      I            to write    (past)
9903060001 5004040000 5005060000 8501111800      0612000100 8803400000
    01          02         03         04             05         06
```

(20) I had eaten.

```
                I       eaten         I          to eat   (past-pant)
[illegible]
```

(21) I have eaten.

```
                I         eaten         I       to eat       (past)
9903060001 5004040000 5005060000 8501111800 0166010100 8800400000
    01         02         03          04         05          06
```

(22) I eat

```
                I          eat          I        to eat      (present)
9903060001 5004040000 5005060000 8501111800 0166010100 8800500000
    01         02         03          04         05          06
```

(23) I shall eat.

```
                I          eat          I        to eat      (future)
9903060001 5004040000 5005050000 8501111800 0166010100 8800600000
    01         02         03          04         05          06
```

(24) I shall have eaten.

```
                I         eaten         I        to eat      (future-future)
9903060001 5004040000 5005060000 8501111800 0166010100 8800700000
    01         02         03          04         05          06
```

(25) Milk arrived.                    [

```
              milk        arrived      milk    no-some-word to arrive      (past)
9903070001 5004050000 5006070000 0332000500 0500011700 0037500100 8800400000
    01         02         03          04         05          06          07
```

(26) Man were working.

```
              man        working       men    no-some-word   to work      (past)
9903070001 5004050000 50060/0000 0311000500 0500012700 0608000100 880140COOO
    01         02         03          04         05          06          07
```

## Part 4. Slunt programs now available.

### Four elementary programs

The following four programs have been written in Slunt Cobol, as described in Part 2 of this paper, and are available. All handle Statement Code 0001 only. Detailed notes of methods used are included at all stages of the programs.

E001 Number Language into English
Input: English dictionary cards in random order
Number Language statements punched into cards
Output: English sentences punched into cards or printed
Contains a verb variation option

E002 English into Number Language
Input: English dictionary cards in random order
English sentences punched into cards Output: Number Language statements punched into cards or printed

E003 English into Number Language
Input: English dictionary cards in sequences which form English sentences Output: Number Language statements punched into cards or printed

G001 Number Language into German
Input: German dictionary cards in random order
Number Language statements punched into cards
Output: German sentences punched into cards or printed
Contains a verb variation option

The verb variation option, if exercised, will first print a translation of the Number Language statement input, and than will translate the same statement in each of the 20 Number Language tenses, both in the affirmative and negative, making 40 sentences in all.

The four programs deal only with Statement Code 0001, the simplest of the Statement Codes, which involves sentences with a subject and a verb, but without any direct or indirect object. There are limitations, however, and not all sentences with Statement Code 0001 can be handled by these programs. These limitations are defined in the. introductory remarks to each program. The limitations were imposed in order to keep the programs simple and to enable them to be released sooner than would have otherwise been possible.

### The purpose of the programs

The four programs have been written with the following objects in mind.

i To demonstrate that Slunt actually works by providing programs which will translate simple English sentences accurately into German.

ii To demonstrate programming methods which are successful. Full explanations of methods are given at all stages in the programs. English programmers can use these or other methods to write programs which will handle more complex English sentences. Programmers who wish to write programs for other spoken languages than English can adapt the programs to allow for the different word order, grammar and Dictionary Design of the other languages.

iii To establish a satisfactory Dictionary Design for English users of Slunt.

iv To provide simple programs in Slunt Cobol which are intelligible even to beginners and which are easily adapted for use on digital computers of any manufacture.

The German program G00l

The German program was written only to provide practical evidence that Slunt is really capable of translating sentences from one spoken language into another. It was written by an English programmer with a very incomplete ; knowledge of the German language, and can be described as experimental. Some of the translations it produces may be incorrect. This is not the fault of Slunt but of errors in the program. A german programmer could easily modify it to produce correct translations. Moreover, the Dictionary Design used is incomplete. Many forms of German words have not been included. Details of the Dictionary Design are given in the introductory notes to program G001.

## The Dictionary Design for English

Details of the Dictionary Design for English used in the three programs E001, E002 and E003 are given below as well as being included in the notes in each of the programs. It is hoped that all Slunt users working in English will use the same Dictionary Design even if they cannot use the same programming language. A uniform Dictionary Design simplifies the distribution of vocabulary. For English one card only is needed for each NLnumber. The arrangements of the card are as follows.

| | |
|---|---|
| Column 1 | Space |
| Columns 2-69 | English words. |

i Nouns. The singular form begins in column 2 and the plural form in column 35.

ii The verb "to be". There are 8 forms of this verb and these are punched to begin in the following columns.

| Verb form | am | is | are | was | were | being | be | been |
|---|---|---|---|---|---|---|---|---|
| Column | 2 | 14 | 18 | 27 | 35 | 42 | 53 | 56 |

iii Adjectival verbs. Verbs like "to be ill" and "to be quick" are adjectival verbs and have only one form. This begins in column 2.

iv Other verbs. All other verbs have 4 or 5 forms. Verbs like "to ask" have 4 forms because "I have asked" has the same forms as "I asked". Verbs like "to see" have 5 forms because "I have seen" has a different form from "I saw". The 4 forms of verbs like "to ask" are punched to begin in the following columns.

| Verb form | ask | asks | asking | asked |
|---|---|---|---|---|
| Column | 2 | 18 | 35 | 53 |

The 5 forms of verbs like "to see" are punched to begin in the following columns.

| Verb form | see | sees | seeing | seen | saw |
|---|---|---|---|---|---|
| Column | 2 | 14 | 27 | 42 | 56 |

v Other words. These usually have only one form which begins in column 2. If there is a second form it begins in column 35. Examples are pronouns like "he" and "him" and the some-word *"a"* and "an".

Column 70 The cardtype. This digit can be 1, 2, 4, 5 or 8 and indicates the number of English words in columns 2-69.

Column 71 Dictionary-item-A. For verbs this is 4 when the first and fourth words on the card are identical, as in "I come" and "I have come" and in "I read" and "I have read". For other verbs it is a space. For adjectives and nouns it is 1 for words like "old" and "hour" which would follow "an" instead of "a" in a sentence. Otherwise it is a space.

Column 72 Dictionary-item-B. Space.

Columns 73-80 The 8-digit Number Language number or NLnumber.

## Apostrophes

Many sentences in the English language include apostrophes. In most cases the apostrophe can be eliminated if desired by rewriting the sentence. Examples are as follows.

| With apostrophe | Without apostrophe |
|---|---|
| I'm coming | I am coming |
| He didn't ask | He did not ask |
| The man's hat is old | The hat of the man is old |
| The boys' shirts are new | The shirts of the boys are new |

There are the following ways of dealing with apostrophes in Slunt.

i Words with apostrophes could be excluded from the dictionary and from the input and output. Any input with an apostrophe would have to be rewritten as shown in the examples above.

ii Words with apostrophes could be excluded from the dictionary but allowed in the input and output. This would involve additional programming to provide for rewriting the input by computer whenever an apostrophe is encountered and for similarly rewriting the output in appropriate circumstances.

iii Words with apostrophes could be introduced into the dictionary and allowed in input and output. This would considerably increase the dictionary for nouns, pronouns and certain verbs, and programs would have to be adjusted.

The first of these three ways has been adopted for the English dictionary, programs, input and output. Apostrophes are eliminated from all of them. This has been done for the sake of simplicity at this early stage. Later on it may be felt that a change is desirable, and the position could then be reviewed.

## ACKNOWLEDGEMENTS

I should like to thank Clerical, Medical & General Life Assurance Society, Liverpool Victoria Friendly Society, Pearl Assurance Company Limited and Wesleyan & General Assurance Society for very kindly providing facilities for the development of Slunt

```
0001 YOU'INFORMAL-SINGULAR= DRESSED TUESDAY.
 9903070001 5004040000 5005070000 8511211800 0157000100 8800400000
 0559009400 0001 HE DID NOT DRESS TUESDAY.
 9903070001 5004040000 5005070000 8501311800 0157000100 8800410000
 05559009400 000l SHE DRESSES.
 9903060001 5004040000 5005060000 8502311800 0157000100 8800500000
0001 WE SHALL DRESS TUESDAY.
 9903070001 5004040000 5005070000 8503121800 0157000100 8800600000
 0559009400 0001 YOU'FORMAL-PLURAL= WILL NOT DRESS.
 9903060001 5004040000 5005060000 8503221800 0157000100 8800610000
0001 YOU'INFORMAL-PLURAL= WILL HAVE DRESSED.
 9903060001 50040040000 5005060000 8513221800 0157000100 8800700000
 0001 THEY WILL HAVE NOT DRESSED.
 9903060001 5004040000 5005060000 8503321800 Ol57000100 8800710000
0001 COLD WINE COMES.
 9903080001 5004060000 5007080000 0598500500 0109000300 0500011700
 0111000100 8800500000 0001
SOME COLD WATER COMES.
 9903080001 5004050000 5007080000 0581000500 0109000300 0500001700
 0111000100 8800500000 0001
SOME COLD DRINKS COME.
 9903080001 5004060000 5007080000 0158000500 0109000300 0500002700
 0111000100 8800500000
0001 MY YOUNG WOMAN COMES QUICKLY,
 9903090001 5004060000 5007090000 0605000500 0619000300 8501111700
 0111000100 8800500000 0428000400
0001 THE YOUNG WOMEN WRITE QUICKLY.
 9903090001 5004060000 5007090000 0605000500 0619000300 0537102700
 0612000100 8800500000 0428000400
0001 THE YOUNG MAN IS QUICK.
 9903080001 5004060000 5007080000 0311000500 0619000300 0537101700
 0428000100 8800500000 0001 THE YOUNG CHILD ARRIVED TUESDAY.
 9903090001 5004060000 5007090000 0103000500 0619000300 0537101700
 0037500100 8800400000 0559009400 0001 THE YOUNG WOMAN HAD USUALLY
BEEN WRITING.
 9903080001 5004060000 5007060000 0605000500 0619000300 0537101700
 0612000100 6803300000 0001 MY YOUNG  CHILDREN WASH QUICKLY.
 9903090001 5004060000 5007090000 0103000500 06l9000300 8501112700
 0578500100 8800500000 0428000400
0001 MY YOUNG CHILD WRITES QUICKLY.
 9903090001 5004060000 5007090000 0103000500 0619000300 8501111700
 0612000100 8800500000 0428000400
OU01 A NEW ANSWER WILL ARRIVE.
 9903080001 5004060000 5007080000 0033000500 0355000300 0000001700
 0037500100 8800600000 0001 A MAN WILL HAVE NOT WASHED.
 9903070001 5004050000 5006070000 0311000500 0000001700 0578500100
 0800710000
```

Figure 1- Printed output from Program E002, which translates English sentences into Number Language statements. The punched card output, produced at the same time, was used as input for Program G001, which translates Number Language statements into German sentences. The printed output from this is shown in Figure 2.

```
9903070001 5004040000 5005070000 6511211800 0157000100 8800400000
0559009400 0001 AM DIENSTAG HAST DU DICH ANGEKLEIDET.
9903070001 5004040000 5005070000 8501311800 0157000100 8800410000
0559009400 0001 AM DIENSTAG HAT ER SICH NICHT ANGEKLEIDET.
9903060001 5004040000 5005060000 8502311800 0157000100 8800500000
0001 SIE KLEIDET SICH AN.
9903070001 5004040000 5005070000 8503121800 0157000100 8800600000
0559009400 0001 AM DIENSTAG WERDEN WIR UNS ANKLEIDEN.
 9903060001 5004040000 5005060000 8503221800 0157000100 8800610000
0001 SIE WERDEN SICH NICHT ANKLEIDEN.
 9903060001 5004040000 5005060000 8513221800 0157000100 8800700000
0001 IHR WERDET EUCH ANGEKLEIDET HABEN.
 9903060001 5004040000 5005060000 8503321800 0157000100 8800710000
0001 SIE WERDEN SICH NICHT ANGEKLEIDET HABEN.
 9903080001 5004060000 5007080000 0598500500 0109000300 0500011700
 0111000100 8800500000
 0001 KALTER WEIN KOMMT.
 9903080001 5004060000 5007080000 0581000500 0109000300 0500001700
 0111000100 8800500000
0001 ETWAS KALTES WASSER KOMMT.
 9903080001  5004060000  5007080000  0158000500  0109000300  0500002700
 0111000100  8800500000 0001  EINIGE
KALTE  GETRAENKE  KOMMEN.
 9903090001  5004060000  5007090000  0605000500  0619000300  8501111700
 0111000100  8800500000  0428000400 0001
MEINE  JUNGE  FRAU  KOMMT  SCHNELL.
 9903090001  5004060000  5007090000  060500G500  0619000300  0537102700
 0612000100  8800500000  0428000400 0001  DIE
JUNGEN  FRAUEN  SCHREIBEN  SCHNELL.
 9903080001  5004060000  5007080000  0311000500  0619000300  0537101700
 0428000100  8800500000 0001  DER
JUNGE  MANN  IST  SCHNELL.
9903090001  5004060000  5007090000  0103000500  0619000300  0537101700
003 7500 100  8800400000  0559009400 0001 AM DIENSTAG  IST  DAS  JUNGE  KIND
ANGEKOMMEN.
 9903080001  5004060000  5007080000  0605000500  0619000300  0537101700
 0612000100  8803300000
0001  DIE  JUNGE  FRAU  WAR  GEWOEHNLICH  BEIM  SCHREIBEN  GEWESEN.'
9903090001  5004060000  5007090000  0103000500  0619000300  8501112700
 0578500100  8800500000  0428000400 0001  MEINE
JUNGEN  KINDER  WASCHEN  SICH  SCHNELL.
 9903090001  5004060000  5007090000  0103000500  0619000300  8501111700
 0612000100  8800500000  0428000400  0001
MEIN  JUNGES  KIND  SCHREIBT  SCHNELL.
 9903080001  5004060000  5007080000  0033000500  0355000300  0000001700
 0037500100  8800600000 0001  EINE  NEUE
ANTWORT  WIRD ANKOMMEN.
 9903070001  5004050000  5006070000  0311000500  0000001700  0578500100
 8800710000 0001  E1N  MANN  WIRD SICH  NICHT
GEWASCHEN  HABEN.
```

Figure 2. Printed output from Program G001, which translates Number Language statements into German sentences. The Number Language statements input were the. punched card output produced at the same time as Figure 1. Figures 1 and *2* taken together illustrate an example of English sentences being translated into German sentences by means of Slunt. German suntences produced by Program G001 may not be correct grammatically. .Any mistakes are due to programming errors which could