The Linguistic Component of METAL

Winfield S. Bennett

Working Paper LRC-82-2
July, 1982

ABSTRACT

This paper presents an overview of the linguistic component of the METAL machine translation system, currently under development at the Linguistics Research Center. A brief description of the monolingual and bilingual lexicons is given, followed by descriptions of grammar rule composition and system operation with regard to the grammar and lexicons.

Table of Contents

1 Introduction

The linguistic component of the METAL system consists of the lexicons and grammar used to translate technical texts. At present, METAL translates from German to English; however, research has already begun for translation using other languages.

This paper presents an overview of the form and function of the METAL linguistic component. Specific details about coding lexical items, writing grammar rules and transformations, etc. are given in the documentation for the system.

2 Lexicons

METAL lexicons are either monolingual or bilingual. There are separate monolingual lexicons for both German and English, each stored in a computer database.

The lexical databases contain German and English entries for:

- function words, i.e., words which are necessary regardless of the type of text to be translated, e.g., prepositions, conjunctions, pronouns, and determiners;

- general words, i.e., those words which may occur in any variety of texts;

- technical words, i.e., those words needed for specific technical texts;

- prefixes, infixes, and inflectional endings; and

- punctuation, symbols, etc.

2.1 German Monolingual Lexicons

Monolingual lexical entries include the canonical form (i.e., a "dictionary" form by which the METAL recognizes the lexical item), followed by the features and values used by the system in analysis or generation. The database entries for the German verb <u>gehen</u> will serve as examples:

```
(gehen          CAT (VST)
   ALO   (gang)
   PLC   (NF NI)
   TAG   (ALL)
   SNS   (1)
   PX    (NIL)
   ARGS  (#PP #NIL)
   TT    (I)
   AX    (sein)
   CL    (PP-GEEN)
)

(gehen          CAT (VST)
   ALO   (ging)
   PLC   (WI)
   TAG   (ALL)
   SNS   (2)
   PX    (NIL)
   ARGS  (#PP #NIL)
   TT    (I)
   AX    (sein)
   CL    (PAI-3 PAS-3)
)

(gehen          CAT (VST)
   ALO   (geh)
   PLC   (WI)
   TAG   (ALL)
   SNS   (3)
   PX    (NIL)
   ARGS  (#PP #NIL)
   TT    (I)
   AX    (sein)
   CL    (IMP-1 INF-EN PRI-1 PRS-1)
)
```

These particular lexical entries provide the following information to the system:

- the canonical form <u>gehen</u> is a verb stem (VST);

- <u>gehen</u> has three stem forms or allomorphs (ALOs): <u>gang</u>, <u>ging</u>, and <u>geh</u>;

- PLC (WI) indicates that the ALOs <u>ging</u> and <u>geh</u> must be preceded by a blank; PLC (NF NI) indicates that the ALO <u>gang</u> must be preceded and followed by non-blank characters, i.e., the past participle marker and an ending;

- gehen is a general term (TAG (ALL));
- each entry for gehen has a unique system-assigned number (SNS) for reference;
- the canonical form gehen takes no prefix (PX (NIL));
- the ARGS feature indicates that gehen may occur either with no non-subject argument or with a single prepositional phrase (PP) argument, in addition to the subject;
- gehen is intransitive (I), indicated by the transitivity type (TT) feature;
- gehen takes sein as its auxiliary (AX); and
- the various allomorphs belong to specific inflectional classes (CL), e.g., the value PP-GEEN indicates that the ALO gang takes a ge- participle marker and an -en ending, while the value PRI-1 indicates that the ALO geh takes the German endings defined for the present indicative paradigm 1;

Verb stems represent complex lexical entries; other lexical entries may be less complex. The database entries for the noun stem (NST) Magnetband and the preposition (PREP) nach illustrate other types of lexical entries:

```
(Magnetband     CAT (NST)
   ALO (Magnetband)
   PLC (WI)
   PRF (2)
   SNS (1)
   TAG (EWS)
   CL  (S-E)
   GD  (N)
   DR  (RD NP)
   SX  (N)
   RC  (LOC)
   MC  (an)
   FC  (PP)

)
```

```
(Magnetband     CAT (NST)
   ALO (Magnetbaend)
   PLC (WI)
   PRF (2)
   SNS (2)
   TAG (EWS)
   CL  (P-4)
   GD  (N)
   DR  (RD NP)
   SX  (N)
   RC  (LOC)
   MC  (an)
   FC  (PP)
)

(nach           CAT (PREP)
   ALO (nach)
   PLC (WI WF)
   SNS (1)
   RO  (TOP TMP LOC)
   PO  (PRE POST)
   GC  (D)
)
```

The entries for Magnetband supply the following information:

- the canonical form Magnetband is a noun stem (NST);
- there are two ALOs: one with and one without an umlauted vowel;
- Magnetband is always preceded by a blank, indicated by PLC (WI); it has a preference factor (PRF) of 2; each entry has a unique, system-assigned number for reference; and it is a telecommunications term (TAG (EWS));
- the ALOs have distinct singular and plural class (CL) values;
- Magnetband is neuter (N) in both grammatical (GD) and natural gender (SX);
- Magnetband may occur with or without a determiner, indicated by DR (RD NP); and
- Magnetband may occur with a locative (RC (LOC)) prepositional phrase complement (FC (PP)) marked with the preposition an (MC (an)), e.g., das Magnetband am Magnetbandgerät.

The entry for the preposition nach provides the following information:

- the canonical form nach is a preposition (PREP);
- there is a single ALO in this category;
- nach occurs with blanks surrounding it, indicated by PLC (WI WF);
- nach in CAT (PREP) has a single SNS;
- the possible semantic roles (RO) for nach are topical (TOP), temporal (TMP), or locative (LOC);
- nach is positioned (PO) either before (PRE) or after (POST) the noun it governs; and
- nach governs dative (D) noun phrases, indicated by the governing case (GC) feature.

Every element which the system must analyze must be entered as a lexical item. This includes punctuation, prefixes, and symbols found in the source texts. The lexical entries for the period, the prefix verloren, and the multiplication symbol (*) provide examples:

```
(period         CAT (PNCT)
   ALO (.)
   SNS (1)
   TAG (ALL)
)

(verloren       CAT (PRFX)
   ALO (verloren)
   PLC (WI)
   SNS (1)
   TAG (ALL)
)

(*              CAT (CONJ)
   ALO (*)
   PLC (WI WF)
   SNS (1)
   TAG (ALL)
   CU  (COR)
   CJ  (COR)
)
```

The multiplication symbol (*) is listed as a conjunction (CONJ) because it is functionally equivalent to a conjunction. Punctuation marks and German prefixes are listed only in the German monolingual database, because they are not translated per se.

In addition to lexical entries for words, it is necessary to include morphological entries in the lexicon. The database entries for German -t will illustrate:

```
(t1             CAT (V-FLEX)
   ALO   (t)
   PLC   (NI)
   SNS   (1)
   CL    (PP-GET PP-T)
   PF    (PAPL)
)

(t1             CAT (V-FLEX)
   ALO   (t)
   PLC   (NI)
   SNS   (2)
   CL    (IMP-1 IMP-3 IMP-7)
   NU    (PL)
   PS    (2)
   MD    (IMP)
   PF    (FIN)
   TN    (PR)
)

(t1             CAT (V-FLEX)
   ALO   (t)
   PLC   (NI)
   SNS   (3)
   CL    (PAS-4)
   NU    (PL)
   PS    (2)
   MD    (SUB)
   PF    (FIN)
   TN    (PA)
)

(t1             CAT (V-FLEX)
   ALO   (t)
   PLC   (NI)
   SNS   (4)
   CL    (PRS-2)
   NU    (PL)
   PS    (2)
   MD    (SUB)
   PF    (FIN)
   TN    (PR)
)
```

```
(tl              CAT (V-FLEX)
   ALO    (t)
   PLC    (NI)
   SNS    (5)
   CL     (PAI-3 PAI-4 PAI-7 PAI-8)
   NU     (PL)
   PS     (2)
   MD     (IND)
   PF     (FIN)
   TN     (PA)
)

(tl              CAT (V-FLEX)
   ALO    (t)
   PLC    (NI)
   SNS    (6)
   CL     (PAI-8)
   NU     (SG)
   PS     (2)
   MD     (IND)
   PF     (FIN)
   TN     (PA)
)

(tl              CAT (V-FLEX)
   ALO    (t)
   PLC    (NI)
   SNS    (7)
   CL     (PRI-1 PRI-2 PRI-4 PRI-5 PRI-6 PRI-16)
   NU     (PL)
   PS     (2)
   MD     (IND)
   PF     (FIN)
   TN     (PR)
)

(tl              CAT (V-FLEX)
   ALO    (t)
   PLC    (NI)
   SNS    (8)
   CL     (PRI-1 PRI-2 PRI-4 PRI-5 PRI-11 PRI-13 PRI-14)
   NU     (SG)
   PS     (3)
   MD     (IND)
   PF     (FIN)
   TN     (PR)
)
```

```
(tl              CAT (V-FLEX)
   ALO    (t)
   PLC    (NI)
   SNS    (9)
   CL     (PRI-4 PRI-5 PRI-10 PRI-14)
   NU     (SG)
   PS     (2)
   MD     (IND)
   PF     (FIN)
   TN     (PR)
)
```

The features indicate that:

- <u>tl</u> is a verb ending (V-FLEX);
- <u>tl</u> must always be preceded by a non-blank character, indicated by PLC (NI);
- each entry has a unique, system-assigned number (SNS) for reference;
- <u>tl</u> is used for verbs of particular inflectional classes (CL), e.g., -<u>t</u> occurs in the past indicative paradigm 8 (PAI-8); and
- the various ALOs of <u>tl</u> have distinct grammatical number (NU), person (PS), mood (MD), predicate form (PF), and tense (TN) values;

Morphological entries do not have transfer lexical entries.

2.2 English Monolingual Lexicons

The principles for the English monolingual lexicons are the same as those for the German. <u>Go</u> is an example of an English verb entry:

```
(go        CAT (VST)
   ALO    (go)
   PLC    (WI)
   TAG    (ALL)
   SNS    (1)
   CL     (PR-ES2 I-0)
   ON     (CO)
   PX     (NIL)
   ARGS   (#PP #NIL)
   PX     (NIL)
)
```

```
(go        CAT (VST)
   ALO     (went)
   PLC     (WI)
   TAG     (ALL)
   SNS     (2)
   CL      (PA-0)
   ON      (CO)
   PX      (NIL)
   ARGS    (#PP #NIL)
   TT      (I)
)

(go        CAT (VST)
   ALO     (gone)
   PLC     (WI)
   TAG     (ALL)
   SNS     (3)
   CL      (P-0)
   ON      (CO)
   PX      (NIL)
   ARGS    (#PP #NIL)
   TT      (I)
)
```

These entries provide the following information:

- the canonical form go is a verb stem (VST);
- go has various allomorphs (ALOs): go, went, and gone;
- PLC (WI) indicates that all ALOs must be preceded by a blank;
- go is a general term (TAG (ALL));
- each entry for go has a unique system-assigned number (SNS) for reference;
- go's various allomorphs belong to specific classes (CL), e.g., the value P-0 indicates that the ALO gone takes no ending as the past participle, and go takes the endings of the PR-ES2 present indicative paradigm;
- onset (ON) for go is consonantal (CO);
- the canonical form go takes no prefix (PX (NIL));
- the ARGS feature indicates that go may occur either with no non-subject argument or with a single prepositional phrase (PP)

argument, in addition to the subject; and

- go is intransitive (I), indicated by the transitivity type (TT) feature;

The entries for magnetic tape and after will show the general form of English noun stem (NST) and preposition (PREP) entries:

```
(magnetic_tape  CAT (NST)
   ALO (magnetic tape)
   PLC (WI)
   PRF (2)
   SNS (1)
   TAG (EWS)
   CL  (S-01 P-S)
   ON  (CO)
   DR  (RD NP)
   SX  (N)
   RC  (LOC)
   MC  (on)
   FC  (PP)
)

(after          CAT (PREP)
   ALO (after)
   PLC (WI WF)
   SNS (1)
   RO  (TOP TMP LOC)
   PO  (PRE)
   ON  (VO)
)
```

Again, the entries are essentially the same as for the corresponding German words. The entry for magnetic tape indicates:

- the canonical form magnetic tape is a noun stem (NST);

- there is a single ALO;

- magnetic tape is always preceded by a blank, indicated by PLC (WI); it has a preference factor (PRF) of 2; each ALO has a unique, system-assigned number for reference; and it is a telecommunications term (TAG (EWS));

- the ALO has particular singular and plural class (CL) values;

- onset (ON) for magnetic tape is consonantal (CO);

- magnetic tape may occur with or without a determiner, indicated by DR (RD NP);

- magnetic tape is neuter (N) in natural gender (SX); and
- magnetic tape may occur with a locative (RC (LOC)) prepositional phrase complement (FC (PP)) marked with the preposition on (MC (on)), e.g., the magnetic tape on the tape drive.

After´s entry supplies the following information:

- the canonical form after is a preposition (PREP);
- there is a single ALO in this category;
- after occurs with blanks surrounding it, indicated by PLC (WI WF);
- after in CAT (PREP) has an SNS (1);
- the semantic roles (RO) of after are topical (TOP), temporal (TMP), or locative (LOC);
- after is positioned (PO) before (PRE) the noun it governs; and
- the onset (ON) for after is vocalic (VO).

As stated above (page 5), punctuation marks and German prefixes are not translated per se, and, thus, need no corresponding English entries. Symbols, such as the multiplication sysmbol, do have English entries:

```
(*        CAT (CONJ)
   ALO (*)
   PLC (WI WF)
   SNS (1)
   TAG (ALL)
   CU  (COR)
   CJ  (COR)
)
```

English endings are coded in essentially the same format as other entries, except the canonical form is the same as the lexical category. The lexical entry for the English verb ending -es is an example:

```
(V-FLEX    CAT (V-FLEX)
   ALO   (-es)
   SNS   (61)
   CL    (PR-ES2)
   PS    (3)
   NU    (SG)
   TN    (PR)
   MD    (IND)
   PF    (FIN)
)
```

This entry supplies the following information:

- es is a verb ending (V-FLEX);
- the ALO is -es;
- the ALO has a unique system-assigned SNS;
- the inflectional class (CL) value is PR-ES2; and
- the ending is 3rd (3) person (PS), singular (SG) number (NU), present (PR) tense (TN); it is indicative (IND) mood (MD); and its predicate form (PF) is finite (FIN);

2.3 Bilingual Lexicons

The bilingual (transfer) lexicons are essentially simple equations in LISP format which indicate correspondence between source and target language canonical forms. An example is the transfer entry for geben:

```
(give                    (geben) VST (CAT VST))
```

This entry simply indicates that the German verb stem (VST) canonical form geben corresponds to the English VST canonical form give.

Bilingual entries may also be more complex, involving special requirements or conditions for transfer. The transfer entries for gehen may be used as examples:

```
(go           (gehen) VST (CAT VST) (PX NIL) (PF FIN INF PAPL))
(outgo        (gehen) VST (CAT VST) (PX NIL) (PF PRPL))
```

The first entry indicates that the German canonical form gehen is equivalent to the English go, if the German stem does not have a prefix (PX) and if its predicate form (PF) is finite (FIN), infinitival (INF) or past participial (PAPL); the second entry indicates that gehen is equivalent to <u>outgo</u>, if it is a present participle (PRPL) and has no prefix.

3 Grammar

3.1 Rule Composition

Grammar rules are used to build words, phrases, clauses, or Ss. In the METAL grammaar an S is any sequence which is defined as such by a grammar rule; this definition allows an S to be any sequence from a single word which is not part of a longer string to a highly complex sentence.

METAL rules consist of a number of parts. These parts are illustrated in the VB -> VST V-FLEX rule:

```
VB       VST        V-FLEX
0        1          2
--       (REQ WI)   (NRQ WI)

TEST     (INT 1 CL 2 CL)
         (OR (NOT (INT 2 PF PAPL))
             (INT 1 CL PP-T PP-ET PP-EN PP-N))
         (OR (INT 2 PF PAPL INF)
             (RET 2 WF))

CONSTR   (CPX 1 ALO CL)
         (CPY 2 PS NU TN MD PF WF)
         (ADD WI)
         (AND (INT 2 PS 3)
              (INT 2 NU SG)
              (PRF 2))
         (* TAG ALL)

TRANSF   (XFM VB)
----------
```

The first line describes the structure to which the rule applies; the second line numbers the elements from 0 - n for reference in the body of the rule. Here the first line indicates that a verb (VB) may consist of a verb stem (VST) and a verb ending (V-FLEX).

In order to limit the scope of a given rule to the appropriate structures, individual right-hand elements may be checked using column tests. All column tests must succeed for the rule to succeed. In the sample rule, the verb stem must be preceded by a blank, stipulated by the placement requirement on the first right-hand element; conversely, the second right-hand element must not be preceded by a blank, as specified in its column test.

Rules may also be constrained using TEST. This part is used to perform tests between right-hand elements. Again, all such tests must succeed for the rule to succeed. In the sample rule the class values for the verb stem and the verb ending are compared using an intersection (INT) to insure that the particular ending may occur on the particular stem. TEST further checks the ending and/or the stem for acceptability on the basis of predicate form, class, and placement values.

After the various tests have succeeded, the CONSTR part of the rule builds the appropriate syntactic tree using the right-hand elements as the "sons" and the left-hand element as the "father" node. CONSTR may also add information to the new node. In the VB -> VST V-FLEX rule CONSTR builds the following syntactic tree, copying (CPX and CPY) and adding (ADD and PRF) necessary information to the new node.

```
          VB
          |
    -------------
    |           |
   VST        V-FLEX
```

In this tree VB is the father node; VST and V-FLEX are the sons. CONSTR is also a convenient place in which to put a comment indicating whether the rule is applicable to German in general (ALL) or to German telecommunications manuals only (EWS).

Transfer (TRANSF) is invoked once the entire sequence has been parsed. In the VB -> VST V-FLEX rule TRANSF invokes a transformation named VB (described more fully on page 23). Transformations in general are described on pages 19 - 20; the transfer process is described beginning on page 23.

3.2 Rule Operation: Analysis

The German sentence nach 3 Stunden geht die Ausgabe auf Magnetband ´the output goes to magnetic tape after 3 hours´ may be used to illustrate the operation of the grammar rules. A complete description of the entire translation process for this sentence is unnecessary; parts of the process will illustrate the operation of the system.

The METAL system parses from left to right beginning with the first element in the sequence and builds structure from right to left. Throughout its operation METAL constantly seeks to build the longest possible string of elements. The parser is an all-paths parser, i.e., it will seek to build all possible structures permitted by the grammar.

Use of rules to build phrases, clauses and sentences is an integral part of any generative grammar, but in METAL such rules have been taken a step farther and are also used to build words. Word-level rules are used to construct words from word components, thus saving lexical storage space in the system, since storage of only word stems and possible endings is required. For the verb gehen, for instance, the system must only store the three allomorphs of the verb stem, given above (page 2), and the endings necessary to build all forms of the verb. Further, the endings may be used for all appropriate stems, e.g., the nine entries for tl (pages 6 - 8) may be used in the construction of all verb forms which use -t in any form. The alternative would be to store all forms of every word; for gehen this would require twenty-six entries, i.e., twenty-four ALOs for the inflected verb forms plus an ALO for the infinitive form and one for the past participle.

The VB -> VST V-FLEX rule will illustrate the operation of a word-level rule:

```
VB        VST       V-FLEX
0         1         2
--        (REQ WI) (NRQ WI)

TEST      (INT 1 CL 2 CL)
          (OR (NOT (INT 2 PF PAPL))
              (INT 1 CL PP-T PP-ET PP-EN PP-N))
          (OR (INT 2 PF PAPL INF)
              (RET 2 WF))

CONSTR    (CPX 1 ALO CL)
          (CPY 2 PS NU TN MD PF WF)
          (ADD WI)
          (AND (INT 2 PS 3)
               (INT 2 NU SG)
               (PRF 2))
          (* TAG ALL)

TRANSF    (XFM VB)
----------
```

The relevant lexical entries are:

```
(gehen          CAT (VST)
   ALO    (geh)
   PLC    (WI)
   TAG    (ALL)
   SNS    (3)
   PX     (NIL)
   ARGS   (#PP #NIL)
   TT     (I)
   AX     (sein)
   CL     (IMP-1 INF-EN PRI-1 PRS-1)
)

(t1             CAT (V-FLEX)
   ALO    (t)
   PLC    (NI)
   SNS    (7)
   CL     (PRI-1 PRI-2 PRI-4 PRI-5 PRI-6 PRI-16)
   NU     (PL)
   PS     (2)
   MD     (IND)
   PF     (FIN)
   TN     (PR)
)

(t1             CAT (V-FLEX)
   ALO    (t)
   PLC    (NI)
   SNS    (8)
   CL     (PRI-1 PRI-2 PRI-4 PRI-5 PRI-11 PRI-13 PRI-14)
   NU     (SG)
   PS     (3)
   MD     (IND)
   PF     (FIN)
   TN     (PR)
)
```

In analyzing <u>geht</u> the rule first applies the column tests, which succeed because the verb stem is preceded by a blank and the verb ending is not. TEST then intersects the values for inflectional class from each of the elements and succeeds with the PRI-1 value. The second test succeeds because at least one of the readings of <u>t1</u> is not past participial; the third test succeeds because <u>t1</u> is followed by a blank. The rule will successfully apply twice, because the form <u>geht</u> may be either third person singular or second person plural; there are, therefore, two interpretations at this point in the process.

CONSTR then builds the following syntactic tree. All features, except allomorph (ALO) and inflectional class (CL), are copied from the verb stem to the VB node, using the CPX function. The values for person (PS), number (NU), tense (TN), mood (MD), predicate form (PF) and word final (WF) are copied (CPY) from the ending; WI (word initial) is added (ADD) to the new node. Since the verb is both third person and singular in one interpretation, a preference factor (PRF) of 2 is added to this interpretation.

```
          VB
          |
     -------------
     |           |
    VST        V-FLEX
     |           |
    geh          t
```

TRANSF aplies only after an S is built; the transfer process is discussed below (beginning on page 23).

The verb (VB) geht becomes a finite predicate (PRED) by means of the PRED -> VB rule and this predicate becomes a right-branching clause (RCL) by the RCL -> PRED rule:

```
RCL           PRED
0             1
(LVL 2)       --

TEST

CONSTR    (CPY 1 CAN TT MD SPX PX)
          (* TAG ALL)

TRANSF    (XFR)
----------
```

Since there are no tests, any finite predicate can become an RCL. CONSTR builds the following tree and copies the transitivity type (TT), mood (MD), separable prefix (SPX) and prefix (PX) features up to the new node:

```
RCL
 |
PRED
 |
 VB
 |
geht
```

The nominal constructions (3 Stunden, die Ausgabe, and Magnetband) in the sentence are analyzed by various rules and each eventually becomes a noun phrase (NP); the two prepositional phrases nach 3 Stunden and auf Magnetband are built up by the PP -> PREP NP rule:

```
PP              PREP  NP
0               1     2
(LVL 1)         --    (OPT KP * DEM)

TEST      (INT 1 GC 2 CA = X1)
          (OR (RET 1 WF)
              (AND (RET 2 BF)
                   (INT 1 CN 2 BF)))

CONSTR    (ADF 1 CAN PR)
          (ADX X1 CA)
          (CPX 2 CA NPCNJ)
          (ADF 1 ADVB)
          (ADF 1 RO)
          (AND (INT 2 KP REL)
               (ADD RELPRN))
          (AND (RET 2 NPCNJ)
               (ADD PPCNJ))
          (* TAG ALL)

TRANSF    (SEF 1 CA GC)
          (OR (RET 1 PRCOM)
              (SEV 1 PRCOM NIL))
          (AND (INT 1 GC A)
               (INT 1 GC D)
               (SEV 1 GC D))
          (XFR)
          (CPY 2 MC)
----------
```

The RCL <u>geht</u> may combine with the noun phrase <u>die</u> <u>Ausgabe</u> to form another RCL using the RCL -> RCL NP rule:

```
RCL             RCL         NP
0               1           2
(LVL 2)         (NRQ CLF)   (REQ CA * G)

TEST

CONSTR    (CPY 1 CAN TT MD SPX PX)
          (XFM FLATL)
          (* TAG ALL)

TRANSF    (XFR)
----------
```

RCL -> RCL NP succeeds because <u>geht</u> is not clause final (CLF) and <u>die</u> <u>Ausgabe</u> is not genitive (G). The CONSTR part of this rule constructs the syntax tree and copies up the listed features from <u>geht</u>. CONSTR also invokes a named transformation, FLATL, which deletes the old RCL node.

The RCL -> RCL NP rule illustrates two important features of METAL grammar rules. One is the use of transformations to alter trees; the other is the principle of recursion. Transformations may be used in TEST, CONSTR and/or TRANSF to change the structures of syntactic trees. These transformations permit context-sensitivity in the grammar rules, and are used to enhance the system's capability to produce high quality translation. METAL transformations may be defined within a grammar rule or defined elsewhere and invoked by name within a rule. FLATL, invoked in the RCL -> RCL NP rule, is a named transformation. Transformations have essentially the same format regardless of whether they are named or defined in a rule. Every transformation consists of two parts: a description of the syntactic tree to which the transformation applies (the structural description) and a description of the tree which the transformation produces (the structural change).

Named transformations are generally used when a transformation is to be used in a number of rules. The FLATL transformation, given below, is a relatively simple transformation, but illustrates how transformations work:

```
(DT (&:1 ((&:2 (--:3)) &:4))
    (&:1 (--:3 &:4))
FLATL)
```

Each ampersand (&) represents one and only one element; the double dashes (--) represent zero or more elements. In this instance, FLATL applies to a tree such as:

```
                    RCL
                     |
  ----------------------------------------
  |                                      |
 RCL                                     NP
  |                                      |
 PRED                                    |
  |                                      |
 geht                              [die Ausgabe]
```

FLATL deletes the lower RCL node and produces a syntactic tree such as:

```
                 RCL
                  |
  ----------------------------------------
  |                                      |
  |                                      NP
  |                                      |
 PRED                                    |
  |                                      |
 geht                              [die Ausgabe]
```

Recursion is the property which permits a grammar rule to apply continuously as long as its structural requirement is met. RCL -> RCL NP is a recursive rule since it may apply as long as it finds any RCL followed by an NP. Such recursiveness permits a finite number of grammar rules to apply to a theoretically infinite number of sequences. The German sequence *gestern gab die Frau dem Kind das Buch* ´yesterday the woman gave the child the book´ may be used to illustrate the recursion principle. The rule RCL -> RCL NP applies first to *gab die Frau* and builds the first tree, given below. The rule can then apply to *gab die Frau dem Kind* and build the next tree. Finally, the same rule analyzes *gab die Frau dem Kind das Buch* and builds the third tree. In each application of the rule the lower RCL node is deleted using the FLATL transformation described immediately above.

```
       RCL
        |
    ----------
    |        |
   RCL       NP
    |        |
   gab   [die Frau]
```

```
              RCL
               |
       -----------------
       |               |
      RCL              NP
       |               |
 [gab die Frau]    [dem Kind]
```

```
                  RCL
                   |
        ----------------------
        |                    |
       RCL                   NP
        |                    |
 [gab die Frau dem Kind]  [das Buch]
```

The sequence of the RCL geht die Ausgabe followed by a prepositional phrase (PP) auf Magnetband becomes an RCL using the RCL -> RCL PP rule:

```
RCL            RCL        PP
0              1          2
(LVL 2)        (NRQ CLF)  --

TEST

CONSTR     (CPY 1 CAN TT MD SPX PX)
           (XFM FLATL)
           (* TAG ALL)

TRANSF     (XFR)
----------
```

This rule builds the tree:

```
                          RCL
                           |
                ---------------------------------
                |                               |
               RCL                              PP
                |                               |
     -------------------------          -------------
     |                       |          |           |
    PRED                     NP        PREP         NP
     |                       |          |           |
    geht              [die Ausgabe]    auf      Magnetband
```

Nach 3 Stunden may then be added to the left of this RCL by the CLS -> PP RCL rule:

```
CLS            PP  RCL
0              1   2
(LVL 3)        --  (REQ MD * IMP)
--             --  (OR (REQ SPX)(REQ PX NIL))

TEST       (FRM 2 1)

CONSTR     (PRF 3)
           (* TAG ALL)

TRANSF     (XFR)
           (ORO)
           (XFM ADV-CLS)
----------
```

The column tests for CLS -> PP RCL succeed because the RCL is not imperative and has a NIL value for the PX feature (i.e., the canonical form gehen does not have a prefix). The case frame (FRM) then applies to the two elements in TEST.

The case frame is a mechanism used to determine the functions of the elements in a clause. It uses the grammatical predicate's arguments (ARGS) as the basis for determining functions. Gehen's ARGS feature specifies either no non-subject argument or a single prepositional phrase argument in addition to the subject (see the entry, page 16). The case frame begins with the longest argument string to insure that all arguments are considered. In this example either prepositional phrase fulfils the requirements; the other is considered to be a peripheral argument. The nominative NP die Ausgabe is recognized as the subject.

The CLS -> PP RCL rule builds the following syntactic tree:

```
                   CLS
                    |
     -------------------------------
     |                             |
     PP                           RCL
     |                             |
 -------------              ----------------------
 |           |              |          |         |
PREP         NP            PRED        NP        PP
 |           |              |          |         |
nach    [3 Stunden]        geht   [die Ausgabe] [auf Magnetband]
```

S -> "$" CLS PNCT "$" is the final rule to apply:

```
S          "$"  CLS  PNCT                                   "$"
0          1    2    3                                      4
(LVL 3)    --   --   (REQ CAN PERIOD EXCLAMATION COLON)     --

TEST

CONSTR     (PRF 3)
           (* TAG ALL)

TRANSF     (XFR 2)
----------
```

As long as the rule finds a clause followed by a period, exclamation point, question mark or colon, and surrounded by dollar sign delimiters, the rule will succeed. The dollar sign delimiters are added as S boundary markers by the system preprocessor. Nach 3 Stunden geht die Ausgabe auf Magnetband meets the necessary conditions and CONSTR builds the tree:

```
                          S
                          |
 -----------------------------------------------------------
 |                        |                            |   |
"$"                      CLS                         PNCT "$"
 |                        |                            |   |
 | --------------------------------------------        |   |
 | |        |          |           |                   |   |
 | PRED     PP         NP          PP                  |   |
 | |        |          |           |                   |   |
 $ geht  [nach 3 Stunden] [die Ausgabe] [auf Magnetband] .  $
```

3.3 Rule Operation: Transfer

After the system has successfully built an S the transfer phase (TRANSF) of the system begins. Transfer starts with the TRANSF portion of the rule which built the highest node of the syntactic tree and works downward recursively. In this instance, transfer begins with the TRANSF portion of the S -> "$" CLS PNCT "$" rule. In this rule TRANSF contains the function XFR, which causes the system to descend the tree to the TRANSF part of the next rule , i.e., CLS -> PP RCL (cited on page 21):

The first function in the TRANSF for CLS -> PP RCL is another XFR which causes the system to descend the tree again, this time to the rules which built the predicate <u>geht</u>, the prepositional phrases <u>nach 3 Stunden</u> and <u>auf Magnetband</u> and the noun phrase <u>die Ausgabe</u>. Each time a XFR is encountered the system will descend the tree until it encounters a terminal node. An example of a rule for terminal nodes is VB -> VST V-FLEX (cited on pages 13 and 15); in this rule, TRANSF invokes the named transformation VB. This named transformation lexically transfers ("translates") the canonical form of the verb from German to English, adding the appropriate ending based on the values for number, tense, predicate form, person and class. In lexical transfer, the bilingual lexicon is used to look up the English equivalent and the English monoligual entry supplies the necessary morphological and syntactic information. At this point in the process, the German word is transferred to the corresponding English word.

All other elements in the sentence are treated in the same manner. After lexical transfer of all the terminal nodes the syntactic tree is:

```
                              S
                              |
 ------------------------------------------------------------
 |    |        |                |            |         |   |
"$"  PRED      PP               NP           PP      PNCT "$"
 |    |        |                |            |         |   |
 $   goes  [after 3 hours]  [the output] [to magnetic   .   $
                                               tape]
```

The system then reascends the tree and applies the remaining functions of TRANSF in the rules which built the tree. When the system returns to the CLS -> PP RCL rule (page 21) the ORO function will be invoked. This function uses the transfer portion of the case frame to order the English words in a predetermined way. The reordered tree is then:

```
                              S
                              |
 --------------------------------------------------------------
 |    |          |            |             |          |   |
"$"   NP        PRED          PP            PP        PNCT "$"
 |    |          |            |             |          |   |
 $ [the output]  goes      [to magnetic  [after 3      .   $
                                tape]     hours]
```

The result of the translation process is thus the English sentence: the output goes to magnetic tape after 3 hours.