TR-I-0365

A Japanese-German Transfer Component for ASURA

Mark Seligman

March, 1993

Abstract

This document describes the transfer component for Japanese-German translation in the ASURA speech-to-speech translation system, emphasizing information needed for maintenance and extension. We give an orientation to the transfer system; we provide a brief primer for the Feature Structure Rewriting System software; we describe a set of useful functions for development; we discuss numerous issues and directions for the future; and finally, we discuss the current set of Japanese-German transfer rules, treating each phase of rule application in turn.

ATR自動翻訳電話研究所
ATR Interpreting Telephony Research Laboratories

Contents

Chapter 1

Introduction

This document describes the Japanese-German transfer component of the ASURA speech-to-speech translation system. J-G transfer uses the Feature Structure Rewriting System (RWS) ([Hasegawa 1990], [Suzuki and Kosaki 1993]) for transforming Japanese analysis output into German generation input.

This chapter gives a brief overview of the transfer process and a first look at transfer rules. In the chapters which follow, we provide a point-by-point primer for the Feature Structure Rewriting System; we describe a set of useful functions for transfer system development; we discuss issues which arose during development and future directions; and we comment on the current set of Japanese-German transfer rules, treating each phase and sub-phase of rule application in turn.

The goal of the current project has been to make a first exploration of the problems of Japanese-German using the RWS technology. Thus we concentrate below (especially in the final two chapters) upon areas in which our present treatment has been incomplete or unsatisfactory, in order to suggest directions for further work.

The Japanese-German transfer facilities function without errors in translating twelve scripted dialogs concerning conference registration ("mset" dialogs DA through D10), with a total of 262 utterances. They also cover the 59 sentences in dialog D2, D3, and D6 which were prepared for the CSTAR demo of January 28, 1993, including numerous variants of the mset sentences. However, because the aim was early exploration rather than robustness or coverage, there has been no attempt yet to harden the current ruleset by experimenting with any other input sentences. Further, within this limited corpus, transfer input and output were strictly controlled: transfer accepted analysis output prepared in advance from a fixed set of scripted Japanese input sentences, and delivered output agreed in advance, so that the German generation component [Tropf 1992] could produce preagreed translations.[1]

Following are pathnames for the relevant input and output files and for the current J-G transfer ruleset:

```
as25:/home/tropf/gen-off/anout/demo6-25.euc          ;analysis output for DA-D10
as25:/home/tropf/gen-off/anout/kaiwa-2-3-6.930118    ;analysis output for CSTAR demo

as25:/home/tropf/gen-off/transout/1-262.921201.1200  ;transfer output for DA-D10
as25:/home/tropf/gen-off/transout/1-59.930119.1745   ;transfer output for CSTAR demo

as25:/home/tropf/gen-off/transfer/<date>             ;latest J-G transfer ruleset

;;;See README files in each subdirectory for further information.
;;;as25:/home/tropf/gen-off/README-gram-trans.text gives generation information.
```

[1] In view of this strictly limited development environment, errors must be expected if the present J-G ruleset is applied to new data. We should also note that because of time limitations it was sometimes necessary to adapt existing Japanese-English transfer rules [Suzuki and Kosaki 1993] for Japanese-German use. Adapted rules which caused no problems were generally not rewritten; however, bugs not found while translating the current corpus may hide within them.

1.1 Overview of Transfer

ASURA is a transfer-based machine translation (MT) system. Thus it carries out translation in three main stages:

- *Analysis* takes a source-language string as input and produces as output a feature structure (the *anout*) which remains close to the structure of the source expression, and whose symbols remain source-language symbols (e.g. 会議-N, *kaigi-N* "conference").
- *Transfer* receives the anout as input and delivers as output a modified feature structure (the *transout*) whose structure is that of the target language, and whose symbols are target language symbols (e.g. KONFERENZ-N).
- *Generation* can then transform the transout into a target-language syntactic structure (the *genout*). *Morphology*, usually considered part of generation, will finally transform the genout into a target language string.

1.1.1 Phases in Transfer

Transfer in ASURA has three main internal phases, which we can call pre-transfer, main-transfer, and post-transfer.

Main-transfer

Main-transfer is the replacement of source-language symbols by target-language symbols. 会議-N (*kaigi-n*).for example, is replaced by KONFERENZ-N for German generation. All source-language symbols must be replaced in this way; and in fact, a large percentage of the rules in the present transfer ruleset are for this purpose.[2]

Pre-transfer

Pre-transfer makes various preparations for main-transfer. In general, it makes transformations which can be specified most easily if they are specified in terms of source-language elements. In the present design, among other functions, pre-transfer must

- supply pronouns and other elided or "missing" elements;
- recognize idioms and replace them by single symbols. For example, if the pattern ホテル の 手配 is found, it is replaced by a single symbol ホテルの手配-1. This symbol can then be replaced by HOTELBUCHUNG-N ("hotel bookings") during main-transfer. (See further below.)
- recognize illocutionary acts, such as REQUEST, INFORM, QUESTIONIF, etc.;
- assign values for pragmatic factors like POLITENESS and INTIMACY;

Post-transfer

Post-transfer performs various structural adjustments after the symbol replacement performed during main-transfer. Generally, these are adjustments which are easiest to specify in terms of target-language elements. Among other functions, post-transfer must

[2] A very small number of language-independent or "interlingual" symbols appear in the analysis output: DESIRE, OBLIGATION, NECESSITY, and a few others. It is not necessary for transfer to replace these symbols, since generation can use them as they are. See below regarding the possible use of more such "interlingual" symbols in future versions of ASURA.

- make some word-sense disambiguations by examining the local target-language context. For instance, 含む-1 (*fukumu*, "include") is normally translated by ENTHALTEN-V, but becomes UMFASSEN-V just in case the subject is KONFERENZ-N;
- supply possessive determiners (*mein, Ihr*) for nouns like *Name* and *Adresse* which lack such determiners in Japanese;
- supply indication of future tense where only present (non-past) tense appears in the original Japanese, e.g. 登録用紙は至急送らせていただきます。("I'll send you a registration form/will have a registration form sent immediately.") gives *Ich* werde *Ihnen sofort ein Anmeldeformular schicken.*
- supply indication of aspect in a few cases, e.g. to differentiate between the two possible forms of the German passive (*wird gemacht* vs. *ist gemacht*);
- replace Japanese particle *no* using appropriate German prepositions for noun noun modification — e.g. "会議の参加料" *kaigi no sanka-ryou*, "attendance fee for the conference" becomes *Teilnahmegebuehr* fuer *die Konferenz*;

In these cases, post-transfer supplies information needed for translation into fluent German but missing in the Japanese source. But post-transfer also makes numerous adjustments which are not directly related to translation; these are intended to permit more regular, economical, or convenient formulation of the rules which generate German syntax.

One example will be enough for now. Consider sentences like *Ich moechte gehen*, often discussed in terms of equi-deletion in the transformational literature: at the logical or "deep structure" level, the speaker is both the EXPR of DESIRE and the AGEN of GEHEN-V; but in surface expression, the speaker is mentioned only once. German syntax rules can be written most simply if the transfer output, too, mentions the speaker only once. Thus a certain post-transfer rule (ARG1000MSS) is designed to delete the EXPR feature and its value from the matrix verb (e.g. DESIRE) in such cases.

By now it should be clear that the transfer stage of translation is broadly defined in ASURA: pre-transfer and post-transfer do jobs which might be done by analysis or generation in other systems.

1.2 Transfer Rules and Structures: a Preview

To complete our orientation, we now give a first look at transfer structures and rules. We use idiom recognition as a convenient example. See the following chapter for a more complete and point-by-point introduction to system syntax.

1.2.1 Transfer Input and Output Structures

Assume that the input to ASURA's analysis program is the Japanese string "ホテルの手配もして頂けるのですか？" ("Could you make the hotel arrangements?") (mset #238).

The analysis output appears below. It shows the typical feature structure pattern for noun-noun modification using *no* (marked ;<<).

```
;>No. d10-14
ホテルの手配もして頂けるのですか
;No.238
[[SEM !X109[[RELN  S-REQUEST]
            [AGEN !X12[[LABEL  *SPEAKER*]]]
            [RECP !X13[[LABEL  *HEARER*]]]
            [OBJE [[RELN  INFORMIF]
```

```
                    [AGEN !X13]
                    [RECP !X12]
                    [OBJE
                     !X25[[RELN  える-POSSIBLE]
                          [AGEN !X09[]]
                          [ASPT  UNRL]
                          [OBJE [[RELN  てもらう-RECEIVE_FAVOR]
                                 [AGEN !X09]
                                 [RECP !X05[]]
                                 [OBJE
                                   !X16[[RELN  する-1]
                                        [AGEN !X05]
                                        [OBJE
                                          !X15[[PARM                          ;<<
                                                 !X08[[PARM !X06[]]           ;<<
                                                      [RESTR [[RELN  手配-1]  ;<<
                                                              [ENTITY !X06]]]] ;<<
                                               [RESTR [[RELN  の-連体修飾]    ;<<
                                                       [ARG-2 !X08]           ;<<
                                                       [ARG-1 [[PARM !X07[]]  ;<<
                                                              [RESTR [[RELN  ホテル-1] ;<<
                                                                      [ENTITY !X07]    ;<<
                     ]]]]]]]]]]]]]]]]]

[PRAG [[RESTR [[IN !X32[[FIRST [[RELN  POLITE]
                                [AGEN !X12]
                                [RECP !X13]]]
                      [REST [[FIRST [[RELN  EMPATHY-DEGREE]
                                     [MORE !X09]
                                     [LESS !X05]]]
                            [REST !X14[[FIRST [[RELN  POLITE]
                                               [AGEN !X12]
                                               [RECP !X13]]]
                                      [REST !X89[]]]]]]]
              [OUT !X89]]]
       [SPEAKER !X12]
       [HEARER !X13]
       [TOPIC [[IN !X31[]]
              [OUT !X110[]]]]
       [PRSP-TERMS [[IN !X29[[FIRST [[FOCUS !X15]
                                     [PRSP-MOD  MO]
                                     [SCOPE !X16]]]
                            [REST !X30[]]]]
                   [OUT !X111[]]]]
       [ASPE [[IN !X17[]]
             [OUT !X17]]]]]]
```

However, to enable a natural German translation using the compound noun *Hotelbuchung* (in *Koennten Sie die Hotelbuchung machen?*), we wish to recognize the pattern ホテル の 手配 (*hoteru no tehai*, "hotel arrangements") as an idiomatic grouping, and thus to replace the pattern with a single symbol, ホテルの手配-1. As mentioned, this recognition takes place during the pre-transfer phase. Then, during main-transfer, this symbol can in turn be replaced by HOTELBUCHUNG-N, giving this input for German generation:

```
;;;No.238

(TRANSOUT :SURFACE
          "NO.238"
```

```
          :STRUCTURE_STRING
          "


;;;=========== Transfer Result ============
 [[SEM [[RELN REQUEST]
        [AGEN !X3[[LABEL *SPEAKER*]]]
        [RECP !X1[[LABEL *HEARER*]]]
        [OBJE [[RELN MACHEN-V]
               [TENSE PRES]
               [AGEN !X1]
               [OBJE [[PARM !X2[]]
                      [RESTR [[RELN HOTELBUCHUNG-N]
                              [ENTITY !X2]]]
                      [INDEX [[DETERM DEFART]
                              [NUMBER SING]
                              [OWNER [[LABEL *UNKNOWN*]]]]]]]]
        [ATTD INTERROGATIVE]]]
  [PRAG [[RESTR [[IN [[FIRST [[RELN POLITE]]]
                     [REST [[FIRST [[RELN EMPATHY-DEGREE]
                                    [LESS !X1]
                                    [MORE !X3]]]
                            [REST [[FIRST [[RELN POLITE]]]
                                   [REST !X4[]]]]]]]]
                [OUT !X4]]]
         [HEARER !X1]
         [INTIMACY LOW]
         [POLITENESS [[DEGREE 3]]]
         [SPEAKER !X3]]]]
")
```

1.2.2 MAINRULE and REWRITE

The changes in question will be made in two phases. Phases within transfer are managed by the obligatory top-level rule, MAINRULE. Like every rule in the system, it is defined by a call to the function RWS:DEFRWSCHEMA2. Part of MAINRULE is shown below. The phases which currently concern us are marked ;<<<.

```
(rws:defrwschema2 mainrule t t
"on <> :main
   set ?SP to input.prag.speaker
   set ?HR to input.prag.hearer
   in= [[sem ?sem]
        ?rest]
   input = [[sem [[reln UNKNOWN-IFT]
                  [agen ?sp]
                  [recp ?hr]
                  [obje ?sem]]]
            ?rest]
   rewrite input.sem with  :PHASE :BAD-ANAL-PATCH              by :RECURSIVE

   rewrite input.sem  with :PHASE :ELLIPSIS-INIT               by :RECURSIVE
   rewrite input.sem  with :PHASE :ELLIPSIS-RESOLUTION         by :RECURSIVE
   rewrite input.prag with :PHASE :ELLIPSIS-RESOLUTION         by :RECURSIVE

   rewrite input.prag with :PHASE :JAPANESE :PRSP :INIT        by :RECURSIVE
   rewrite input.sem with  :PHASE :JAPANESE                    by :LOOP :RECURSIVE  ;<<<
```

```
   rewrite input.sem with  :PHASE :IF-REDUCE :TYPE :GENERAL     by :LOOP :RECURSIVE
   rewrite input     with  :PHASE :IF-REDUCE :TYPE :DEFAULT     by :RECURSIVE
   rewrite input     with  :PHASE :IF-REDUCE :TYPE :OMOU-FILTER by :RECURSIVE

   rewrite input.prag with :PHASE :PRAG-INIT                    by :LOOP :RECURSIVE
   rewrite input.prag with :PHASE :PRAG-POLITENESS              by :RECURSIVE
   rewrite input.prag with :PHASE :PRAG-FILTER                  by :RECURSIVE
   rewrite input.prag with :PHASE :PRAG-INTIMACY-DEFAULT        by :ONCE
   rewrite input.prag with :PHASE :PRAG-POLITENESS-DEFAULT      by :ONCE

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   rewrite input.sem with  :PHASE :J-G :TYPE :IDIOM             by :LOOP :RECURSIVE
   rewrite input.sem with  :PHASE :J-G :TYPE :GENERAL           by :LOOP :RECURSIVE
   rewrite input.sem with  :PHASE :J-G :TYPE :DEFAULT           by :LOOP :RECURSIVE  ;<<<
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

   ...

   return input
end")
```

Each use of the REWRITE operator in MAINRULE defines an ordered micro-phase of rule application. The first 14 REWRITE calls jointly make up the phase we have been calling pre-transfer. The three calls emphasized by comments make up main-transfer. (Post-transfer, which also contains many microphases, is not shown here since it is not relevant for our example.)

Arguments to a REWRITE operation are in the following format:

```
REWRITE <feature structure>
   WITH <application constraints>
      BY <rule control keywords>
```

In the Japanese-German transfer system, the feature structure to be rewritten can be (a) input (the entire anout) or (b) input.sem (the SEM feature within the entire input) or (c) input.prag (the PRAG feature within the entire input).

For the moment, let us assume that a given REWRITE operation can use only rules which are tagged with the same *application constraints* (specified after WITH). For example, the first REWRITE can use only rules which are marked :PHASE :BAD-ANAL-PATCH.[3]

We postpone discussion of the BY operator until the next chapter.

1.2.3 Rules

The following rules are needed to rewrite the idiom in question.

```
(rws:defrwschema2 jap056 t t
"on <parm restr reln> 手配-1 in :PHASE :Japanese
in=
[[PARM !X4[[PARM !X2[]]
           [RESTR [[RELN 手配-1]
                   [ENTITY !X2]]]]]
 [RESTR [[RELN の-連体修飾]
```

[3]This description is actually oversimplified. More flexible use of application constraints will be discussed in the next chapter.

```
        [ARG-1 [[PARM !X3[]]
                [RESTR [[RELN ホテル-1]
                        [ENTITY !X3]]]]]
        [ARG-2 !X4]]]]

out=
[[parm !X1[]]
 [restr [[reln ホテルの手配-1]
        [entity !X1]]]]
end")

------------------------------------------------------------------

(rws:defrwschema2 defn2000 t t
"on <restr reln> ホテルの手配-1  in :PHASE :J-G :TYPE :default
in=
[[PARM !X[]]
 [RESTR [[RELN ホテルの手配-1]
        [ENTITY !X]]]
 ?rest]
out=
[[PARM !X[]]
 [RESTR [[RELN Hotelbuchung-n]
        [ENTITY !X]]]
 [INDEX [[NUMBER SING]
        [DETERM DEFART]
        [OWNER [[LABEL *UNKNOWN*]]]]]
 ?rest]
end")
```

If all of a rule's application constraints are met (because they have been set by a previous or currently running rule) it passes the first level of testing for possible activation. The first transfer rule, JAP056, is tagged IN :PHASE :JAPANESE, and will thus pass this first test during the :PHASE :JAPANESE micro-phase of pre-transfer. Similarly, the second rule, DEFN2000, will pass the first testing level during the :PHASE :J-G :TYPE :DEFAULT micro-phase of main-transfer.

After a rule is found to meet the current constraints, three further levels of testing are applied.

- The feature path which follows the rule's ON keyword is examined. The ON syntax can be read as follows for JAP056: "Keep this rule for further testing if there is a feature path `<parm restr reln>` within the input FS whose atomic value is 手配-1." The current anout does have such a path and value, so JAP056 passes this test. Rules are indexed according to their ON paths, so that the rules which pass this level of testing can be found efficiently.

- Unification-based pattern matching determines whether the input pattern of a candidate rule (marked with the keyword IN=) matches against part of the input feature structure. The input pattern for JAP056 does match against the current anout; so this rule passes the second test.

- Programmatic if/then tests can be used to further test the input feature structure and manipulate the output feature structure. The keyword FAIL can be used to programatically abort rule operation if desired.

Winning rules (those which have the right constraints, ON path values, IN= patterns, and perhaps programmatic tests) are *activated*: the rewriting system destructively replaces the matched sub-structure using their output pattern (marked with keyword OUT=).[4]

[4]In the next chapter, we will discuss pattern matching details: how are feature structure tags interpreted? Must sub-structures be at the top level of the input feature structure, or can they be embedded? and so on.

Rule DEFN2000 will likewise pass all tests and apply during phase :PHASE :J-G :TYPE :DEFAULT. After the application of many other rules not discussed here, the desired transout will be formed.

Having finished our orientation tour, we can consider ASURA's rewrite system point-by-point and in greater detail in the next chapter.

Chapter 2

Feature Structure Rewriting System Primer

ASURA's present transfer components use the Feature Structure Rewriting System (RWS) for transforming Japanese analysis outputs into German (or English) generation inputs. The system was developed by Toshiro Hasegawa [Hasegawa 1990] and updated by Masami Suzuki and Hirohisa Kosaki [Suzuki and Kosaki 1993].

The only English discussion of the rewriting system is Hasegawa [1990], "A Rule Application Control Method in a Lexicon-driven Transfer Model of a Dialog Translation System." All versions of the system manual are in Japanese. This chapter provides a brief English reference resource, especially for those who read no Japanese. Its aim is to provide enough information to understand the syntax of transfer rules.

We emphasize features of the system used in the Japanese-German system, and thus do not follow the order of the system manuals. The previous chapter contains an introduction to the transfer system with examples.

2.1 RWS:DEFRWSCHEMA2: Defining a Transfer Rule

```
=====================================================================
rws:defrwschema2 (name type sortkey body)
=====================================================================
```

Each transfer rule is defined using a call to the system function rws:defrwschema2. The name is a unique rule identifier composed by hand. *Type and sortkey are not used in the Japanese-German system, and default to T for every rule.* The rule body is a string (in double quotes) whose elements are described throughout this chapter. For example, here is part of the top-level rule:

```
(rws:defrwschema2 mainrule t t
"on <> :main
   set ?SP to input.prag.speaker
   set ?HR to input.prag.hearer
   in= [[sem ?sem]
        ?rest]
   input = [[sem [[reln UNKNOWN-IFT]
                  [agen ?sp]
                  [recp ?hr]
                  [obje ?sem]]]
            ?rest]
   rewrite input.sem with  :PHASE :BAD-ANAL-PATCH  by :RECURSIVE
```

```
   ...

   return input
end")
```

2.2 The REWRITE Operator: Rewrite What, Under Which Constraints?

```
=====================================================================
REWRITE <FeatureStructure> WITH <ApplicationConstraints> BY <Keywords>
=====================================================================
```

This operator orders a rewriting operation on a specified feature structure. Following WITH, it sets application constraints in order to specify which rules can be considered. (See below.) The constraints remain in effect only during the current REWRITE. The BY keyword specifies how those rules should be applied: Should each rule be applied only once, or should looping be allowed? Should rules apply only to the top-level feature structure, or to embedded structures as well? etc. (See below.)

Numerous calls to REWRITE can be found in MAINRULE, the top-level rule. For example,

```
rewrite input.sem with  :PHASE :JAPANESE                    by :LOOP :RECURSIVE
```

Concerning the notation input.sem, see just below. (See also Embedded Rewrites, below.)

2.3 Specifying a Feature Structure

2.3.1 System Variables

```
=====================================================================
input   = the current FS before any rules are applied
?input  = the current FS after some rules have applied
root    = the root FS --- the very top level, no matter what the current FS may be
?it     = context-dependent indication of success or failure of the last call to rewrite;
          when rewrite call succeeds, ?it gets the resulting value, true or false;
          if rewrite fails, ?it gets the default value "empty"
=====================================================================
```

In most rules, root and input are sufficient. The "current feature structure" for a rule (system variable input) is specified using the rule's ON path. (See below.)

2.3.2 Dot Notation for Feature Paths

```
=====================================================================
feature.feature.etc
=====================================================================
```

Dots are used to specify paths within a feature structure. Paths begin with system variables (usually `root` or `input`). Thus `root.sem` points to the value of the SEM feature of the top-level feature structure; and `input.obje.reln` points to the value of the RELN feature within the OBJE feature of the `input` feature structure (before any rules have applied).

2.4 Application Constraints: When Should This Rule Be Used?

```
=================================================================
  attrib1 value1 ... attribN valueN
=================================================================
```

Application constraints should be written as attribute-value pairs, for example

```
:PHASE :PHASE1 :SUBPHASE :SUBPHASE3
```

Rule R can be used at time T only if all of its application constraints are in effect — that is, if the constraints have been set by some previous or currently-running rule. (The top-level rule MAINRULE is an exception in that its one constraint, :MAIN, is set by the transfer system at start time.)

Constraints are specified for a rule using the IN operator (see below).

In the Japanese-German system, most constraints are set using the REWRITE ... WITH ... operator combination (see above). Such settings remain in effect only during the REWRITE call.

The constraints have the effect of classifying the transfer rules: the rules which can currently apply form a subset within the entire ruleset. Constraints can classify hierarchically, since (a) below gives a more general rule subset than (b).

```
(a) :x :y

(b) :x :y :a :b.
```

2.4.1 Explicit Setting of Application Constraints

```
=================================================================
set
unset
=================================================================
```

ApplicationConstraint parameters can be set and unset explicitly using the SET and UNSET operators, but such setting is avoided in the Japanese-German transfer system. Instead, as already explained, constraints are usually set via explicit top-level calls to REWRITE (above); or less commonly by calls to embedded rewrites using ->, =>, etc (below).

The system can in theory extract constraint parameters from the discourse environment, but this potential is now unused.

2.5 The ON Operator

```
=====================================================================
ON <FeaturePath> AtomicFeatureStructure [IN ApplicationConstraints]
  body
END
=====================================================================
```

The ON operator has several functions: it specifies a test which a rule must pass before application; it provides a rule indexing mechanism; and it specifies the current feature structure (system variable input).

2.5.1 Which Rules Apply to This FS?

Concerning the test applied using the ON operator, assume we have this FS,

```
[[reln request]
 [agen ...]
 [recp ...]
 [obje [[reln SURU-V] ;<<<   note path <obje reln> SURU-V
        [agen ...]
        [obje ...]]]]
```

and we are asking whether the following transfer rule should be kept for further testing (we assume that it does at least have appropriate application constraints):

```
on <obje reln> SURU-V              ;<<< note path <obje reln> SURU-V
   in <application constraints> ;<<< assumed OK
   <body>
end
```

The test applied by the ON syntax is as follows: "Is there a feature path `<obje reln>` within the input FS whose value is SURU-V?" (By default, this path must be traced from the top level of the feature structure supplied by the REWRITE call. However, see below concerning :RECURSIVE rule application, which enables examination of embedded paths.)

There is such a path; so the rule would in fact be kept for further testing, using the rule BODY. These additional tests are typically pattern matching tests (below) and/or programmatic IF ... THEN ... tests (below) on the FS.

2.5.2 Rule Indexing

The ON syntax provides an indexing mechanism so that, given an FS, the system can quickly discover (among the rules having the right application constraints) all the rules which deserve further testing against the current feature structure using the rule BODY. The system maintains a discrimination net for this purpose.

2.5.3 The Input Feature Structure

The feature structure referred to by the system variable input, usable within a rule, is defined by the rule's ON path: input is the smallest feature structure which contains that path. For the last feature structure shown, the smallest feature structure containing the path `<obje reln>` SURU-V is the entire feature structure.

By contrast, if the ON path were instead <reln> SURU-V, the value of input would be

```
[[reln SURU-V]
 [agen ...]
 [obje ...]]
```

2.6 The BY Operator: How Should Rules BE Applied?

```
=====================================================================
rewrite <feature structure> with <constraints> BY [:loop :recursive :once]
=====================================================================
```

BY specifies rule application modes:

```
:once             = Apply each rule at most one time.

:loop             = Continue applying rules until all rules are exhausted;
                    thus, chaining is enabled: output of R1 can become input for R2;
                    endless looping (involving one or several rules) is a danger.

:recursive        = Seek applicable rules not only for the top-level feature
                    structure, but also recursively for embedded structures.

:loop :recursive  = Allow repeated rule use AND attempt to rewrite embedded
                    feature structures; rewriting with chaining may occur
                    at several levels.
```

Regarding :RECURSIVE mode, some additional explanation is needed. Normally, rewrite rules are sought only for the top-level FS. However, in :RECURSIVE mode only, rewrite rules are sought for all embedded FS's. Reconsider this rule:

```
on <reln> SURU-V
   in :language :japanese
  body
end
```

For the following top-level input,

```
[[reln request]
 [agen ...]
 [recp ...]
 [obje [[reln SURU-V]
        [agen ...]
        [obje ...]]]]
```

the rule would apply to this embedded feature structure

```
[[reln SURU-V]
 [agen ...]
 [obje ...]]
```

only if :RECURSIVE mode is in effect. Otherwise, since there is no path <reln> SURU-V from the top level, the rule would not be kept for further testing.

If both :RECURSIVE and :LOOP modes are in effect, rewriting may occur at several levels of embedding; furthermore, at each level, rule application will continue until no further rules can apply. Thus, at any level, the output of R1 may be the input for R2.

2.6.1 Rule Ordering

In the current version of the RWS system, if several rules are found which can apply to the same feature structure, all of them are tried in parallel. There will then be one output for each possible rule ordering. (Our test corpus, however, is simple enough that multiple transouts do not occur.)

In future implementations of the system, *conflict resolution* mechanisms will be introduced to decide among competing rules, e.g. by preferring the most specific rule (the one matching the most of the input, the one matching most constants, etc.) In the current implementaion, however, one can control rule ordering only by using application constraints. In the top-level rule MAINRULE, for example, the ordered series of REWRITE calls defines phases of rule operation.

It is true that one can "program" low-level rule ordering by exploiting the system's ability to explicitly set and unset application constraints; but this amounts to unstructured or "spaghetti" programming, and is discouraged as bad style.

2.7 Rule Input and Output

```
=========================================================================
in= FS1
out= FS2
return
=========================================================================
```

The IN= operator can be used in a rule if pattern matching tests are desired. The meaning is: "Does the input structure unify with (match against) the following feature structure?" If so, the feature structure which should be used to replace the input structure can be specified using OUT=. Here, for example, we replace the pattern ホテル の 手配 with a single symbol.

```
(rws:defrwschema2 jap056 t t
"on <parm restr reln> 手配-1 in :PHASE :Japanese
       in=

[[PARM !X4[[PARM !X2[]]
           [RESTR [[RELN 手配-1]
                   [ENTITY !X2]]]]]
 [RESTR [[RELN の-連体修飾]
         [ARG-1 [[PARM !X3[]]
                 [RESTR [[RELN ホテル-1]
                         [ENTITY !X3]]]]]
         [ARG-2 !X4]]]]

       out=
[[parm !X1[]]
 [restr [[reln ホテルの手配-1]
         [entity !X1]]]]
       end")
```

2.8 Tags for Feature Structures

```
=========================================================================
@ marks a "global" tag for an FS
! marks a "local" tag for an FS
? marks a pattern matching variable
=========================================================================
```

The first two tag markers (@ and !) are used to tag feature structures used in a rule, so that any feature structure must be written in full only once. The scope of a "local" tag is one feature structure; the scope of a "global" tag is the entire rule, and allows tag references between feature structures. Below, several local tags are shown. Notice that the empty structure is also tagged.

```
(rws:defrwschema2 jap056 t t
"on <parm restr reln> 手配-1 in :PHASE :Japanese
       in=

[[PARM !X4[[PARM !X2[]]
           [RESTR [[RELN 手配-1]
                   [ENTITY !X2]]]]]
 [RESTR [[RELN の-連体修飾]
         [ARG-1 [[PARM !X3[]]
                 [RESTR [[RELN ホテル-1]
                         [ENTITY !X3]]]]]
         [ARG-2 !X4]]]]

       out=
[[parm !X1[]]
 [restr [[reln ホテルの手配-1]
         [entity !X1]]]]
       end")
```

The ? convention makes it possible to specify locations within rules where any feature structure will match. (Type restrictions can be used to constrain variable matches. See below.) The scope of such a variable is the entire rule.

```
;;;filtering expr under desire
(rws:defrwschema2 arg1000mss t t
"on <reln> DESIRE in :PHASE :final-filter
   in= [[reln DESIRE]
        [EXPR ?EXPR]
        ?rest]
   out= [[reln desire]
         ?rest]
end")
```

Note the frequent use of "rest variables" like ?rest, ?rest1, etc. to match against any number of features. (Such names are conventional, but in fact any names can be used.)

```
[[reln SURU-V]
 [agen ?agen]
 [obje ?obje]
 ?rest]
```

2.9 Programmatic Manipulation of Feature Structures

We have seen the use of IN= to perform pattern matching tests on a rule's input feature structure, and the use of OUT= to specify the feature structure which should replace the input feature structure. The system also enables programmatic testing of the input and programmatic construction of the output.

For example, one can write

```
IF <input feature structure> HAS <feature>
THEN ADD <feature value list> TO <output feature structure>
ENDIF
```

Actions (like ADD, see below) can also be taken unconditionally.

2.9.1 The If/Then Construction

```
==========================================================================
if ... then ... [else] ... endif
==========================================================================
```

These are the standard controls. If/then constructions can be embedded.

2.9.2 Tests on Feature Structures

```
==========================================================================
FS1 IS FS2      (or as notational variant FS1 =? FS2) "FS1 identical to FS2?"
FS1 IS NOT FS2  (or as notational variant FS1 =! FS2) "FS1 not identical to FS2?"
FS HAS feature
FS HAS NOT feature (i.e. does not have)

AND
OR
NOT
==========================================================================
```

The above tests can be performed within if/then constructions.

2.9.3 Rewriting Operations on Feature Structures

```
==========================================================================
FS1 = FS2       (set FS1 to FS2)
FS1 == FS2      (set FS1 to unification of FS1 and FS2) [not used]

ADD FeatureValueList TO FeatureStructure
DELETE feature FROM FeatureStructure
TYPE OF FeatureStructure [not used]
SET variable TO FeatureStructure
==========================================================================
```

The above actions can be performed unconditionally or within if/then constructions.

2.9.4 The SWITCH Operator

```
=====================================================================
switch ... case ... default ... endswitch
=====================================================================
```

SWITCH is a specialized IF construction (cf. Common Lisp CASE) which facilitates repeated tests of the value of a given symbol. The following rule states that if the value of ?obje is [], the output (OUT=) will use VORTRAGEN-V; otherwise it will default to HALTEN-V.

```
(rws:defrwschema2 gen133mss t t
                "on <reln> 発表する-1  in :PHASE :J-G :TYPE :GENERAL
   in= [[reln 発表する-1]
        [agen ?agen]
        [obje ?obje]
        ?rest]
   switch ?obje
   case []
        out=
[[RELN vortragen-v]
 [agen ?agen]
 [obje []]
 ?rest]

   default
       out=
[[reln halten-v]
 [agen ?agen]
 [obje ?obje]
 ?rest]
   endswitch
end")
```

2.10 Embedded Rewrite Calls

```
=====================================================================
->      [with AttributeValueList] forgiving non-recursive embedded rewrite
-->     [with AttributeValueList] strict non-recursive embedded rewrite

=>      [with AttributeValueList] forgiving recursive embedded rewrite
==>     [with AttributeValueList] strict recursive embedded rewrite
=====================================================================
```

Embedded rewrite calls are like the explict REWRITE used in the top-level rule, except that they are used in lower-level rules. As in the case of top-level REWRITE, one can specify application constraints to constrain the rules to be considered.

The arrows differ along two dimensions: (1) the action which is taken if embedded rewriting fails and (2) whether the embedded rewrite call is or is not in :RECURSIVE mode.

- The short arrow rewrite calls (-> or =>) are *forgiving*, and allow a calling rule to continue after the rewrite fails. (A failing rewrite is one returning the reserved value "empty" because no rules at all succeeded.) The long arrow calls (--> or ==>) are *severe*, and force failure of the calling rule as a whole.

- The single arrow rewrite calls (-> and -->) are made without :RECURSIVE mode; the double arrow calls (=> and ==>) are made with :RECURSIVE mode.

An example of embedded rewriting:

```
on <reln> suru-v
  in :language :japanese

in= [[reln suru-v]
     [agen ?agent]
     [obje ?object]]
--> ?object                                    ;<<<<<<
       with :language :japanese :POS :Verbal   ;<<<<<<
out= ?object
end
```

That is, assuming the pattern-matching (following IN=) succeeds, rewrite ?object (the value of the OBJE feature), temporarily putting the constraints :LANGUAGE :JAPANESE :POS :VERBAL into effect. In this particular rule, the result (the rewritten ?object) becomes the output of the rule as a whole, replacing the pattern in which ?obje was embedded.

Assume the rule were invoked (in :RECURSIVE mode) on the following FS.

```
[[reln request]
 [agen !sp *speaker*]
 [recp *hearer*]
 [manner indirect]
 [obje [[reln suru-v]                              ;;do
        [agen !sp]
        [obje [[parm !x touroku-n]                ;;application
               [restr [[reln modify]
                       [arg1 !x]
                       [sloc kaigi-n]]]]]]]]]]   ;;conference
```

The FS which is the value of obje, [[reln suru-v] ...], would then be replaced by the result of a lower-level computation (using rules not shown). Here is one possible replacer:

```
[[reln tourokusuru-v]      ;;apply (for)
 [agen !sp]
 [obje []]
 [sloc kaigi-n]]           ;;conference
```

Embedded rewrites are sparsely used in the Japanese-German system.

2.11 Typed Feature Structures

The RWS contains facilities for specifying the type of a feature structure. This capacity is not used in the Japanese-German system. However, in a later chapter, we will recommend its future use, so we provide a very brief description here.

Using the system function DEFFSTYPE, a hierarchy of types can be defined. :HUMAN, for instance, can be defined as a subtype of :CREATURE, which might in turn be a subtype of :CONCRETE-OBJECT; :TEACHER could be defined as a subtype of :HUMAN; and so on [Hasegawa 1990: page 44].

One can then use the type as a prefix for feature structures in pattern matching. A variable, for instance, can be typed, so that unification is allowed only if the proposed binding has the proper type.

```
:human ?variable
```

Many current transfer rules in the Japanese-German system are too specific: they mention specific symbols when they should mention typed variables instead. For instance, one rule marks the specific symbols NAME-N and ADRESSE-N to indicate that they require possessive German determiners. (Thus NP's like *mein Name* and *Ihr Adresse* can be produced as translations for Japanese expressions lacking determiners.) In a later chapter, we will suggest that many current rules mentioning such specific noun symbols should eventually be replaced by more general rules using typed variables. (In the present example, a type like :PERSONAL-ID-INFO might be appropriate.)

This proposal assumes that the analysis system can also be upgraded to deliver anouts containing type information, or that type information can be added to anouts during pre-transfer. Preliminary experiments are now being performed with both techniques.

Chapter 3

Useful Transfer Functions

For our development enviroment, we have prepared several functions to simplify loading transfer rules and input structures, executing transfer, etc. These are stored in the following file unless otherwise indicated:

`as25:/home/tropf/gen-off/transfer/fns/transferfns.lisp`

If this file is loaded into Lisp, the transfer system loading described just below will be done automatically.

3.1 Loading

The transfer system is loaded into Lisp by loading the current `load.lisp` file. Presently (March 15, 1993):

```
(load "as26:/home2/nadine91/transfer/rws-v2/load.lisp")
```

The following new functions are useful for simplifying system loading calls. Path names are hard-coded to avoid typing. Of course, these must be updated before use.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
load-trules ()
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(defun load-trules ()
  (rws::remove-all-rw-rules2)
  (load <path for current ruleset>))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
load-anouts ()
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(defun load-anouts ()
  (setq fs (rws::push-fs-from-file <path for current anouts>)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
load-trules-and-anouts ()
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(defun load-trules-and-anouts ()
  (load-trules)
  (load-anouts))
```

3.2 Executing and Debugging

In order to use the functions in this section, the following files containing auxiliary functions must first be loaded. (They will be loaded automatically if `transferfns.lisp` is loaded.)

```
"as26:/home/tropf/generate-off/transfer/fns/readfns.lisp"
"as26:/home/tropf/generate-off/transfer/fns/debug-patch.lisp"
```

```
*****************************************************************************
dt (number &optional debug)
*****************************************************************************

(defun dt (number &optional debug)
  (if debug (bug-on)(bug-off))
  (rws::extrans fs number))
```

"Do transfer". (Assumes anouts are loaded.) Gives a quick way to execute transfer on a numbered anout, with or without debug traces. If the debug optional argument is non-nil, a useful set of debug traces are turned on (and turned off when execution is finished).

The following aux functions are used:

```
;;;Turn on useful set of debug traces.
(defun bug-on ()
  ;(setq rws::*debug-transfer-input* t)
  (setq rws::*debug-transfer* t)
  (setq rws::*debug-apply-rw-rule* t)
  (setq rws::*debug-rewrite-subfs* t)
  (setq rws::*debug-fs-match* t)
  (setq rws::*debug-parameter-setting* nil)
  (setq rws::*preference-fig* nil))

;;;Turn off most debug traces.
(defun bug-off ()
  ;(setq rws::*debug-transfer-input* t)
  (setq rws::*debug-transfer* t)
  (setq rws::*debug-apply-rw-rule* nil)
  (setq rws::*debug-rewrite-subfs* nil)
  (setq rws::*debug-fs-match* nil)
  (setq rws::*debug-parameter-setting* nil)
  (setq rws::*preference-fig* nil))
```

```
*****************************************************************************
trans (start end file &optional start-no)
*****************************************************************************

(defun trans (start end file &optional start-no)
  (transouts-on file)

  ;;;write unnumbered transouts to output file
  (dmt start end)

  ;;;add numbers to output file
  (if start-no (copy-loop file file start-no))

  (transouts-off))
```

(Assumes anouts are loaded.) Executes transfer on numbered anouts from `start` to `end`, and writes results to file. Results are numbered consecutively using comments, beginning with optional argument `start-no` (which can be different from `start`). (*NOTE! Results can be fed directly to generation programs using the development environment.* Existing transfer system functions, by contrast, produce output which must be modified before such use.)

The following aux functions are used:

```
;;;''Do multiple transfer''. Multiple version of dt, above,
;;;but no debug option. Execute transfer on numbered anouts
;;;between start and end.
(defun dmt (start end)
  (rws::trans-range-exec fs start end))

;;;Open file for transfer output.
(defun transouts-on (file)
  (input-off)
  (loop-on)
  (open_output file))

;;;Close file for transfer output.
(defun transouts-off ()
  (input-on)
  (loop-off)
  (close_output))

;;;Input to transfer, i.e. anout, should be printed.
(defun input-on ()
  (setq rws::*debug-transfer-input* t))

;;;Input to transfer, i.e. anout, should not be printed.
(defun input-off ()
  (setq rws::*debug-transfer-input* nil))

;;;In file debug-patch.lisp
;;;Turns on flag for function (also in debug-patch.lisp)
;;;which reformats transfer output.
(defun loop-on ()
  (setq rws::*debug-transfer-loop-format* t))

;;;In debug-patch.lisp
;;;Turns off flag for function (also in debug-patch.lisp)
;;;which reformats transfer output.
(defun loop-off ()
  (setq rws::*debug-transfer-loop-format* nil))
```

3.3 Creating and Changing Rules

A new rule is typically written in a rule file. However, it must be loaded into Lisp before use. Functions shown above permit the loading of an *entire rule file* at once. Loading of a *single new rule* during development is performed by copying an entire RWS:DEFRWSCHEMA2 definition from the rule file buffer to the Lisp buffer using a text editor (usually emacs). This definition function is interactively executed at the Lisp input prompt.[1]

If the rule is then changed, the same process must of course be repeated. The changed definition will override the earlier one.

[1]Warning! A known bug prevents the interactive or batch loading of MAINRULE when it is quite large. Consult the system manager.

3.4 Deleting Rules

The following function permits the removal of all rules at once.

```
*****************************************************************************
remove-all-rules ()
*****************************************************************************

(defun remove-all-rules ()
  (rws::remove-all-rw-rules2))
```

When it is necessary to delete a single rule, the following function is useful.

```
*****************************************************************************
rr (rule)
*****************************************************************************

(defun rr (rule)
  (rws::remove-rw-rule2 rule))
```

Chapter 4

Issues and Directions for Development

In this chapter, we discuss several issues which arose during implementation of the current Japanese-German transfer system and their implications for future work.

The most general issue concerns the level at which transfer operates: How are the symbols now interpreted, and what changes are possible and desirable? How is the boundary between transfer and its neighbors, analysis and generation, to be defined?

Second, we consider the possible use of language-neutral symbols in transfer. The aim is to minimize the number of transfer rules, and to enable some rule sharing among transfer components for different target languages.

Third, we consider several issues of semantic representation. Sometimes problems arose because information — often thematic information — was missing or insufficient in the analysis output. In other cases, all the necessary information was present, but its proper presentation or formatting was uncertain, e.g. for representing prepositional relations.

Lastly, we examine various *ad hoc* aspects of the current system. In many cases, greater generality can be achieved in the future by taking full advantage of the RWS' typing system for feature structures. Other problems, however, will require extensive reasoning — expert systems or other knowledge-based processing — for their solution.

4.1 Level of Transfer

ASURA's transfer components are often described as "semantic-level" transfer systems. But what does this phrase mean? We need to examine the level at which transfer operates.

The feature structures received and delivered by transfer mix a few purely semantic terminal symbols (NECESSITY, OBLIGATION, NEGATE) with a majority of terminal symbols bearing part-of-speech labels (GEHEN-V, KONFERENZ-N).

Presently, the symbol-to-word mapping is one-to-one. That is, only one German word is supplied during generation to express a given transout symbol, and it always respects the part-of-speech label.[1] The symbol GEHEN-V, for instance, is always expressed using the verb *gehen*. So the suffixed symbols presently represent dictionary forms of meaning-bearing *syntactic* objects, rather than purely semantic objects. (Of course, transfer can explicitly change parts of speech when it transforms input structures into output structures. But the point is that parts of speech are kept unless explicitly changed in this way.)

Thus the most accurate description of the current transfer system's level might be "syntactic-semantic" transfer:

[1]The current English transfer system is like the German system in this respect.

most of the symbols which transfer manipulates carry syntactic constraints, but a few do not. The symbols without syntactic constraints are derived from idiomatic Japanese constructions expressing illocutionary force (e.g. REQUEST), modality (DESIRE, POSSIBILITY, OBLIGATION, NECESSITY), etc. These may thus be expressed by target structures quite different from the source structures, even without explicit transformation during transfer. And these are the symbols which justify the description "semantic transfer".

The one-symbol-one-word generation policy was adopted for practical rather than theoretical reasons: it simplified processing during early stages of experimentation with ASURA. In future versions of the system, different policies might be adopted. The suffixed symbols could be interpreted more abstractly: they could be understood as indications of the original part of speech in the source, but not as restrictions on choice of expression. Generation, that is, could freely vary the part of speech as long as the entire utterance remained well-formed. Or a changed policy could go even further toward semantic interpretation of these symbols: during generation, not only free choice of part-of-speech but also free choice of lexical item could be allowed.

Of course, if generation ambiguity is allowed in the future, it will be necessary to provide mechanisms for choosing among competing genouts. Such mechanisms are difficult to design; but the naturalness of the output could justify the effort. [Hutchins and Somers 1992: page 137] give an excellent discussion of this tradeoff.

4.1.1 Dividing the Work

The issue of transfer level is in part a question about how the work should be divided among ASURA's components. As mentioned, transfer in ASURA is broadly defined: it performs tasks that might be performed by analysis or generation in other systems. Disambiguation, for instance, might be viewed as a job for analysis, rather than for transfer (or even pre-transfer). However, during the preparation of the Japanese-German transfer system, analysis was almost fixed; so most practical issues related to the interface between transfer and generation.

In general, the current German generation grammar is more economical and linguistically motivated than ASURA's English generation grammar [Ueda and Kogure 1990], in which convenience, speed, coverage, and modularity were stressed instead. (For German, an HPSG-like design was adopted. See [Tropf 1992].)

However, in some cases, the economy and principle which were gained in German syntax were lost in Japanese-German transfer: additional transfer rules or subphases often became necessary to provide the most convenient input for syntax, some of them *ad hoc*. The general tendency, when in doubt, was to complicate transfer to clarify generation.

For example, generation produces *auf Englisch* (where the noun lacks a determiner), but *ins Englische* (where the contracted determiner *das* is present). In order to give generation its usual signal for using or not using a determiner, transfer now assigns definiteness values to language noun symbols (ENGLISCH-N) according to the relevant prepositional symbols (AUF-LOC-P, IN-DIR-P). But transfer should in general operate on semantic symbols; and in this case, it seems unlikely that there is any real semantic difference between the two mentions of ENGLISCH-N. More likely, the usages differ only idiomatically depending on syntactic and lexical context. If so, perhaps the context-sensitive alternation should be handled by generation rather than transfer.

A second example: German does not distinguish the meanings which in Japanese are expressed using postpositions *made* and *made ni* ("until" and "by" in English): both are expressed using preposition *bis*. Should transfer neutralize this semantic difference (which appears as a difference between feature names in the anout)? Or should the semantic difference remain, so that generation must produce identical output for two different representations (i.e., must provide a many-to-one mapping)? Presently, transfer does neutralize the two.

A third example: The current syntax provides determiners only for common nouns. Months usually require determiners in our corpus, as in *ab dem vierten August*. Thus transfer now specifies months as common nouns (by providing the appropriate semantic format). However, this date construction may again be idiomatic, and thus a candidate for special handling in syntax. (The date expression is not a normal ordinal expression — were there three previous Augusts? Rather, it is clear that the original meaning is "the fourth *day* of August". The determiner is probably that of the noun *Tag*, which is now conventionally elided.)[2]

[2]By the way, our current transfer-based specification of a particular noun as always common or always proper is in any case

In such cases, the transfer output, which already has many syntactic qualities, comes to look more and more like a deep syntactic representation rather than a shallow semantic one. In the future, it will be worthwhile to clarify the depth at which transfer operates and to define as sharply as possible the line between transfer and its neighbors.

4.2 Avoiding Unnecessary Transfer

No matter how transfer symbols are interpreted in the future, perhaps a more language-independent representation can be considered while retaining the transfer paradigm.

Presently, a lexical symbol like 会議-N is transferred to a symbol like KONFERENZ-N. But when direct translation is possible (as it usually is within our corpus) this step is unnecessary. Analysis could instead produce a "neutral" symbol like *CONFERENCE-N, and generation could accept this as input without change. ("Neutral" here means "usable for one or more languages, but not necessarily all languages". The set of neutral semantic symbols could be developed step by step: new symbols could be added whenever necessary.)[3] A large number of the current transfer rules would then simply disappear: the transfer component for both Japanese-English and Japanese-German would shrink.

Of course, not all transfer rules can disappear, because direct translation is not always possible. The target language may prefer a more precise symbol, or a more general symbol, or a semantic reorganization involving several symbols: in such cases, the transfer component must go to work. But if transfer works only when necessary, it should work faster and its task may be clarified.

Further, since the remaining rules would be written in terms of neutral symbols, some of them might be usable for several target languages, especially when targets are as similar as English and German. It might therefore become possible to share some rules among transfer systems.

4.3 Semantic Representation

We now turn to semantic representation issues which arose during development. Sometimes not enough information is present to enable necessary syntactic choices. Sometimes information is all there, but its proper format is uncertain.

4.3.1 Missing Semantic Information

Additional semantic information sometimes seems necessary in order to permit generation to make necessary choices. Thematic information — information concerning focus and presupposition — often seems to be involved.

Precedence

The most obvious case relates to *surface ordering* (precedence) in German. Transfer currently gives generation no information regarding ordering. Generation should thus produce all possible orderings; but this result was unacceptable, so generation temporarily introduced a fixed ordering into its subcategorization statements ([Tropf 1992]).

---

oversimplified. A particular noun which is normally proper can be treated as common in marked contexts, and vice versa: compare *Mayumi* as a proper noun and *der Mayumi von dem ich rede*, in which *Mayumi* is treated as common.

[3] A given symbol S1 may not be directly expressible in a given language L1; so the symbol set is not exactly an interlingua. However, it will be possible to examine the *subset* of symbols which can be directly expressed in language L1, or the *intersection* between the respective subsets of L1 and L2.

Kein

There is a similar problem with our treatment of *kein*: we receive no thematic information from analysis which would allow us to distinguish *Es gibt nicht ein Rabatt* from *Es gibt keinen Rabatt.* Thus, where the *kein* pattern is desired, we produce it *ad hoc* according to the specific surrounding symbols.[4]

The Copula

As a final, more extended example of problems arising from missing thematic information, consider relations with or without the copula. On the logical level (and ignoring thematic differences), the following two utterances should perhaps be represented identically: (1) *ein weites Forschungsgebiet* ("a broad research field") (2) *ein Forschungsgebiet, das weit ist* ("research field which is broad"). But once again, we did need to control the choice between these two outputs; so distinctive transout structures were needed. We adopted the following conventions:

```
[[parm !X7[[parm !X4[]]
           [restr [[reln Forschungsgebiet-n]
                   [entity !X4]]]]
 [restr [[reln weit-a]
         [obje !X7]]]]]

weites Forschungsgebiet
(NOTE: No TENSE, ASPT, etc. is provided under WEIT-A in this case.)

-------------------------------------------------------------------------

[[parm !X7[[parm !X4[]]
           [restr [[reln Forschungsgebiet-n]
                   [entity !X4]]]]
 [restr [[reln sein-v] ;<<<<<<<COPULA represented in transout
         [tense ...]
         etc.
         [obje !X7]
         [iden [[reln weit-a]
                [obje !X7]]]]]]]

Forschungsgebiet, das weit ist
(NOTE: TENSE and other aux features are included under SEIN-V.)
```

The relative clause representation is derived from the following representation for an independent clause, *Das Forschungsgebiet ist weit.*

```
[[reln sein-v] ;<<<<<<<COPULA represented in transout
 [tense ...]
 etc.
 [obje !X7[[parm !X4[]]
           [restr [[reln Forschungsgebiet-n]
                   [entity !X4]]]]
 [iden [[reln weit-a]
        [obje !X7]]]]
```

Note that the following representation presently cannot be expressed as an independent clause:

[4] In the transout, we distinguish the two patterns using the scope of the relation NEGATE: it dominates the main verb (ES_GEBEN-IDIOM) in the first case, but dominates only its OBJE value RABATT-N in the second case.

```
[[reln weit-a]
 [obje !X7[[parm !X4[]]
          [restr [[reln Forschungsgebiet-n]
                  [entity !X4]]]]]]
```

These conventions are consistent with those used in the Japanese anout, where the Japanese copula *da* is similarly shown as a relation. Even so, they seem artificial, and probably introduce syntactic elements into a representation which is intended to remain semantic.

The copula problem is a broad one, which affects not only adjectival constructions but nominal and prepositional constructions as well:

- *Johann, der Mann*
  Johann, der ein Mann ist
  Johann ist ein Mann
- *der Tisch in dem Zimmer*
  der Tisch, der in dem Zimmer ist
  der Tisch ist in dem Zimmer

4.3.2 Problems with Semantic Format

We have seen that problems sometimes occur when not enough semantic information is provided. In other cases, all of the necessary information is present, but its format, scoping, etc. may be uncertain or inconsistent.

Prepositional Relations

The most important format problem concerns the representation of prepositional relations. In the current system, adjunct prepositional phrases are often represented using specially defined features. For instance, *Ich esse in dem Zimmer* would be represented using the feature SLOC ("spatial location") as follows:

```
[[reln essen-v]
 [tense ...]
 ...
 [agen !X7[[label *speaker*]]]
 [sloc [[parm !X4[]]
        [restr [[reln Zimmer-n]
                [entity !X4]]]]]]
```

But if there is a special feature for the meaning "in", why not one for "outside of"? We are tempted to allow a new feature (in effect, a new deep case) for every valence-bound or adjunct preposition. But then syntax must handle an open set of case representations; and the generality and robustness of the generation grammar is damaged.

Instead, a small and fixed set of features representing deep case should contain a large and more flexible set of relations as values. At the same time, the distinction between valence-bound and adjunct prepositional relations should be clarified: valence-bound prepositional expressions should be represented via deep cases (i.e. dedicated features); and adjunct expressions can be collected under a single deep case, e.g. ADJUNCTS,[5] giving the following revised representation for the sentence above:

[5] A similar ADJUNCTS feature convention was considered for early versions of ASURA, but was not adopted. See [Kume and Nagata: page 13].

```
[[reln essen-v]
 [tense ...]
 ...
 [agen !X7[[label *speaker*]]]
 [adjuncts [[reln in-loc-p]
            [obje !X8[[parm !X4[]]
                     [restr [[reln Zimmer-n]
                             [entity !X4]]]]]]]]
```

If there are several adjuncts, the value of the ADJUNCTS feature would be a conjoined expression.

Other Semantic Representation Problems

We can briefly mention a few more problems in semantic representation:

- We are uncertain about the proper representation for modification of NEGATE (e.g. for *ueberhaupt nicht, nicht viel*).
- Our treatment of relations between clauses is inexact: the imprecise representation of *no de* as CAUSE in mset 104 gives one example: 会議の案内書をお送り致しますので、それをご覧下さい. ("I'll send the Conference Announcement to you, *so* please have a look at it.") For the present, we have chosen to express the symbol using a full stop in German: *Ich werde Ihnen die Konferenzankuendigung schicken. Bitte werfen Sie einen Blick hinein!*[6]
- The scope of modifiers is often debatable. Consider for instance mset sentence 34, *Als Anmeldegebuehr werden fuenfunddreissigtausend Yen pro Person verlangt.* (from登録費としてお一人三万五千円が必要です。"As the registration fee, thirty-five thousand yen per person is necessary."). Analysis views *tourokuhi to shite* ("as the resitration fee") as a modifier of the top-level verb *hitsuyo desu.* For German, the phrase was instead seen as a modifier of *Yen.*
- We use a specialized representation to indicate, for German noun symbols, the definiteness or determiner, number, and owner if any: an INDEX feature is included as part of the noun symbol pattern. It can be considered shorthand for a more semantically consistent but less convenient representation. We are in any case unsure about the proper representation.

  The INDEX feature is included along with the features PARM and RESTR, which together denote the semantic structure of common nouns. PARM names an unrestricted variable, e.g. !V[], and RESTR indicates the restrictions on its range, in the manner of the logical operator SUCH-THAT. Thus the following expression denotes a variable !V such that the relation JAHR-N is true of !V — that is, a year. Further, for this particular instance of the class of years, DETERM, NUMBER, and OWNER have the values shown.

```
[[PARM !V[]]
 [RESTR [[RELN Jahr-n]
         [ENTITY !V]]]
 [INDEX [[DETERM DEFART]
         [NUMBER SING]
         [OWNER [[LABEL *UNKNOWN*]]]]]]
```

4.4 Ad Hoc Aspects of the Present Transfer System

We now examine various *ad hoc* aspects of the current system. In many cases, greater generality can be achieved in the future by exploiting the RWS' mechanisms for typing feature structures. In other cases, extensive reasoning — expert systems or other knowledge-based processing — will be required.

[6] German morphology required modifications to permit multi-sentence output. See [Seligman 1993].

4.4.1 Possible Generalizations Using Types

We first discuss possible uses of typed feature structures to reduce the *ad hoc* character of several sorts of transfer rules. Typing facilities were still incomplete and thus unavailable during our development period.

Disambiguations

In several cases, a Japanese expression has only one translation in our corpus, but would have more potential translations in a larger corpus. We now ignore such potential ambiguity. (However, we do provide comments on specific *ad hoc* rules.)

In the future, instead of translating input symbols unconditionally, translations can be made sensitive to the symbols in the local context. For maximum generality, such pattern matching can use typed variables rather than constant symbols.

A partial listing of current *ad hoc* translations:

```
待つ-1                          --> erwarten-v

方-2                            --> Leute-n

誰か                            --> jemand_anders-idiom

発表する-1/zero object          --> vortragen-v (etwas)
         /non-zero object       --> halten-v (Vortrag-n)

割引する-1                      --> es_geben-idiom (Rabatt-n)

ない-adjective                  --> negate/es_geben-idiom

書く-1                          --> stehen-v

聞く-3                          --> wenden-v

取る-1                          --> bekommen-v

取れる-1                        --> possibility/bekommen-v

見る-1                          --> hineinwerfen-v (Blick-n)

要る-1                          --> passive/verlangen-v

こちら-1                        --> hier
```

Other Opportunities for Typing

Other possible uses of the type system for greater generality:

- The Japanese particle *no* presents an especially difficult and common disambiguation problem. Analysis retains the information that *no* appeared in the source by using the special relation symbol の-連体修飾. In our transfer system, this symbol is replaced by the general-purpose relator MODIFY (which is expressed as a genitive construction) in the default case; or by specific prepositional symbols, depending on the specific word symbols in the context. For example, we now rewrite MODIFY as FUER-P if the related symbols are TEILNAHMEGEBUEHR-N and KONFERENZ-N, thus translating 会議の参加料

("the attendance fee for the conference") as *die Teilnahmegebuehr fuer die Konferenz.* In the future, type specifications of the related symbols should be used instead.

- We must sometimes provide German possessive determiners for words like *Name* and *Adresse* when determiners are absent in the source. Presently, in specifying the symbols requiring possessive modifiers, we mention the specific symbols NAME-N, ADDRESS-N, PHONE_NUMBER-N. A type specification, e.g. :PERSONAL-INFO, should be used instead. Possessive determiners are also needed for nouns derived from verbs, e.g. *Teilnahme.* We presently specify the particular noun TEILNAHME-N. A type, e.g. :VERBAL-NOUN, should be substituted.

- Presently, by request of the generation component, the TENSE feature is deleted under stative relation symbols. We now mention such symbols (like WEIT-A) specifically. A type specification, e.g. :STATE, should be used.

- Numerous rules move TENSE, WH elements, or other features according to the preference of the German generator. These rules presently mention specific semantic symbols. For instance, the TENSE movement rules mention specific "abstract verbs" like OBLIGATION, POSSIBILITY, etc. Types, e.g. :MODAL, should be used.

- Certain rules require the recognition of simple or compound noun-type symbols. (Rules relating to illocutionary force type REQUEST-PERMISS — the requesting of permission, as in *Koennte ich ... haben?* — have this requirement, for instance.) Presently, the rules refer to special symbols which represent noun-like configurations (PARM plus RESTR), or to specific conjunction symbols like UND-CONJ. Types like :ENTITY or :NOUN-RELATOR should be used instead.

4.4.2 Remaining Hard Problems

We should not give the impression that every problem with generality can be solved using typed features. Several well-known difficulties in Japanese-German or Japanese-English translation will require extensive knowledge processing. Presently, we rely on temporary fixes: heuristics which are greatly oversimplified, or patches which we fully realize are *ad hoc.* In these cases, we do not pretend to solve the problems in question; instead, we simulate their solution, providing nothing more than placeholders ("hooks") for future programs. Let us briefly survey some hard problems and our temporary methods of handling or avoiding them.

Definiteness and Number

Most nouns in our corpus are either definite or indefinite and either singular or plural throughout the corpus. For these nouns, we treat definiteness and number as fixed attributes of the semantic symbol. The few case in which both definite and indefinite or both singular and plural occur for a single noun are handled by *ad hoc* rules which assign definiteness or number based on specific symbols in the context.

Tense

Several German target sentences should ideally use the future tense. But of course the Japanese source sentences employ only the present (non-past) tense. We hoped that adverbials in the context might supply sufficient clues to select future tense, but a study produced disappointing results: almost all adverbials which occurred with the relevant verbs in our corpus could indicate other tenses equally well.

Thus we adopted a simpler heuristic: future tense is now given only for relations dominated by illocutionary force type PROMISE. This IFT, in turn, is produced by analysis only when the source contains *-sasete itadakimasu* (lit. "I will have it done"), e.g. in mset 15: 登録用紙は至急送らせていただきます。("I'll send you a registration form/will have a registration form sent immediately."), giving *Ich* werde *Ihnen sofort ein Anmeldeformular schicken.* Of course, the problem of determining the proper target tense may in general be very complex.

Aspect

We must distinguish between German passive with *sein* and passive with *werden*, for instance in translating mset 65, 参加料には、予稿集代と歓迎会費が含まれています。("The proceedings and the reception fee are included in the attendance fee.") giving *Die Tagungsbaende und die Empfangskosten* sind *in der Teilnahmegebuehr enthalten.* and 101, 会議は 8 月 22 日から 25 日まで京都国際会議場で開催されます。("The Conference will take place from August 22nd to 25th at the Kyoto International Conference Center.") giving *Die Konferenz wird ... abgehalten werden.*

We use the feature SEM-ASPE for this purpose, with values STAT and PROG respectively. (This use of the SEM-ASPE feature is in fact the only use of aspect information at present.) Unfortunately, the values of SEM-ASPE now recieved from analysis do not correspond cleanly to the desired passive types in the targets. So, to give the desired target values for now, *ad hoc* rules were written. They should of course be revised. At the same time, the analysis of Japanese aspect should be reconsidered.

Intimacy

German needs an indication of the degree of intimacy of an utterance to distinguish between *Sie* and *du*, but presently there is no such indication in the analysis results. (Politeness is indicated, but this is a different matter.) Thus we have simply written a transfer rule giving a default INTIMACY of LOW (giving *Sie*) for every utterance. This is in fact a realistic assumption for the present corpus.

Politeness

ASURA's treatment of politeness needs study. Currently, analysis identifies and lists politeness markers, but does not assign a politeness degree. This assignment is made by transfer — rather crudely, at present. Following the method developed for Japanese-English transfer, PRAG POLITENESS DEGREE is determined by counting the number of explicit politeness markers: degree 1 means "one explicit politeness marker is present". But of course this measurement method largely depends on the length of an utterance, and ignores qualitative differences.

Further, there is a mismatch in the interpretation of levels between Japanese and German. Japanese sees degree 1 as "slightly polite"; but German interprets 1 as as "neutral". For the present corpus, the German target requires at least degree 2 in all utterances to avoid rudeness. And so, as a temporary adjustment, we have rewritten the transfer rules so that degree 2 is given for either one or two politeness markers.

Possessive Determiners

Often the German target requires possessive determiners (*mein*, *Ihr*) but the Japanese source has no determiners at all. Our method of supplying these determiners is temporary. We now break the job into two phases:

First, as already described, we mark noun-type symbols which require possessives, e.g. NAME-N, ADDRESS-N, PHONE_NUMBER-N, deverbal nouns like TEILNAHME-N. (We assign [[label *OWNER*]] as a temporary value of the INDEX OWNER path of such symbols.) We have already recommended possible methods of generalizing this pre-marking phase.

Second, we must identify the specific owner in the current utterance, overwriting the temporary marker. We temporarily use a very simple heuristic: if the IFT is INFORM, *OWNER* becomes *SPEAKER*; else it becomes *HEARER*. This heuristic is not unreasonable, and works well in our limited corpus; but it will certainly fail in larger corpora, especially when there are several parties to the discourse.

Chapter 5

The Current Japanese-German Transfer Ruleset

This chapter provides brief notes on each micro-phase of the present Japanese-German transfer ruleset. Please refer also to the more general comments above, which will not be repeated here.

5.1 MAINRULE: Top-level Control

We have simplified problems of rule ordering by specifying a large number of micro-phases (calls to the REWRITE operator). Fewer phases could be used in the future.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                         ;;;
;;;=========================================================================;;;
;;;                Japanese-German Transfer Rules for ASURA                 ;;;
;;;=========================================================================;;;
;;;                                                                         ;;;
;;;                             Mark Seligman                               ;;;
;;;                               1992-1993                                 ;;;
;;;                       seligman@atr-la.atr.co.jp                         ;;;
;;;                                                                         ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(rws:defrwschema2 mainrule t t
"on <> :main
   set ?SP to input.prag.speaker
   set ?HR to input.prag.hearer
   in= [[sem ?sem]
        ?rest]
   input = [[sem [[reln UNKNOWN-IFT]
                  [agen ?sp]
                  [recp ?hr]
                  [obje ?sem]]]
            ?rest]

;;;PRE-TRANSFER
```

```
   rewrite input.sem with  :PHASE :BAD-ANAL-PATCH                  by :RECURSIVE

   rewrite input.sem  with :PHASE :ELLIPSIS-INIT                   by :RECURSIVE
   rewrite input.sem  with :PHASE :ELLIPSIS-RESOLUTION             by :RECURSIVE
   rewrite input.prag with :PHASE :ELLIPSIS-RESOLUTION             by :RECURSIVE

   rewrite input.prag with :PHASE :JAPANESE :PRSP :INIT            by :RECURSIVE
   rewrite input.sem with  :PHASE :JAPANESE                        by :LOOP :RECURSIVE

   rewrite input.sem with  :PHASE :IF-REDUCE :TYPE :GENERAL        by :LOOP :RECURSIVE
   rewrite input      with  :PHASE :IF-REDUCE :TYPE :DEFAULT        by :RECURSIVE
   rewrite input      with  :PHASE :IF-REDUCE :TYPE :OMOU-FILTER    by :RECURSIVE

   rewrite input.prag with :PHASE :PRAG-INIT                       by :LOOP :RECURSIVE
   rewrite input.prag with :PHASE :PRAG-POLITENESS                 by :RECURSIVE
   rewrite input.prag with :PHASE :PRAG-FILTER                     by :RECURSIVE
   rewrite input.prag with :PHASE :PRAG-INTIMACY-DEFAULT           by :ONCE
   rewrite input.prag with :PHASE :PRAG-POLITENESS-DEFAULT         by :ONCE

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;MAIN-TRANSFER

   rewrite input.sem with  :PHASE :J-G :TYPE :IDIOM                by :LOOP :RECURSIVE
   rewrite input.sem with  :PHASE :J-G :TYPE :GENERAL              by :LOOP :RECURSIVE
   rewrite input.sem with  :PHASE :J-G :TYPE :DEFAULT              by :LOOP :RECURSIVE

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;POST-TRANSFER

   rewrite input.sem with  :PHASE :VOCAB-SPECIFICATION             by :LOOP :RECURSIVE
   rewrite input.sem with  :PHASE :OWNER-SPECIFICATION-I           by :RECURSIVE
   rewrite input.sem with  :PHASE :OWNER-SPECIFICATION-II          by :RECURSIVE
   rewrite input.sem with  :PHASE :TENSE-SPECIFICATION             by :RECURSIVE
   rewrite input.sem with  :PHASE :SEM-ASPE-SPECIFICATION          by :RECURSIVE
   rewrite input.sem with  :PHASE :MODIFY-SPECIFICATION            by :RECURSIVE
   rewrite input.sem with  :PHASE :DETERM-SPECIFICATION            by :RECURSIVE
   rewrite input.sem with  :PHASE :IDEN-SPECIFICATION              by :RECURSIVE
   rewrite input.sem with  :PHASE :COORD-SPECIFICATION             by :RECURSIVE
   rewrite input.sem with  :PHASE :TLOC-SPECIFICATION              by :RECURSIVE

   rewrite input.sem with  :PHASE :TENSE-MOVEMENT                  by :RECURSIVE
   rewrite input.sem with  :PHASE :NEGATE-MOVEMENT                 by :RECURSIVE
   rewrite input.sem with  :PHASE :WH-MOVEMENT                     by :LOOP :RECURSIVE
   rewrite input.sem with  :PHASE :A-MOVEMENT                      by :ONCE
   rewrite input.sem with  :PHASE :CONECT-MOVEMENT                 by :RECURSIVE
   rewrite input.sem with  :PHASE :DATE-MOVEMENT                   by :RECURSIVE
   rewrite input.sem with  :PHASE :ORDINAL-MOVEMENT                by :RECURSIVE
   rewrite input.sem with  :PHASE :COMPOUND-MOVEMENT               by :RECURSIVE
   rewrite input.sem with  :PHASE :LONG-MOVEMENT                   by :RECURSIVE
   rewrite input.sem with  :PHASE :ZU-MOVEMENT                     by :RECURSIVE
   rewrite input.sem with  :PHASE :UNGEFAEHR-MOVEMENT              by :RECURSIVE
   rewrite input.sem with  :PHASE :PREIS-MOVEMENT                  by :RECURSIVE
   rewrite input.sem with  :PHASE :SUBORD-MOVEMENT-AND             by :RECURSIVE

   rewrite input.sem with  :PHASE :FINAL-FILTER                    by :RECURSIVE

   return input
end")
```

5.2 Pre-transfer

5.2.1 Bad-analysis-patch

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                          ;;;
;;;    :PHASE :BAD-ANAL-PATCH                    by :RECURSIVE               ;;;
;;;                                                                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;===============================================================================
;    :PHASE :BAD-ANAL-PATCH                    by :RECURSIVE
;===============================================================================
```

In a small number of cases, errors appeared in the analysis output. There was no opportunity to revise the analysis, so this phase was inserted to permit temporary patching.

Some examples:

- BADANALPATCH1000MSS. Patches a scoping problem. The anout now reads "single room for [7000 yen which is in the Kyoto Prince Hotel]". Should be "[single room for 7000 yen] which is in the Kyoto Prince Hotel".
- BADANALPATCH1001MSS. Patches an error in which the PASSIVE form was not properly recognized. "Papers which present" in the anout should be "papers which are presented".
- ARG1002MSS. For 169 *Moushiwake arimasen ga, kochira DEWA senmonteki-na shitsumon ni o-kotae dekimasen.* DE should not be analyzed as indicating LOCT: this is special idiomatic use of *de*, equivalent to *WA/GA*. ARG1004MSS patches a similar problem for 231.
- OWNERFILTER1000MSS. A last-minute version of analysis sometimes supplies the OWNER feature, disrupting transfer's established handling of the same feature (see below). For now, we avoid revision by simply deleting OWNER when it is given by analysis.

5.2.2 Ellipsis-resolution

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                          ;;;
;;;    :PHASE :ELLIPSIS-INIT             by :RECURSIVE                       ;;;
;;;    :PHASE :ELLIPSIS-RESOLUTION       by :RECURSIVE                       ;;;
;;;                                                                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Most ellipsis resolution rules were adapted from the existing Japanese-English transfer rule system. Some embody interesting heuristics, but most are relatively *ad hoc.*

```
;===============================================================================
;    :PHASE :ELLIPSIS-INIT             by :RECURSIVE
;===============================================================================
```

ELPS001 initializes embedded rewrite with :MOOD :INTERROGATIVE or :MOOD :DECLARATIVE depending on the presence or absence of relation S-REQUEST.

A typical :MOOD :INTERROGATIVE rule is ELPS002, below, which supplies *SPEAKER* as the AGEN of any action under reln ば-CONDITIONAL ("if", often part of idiomatic "should" constructions). Effect is to supply "I" in interrogative contexts like "What should () do?"

```
;;;What should () do? --> What should I do?
(rws:defrwschema2 elps002 t t
"on <reln> ば-conditional IN :MOOD :INTERROGATIVE
   in= [[reln ば-conditional]
        [iden [[reln ?action]
               ?rest1]]
        ?rest2]

   if input.iden.agen =? [] then
     input.iden.agen = root.prag.speaker
   endif
   RETURN INPUT
END")
```

```
;=============================================================================
;    :PHASE :ELLIPSIS-RESOLUTION     by :RECURSIVE
;=============================================================================
```

Ellipsis resolution rules in which it does not matter whether the utterance is declarative or interrogative.

Typical rule is ELPS011, which supplies "I" and "you" in the pattern "() receive the the favor from () of () letting () [action]", giving "I receive the favor from you of you letting me [action]".

```
;;;() receive the the favor from () of () letting () <action> -->
;;;  I receive the favor from you of you letting me <action> -->
(rws:defrwschema2 elps011 t t
"on <obje reln> させる-PERMISSIVE IN :PHASE :ELLIPSIS-RESOLUTION
    in=       [[reln てもらう-RECEIVE_FAVOR]
               [agen ?AGEN]
               [recp ?recp]
               [obje [[reln させる-PERMISSIVE]
                      [agen ?agen2]
                      [recp ?recp2]
                      [obje ?obje]
                      ?rest1]]
               ?rest2]

   if input.AGEN =? [] then
     input.agen = root.prag.speaker
   endif
   if input.recp =? [] then
     input.recp = root.prag.hearer
   endif

   if input.obje.agen =? [] then
     input.obje.agen = root.prag.hearer
   endif
   if input.obje.recp =? [] then
     input.obje.recp  = root.prag.speaker
   endif

   if input.obje.obje.recp =? [] then
     input.obje.obje.recp = root.prag.hearer
```

```
   endif
   RETURN INPUT
end")
```

5.2.3 Rewriting Japanese Symbols

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                          ;;;
;;;   :PHASE :JAPANESE :PRSP :INIT         by :RECURSIVE                     ;;;
;;;   :PHASE :JAPANESE                     by :LOOP :RECURSIVE               ;;;
;;;                                                                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Many rules which rewrite Japanese structures were adapted from the existing Japanese-English transfer rule system. Several rules contain embedded rewrites.

```
;===============================================================================
;    :PHASE :JAPANESE :PRSP :INIT          by :RECURSIVE
;===============================================================================
```

Initializes path <root.pragmatic.prsp> (presupposition) for utterances with *mo* ("also").

```
;===============================================================================
;   :PHASE :JAPANESE                       by :LOOP :RECURSIVE
;===============================================================================
```

Rules which reduce Japanese structures, or regularize them, or make them more abstract. Idiom consolidation (example in introduction) occurs here.

- A reduction example: JAP013, which simplifies the idiomatic construction ことはできる (*koto wa dekiru*), leaving only できる-POSSIBLE.
- Regularization: JAP015 transforms 欲しい-DESIRE into same format as たい-DESIRE plus てもらう-RECEIVE_FAVOR ("want to recieve favor").
- Idiom recognition: JAP056 consolidates pattern ホテル の 手配 ("hotel arrangements") into single symbol ホテルの手配-1, suitable for later replacement by German symbol HOTELBUCHUNG-N. Discussed in the introduction.

Note a confusing use of embedded rewrites for some idiom consolidation: e.g. to replace the pattern 行なう／割引 (*okonau/waribiki*, "make a discount") with the single symbol 割引する-1. When 行なう is recognized, its OBJE, 割引-1, is rewritten (using special application constraints) as 割引する-1. This result then replaces the input pattern containing 行なう in the higher-level rule.

```
;;;<object> okonau
(rws:defrwschema2 jap009 t t
"on <reln> 行なう -1 in :phase :japanese
   in= [[reln 行なう -1]
        [obje ?obje]
        ?rest]
```

```
   -> ?obje with :phase :japanese :event-or-object :event
   if it is false then fail endif
   add ?rest to ?obje

   if ?input.agen then
      ?obje.agen = ?input.agen
   endif

   out= ?obje
end")
```

```
;;;waribiki --> waribikisuru
(rws:defrwschema2 jap047 t t
"on <restr reln> 割引-1 in :Phase :Japanese :event-or-object :event
       in= [[parm !X[]]
        [restr [[reln 割引-1]
                [entity !X]]]
        ?rest]
       out= [[reln 割引する -1]
         [agen []]
         [obje []]
         ?rest]
       end")
```

5.2.4 Identifying Illocutionary Force

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                            ;;;
;;;  :PHASE :IF-REDUCE :TYPE :GENERAL         by :LOOP :RECURSIVE              ;;;
;;;  :PHASE :IF-REDUCE :TYPE :DEFAULT         by :RECURSIVE                    ;;;
;;;  :PHASE :IF-REDUCE :TYPE :OMOU-FILTER     by :RECURSIVE                    ;;;
;;;                                                                            ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

For reducing and identifying illocutionary force types such as INFORM, QUESTIONIF, REQUEST, etc. Most rules adapted from existing Japanese-English ruleset.

```
;=============================================================================
;   :PHASE :IF-REDUCE :TYPE :GENERAL         by :LOOP :RECURSIVE
;=============================================================================
```

A typical rule is IFT015, which transforms pattern UNKNOWN-IFT/S-REQUEST/INFORMIF into QUESTIONIF. Thus "would you inform me whether X is true" becomes "Is X true?"

```
(rws:defrwschema2 ift015 t t
"on <obje reln> S-REQUEST in :PHASE :IF-REDUCE :Type :General
   in= [[reln UNKNOWN-IFT]
        [agen ?agen]
        [recp ?recp]
        [obje [[reln S-REQUEST]
               [AGEN ?AGEN]
               [RECP ?RECP]
```

```
               [OBJE [[reln INFORMIF]
                      [agen ?recp]
                      [recp ?agen]
                      [obje ?obje]]]
               ?rest]]]

   out= [[reln QUESTIONIF]
         [AGEN ?AGEN]
         [RECP ?RECP]
         [OBJE ?OBJE]
         ?rest]
end")
```

Note the transformation of numerous Japanese indirect request patterns into plain REQUEST for easier translation. E.g. IFT031MSS recognizes "Is it not possible for me to recieve the favor of your doing [action]?" and produces REQUEST with [attd interrogative]. Generation will express this pattern as *Koennten Sie [action], bitte?*

```
;;;# 171: "Sochira o mite-itadakenai deshou ka?"
;;;This transformation happens after う-GUESS is reduced via ift013
(rws:defrwschema2 ift031mss t t
"on <reln> QUESTIONIF in :PHASE :IF-REDUCE :Type :General
   in=
[[RELN QUESTIONIF]
 [AGEN ?agen]
 [RECP ?recp]
 [OBJE [[RELN NEGATE]
[aspt ?aspt]
[OBJE [[RELN える-POSSIBLE]
       ?rest3
       [OBJE [[RELN てもらう-RECEIVE_FAVOR]
      ?rest4
      [OBJE [[reln ?obje-reln]
     ?rest5]]]]]]]]]

   root.prag.politeness = [[degree 3]]

   out=
[[reln REQUEST]
 [attd interrogative]
 [agen ?agen]
 [recp ?recp]
 [obje [[reln ?obje-reln]
[aspt ?aspt]
?rest5]]]
end")
```

```
;=============================================================================
;   :PHASE :IF-REDUCE :TYPE :DEFAULT           by :RECURSIVE
;=============================================================================
```

Once the basic IFT type has been set by the previous phase, makes numerous adjustments. For example, for an INFORM ift, IFT039 moves any CONECT feature (from *sore de*, "well, then" etc.) from with an embedded OBJE feature to the top level.

```
;=============================================================================
```

```
;    :PHASE :IF-REDUCE :TYPE :OMOU-FILTER        by :RECURSIVE
;================================================================================
```

Reduces hyper-polite Japanese 思う -1 under たい -DESIRE, so that "I think I want/would like" becomes simply "I want/would like".

5.2.5 Pragmatic Factors

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                          ;;;
;;; :PHASE :PRAG-INIT                  by :LOOP :RECURSIVE                   ;;;
;;; :PHASE :PRAG-POLITENESS            by :RECURSIVE                         ;;;
;;; :PHASE :PRAG-FILTER                by :RECURSIVE                         ;;;
;;; :PHASE :PRAG-INTIMACY-DEFAULT      by :ONCE                              ;;;
;;; :PHASE :PRAG-POLITENESS-DEFAULT    by :ONCE                              ;;;
;;;                                                                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Rewrite the PRAG(matic) rather than SEM(antic) area of the input.

```
;================================================================================
;   :PHASE :PRAG-INIT                  by :LOOP :RECURSIVE
;================================================================================
```

Reduces unnecessary arguments under the RESPECT, POLITE, and CONDESCEND features within PRAG.[1]

```
;================================================================================
; :PHASE :PRAG-POLITENESS              by :RECURSIVE
;================================================================================
```

Sets PRAG POLITENESS DEGREE according to the number of Japanese politeness indicators. We give degree 2 to utterances having one or two indicators, and 3 to utterances having three. If these rules fail (because there are no indicators at all), a default POLITENESS DEGREE of 2 is assigned below.

```
;================================================================================
;   :PHASE :PRAG-FILTER                by :RECURSIVE
;================================================================================
```

These rules deliver a cleaner and smaller transout for German generation by filtering features from the PRAG area which are not used by German generation. However, they are not required, and can be omitted if a slight gain in transfer speed is desired.

```
;================================================================================
; :PHASE :PRAG-INTIMACY-DEFAULT        by :ONCE
;================================================================================
```

One rule giving PRAG INTIMACY LOW as a default for every transout. Used to generate *Sie* rather than *du* throught the corpus.

[1]The CONDESCEND feature should perhaps be renamed HUMILITY. It relates to Japanese indications that the speaker has lowered, not raised, him- or herself.

```
;==============================================================================
; :PHASE :PRAG-POLITENESS-DEFAULT      by :ONCE
;==============================================================================
```

If rules in phase :PRAG :POLITENESS fail (because there are no indicators at all), a default POLITENESS DEGREE of 2 is assigned here.

5.3 Transfer

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                          ;;;
;;;   :PHASE :J-G :TYPE :IDIOM               by :LOOP :RECURSIVE             ;;;
;;;   :PHASE :J-G :TYPE :GENERAL             by :LOOP :RECURSIVE             ;;;
;;;   :PHASE :J-G :TYPE :DEFAULT             by :LOOP :RECURSIVE             ;;;
;;;                                                                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Japanese symbols (and some symbol configurations) become German symbols and configurations in this central phase of transfer.

The three micro-phases were separated during the preparation of the Japanese-English transfer system. To avoid unnecessary confusion during development, we retained the phases and usually Germanized existing rules without moving them. However, some rules now seem out of place: for instance, rules translating idioms can be found in the :GENERAL or :DEFAULT phases as well as the :IDIOM phase. Further, it is no longer clear whether three phases are needed. It may well be possible to collapse phases and move rules in the future.

```
;==============================================================================
;  :PHASE :J-G :TYPE :IDIOM              by :LOOP :RECURSIVE
;==============================================================================
```

Idiomatic constructions are replaced by German equivalents.

IDIOM024, for example, translates そうだ-REPORT using *ich habe gehoert, dass ...*, thus translating mset 141 会議の間に市内観光があるそうですが。 ("I've heard that there is a city tour during the conference.") as *Ich habe gehoert, dass es eine Stadtrundfahrt waehrend der Konferenz gibt.*.

```
;;;souda --> ich habe gehoert ...
(rws:defrwschema2 idiom024 t t
"on <reln> そうだ-REPORT in :PHASE :J-G :TYPE :IDIOM
   in= [[reln そうだ-REPORT]
        [ASPT ?aspt]
        [OBJE ?OBJE]
        ?rest]

      set ?AGEN to root.prag.speaker

      if input.obje.aspt =? [] then
         input.obje.aspt = ?aspt
      endif

      if input.obje has not aspt then
         set ?OBJE2 to [[aspt ?aspt]]
         add ?OBJE to ?OBJE2
```

```
      endif

   out= [[reln hoeren-v]
         [ASPT past]
         [AGEN ?AGEN]
         [OBJE ?OBJE2]
         ?rest]
end")
```

IDIOM028 (not shown) translates the Japanese configuration 他に何か (*hoka ni nani ka*, "something else") as SONST_NOCH_IRGENDETWAS-IDIOM.[2]

```
;=============================================================================
;  :PHASE :J-G :TYPE :GENERAL           by :LOOP :RECURSIVE
;=============================================================================


;=============================================================================
;  :PHASE :J-G :TYPE :DEFAULT           by :LOOP :RECURSIVE
;=============================================================================
```

In these two micro-phases are the majority of rules replacing Japanese symbols with German symbols in appropriate contexts.

The output of a general rule may serve as input for a more specific one: thus the general rule provides a default translation which may be overridden. For instance, GEN003 translates だ-IDENTICAL as SEIN-V; but GEN1000MSS replaces SEIN-V by BETRAGEN-V just in case the verb's IDEN value is money. Using the more specific rule for mset 64 来月お申込みになりますと4万円です。("If you apply next month, it will be 40,000 yen.") permits translation as *Wenn Sie sich naechsten Monat anmelden, wird es vierzigtausend Yen betragen.*

```
(rws:defrwschema2 gen003 t t
"on <reln> だ-IDENTICAL in :PHASE :J-G :type :GENERAL
   in= [[reln だ-IDENTICAL]
        [IDEN ?IDEN]
        [OBJE ?OBJE]
        ?rest]

   out= [[reln sein-v]
         [IDEN ?IDEN]
         [OBJE ?OBJE]
         ?rest]
end")

-------------------------------------------------------

;;;{\em ad hoc}: uses reln money for now; should use type
(rws:defrwschema2 gen1000mss t t
"on <reln> sein-v in :PHASE :J-G :type :GENERAL
   in= [[reln sein-v]
        [IDEN [[reln money]
               [quant ?quant]
               [unit ?unit]]]
        [OBJE ?OBJE]
```

[2]We often reduce source configurations to single symbols before this phase. We did not do so in this case, however, because it was convenient to adapt an existing rule from the Japanese-English system. Policy should be clarified and rules should be regularized in the future.

```
        ?rest]

   out= [[reln betragen-v]
         [IDEN [[reln money]
                [quant ?quant]
                [unit ?unit]]]
         [OBJE ?OBJE]
         ?rest]
end")
```

Notice carefully the inclusion of an INDEX feature in German noun symbols, giving information concerning definiteness or determiner, number, and owner if any. See our comments on this convention in Chapter 4.

```
(rws:defrwschema2 defn033 t t
"on <restr reln> 会員-1 in :PHASE :J-G :TYPE :default
   in= [[PARM !X[]]
        [RESTR [[RELN 会員-1]
                [ENTITY !X]]]]
   out= [[PARM !X[]]
         [RESTR [[RELN Mitglied-n]
                 [ENTITY !X]]]
         [INDEX [[NUMBER SING]
                 [DETERM INDEFART]
                 [OWNER [[LABEL *UNKNOWN*]]]]]]
end")
```

5.4 Post-transfer

5.4.1 Specification

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                          ;;;
;;;     :PHASE :VOCAB-SPECIFICATION          by :LOOP :RECURSIVE             ;;;
;;;     :PHASE :OWNER-SPECIFICATION-I        by :RECURSIVE                   ;;;
;;;     :PHASE :OWNER-SPECIFICATION-II       by :RECURSIVE                   ;;;
;;;     :PHASE :TENSE-SPECIFICATION          by :RECURSIVE                   ;;;
;;;     :PHASE :SEM-ASPE-SPECIFICATION       by :RECURSIVE                   ;;;
;;;     :PHASE :MODIFY-SPECIFICATION         by :RECURSIVE                   ;;;
;;;     :PHASE :DETERM-SPECIFICATION         by :RECURSIVE                   ;;;
;;;     :PHASE :IDEN-SPECIFICATION           by :RECURSIVE                   ;;;
;;;     :PHASE :COORD-SPECIFICATION          by :RECURSIVE                   ;;;
;;;     :PHASE :TLOC-SPECIFICATION           by :RECURSIVE                   ;;;
;;;                                                                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;===============================================================================
;      :PHASE :VOCAB-SPECIFICATION          by :LOOP :RECURSIVE
;===============================================================================
```

Several context-sensitive lexical specifications are carried out here, in addition to those performed during main transfer. (In the future, this phase can perhaps be collapsed into main transfer.)

For example, DEFV273MSS specifies ENTHALTEN-v as UMFASSEN-V if LOCT value is KONFERENZ-N.

```
;=============================================================================
;      :PHASE :OWNER-SPECIFICATION-I        by :RECURSIVE
;=============================================================================

;=============================================================================
;      :PHASE :OWNER-SPECIFICATION-II       by :RECURSIVE
;=============================================================================
```

Often the German target requires possessive determiners (*mein, Ihr*) but the Japanese source has none. We supply the dets in two phases:

In :OWNER-SPECIFICATION-I, we mark noun-type symbols which require possessives, e.g. NAME-N, ADDRESS-N, PHONE_NUMBER-N, deverbal nouns like TEILNAHME-N. (We assign [[label *OWNER*]] as a temporary value of the INDEX OWNER path.)

In :OWNER-SPECIFICATION-II, we identify the specific owner in the current utterance, overwriting the temporary marker. The present heuristic: if the IFT is INFORM, *OWNER* becomes *SPEAKER*; else it becomes *HEARER*.

```
;=============================================================================
;     :PHASE :TENSE-SPECIFICATION          by :RECURSIVE
;=============================================================================
```

The single rule TENSESPEC1000MSS relates Japanese ASPT specification to German TENSE: if ASPT is PAST, or if ASPT is ITER and the adverb SCHON-ADV is found, TENSE is PAST; else if IFT is PROMISE, TENSE is FUTURE; else TENSE is PRES.

```
;;;this rule covers all tense assignment
(rws:defrwschema2 tensespec1000mss t t
"on <aspt> :unspecified in :PHASE :tense-specification
     in= [[aspt ?aspt]
          ?rest]

     if ?aspt =? $past
     then
     input.tense = $past
     else
       if ?aspt =? $iter and
       input.TLOC.restr.reln =? $schon-adv
       then
       input.tense = $past
       else
          if root.sem.reln =? $promise then
          input.tense = $future
          else input.tense = $pres
          endif
       endif
     endif
return input
end")
```

```
;=============================================================================
;     :PHASE :SEM-ASPE-SPECIFICATION       by :RECURSIVE
;=============================================================================
```

The SEM-ASPE feature is used only for determining the proper form for German passive, as explained in Chapter 4. A single *ad hoc* rule SEMASPESPEC1000 relates Japanese ASPT to SEM-ASPE.

```
;============================================================================
;      :PHASE :MODIFY-SPECIFICATION          by :RECURSIVE
;============================================================================
```

Specifies the proper preposition for noun-noun modification, according to specific word symbols in the context. For example, MOD1001MSS rewrites the default symbol MODIFY (expressed using a genitive construction) as FUER-P if the related symbols are TEILNAHMEGEBUEHR-N and KONFERENZ-N, thus translating 会議の参加料 ("the attendance fee for the conference") as *die Teilnahmegebuehr fuer die Konferenz.* In the future, type specifications of the related symbols should be used instead.

```
;;;#60
(rws:defrwschema2 mod1001mss t t
"on <restr arg-1 restr reln> KONFERENZ-N in :PHASE :modify-specification
   in=
[[PARM !X5[[PARM !X4[]]
           [RESTR [[RELN TEILNAHMEGEBUEHR-N]
                   [ENTITY !X4]]]
           ?rest1]]
 [RESTR [[RELN MODIFY]
         [ARG-1 [[PARM !X3[]]
                 [RESTR [[RELN KONFERENZ-N]
                         [ENTITY !X3]]]
                 ?rest2]]
         [ARG-2 !X5]]]]

   out=
[[PARM !X5[[PARM !X4[]]
           [RESTR [[RELN TEILNAHMEGEBUEHR-N]
                   [ENTITY !X4]]]
           ?rest1]]
 [RESTR [[RELN FUER-p]
         [modifier [[PARM !X3[]]
                    [RESTR [[RELN KONFERENZ-N]
                            [ENTITY !X3]]]
                    ?rest2]]
         [modified !X5]]]]
end")
```

```
;============================================================================
;      :PHASE :DETERM-SPECIFICATION          by :RECURSIVE
;============================================================================
```

Most nouns in our corpus are either definite or indefinite and either singular or plural throughout the corpus. For these nouns, we treat definiteness and number as a fixed attributes of the semantic symbol, supplied during main transfer. The few case in which both definite and indefinite or both singular and plural occur for a single noun are handled in :DETERM-SPECIFICATION by *ad hoc* rules which assign definiteness or number based on specific symbols in the context.

```
;============================================================================
;      :PHASE :IDEN-SPECIFICATION            by :RECURSIVE
;============================================================================
```

To avoid overloading the rule relating SEIN-V and its argument IDEN in the current German generator, we specialize IDEN feature to IDEN-1 when a prepositional relation is the value. We must currently list such relations; types should be used in the future.

```
;=============================================================================
;     :PHASE :COORD-SPECIFICATION          by :RECURSIVE
;=============================================================================
```

To aid the generation of noun series like *X, Y, und Z*, we rewrite an embedded nest of PAUSE-COORDINATION relations. The top-level PAUSE-COORDINATION only is rewritten as UND-CONJ.

```
;=============================================================================
;     :PHASE :TLOC-SPECIFICATION           by :RECURSIVE
;=============================================================================
```

The adverbial feature TLOC is rewritten as TLOC-1 for certain adverbs to artificially fix the adverb's location. This micro-phase is *ad hoc*, and for demo purposes only. It should be omitted in a more general treatment, and the corresponding generation rules should be revised.

5.4.2 Movement

These rules move TENSE, WH elements, or other features according to the preference of the German generator. The rules presently mention specific semantic symbols. For instance, the TENSE movement rules mention specific verbs like OBLIGATION, POSSIBILITY, etc. Types, e.g. :MODAL, should be used in the future.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                         ;;;
;;;     :PHASE :TENSE-MOVEMENT              by :RECURSIVE                   ;;;
;;;     :PHASE :NEGATE-MOVEMENT             by :RECURSIVE                   ;;;
;;;     :PHASE :WH-MOVEMENT                 by :LOOP :RECURSIVE             ;;;
;;;     :PHASE :A-MOVEMENT                  by :ONCE                        ;;;
;;;     :PHASE :CONECT-MOVEMENT             by :RECURSIVE                   ;;;
;;;     :PHASE :DATE-MOVEMENT               by :RECURSIVE                   ;;;
;;;     :PHASE :ORDINAL-MOVEMENT            by :RECURSIVE                   ;;;
;;;     :PHASE :COMPOUND-MOVEMENT           by :RECURSIVE                   ;;;
;;;     :PHASE :LONG-MOVEMENT               by :RECURSIVE                   ;;;
;;;     :PHASE :ZU-MOVEMENT                 by :RECURSIVE                   ;;;
;;;     :PHASE :UNGEFAEHR-MOVEMENT          by :RECURSIVE                   ;;;
;;;     :PHASE :PREIS-MOVEMENT              by :RECURSIVE                   ;;;
;;;     :PHASE :SUBORD-MOVEMENT-AND         by :RECURSIVE                   ;;;
;;;                                                                         ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;=============================================================================
;      :PHASE :TENSE-MOVEMENT               by :RECURSIVE
;=============================================================================
```

When a modal-type relation dominates a tensed verb relation, move the tense feature from the verb up to the modal level. For example:

```
(rws:defrwschema2 tensemove1000mss t t
"on <reln> possibility in :PHASE :tense-movement
     in= [[reln possibility]
          [obje [[reln ?reln]
                 [tense ?tense]
                 ?rest1]]
```

```
          ?rest]

     out= [[reln possibility]
           [tense ?tense]
           [obje [[reln ?reln]
                  ?rest1]]
           ?rest]
end")


;============================================================================
;     :PHASE :NEGATE-MOVEMENT              by :RECURSIVE
;============================================================================
```

Three rules handling the relation NEGATE. GEN1004MSS corrects an analysis error (and might be moved up to :BAD-ANALYSIS-PATCH); the other two give special and *ad hoc* treatment of *kein* (below) and *nichts*.

```
;;;ad hoc, overspecific??
;;;transforms negate es-geben obje --> es-geben negate obje
;;;Thus "es gibt nicht ein Rabatt" --> "es gibt keinen R."
(rws:defrwschema2 gen1005mss t t
"on <reln> negate  in :PHASE :negate-movement
   in= [[reln negate]
        [aspt ?aspt]
        [OBJE [[reln es_geben-idiom]
               [obje ?obje]
               ?rest1]]
        ?rest]

   out= [[reln es_geben-idiom]
         [aspt ?aspt]
         [obje [[reln negate]
                [obje ?obje]
                ?rest]]
        ?rest1]
end")


;============================================================================
;     :PHASE :WH-MOVEMENT                  by :LOOP :RECURSIVE
;============================================================================
```

The German generator prefers to see WH question elements in the same locations as non-WH elements. The analysis output, however, delivered an extraposed PARM/RESTR structure. The rules in this phase make the adjustment.

```
;;;#86
(rws:defrwschema2 wh1001mss t t
"on <parm restr reln> was-pronp in :PHASE :wh-movement
   in=
[[PARM @wh [[PARM !X1[]]
            [RESTR [[RELN WAS-PRONP]
                    [ENTITY !X1]]]]]
 [RESTR [[RELN sein-v]
         ?rest1
         [iden @wh]]]]
```

```
   out=
[[RELN sein-v]
 ?rest1
 [iden @wh]]
end")


;=============================================================================
;     :PHASE :A-MOVEMENT                    by :ONCE
;=============================================================================
```

Rewrites a top-level adjectival relation as SEIN-V dominating the same adjectival. See the discussion of representational issues involving the copula, Chapter 4.

```
;;;ad-hoc mention of nah-a specifically; should mention adj-type reln
(rws:defrwschema2 amove1000mss t t
"on <obje reln> nah-a in :PHASE :a-movement
   in=
[[OBJE @nah[[reln nah-a]
    [obje ?obje]
    ?rest2]]
 ?rest1]

out=
[[OBJE [[reln sein-v]
[tense pres]
[obje ?obje]
[iden @nah]]]

 ?rest1]
end")


;=============================================================================
;     :PHASE :CONECT-MOVEMENT               by :RECURSIVE
;=============================================================================
```

Two rules correct scoping error in analysis relating to connectives (giving e.g. *es tut mir sehr leid*). They could be moved to :BAD-ANALYSIS-PATCH.

```
;=============================================================================
;     :PHASE :DATE-MOVEMENT                 by :RECURSIVE
;=============================================================================
```

Three rules convert dates to a regularized special format including a PARTOFDAY feature to give postmodifier *abends*, *morgens*, or *nachmittags* e.g. *Das ist ab dem vierten August, abends* for mset 248.

```
;=============================================================================
;     :PHASE :ORDINAL-MOVEMENT              by :RECURSIVE
;=============================================================================
```

One rule, ORDMOVE1001MSS, converts ordinal number, e.g. *zweite*, to preferred format. For mset 170. Unless convention changes, one rule will be needed for each ordinal.

```
;=============================================================================
;      :PHASE :COMPOUND-MOVEMENT              by :RECURSIVE
;=============================================================================
```

Breaks single sentences into two sentences, each with its own illocutionary force type. Enables new special relations COMPOUND-IFT and COMPOUND-IFT-UND, which can dominate two ifts. For example, mset 174, 送り先は大阪市東区城見２の１の６１渡辺明です。 is rewritten with COMPOUND-IFT and thus gives *Meine Adresse ist zwei, eins, einundsechzig Shiromi, Higashi-ku, Osaka. Mein Name ist Akira Watanabe.* Mset 189, 投稿が受理された場合、原稿用紙を同封いたします。 is rewritten with COMPOUND-IFT-UND and thus gives *Ich werde das hier begutachten und ich werde Ihnen das Ergebnis bis zum zwanzigsten Mai schicken.*

Note: German morphology required modifications to permit multi-sentence output. See [Seligman 1993].

```
;=============================================================================
;      :PHASE :LONG-MOVEMENT                  by :RECURSIVE
;=============================================================================
```

Gives the preferred format for the NP *eine zweihundert Woerter lange Zusammenfassung* in mset 188.

```
;=============================================================================
;      :PHASE :ZU-MOVEMENT                    by :RECURSIVE
;=============================================================================
```

Converts "a bus which goes to X" to "a bus to X" for mset 215. Thus そこから国際会議場へ行くバスが利用できます。 gives *Von dort koennen Sie einen Bus zum Internationalen Konferenzzentrum nehmen.* Mention of BUS-N specifically is *ad hoc*, should be replaced by type.

```
;=============================================================================
;      :PHASE :UNGEFAEHR-MOVEMENT             by :RECURSIVE
;=============================================================================
```

Gives preferred format for "about [money amount]" for mset 218. Gives *Wenn es ab dem Bahnhof Kyoto ist, kostet es ungefaehr sechstausend Yen.*

```
;=============================================================================
;      :PHASE :PREIS-MOVEMENT                 by :RECURSIVE
;=============================================================================
```

Gives preferred format for "between [money amount] and [money amount]" for mset 232 and 233. See discussion of representation of adjunct prepositions, Chapter 4.

```
;=============================================================================
;       :PHASE :SUBORD-MOVEMENT-AND           by :RECURSIVE
;=============================================================================
```

Includes several temporary fixes for problems which otherwise cannot be handled at the present stage of development:

- For mset 71, an "unagi" ellipsis which cannot be handled without checking previous sentences. Using (SUBORDMOVE1001MSS), 参加料は銀行振り込みです。 is forced to give *Sie sollten die Teilnahmegebuehr mit einer Bankueberweisung bezahlen.*

- For mset 249, a REQUEST in which the main verb is elided: *Bis zum achten, morgens, bitte.* Uses (DATEREQUESTMOVE1000MSS).
- Post-modifications for INDEX features NUMBER and DETERM in nouns which have no fixed values for these features. (Could be moved to :DETERM-SPECIFICATION.)

5.4.3 Filtering

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                             ;;;
;;;      :PHASE :FINAL-FILTER                    by :RECURSIVE                  ;;;
;;;                                                                             ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;==================================================================================
;      :PHASE :FINAL-FILTER                   by :RECURSIVE
;==================================================================================
```

Various features are filtered according to the preferences of the German generator.

- TENSE is filtered under state and prepositional relations (TENSEFILTER1004MSS and others).
- SEM-ASPE is filtered except under PASSIVE, where it is used to differentiate two German passive types as explained in Chapter 4 (SEMASPEFILTER1000).
- EXPR is filtered under DESIRE (ARG1000MSS).
- ASPT feature is filtered throughout (ASPTFILTER1000).
- INFATTD feature ("inferior attitude", related to Japanese expression *chotto*) is filtered throughout (ASPTFILTER1000).
- よろしく-1 as a manner adverb in *yoroshiku onegai shimasu* is filtered (ARG1003MSS).

Appendix A

References

Hasegawa, Toshiro. 1990. *Feature Structure Rewriting System Manual (Revised Version).* ATR Technical Report, TR-I-0187.

Hasegawa, Toshiro. 1991. "A Rule Application Control Method in a Lexicon-driven Transfer Model of a Dialog Translation System." In *Research Activities of the Natural Language Understanding Department and the Data Processing Department for Nov. 1989 - Mar. 1991*, page 278. ATR Technical Report TR-I-0231.

Hutchins, W. and Harold Somers. 1992. *An Introduction to Machine Translation.* Academic Press, London.

Kume, Masako and Masaaki Nagata. 1990. *Semantic Representations Used in Japanese Analysis Grammar.* ATR Technical Report, TR-I-00155.

Suzuki, Masami and Hirohisa Kosaki. 1993. *Synopsis of Language Transfer Engine for ASURA.* ATR Technical Report, TR-I-0330.

Seligman, Mark. 1993. *Morphological Facilities for German Generation in ASURA.* ATR Technical Report, TR-I-0362.

Tropf, Herbert. 1992. *The German Grammar for ASURA.* ATR Technical Report, TR-I-0289.

Ueda Yoshihiro and Kiyoshi Kogure. 1990. "Generation for the Dialog Translation Using Typed Feature Structure Unification." In *Research Activities of the Natural Language Understanding Department and the Data Processing Department for Nov. 1989 - Mar. 1991*, page 245. ATR Technical Report TR-I-0231.