[*International Conference on the State of Machine Translation in America, Asia and Europe*. Proceedings of IAI-MT86, 20-22 August 1986, Bürgerhaus, Dudweiler]

Peter Hellwig

Program for Language Analysis and Inference (PLAIN)

Germanistisches Seminar
Universität Heidelberg
Hauptstraße 207-209
6900 Heidelberg 1

PROGRAMS FOR LANGUAGE ANALYSIS AND INFERENCE (PLAIN) A WORKBENCH FOR MACHINE TRANSLATION

Peter Hellwig

0. Data about the system

Name: PLAIN.
Type: knowledge based system for natural language analysis and processing.
Components: analysis, deduction, transfer.
Data bases for MT: morphological lexicon, valency lexicon, transfer rules.
Implementation language: PL/1.
Storage requirements: 384 KB maximum.
Operation: dialog and batch.
Hardware and operating systems: IBM/MVS, Siemens/BS 2000.

1 • Characteristics

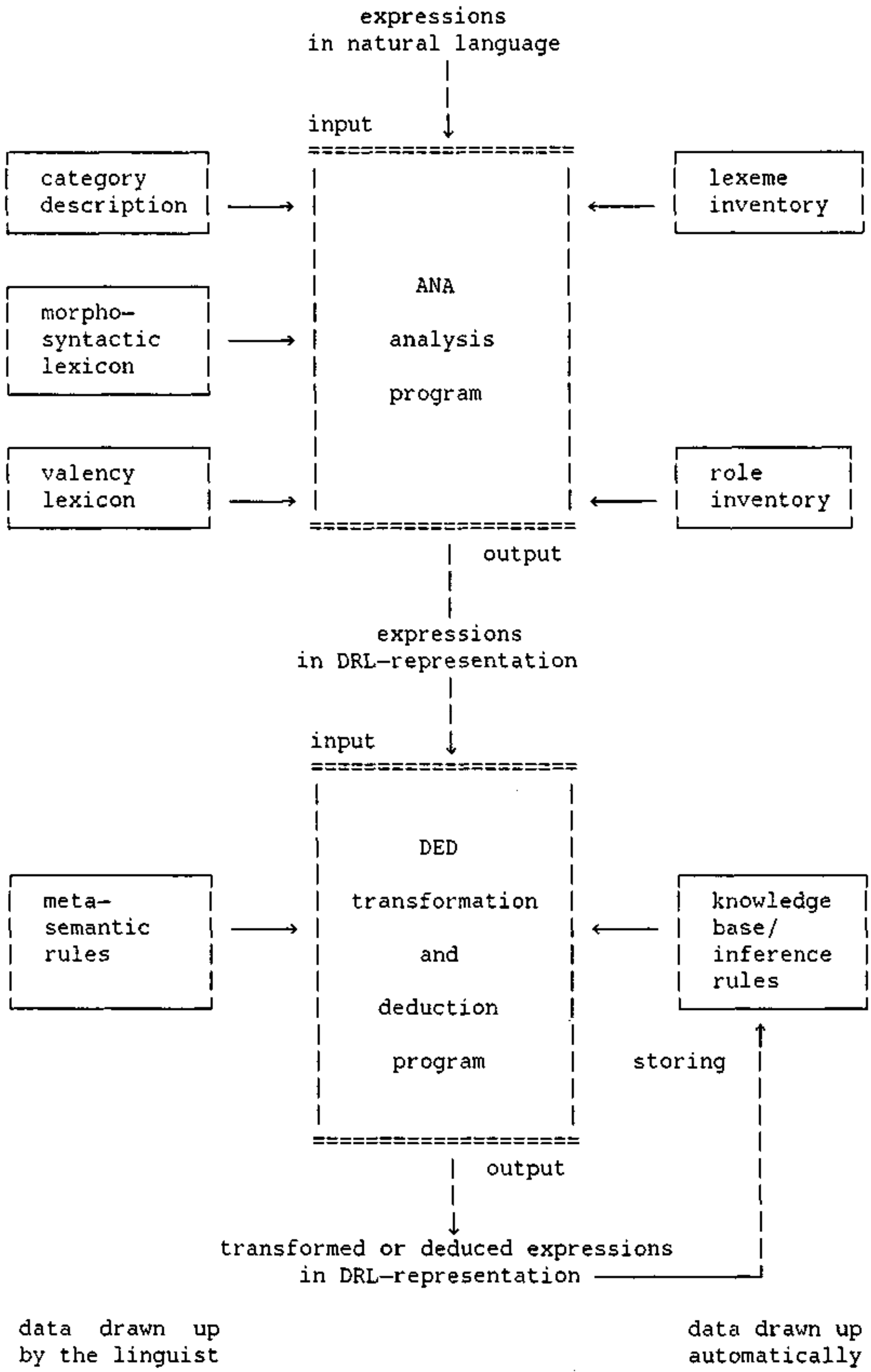
PLAIN ("Programs for Language Analysis and Inference") is a comprehensive system for the analysis and processing of natural languages. It has the following features:

(1) PLAIN is based on a theory of grammar called "Dependency Unification Grammar" (DUG) resulting in a new parsing device. DUG incorporates many ideas from traditional linguistics. At the same time, it fits with the recent developments in grammar theory. It shares with other modern formalisms the concept of "governing", the use of complex categories, the orientation towards syntactic functions, the lexicalistic approach to grammar, the unification method as the basis of the parsing algorithm. The unique feature of DUG is its augmented dependency structure which results in the parallel treatment of three linguistic dimensions: the individual lexical meanings, the grammatical functions, and the outward form.

(2) PLAIN uses a special language for representing linguistic facts called "Dependency Representation Language" (DRL). DRL is a language ideally suited for the linguistic domain. Individual grammars can be programmed in it in declarative form. The design of DRL - independent of its currently being embedded in PLAIN - is an alternative to the other languages used in computational linguistics, e.g. LISP and PROLOG.

(3) PLAIN is a package of programs for developing linguistic software. Its architecture has been designed with the linguist in mind who is drawing up a DUG for an individual language. In addition, PLAIN is a knowledge-based system that supplies a shell for prototypes of translation systems and natural language expert systems. Commands, options and parameters offer a maximum of comfort, transparency and control during the processing of large volumes of linguistic data. PLAIN is implemented at several universities in Germany and is also available at the University of Cambridge, U.K. It can be provided to other research groups for experimental use.

2- Survey of the main components of PLAIN



The two main components of PLAIN are the programs ANA and DED. ANA analyses natural language input and converts it into expressions of DRL (Dependency Representation Language). For this purpose the following data on the input language is needed: a definition of the morpho-syntactic categories used, a morpho-syntactic lexicon, and a valency lexicon. DED transfers DRL expressions according to given rules into new DRL expressions. It is suited to cover essential functions within automatic translation, theorem proving and deductive question answering. So-called meta-semantic rules determine the logical form of the input; these rules result in DRL expressions that are themselves rules according to which DED subsequently carries out deductions. The meta-semantic rules also include translation rules which convert DRL representations of one language into DRL representations of another language. The data sets on the left of the diagram have to be compiled by linguists for the particular input languages. Taken together they form a complete single-language or bilingual lexicon with a specification of morphology, syntax and semantics. The data sets on the right of the diagram result from the actual use of PLAIN. They are built up automatically. The lexeme and role inventory is created by any lexeme and role names being used in DRL expressions. The knowledge base is set up as follows: any statements on an object domain are entered, analysed by ANA, transformed by DED and finally stored in a database. In this process, general statements on the object domain are converted into inference rules. In addition to the components shown, PLAIN has further programs to define categories and other special symbols (DFN), to update the lexicon (LEX), to organise network-like databases (UBS) and to make various print-outs (DFN, LEX, PBS).

Below, a demonstration of some of PLAIN'S components is given. The listings are the same as the output which appears on the screen.

3. Analysis Example

The example shows the analysis of the sentence "Sheila persuades Arthur to attend the conference on machine translation." The result is a list (in the sense of list-processing) or a "tree" representing the linguistic structure of the sentence. The parentheses as well as the indentation show the dependency relationships of the elements. Each element with all of the elements it dominates, i.e. everything that is between two complementary parentheses, forms a partial tree. The composition of a list by partial trees implicitly reflects the constituent structure of the input phrase. Every list element contains a role marker, a lexeme and a complex category, separated by a colon. In the example the role markers have capital letters. Lexemes follow in the second position. The sentence semantics is also represented by lexemes, cf. the lexeme "assertion". The third position is for morpho-syntactic categories which consist of a main category (generally a word class) and a sequence of subcategories representing grammatical features. The notation of subcategories comprises a name (such as, for example, "num" for number) and so-called values in parentheses (in the case of "num", for instance, "1" for singular and "2" for plural). Morpho-syntactic categories include positional features, such as adjacency ("adj") and sequence ("seq").

```
*** ANALY - INPUT FOR ANALYSIS:

'SHEILA PERSUADES ARTHUR TO ATTEND THE CONFERENCE ON MACHINE
TRANSLATION. '


*** ANALY - RESULT OF ANALYSIS:

LIST  58:

 1   (ILLOCUTION: assertion: clse typ<l>
 2         (PREDICATE: persuade: verb fin<l> adj<l>
 3               (SUBJECT: sheila: noun per<3,C> num<l,C> adj<l>)
 4               (OBJECT: arthur: noun adj<2>)
 5               (OBJECOMPL: attend: verb fin<2> seq<2>
 6                     (OBJECT: conference: noun adj<2>
 7                           (DETERMINATION: the: dete seq<l>)
 8                           (PREPCOMPL: on: prep adj<2>
 9                                 (: translation: noun adj<2>
10                                       (ATTRIBUTE: machine: noun
                                         adj<l>)))))));

*** ANA - STATISTICS:

      WORDS=  11, LISTS BUILT BY LEXICON LOOKUP       =  12
      AVERAGE  READINGS  PER WORD  (HOMONYMY)    = 1.09
      AVERAGE MORPHEMES  PER WORD (COMPLEXITY)= 1.00
      SLOTS  EXAMINED=    150,  DERIVED  LISTS             =  58
      DERIVED  LISTS / AV. MINIMUM (EFFICIENCY)        = 5.80
      RESTARTS (DISCONTINUITY)=  0, CPU-SECONDS      = 1.11
```

4. Representation for further processing

This is the analysis result without the morpho-syntactic categories. After all, the latter describe surface features of the input languages. After the analysis has been completed, they can be removed.

```
 1  (ILLOCUTION: assertion
 2        (PREDICATE: persuade
 3              (SUBJECT: sheila)
 4              (OBJECT: arthur)
 5              (OBJECOMPL: attend
 6                    (OBJECT: conference
 7                          (DETERMINATION: the)
 8                          (PREPCOMPL: on
 9                                (: translation
10                                      (ATTRIBUTE: machine)))))));
```

5. Definition of the morpho-syntactic categories

Categories must be defined before they can be used in terms. In particular, for each subcategory the number of values and the operation which is to be responsible for processing the values must be specified. At present fourteen operations are available which correspond to a typology of the categories which linguists have used in formal and non-formal grammars and dictionaries. The category definition which be was used for the processing the above example is the following:

clse	typ 1 2					
verb	fin 1 3	per	1 3	num 1 2	seq 11 2	adj 12 2
noun	gen 1 3	per	1 3	num 1 2	seq 11 2	adj 12 2
dete	seq 11 2	adj	12 2			
prep	seq 11 2	adj	12 2			

The main categories are clse (clause), verb, noun, dete (determiner), prep (preposition). The explanation of the subcategories is as follows:

typ = type of clause - operation 1 (Boolean)
values : 1 main, 2 subordinate
fin = finiteness - operation 1 (Boolean)
values : 1 finite, 2 infinitive,
3 past participle
per = person - operation 1 (Boolean)
values : 1 first, 2 second, 3 third
num = number - operation 1 (Boolean)
values : 1 singular, 2 plural
gen = gender - operation 1 (Boolean)
values : 1 masculine, 2 feminine, 3 neuter
seq = sequence - operation 11 (positional)
values : 1 preceding, 2 following
adj = adjacency - operation 12 (positional)

values : 1 left-adjacent, 2 right-adjacent

6. Morpho-syntactic lexicon

At the beginning of the analysis, the input character string is broken down into basic segments each of which is assigned one or more DRL elements with a lexeme and a morpho-syntactic category. This happens by consulting the morpho-syntactic lexicon. The lexicon is arranged in so-called sections, especially into sections for stems and for endings. References lead from one section to another which corresponds to the possible continuation of the character strings in the words. A backtracking mechanism makes sure that all of the possible decompositions, including those of compounds, are found. For example, the German equivalent for "to attend", the word 'TEILNEHMEN ', is processed according to the part of the lexicon printed here as follows. First the segment 'TEIL' is found in the section of the stems; a DRL term is formed for this segment with the lexeme "teil". Section 128 is referred to for continuation. In Section 128 under ' ', i.e. without another character being added, the category "prfx kmp<l>" is found, i.e. prefix in compound. Since there is no reference to another section, the analysis of the first word component 'TEIL' ends here. Now the section of the stems is tried again and 'NEHM' is found. After a new DRL element "(*:nehmen)" has been created, the processing continues in section 105. This section contains the endings of the infinitive and of the present tense of those forms which do not change the stem (while section 107, which is also listed, contains the endings corresponding to stems with Ablaut, e.g. 'NIMM'). The end of the word is reached with the ending 'EN ' (which occurs twice). The terms and categories associated with the ending are added to the DRL representation for the segment 'NEHMEN '.

*** PUTLX - LEXICON SECTION NO. 3:
Testvokabular aus "Bausteine des Deutschen"

'TEIL'
(*: teil);
SECTION 128

'NEHM'
(*: nehmen);
SECTION 105
SECTION 107

*** PUTLX - LEXICON SECTION NO. 128:
Abtrennbare Praefixe des Verbs

' '

prfx kmp<l>

' '

prfx

*** PUTLX - LEXICON SECTION NO. 105:
Verben -
Infinitiv, Praesens Indikativ ohne 2. und 3. Person Singular
(d.s.die nicht umlautenden Formen bei Umlaut), Imperativ Plural

---

' '
verb kmp<l>

'E'
verb kmp<l>

'E '
((TEMPS: praesens')(MODUS: indikativ'));
verb fin<l> num<l> per<l> tmp<l>

'EN '
((TEMPS: praesens')(MODUS: indikativ'));
verb fin<l> num<2> per<l,3> tmp<l>

'EN '
verb fin<2>

' T '
((TEMPS: praesens')(MODUS: indikativ'));
verb fin<l> num<2> per<2> tmp<l>

' T '
((MODUS : imperativ_pl')) ;
verb fin<4> num<2> per<2>

*** PUTLX - LEXICON SECTION NO. 107:
Verben -

Praesens Indikativ 2. und 3. Person Singular
(d.s. die umlautenden Formen bei Umlaut)

'ST '
((TEMPS: praesens')(MODUS: indikativ'))•
verb fin<l> num<l> per<2> tmp<l>

'T '
((TEMPS: praesens')(MODUS: indikativ'));
verb fin<l> num<l> per<3> tmp<l>

7. Updating the morpho-syntactic lexicon by paradigm

In order to enlarge the morpho-syntactic lexicon, one can enter a paradigm of inflectional and derivative forms by which PLAIN recognizes the morphology of the word and then carries out an appropriate entry in the lexicon. The prerequisite for this is the presence of patterns which indicate both the expected form of the input as well as the ending sections with which the stems of the words are to be linked. In the example, the input format for the verb "NEHMEN" is documented and so is the pattern that fits this paradigm. The pattern indicates that the word stem "NEHM" is to be linked to section 105, the stem 'NIMM' to section 107, the stem 'NAHM' to section 118, which is the past tense in the indicative, the stem 'NAEHM' to section 123, which is the past conjunctive, and the stem 'NOMM' to section 126, which contains the ending of the past participle. The other patterns shown are part of the total number of 230 paradigms which completely account for the German morphology. The underscore in the patterns always stands for the stem of the basic form. Included in the parentheses, the stem changes from Ablaut and Umlaut are specified. The device does not only serve to simplify drawing up the lexicon. The patterns also represent a theoretically important systematization of the morphology, since they show the relationships of the various forms.

Input:

NEHMEN NIMMST NAMST NAEMEST GENOMMEN

Pattern matching with this input:

_EN |105| _(EHM/IMM)ST |107| _(EH/A)ST |118| <_(EH/AE)EST |123|>
_(EHM/OMM)EN |126| ;

Examples of other patterns (a noun and an adjective):

DER _ |132| _ES _(+E)ER |168|;
(DER WALD WALDES WAELDER)

|195| <_(+E)ER AM _(+E)STEN |193|> <IST _ |190|> <DAS _E |191|> ;
(JUNG JUENGER AM JUENGSTEN IST JUNG DAS JUNGE)

8. Valency lexicon

In a dependency representation the syntactic relationships between the basic elements of a language are represented directly. This corresponds to the valency principle which says that the structure of complex expressions are determined by the combination capability of the basic elements. According to this approach, the syntax of a language is a matter of a lexicon in which the combination capabilities of the language elements are described. A possible way to formulate the combination capabilities is to assign slots to each lexeme for all of the elements which it is to be superordinated to in the structural tree, thereby also taking into account the morpho-syntactic categorization. In the DRL formalism, slots are represented by list elements with a variable in the place of the lexeme. The classes of elements which are allowed to fill a slot can be delimited as precisely as necessary by morpho-syntactic categorization. The slots themselves are made functionally identifiable by a role marker. If the same complement can go with different lexemes, a completion pattern is set up and references are made to this from the individual lexicon entries. Apparently this way of describing the syntactic relationships is in agreement with the "frame" approach advocated by Artificial Intelligence. The following completion patterns and valency references suffice for the syntactic analysis of the above example.

Completion patterns:

```
(*: +subject: verb fin<l>
        (SUBJECT: = : noun per<C> num<C> adj<l>));

(*: +object: verb
        (OBJECT: _ : noun adj<2>));

(*: +object_complement: verb
        (OBJECOMPL: _ to_infinitival: verb fin<2> seq<2>));

(: +countable
        (DETERMINATION: _ : dete seq<l>));

(: +attribute: noun
        (ATTRIBUTE: _ : noun adj<l>));

(ILLOCUTION: assertion: clse typ<l>
        (PREDICATE: - : verb fin<l> adj<l>));
```

Valency references:

```
(: ->  (*: persuade)            (& (: +subject)
                                   ( : +object)
                                   (: +object_complement)));

(: ->  (*: to_infinitival)         (*: = : verb fin<2> adj<2>));

(: ->  (*: attend)              (& (: +subject)
                                   (: +object)));

(: ->  (*: conference)          (& (: +countable)
                                   (: +attribute)));

(: ->  (*: translation)         (& (: +countable)
                                   ( : +attribute)));
```

9. The parsing process

After the base lexicon has been consulted and the terms have been augmented by slots from the valency lexicon the following lists exist. (We disregard grammatical ambiguities which results in more than one list being associated with one word.)

```
*** ANALY - HISTORY OF LIST  58:

LIST   1 (BUILT BY LEXICON LOOKUP):
'SHEILA '
MORPHEM(S): 1, WORD(S): 1

   1  (*: sheila: noun gen<2> per<3> num<l>);
```

```
LIST   2 (BUILT BY LEXICON LOOKUP):
'PERSUADES '
MORPHEM(S): 1, WORD(S): 01

   1    (*: persuade: verb fin<l> per<3> num<l>
   2          (SUBJECT: = <SLOTOBL>: noun per<C> num<C> adj<l>)
   3          (OBJECT: _ <SLOT>: noun adj<2>)
   4          (OBJECOMPL: _ to_infinitival <SLOT> : verb fin<2>
              seq<2>));
```

```
LIST   3 (BUILT BY LEXICON LOOKUP):
'ARTHUR '
MORPHEM(S): 1, WORD(S): 001

   1  (*: arthur: noun gen<l> per<3> num<l>);
```

```
LIST   4 (BUILT BY LEXICON LOOKUP):
'TO '
MORPHEM(S): 1, WORD(S): 0001

   1    (*: = <SLOTOBL>: verb fin<2> adj<2>);

SATISFYING: to_infinitival;

- LOOKING FOR SLOTS WAS SKIPPED FOR THIS LIST -
```

```
LIST   5 (BUILT BY LEXICON LOOKUP):

 'ATTEND '

MORPHEM(S): 1, WORD(S): 00001

   1    (*: attend: verb fin<2>
   2          (OBJECT: _ <SLOT>: noun adj<2>));
```

```
LIST   6 (BUILT BY LEXICON LOOKUP):
 'THE '
MORPHEM(S): 1, WORD(S): 000001

   1  (*: the: dete);
```

```
LIST   8 (BUILT BY LEXICON LOOKUP) :
'CONFERENCE '
MORPHEM(S): 1, WORD(S): 0000001

(*: conference: noun per<3> num<l>
        (DETERMINATION: _ <SLOT> : dete seq<l>)
        (PREPCOMPL: _ ? <SLOT>: prep adj<2>
                (: ? on)));
```

```
LIST   9 (BUILT BY LEXICON LOOKUP):
'ON '
MORPHEM(S): 1, WORD(S): 00000001

(*: on: prep
        (: = <SLOTOBL>: noun adj<2>));

SATISFYING: local' ;
```

```
LIST  10 (BUILT BY LEXICON LOOKUP):
'MACHINE '
MORPHEM(S): 1, WORD(S): 000000001

(*: machine: noun per<3> num<l>);
```

```
LIST  11 (BUILT BY LEXICON LOOKUP):

'TRANSLATION'
MORPHEM(S): 1, WORD(S): 0000000001

(*: translation: noun per<3> num<l>
        (DETERMINATION: _ <SLOT> : dete seq<l>)
        (ATTRIBUTE: _ <SLOT>: noun adj<l>));
```

```
LIST  12 (BUILT BY LEXICON LOOKUP):
' . '

MORPHEM(S): 1, WORD(S): 00000000001
(ILLOCUTION: assertion: clse typ<l>
        (PREDICATE: = <SLOTOBL> : verb fin<l> adj<l>));
```

Once the lexical terms; are associated with completion patterns the parsing process is fairly simple. Every list, beginning with the first one, looks for a slot in another list. All the variables and category values of filler lists and slot lists are unified. If the unification is successful, then a new list is formed by insertion of the filler list in the slot. The new list is added to the end of the line. When its turn comes, this list also looks for a slot, and so on until no further insertions of filler lists in slots are possible. Parsing has been successful when a list has been constructed which comprises a term for every segment of the input string. The following is the record of this procedure for the above example:

```
LIST  13 (DERIVED FROM LIST   1 AND LIST  2):
'SHEILA PERSUADES '
MORPHEM(S): 1, WORD(S): 11

  1    (*: persuade: verb fin<l> per<3> num<l>
  2          (SUBJECT: sheila: noun per<3,C> num<l,C> adj<l>)
  3          (OBJECT: _ <SLOT>: noun adj<2>)
  4          (OBJECOMPL: _ to_infinitival <SLOT> : verb fin<2>
             seq<2>));
```

```
LIST  15 (DERIVED FROM LIST   3 AND LIST  13):
'SHEILA PERSUADES ARTHUR '
MORPHEM(S): 1, WORD(S): 111

  1    (*: persuade: verb fin<l> per<3> num<l>
  2          (SUBJECT: sheila: noun per<3,C> num<l,C> adj<l>)
  3          (OBJECT: arthur: noun adj<2>)
  4          (OBJECOMPL: _ to_infinitival <SLOT> : verb fin<2>
             seq<2>));
```

```
LIST  16 (DERIVED FROM LIST   5 AND LIST  4):
'TO ATTEND '
MORPHEM(S): 1, WORD(S): 00011

  1    (*: attend: verb fin<2>
  2          (OBJECT: _ <SLOT>: noun adj<2>));

SATISFYING: to_infinitival;
```

```
LIST  17 (DERIVED FROM LIST   6 AND LIST  8):
'THE CONFERENCE '
MORPHEM(S): 1, WORD(S): 0000011

  1    (*: conference: noun per<3> num<l>
  2          (DETERMINATION: the: dete seq<l>)
  3          (PREPCOMPL: _ ? <SLOT>: prep adj<2>
  4                (: ? on)));
```

```
LIST  19 (DERIVED FROM LIST  10 AND LIST  11):
'MACHINE TRANSLATION'
MORPHEM(S): 1, WORD(S): 0000000011

  1    (*: translation: noun per<3> num<l>
  2          (DETERMINATION: _ <SLOT>: dete seq<l>)
  3          (ATTRIBUTE: machine: noun adj<l>));
```

```
LIST  26 (DERIVED FROM LIST  19 AND LIST  9):
'ON MACHINE TRANSLATION'
MORPHEM(S): 1, WORD(S): 0000000111

  1    (*: on: prep
  2          (: translation: noun adj<2>
  3                (ATTRIBUTE: machine: noun adj<l>)));

SATISFYING: local';
```

LIST 35 (DERIVED FROM LIST 26 AND LIST 17):
'THE CONFERENCE ON MACHINE TRANSLATION'
MORPHEM(S): 1, WORD(S): 0000011111

```
1  (*: conference: noun per<3> num<l>
2          (DETERMINATION: the: dete seq<l>)
3          (PREPCOMPL: on: prep adj<2>
4                  (: translation: noun adj<2>
5                          (ATTRIBUTE: machine: noun adj<l>))));
```

LIST 44 (DERIVED FROM LIST 35 AND LIST 16):
'TO ATTEND THE CONFERENCE ON MACHINE TRANSLATION'
MORPHEM(S): 1, WORD(S): 0001111111

```
1  (*: attend: verb fin<2>
2          (OBJECT: conference: noun adj<2>
3                  (DETERMINATION: the: dete seq<l>)
4                  (PREPCOMPL: on: prep adj<2>
5                          (: translation: noun adj<2>
6                                  (ATTRIBUTE: machine: noun adj<l>)))));
```

SATISFYING: to_infinitival;

LIST 52 (DERIVED FROM LIST 44 AND LIST 15):
'SHEILA PERSUADES ARTHUR TO ATTEND THE CONFERENCE ON MACHINE
TRANSLATION'
MORPHEM(S): 1, WORD(S): 1111111111

```
1  (*: persuade: verb fin<l> per<3> num<l>
2          (SUBJECT: sheila: noun per<3,C> num<l,C> adj<l>)
3          (OBJECT: arthur: noun adj<2>)
4          (OBJECOMPL: attend: verb fin<2> seq<2>
5                  (OBJECT: conference: noun adj<2>
6                          (DETERMINATION: the: dete seq<l>)
7                          (PREPCOMPL: on: prep adj<2>
8                                  (: translation: noun adj<2>
9                                          (ATTRIBUTE: machine: noun
                                           adj<l>))))));
```

LIST 58 (DERIVED FROM LIST 52 AND LIST 12):
'SHEILA PERSUADES ARTHUR TO ATTEND THE CONFERENCE ON MACHINE
TRANSLATION. '
MORPHEM(S): 1, WORD(S): 11111111111

```
 1  (ILLOCUTION: assertion: clse typ<l>
 2          (PREDICATE: persuade: verb fin<l> adj<l>
 3                  (SUBJECT: sheila: noun per<3,C> num<l,C> adj<l>)
 4                  (OBJECT: arthur: noun adj<2>)
 5                  (OBJECOMPL: attend: verb fin<2> seq<2>
 6                          (OBJECT: conference: noun adj<2>
 7                                  (DETERMINATION: the: dete seq<l>)
 8                                  (PREPCOMPL: on: prep adj<2>
 9                                          (: translation: noun adj<2>
10                                                  (ATTRIBUTE: machine: noun
                                                    adj<l>)))))));
```

10. Translation German - English

After the analysis is finished, the morpho-syntactic categories are removed from the lists turning DRL into a semantic representation language which consists of terms with roles and lexemes, either constant or variable, and of variables covering lists and sequences of lists. DRL expressions of this type are processed by PLAIN'S deduction component DED. The following listings illustrate the application of DED in the framework of machine translation. The DRL representation of two German sentences with the ambiguous verb "erinnern", one with the reading "remind" the other one with the reading "remember", is transformed here into a DRL representation for English sentences. (Within the framework of an automatic translation system, a generation component would have to make English sentences out of these lists. Such a component has not yet been implemented).

*** DED - LIST_1:

(Du erinnerst mich an meine Freundin.)

```
(ILLOC: aussage'
        (PRAED: erinnern
                (TEMPS:   praesens')
                (MODUS:   indikativ')
                (SUBJE:adressat_sg')
                (TRANS:   sprecher_sg')
                (CASPP:   an
                        (: freundin
                                (REFER: definit')
                                (NUMRS: singular')
                                (ZUORD: sprecher_sg')))));
```

*** REPLC - LIST AFTER REPLACEMENT

(You remind me of my girl friend.)

```
(ILLOC: assertion'
        (PRAED: remind
                (TEMPS:   praesens')
                (MODUS:   indikativ')
                (SUBJE:you)
                (TRANS:   me)
                (CASPP:   of
                        (: girl friend
                                (REFER: definit' )
                                (NUMRS: singular')
                                (ASSOC: my)))));
```

```
*** DED - LIST_1:
(Ich erinnere mich an meine Freundin.)

(ILLOC: aussage'
        (PRAED: erinnern
                (TEMPS: praesens')
                (MODUS: indikativ')
                (: reflexiv')
                (SUBJE: sprecher_sg')
                (CASPP: an
                        ( : freundin
                                (REFER: definit')
                                (NUMRS: singular')
                                (ZUORD: sprecher_sg')))));

*** REPLC - LIST AFTER REPLACEMENT
(I remember my girl friend.)

(ILLOC: assertion'
        (PRAED: remember
                (TEMPS: praesens')
                (MODUS:     indikativ')
                (SUBJE:  I)
                (TRANS: girl friend

                        (REFER: definit')
                        (NUMRS: singular')
                        (ASSOC: my))));
```

11. The translation rules used

These are the rules used for the transformation in 10. Each rule is made up of a rule symbol (in the first list element). In the case here, we are dealing with replacement rules. Every rule is made up of two partial trees, subordinated to the rule symbol. The first tree *is a* pattern for an initial structure, the second one is a pattern for a target structure. Roles, lexemes, and entire partial trees can be substituted in rules by variables. When the rules are applied, variables in the pattern for the target structure are replaced by precisely the elements or trees that corresponded to the same variables in the initial structure. Variables allow for sufficient generalization, roles and lexemes for sufficient restriction of the rule applications.

```
(: D-E= => <REPLACE>
        (SUBJE: sprecher_sg')
        (SUBJE: I));

(: D-E= => <REPLACE>
        (ZUORD: sprecher_sg')
        (ASSOC: my));

(: D-E= => <REPLACE>
        (*: freundin
                (-))
        (*: girl friend
                (-)));
```

```
(: D-E==> <REPLACE>
          (TRANS: sprecher_sg')
          (TRANS: me));
```

```
(: D-E==> <REPLACE>
          (SUBJE: adressat_sg')
          (SUBJE: you));
```

```
( : D-E"> <REPLACE>
          (*: erinnern
                    (: reflexiv')
                    (CASPP: an
                              (: $1))
                    (-1))
          (*: remember
                    (-1)
                    (TRANS: $1)));
```

```
(: D-E==> <REPLACE>
          (*: erinnern
                    (TRANS: $1)
                    (CASPP: an
                              (-1))
                    (-2))
          (*: remind
                    (-2)
                    (TRANS: $1)
                    (CASPP: of
                              (-D)));
```

```
(: D-E==> <REPLACE>
          (*: aussage'
                    (-))
          (*: assertion'
                    (-)));
```

12. Documentation of part of the transformation of 11

In DRL one can easily write rules according to which every DRL expression can be freely transformed into almost any other one. The structure of the target expression can be completely different from that of the initial expression. The following printout shows the phase of the translation in 11 in which the German reflexive verb "erinnern" with a prepositional object is transformed into the English verb "remember" with a direct object. The correspondence of variables and elements is kept in a register (see "LIST OF VARIABLES").

```
*** REPLC - LIST AFTER REPLACEMENT:

(ILLOC: aussage'
          (PRAED: erinnern
                    (TEMPS : praesens')
                    (MODUS: indikativ')
                    (: reflexiv')
                    (SUBJE: I)
                    (CASPP: an
                              ( : girl friend
                                        (REFER: definit')
                                        (NUMRS: singular')
                                        (ASSOC: my)))));
```

```
*** REPLC - RULE USED FOR REPLACEMENT:

    1   ( : D-E==> <REPLACE>
    2           (*: erinnern
    3                   (: reflexiv')
    4                   (CASPP: an
    5                           (= $1))
    6                   (-1))
    7           (*: remember
    8                   (-1)
    9                   (TRANS: $1)));

*** PVLST - LIST OF VARIABLES:

        TYPE:  11   NAME: *     REFERS TO:   2
        TYPE:  12   NAME: -1    REFERS TO:   3
        TYPE:  12   NAME: -1    REFERS TO:   4
        TYPE:  12   NAME: -1    REFERS TO:   6
        TYPE:  22   NAME: $1    REFERS TO:   8

*** REPLC - LIST (  2) REPLACED ACCORDING TO RULE (  7)
```

```
*** REPLC - LIST AFTER REPLACEMENT:

    1   (ILLOC: aussage'
    2           (PRAED: remember
    3                   (TEMPS:  praesens')
    4                   (MODUS: indikativ')
    5                   (SUBJE:   I)
    6                   (TRANS:  girl friend

    7                           (REFER: definit')
    8                           (NUMRS: singular')
    9                           (ASSOC: my))));
```

12. Reports on PLAIN

Peter Hellwig, University of Heidelberg

Formal-desambiguierte Repraesentation. Vorueberlegungen zur maschinellen Bedeutungsanalyse auf der Grundlage der Valenzidee. Stuttgart 1978.

"PLAIN - A Program System for Dependency Analysis and for Simulating Natural Language Inference." In: Leonard Bolc (Hg.): Representation and Processing of Natural Language. Muenchen, Wien, London 1980, pp. 271 - 376.

Programmsystem PLAIN - "Programs for Language Analysis and Inference" - Benutzungsanleitung. Germanistisches Seminar der Universitaet Heidelberg, Postfach 105760, D-69 Heidelberg. (PLAIN User's Guide)

"Bausteine des Deutschen". Daten fuer das Programmsystem PLAIN. Germanistisches Seminar der Universitaet Heidelberg, Postfach 105760, D-69 Heidelberg.

Program System PLAIN (Programs for Language Analysis and and Inference) - Examples of Application. Computing Unit and Linguistics and International Studies Department, University of Surrey, Guildford/U.K. October 1985.

"The PLAIN Approach to Natural Processing". In the Proceedings of the 1986 Spring Meeting of the SHARE European Association(SEAS).

"Dependency Unification Grammar (DUG)", 1986. To appear in the proceedings of COLING 1986.