gefördert vom
Bundesministerium
für Forschung
und Technologie

Verbmobil
Verbundvorhaben

Transfer and Architecture: Views from Chart Parsing

Jan W. Amtrup

FBI/UHH

VM-Report 8
March 1994

March 1994

Jan W. Amtrup

Arbeitsbereich Natürlichsprachliche Systeme
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Str. 30
22527 Hamburg

Tel.: (040) 54 715 - 519
e-mail: `jan@nats2.informatik.uni-hamburg.de`

Gehört zum Antragsabschnitt: 15.9: Interaktionsphänomene für Auswertung und Transfer

Das diesem Bericht zugrundeliegende Forschungsvorhaben wurde mit Mitteln des Bundesministers für Forschung und Technologie unter dem Förderkennzeichen 01 IV 101 A/O gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei dem Autor.

Abstract

The objective of this report is to describe the embedding of a transfer module within an alternative architectural approach for machine translation of spontaneous spoken language. The approach is cognitively oriented, i.e. it adapts some of the assumed properties of human language comprehension and production. The aspects to be modeled will include incrementality and robustness with respect to disturbances caused by the environment and performance phenomena of speech. Interaction between software modules is used to reduce ambiguity.

The transfer stage of a translation system clearly has to obey these requirements to be an integral part of such a system. This paper outlines the kind of demands to be placed on the transfer module. Relations between the basic formalisms representing linguistic knowledge on the one hand and transfer on the other hand are demonstrated as well as the consequences for algorithms and data structures.

Contents

Chapter 1

Introduction

The long-term target of the joint research project Verbmobil is the development of a mobile interpreting machine which supports two business persons speaking different languages during their discussions. Verbmobil aims at a speaker-adaptive processing of spontaneous spoken language and uses English as dialogue language. If one of the dialogue partners runs into problems, he falls back into his mother tongue (German or Japanese) and requests a translation from Verbmobil. The domain for Verbmobil is the field of general negotiations, for the first few years work concentrates on the agreement upon scheduling meetings.

We focus on architectural issues within the framework of automatic interpreting systems requiring fast system responses and the processing of spontaneous spoken language input. This paper deals with general properties that can be derived for a transfer module when integrating it into an architecture for such a system. The transfer stage is not just an add-on on top of a language understanding system. Rather it should be an integral part that is coupled with other components processing different levels of linguistic representation. Properties of the system as a whole draw certain consequences onto the transfer level. Other dependencies arise when recognizing that connections between the subsystems constrain the functionality of the connected partners. We investigate these properties and constraints in more detail in chapter 2.

Chapter 3 discusses similarities and differences of models of unification-based transfer used so far. Based on observations of this kind we present a formalism for unification-based transfer in chapter 4. It is easy and extendible and thus should be a well-suited tool for the development of an experimental transfer system.

Chapter 5 shows that techniques derived from chart-parsing can be relevant for the implementation of a transfer relation, especially when taking into account the architectural constraints stated in the chapters before. The transfer algorithms can be equipped with different processing and search strategies. This enables them to obtain the translation of a source language utterance in an incremental way. Interactions between transfer and other levels of linguistic description provide a source for multi-level-transfer.

The following chapter deals with recent trends to formulate any-time algorithms that can be interrupted to present output at request. Chapter 7 draws some conclusions.

Chapter 2

Architectural Constraints for Transfer

Traditionally, the way to look at the transfer stage of a machine translation system based on the transfer paradigm is the following: Transfer operates on completely analyzed source language utterances. The input is a representation of the utterance on a syntactic or semantic level. This representation is often required to be unique as most transfer models can not easily cope with ambiguity. The input is traversed top-down and transferred to the target language beginning at the topmost level of description. The search space is explored with a breadth first search strategy and the next level of representation is considered only after all higher parts have been processed.

Within this chapter we will argue for a more detailed view for the architecture of a transfer stage. For the purposes of a system interpreting spontaneous spoken input it is not sufficient to specify a transfer relation which (somehow) determines the correct translation for a given source language structure. From an architectural point of view it is crucial for the relation — or, more specifically, the algorithmic implementation of the transfer relation — to have certain features allowing it still to be used under certain conditions. The main arguments that we will use below to call for certain features of the implementation stem from

- the existence of transfer problems on almost all levels of linguistic description and the way human interpreters handle such problems,
- work done in the field of architecture of natural language processing systems in general which aim at a strict modularization and

- the target of neatly integrating the transfer process into a special architectural model described below.

2.1 Levels of transfer

Assuming that the transfer stage in an automatic interpreting system is located behind the modules for signal analysis, morphosyntax and probably semantic analysis, one has to decide what the major input source for transfer should be. Depending on the view of modeling, it could be either syntactic descriptions of utterances to carry out structural transfer or semantically oriented structures such as DRSs. But clearly this positioning of transfer is not completely correct. Transfer problems arise on different levels of linguistic representation:

- *Idioms* or *routine formulas* are examples roughly on the lexical level,
- *head switching* could be described as a structural transfer problem[1] and
- the correct translation of *tense and aspect* is semantic in nature. Even extralinguistic knowledge may be required for the translation of these phenomena.

It is argued that there are as many levels of transfer as there are levels of linguistic description for the source language.[2] One could solve the dilemma of having one input representation and several levels participating in a translation by allowing the incorporation of knowledge from different linguistic levels into transfer specifications and by collecting the respective data for an utterance during the course of analysis. But this doesn't prevent an idiomatic utterance or a routine formula (like several greetings) from being analyzed semantically before recognizing them as transfer issues on a lexical level.

It also does not correspond to the way human interpreters work. Often humans interpret "automatically" without using a deep analysis up to a semantic or pragmatic level. Only in case of the speech material being complex or containing ambiguities a detailed and conscious processing of the source language text is necessary. *Hauenschild 1985; Hauenschild and Prahl 1994* propose the notion of a "variable depth of analysis" to model the different

[1] *Dorr 1993* explains this phenomenon at a lexical semantic level

[2] *Hauenschild 1985; Hauenschild and Prahl 1994*

ways distinct utterances are processed. Until now, no workhas been done focussing on the characterization of the complexity of an utterance that results in a measure which allows to set a maximum level of interpretation sufficient for a correct transfer into the target language.

A prerequisite for the formulation of a multi-level-transfer is to split transfer in parts. First estimations can result from work continuing the work of *Eberle et al. 1992*. On almost all levels of representation there have to be connections between source language and target language data structures. For example, the separation of *lexical transfer* from *structural transfer* could derive benefits, although the mechanisms to treat both kinds of knowledge are almost the same.

2.2 A cognitively oriented architecture model

We will now describe the main properties of a "cognitively oriented architecture model" and show the consequences for the implementation of a transfer relation when integrating it into such a schema. The cognitively oriented model which is based on *Briscoe 1987* and extended in *Görz 1993* is primarily concerned with the modularization of and interaction among subsystems within a speech understanding system. The assumption made is that computer-based models of speech processing should reflect the basic structures of human speech understanding. The realization of this requirements leads to an *incremental, interactive system architecture.*

The basic building blocks are a number of modules operating mainly autonomous, but being able to interact in order to constrain meaning and resolve ambiguities as soon as possible. This corresponds to the *"weak modularity hypothesis"* which is assumed to be a basis for human speech comprehension[3]. The modules are interrelated by means of two types of connections:

- The main, bottom-up data path carries linguistic information according to the assumed standard hierarchy.
- Alternative connections are of lower bandwidth and more rarely used. They play a crucial role for disambiguation, however. They function as hypotheses channels through which one module's assumptions about

[3] cf. *Görz 1993, p. 8*

> properties of some units of the utterance are propagated to another module that uses this data to constrain its search space. The direction of these hypotheses is top-down thus allowing higher linguistic levels to influence the processing of lower levels. Another kind of secondary channels works in a question-answering mode: Upon the appearance of an ambiguity that can not be solved within a certain module it is possible to send a simple question to a different module. The answer could then possibly be used as a guideline whether one alternative has to be favored against another.
>
> These interactions are the ones used to eliminate ambiguities as soon as possible during the course of processing.

The utilization of the second type of connections is crucial for transfer, too. *Eberle et al. 1992* show the importance of contextual and even extralinguistic knowledge for correct translation of tense or gender of pronomina. Therefore, a transfer module should maintain such a communication device to be able to explore interaction if necessary.

Incrementality, *time-synchronicity* and the *possibility for parallelization* are the main properties of the system as a whole, constructed from separate modules. The notion of a *limited memory*[4] is modeled using a constraint for the access of hypotheses within the system. Every module can only read information concerning a definite interval of the input data. Thus it can not wait indefinitely before incorporating a hypothesis but has to work time-synchronously in principle.

The architectural freedom for the implementation of individual modules is constrained by these properties. But they may not be strict consequences from the application a system is designed for. For example, a system for interpretation within dialogues is in principle able to wait until utterances are complete before processing them and thus has complete analyses at hand at every stage of processing. An incremental and time-synchronous system for interpretation aims at the step towards simultaneous interpretation[5]. The system has to start processing as soon as data arrive. Consequently, every module is forced to have only a minimal delay compared to the input flow

[4] cf. *Pyka 1991*

[5] This is not quite correct: There are certain different aspects of simultaneous interpretation that do not arise in conversational interpretation. An example of these phenomena is the use of anticipations by an interpreter who in some cases has to guess what the source language speaker will utter in the future to be able to continue translating.

of data. Therefore, algorithms relying on the existence of entities covering a greater amount of the speech signal are not useful concerning that purpose[6]. A strict left-to-right processing and the ability to cope with partial information and quality-valued data follow from this.

Incremental transfer has to start with fragmentary descriptions of source language utterances, the completion of these being delivered during the course of processing. Even worse, according to the limited memory approach, the transfer module is not able to wait, but has to embody early hypotheses almost immediately.

The weak modularity hypothesis demands reusing already translated partial constituents for transfer. In case the translation is executed in a compositional manner it is not necessary to recompute the results. They can be inserted directly. The elements of the transfer relation are therefore memorized in a special storage. Before computing the correct translation of a constituent, consultation of this memory eliminates duplication of work. This notion can be further extended to come out with something called here a *transfer chart*, which — analogous to syntactical analysis — holds complete and incomplete translated utterance parts. This extension will be presented in detail in chapter 5.

The possibility for parallelization is an additional aspect of the architecture in question. Due to the loose coupling of individual modules, a global parallelization is possible per se. In addition, the possibility to induce parallel models of computation into specific modules is desirable. There have been proposals for some parts of an interactive architecture (e.g. for the parser or lexicon module), but at the moment a correct and meaningful application to transfer seems out of sight.[7]

[6] *Görz 1993, p. 11*

[7] A potential application for parallel processing grounds on the incrementality that the transfer stage has to show. Incremental transfer directly leads to the occurrence of multiple hypotheses — the input to the transfer module is no longer one single analysis of an utterance but rather a stream of partial results that describe parts of the signal. These descriptions can overlap and compete. The interface between a transfer module and previous stages of analysis will consist of a directed graph of partial analysis within our application. The workload of processing could be distributed among the paths within this graph.

2.3 Transfer as a software module

To conclude this architectural motivation, the construction of a separate transfer module seems to be appropriate. Another solution is possible: to annotate rules of a different analysis component — e.g. rules of the syntactic parser — with specifications of how to compute a translation. But there are several drawbacks of such an architecture:

- First, NLP-systems are often organized according to the classic linguistic levels of morphology, syntax, semantics and pragmatics. To associate the transfer module at any of these stages contradicts the intuition of distinct processing steps which seems to be at least partly real[8]: it is difficult to say which level transfer belongs to. In addition, human processing of a natural language utterance on the one hand and human interpretation of this utterance on the other hand are no identical but separate mechanisms — although they can overlap in time. The just mentioned property of variable analysis depth is a further cue for an independent transfer module that is not attached to a specific level of analysis: The annotation of e.g. syntactic rules can handle neither the flat transfer of idioms nor the very deep analysis of certain phrases.

- Second, transfer does not really correspond to other kinds of processing. If we again look at the example of syntactic parsing, it should be clear that neither there exists a corresponding transfer rule for each well-formed syntactic constituent, nor is it necessary to carry out a transfer operation with every parsing operation. An example of the 1:n-connection between parsing and transfer is *structure shift*, the change of grammatical function. The English nominal phrase MOUNTING OF THE DRILL is translated into German as MONTAGE DES BOHRERS by changing the prepositional phrase into a genitival nominal phrase. However it seems unnatural to provide two translations as soon as the PP is encountered, only because there are different translations in specific cases. One of the alternatives is normally suppressed.

  The other difference mentioned between parsing and transfer concerns the lack of synchronicity. When analyzing the nominal phrase A BIG

[8]There is psycholinguistic evidence for the existence of a separate but not autonomous morphosyntactic module based on studies of aphasia (cf. *Görz 1993, p. 11*).

DOG the parser builds up syntactic structures that are growing gradually when working from left to right. Transfer of the adjective can only start when the head of the phrase is processed. On the one hand the gender of the NP can not be identified earlier, on the other hand the correct translation of BIG can not be computed in advance: within a different context it could have been translated with the meaning of DISTINGUISHED, i.e. within the phrase A BIG STATESMAN[9].

- Third, from a point of view of software engineering there are arguments supporting an independent implementation of the transfer module. As mentioned above the transfer module has to communicate with other modules in order to reduce or eliminate ambiguity. Such complex requirements encourage its encapsulation as a separate module with interfaces to other software parts. Similarly, it seems to be appropriate formulating the tasks of modules relatively narrow: because transfer mechanisms are complex by themselves and at the same time reasonably independent from other mechanisms.

[9] Given the framework of chart parsing a similar behaviour could be realized when activating the transfer with every *inactive* edge inserted into the chart.

Chapter 3

Models of unification-based transfer

Soon after the introduction of unification-based formalism into natural language processing[1] they were supposed to be useful for machine translation, specifically for transfer. *Kay 1984* suggests the inclusion of transfer knowledge into Functional Unification grammar (FUG), other authors use similar approaches. The most important expansion of formalisms that simply use unification to build up larger structures is the necessity for at least two relations. One of them is usually not included within such mechanisms. Apart from coreference between nodes of graphs which is used to express identity and is common among normal graph unification, one needs the ability to express subsequent transfer, to denote elements of the transfer relation. This can be done in several ways:

- LFG: *Kaplan et al. 1989* use so called *correspondences* of LFG that realize relations between different levels of representation of an utterance. Φ is the relation between the constituent structure (*c-structure*) and the level of grammatical functions (*f-structure*), σ the prolongation up to the level of semantic description. In order to formulate a transfer relation an additional mechanism named τ is introduced relating source and target language structures by means of the transfer relation. Through the composition of, say, Φ and τ contrastive phenomena can be modeled within transfer rules:

[1] see, e.g., *Kaplan and Bresnan 1982*

beantworten V
$(\uparrow \text{PRED}) = \text{'beantworten} \langle (\uparrow \text{SUBJ}) (\uparrow \text{OBJ})\rangle \text{'}$
$(\tau \uparrow \text{PRED FN}) = \text{repondre}$
$(\tau \uparrow \text{SUBJ}) = \tau (\uparrow \text{SUBJ})$
$(\tau \uparrow \text{AOBJ OBJ}) = \tau (\uparrow \text{OBJ})$

All coreferences that hold between source and target language models are again used as input to the transfer machinery. The example lexical entry for the German verb BEANTWORTEN (*to answer*) exemplifies (via the additional AOBJ within the path) the insertion of the French À in the translation:

Der	*Student*	*beantwortet*	*die*	*Frage.*
Le	étudiant	répond	la	question.

"L'étudiant répond à la question."

"The student answers the question"

- Eurotra (*Schütz 1988*) does not use complete feature structures to represent linguistic knowlegde. Rather the input structures for transfer are trees. Unification is used during transfer: Within transfer rules the modeler can introduce unification variables which are used to carry out e.g. agreement checks or guarantee other identities:

```
np.[^(art, {def=D}), $1:n] => (?, {cat=np, def=D}).[$1]
```

- *Noord 1990* uses a PATR-style notation to express transfer regularities. He provides special features GB and SP for the two languages English and Spanish to distinguish the respective data. He specifies a bilingual lexicon and operates with rules like the following which is a fairly general one capable of compositionally transferring verbal phrases with two arguments:

$0 \rightarrow 1\ 2\ 3$
$\langle$ 0 gb pred$\rangle = \langle$1 gb$\rangle$
$\langle$ 0 gb arg1$\rangle = \langle$2 gb$\rangle$
$\langle$ 0 gb arg2$\rangle = \langle$3 gb$\rangle$
$\langle$ 0 sp pred$\rangle = \langle$1 sp$\rangle$
$\langle$ 0 sp arg1$\rangle = \langle$2 sp$\rangle$
$\langle$ 0 sp arg2$\rangle = \langle$3 sp$\rangle$

- *Carlson and Vilkuna 1990* integrate transfer rules into feature structures. The result of their processing is a description of the source language utterance and the corresponding translation, but the features describing parts of the properties of the languages are mixed at several levels in the features structure. The singular target language description has to be extracted from there. A rule partially describing the Verb DISCUSS and its Finnish counterpart KESKUSTELLA is of the form:

```
[E: [LEX: DISCUSS
     CAT: VERB
     SUBJ: #2[E:[DUMMY: F]]
     OBJ:  #3[F: [CASE: ELA]]
     PRED: [ARG1: #2
            ARG2: #3
            ARG3: *NONE*]]
 F: [LEX: KESKUSTELLA
     CAT: VERB
     SUBJ: #2
     OBL:  #3
     PRED: [ARG1: #2
            ARG2: #3
            ARG3: *NONE*]]]
```

  Carlson and Vilkuna conclude that in principle it is possible to carry out transfer in an incremental way. The reason for this is the monotonicity of unification-based methods which guarantees the persistence of partial results. In their model every sentence is completely analyzed before being transferred, though. It is argued that knowledge of the target language, as it is described within transfer, *"is not likely to influence parsing of the source language in any significant fashion"*[2].

- *Zajac 1992* uses a typed feature structure formalism which simplifies transfer rules in many cases by using an inheritance hierarchy for them. Entries on lower levels of that hierarchy specialize the entries above them and need not contain as much information as if the structure of rules would be flat. Zajac lets coreferences denote identity even across

[2] *Carlson and Vilkuna 1990, p. 3/60*

the language boundary and makes use of two special features, namely ENG and FR to differentiate between source and target language. Subsequent transfer is expressed with *conditions* that are language elements of the formalism he uses, the Typed Feature Structures (TFS, *Emele and Zajac 1990*). An example for a translation using Zajac's model could be

A student falls.
Un étudiant tombe.

that is rendered with a lexical transfer rule for FALL:

$$\begin{bmatrix} \textit{tau-lex-fall} & \\ \text{ENG} & \begin{bmatrix}\text{PRED} & \text{fall}\\ \text{SUBJ} & \boxed{\text{e-subj}} \\ \text{TENSE} & \boxed{\text{e-tense}}\end{bmatrix} \\ \text{FR} & \begin{bmatrix}\text{PRED} & \text{tomber}\\ \text{SUBJ} & \boxed{\text{f-subj}} \\ \text{TENSE} & \boxed{\text{f-tense}}\end{bmatrix}\end{bmatrix} :\text{-} \begin{bmatrix}\textit{tau-lex} & \\ \text{ENG} & \boxed{\text{e-subj}} \\ \text{FR} & \boxed{\text{f-subj}}\end{bmatrix}, \begin{bmatrix}\textit{tau-tense} & \\ \text{ENG} & \boxed{\text{e-tense}} \\ \text{FR} & \boxed{\text{f-tense}}\end{bmatrix}$$

- *Morimoto et al. 1992* use feature structures describing the semantic content of an utterance as input to transfer within a speech-to-speech translation system from Japanese to English. A rewriting mechanism traverses the source language structure and transfers it. The transfer rules contain a source and a target partial feature description as well as a condition part that controls the application of rules. Coreference across the language boundary again triggers subsequent transfer.

```
on <reln> j-have in:phase J-E
  in = [[reln j-have]
        [agen ?agen]
        [obje ?object]
        ?rest]
  out= [[reln have]
        [agen ?agen]
        [obje ?object]
        ?rest]
```

- *Beskow 1993* divides transfer rules into four parts and thus stresses the different parts of the application of a transfer rule. A rule for definite nominal phrases shows this strong separation:

```
Label
    NP-def
Source
    <* cat>     = NP
    <* def>     = DEF
    <* num>     = ?Num
    <* head>    = ?head1
Target
    <* cat>     = NP
    <* def>     = DEF
    <* num>     = ?Num
    <* det lex> = 'the'
    <* head>    = ?head2
Transfer
    ?head1     <=> ?head2
```

This rule handles the insertion of the definite determiner THE in the English translation of the following Swedish noun phrase:

Montering *af* *bergborrmaskin*
Mounting of rock drill.DEF
"Mounting of the rock drill"

The four parts of each rule are:

  - A label that identifies each rule uniquely.
  - A source language structure that is the trigger for application of a transfer rule. The rule is only applicable if the source structure can be unified with the description of the constituent in question. The star ('*') denotes the root of the actual graph, variables start with '?'.

- A target structure that is either used to check the translation or to construct the target language representation (The model can either be used affirmatively to check certain translations or constructively to translate sentences).
- Transfer rules that make explicit the subsequent pairs of members of the transfer relation. They initiate the recursive call to the transfer mechanism.

The methods mentioned above all have several aspects in common. They all separate the procedural component which is formulated as a control structure and the declarative component which is used to describe contrastive knowledge of the two languages. This seems quite natural when utilizing unification-based formalisms, however. Transfer rules are most often stated as features structures, be they typed or not, although the actual notation varies. In addition to the relation of identity expressed as coreference the relation of subsequent transfer is necessary. It enables each model to translate large constituents by decomposing them and translating each part in turn. This relation is in some cases implicit, some make it explicit in separate transfer equations, e.g. *Beskow 1993*. All formulations use a top-down approach for transfer. The algorithms start at the outermost level of linguistic description and descend recursively into deeper levels as subsequent transfer equations are encountered. However, such an approach does not imply the impossibility of transferring incomplete input structures. A bottom-up approach is realized in none of the models.

Chapter 4

A formalism for experimental transfer systems

In the first part of this chapter we will outline some proposals for the properties of a formalism that is suitable to implement a novel architectural schema. Our work depends on the basic formalism being a well-suited tool for the implementation of an incremental, interactive architecture. After the introduction of these properties we show that a feature structure formalism as described in *Euler 1992b; Euler 1992c* can be used to formulate transfer statements of sufficient complexity although it has not all properties which we regard necessary below.

4.1 Features of a formalism for architecture

For the experimentation with several components of a system based on a specific formalism and interacting to a certain degree, the implementation of that formalism should have a tool character in order to guarantee maximal flexibility for the researchers. During earlier work a formalism for typed feature structures with appropriateness and disjunction has been developed[1]. This formalism simply supported a functional interface. The user could construct and unify feature structures while at the same time no regulations concerning the order of application of operations were made. Algorithms are completely a matter of the user of the formalism. The following properties

[1]see *Euler 1992a*

are desirable for a formalism[2]:

- fine-grained control structure. Assuming that the main device of the formalism is a constraint-solver it should be possible to utilize several levels of granularity. On the lowest level the user should have control over every single unification while at an appropriate high level sets of constraints could be applied within one operation of the formalism yielding a solution for a set of feature terms[3].

- User-driven extension. The meaning of this is twofold: First, the formalism should have the ability to formulate function calls resp. relations which are implemented outside of the formalism. That goes beyond what is necessary to develop HPSG-like grammars, namely relations like *append()*; the drawing of inferences that may be extralinguistic in nature or the handling of communication tasks could be typical applications of such a schema. Second, the steering of such an extension is completely within control of the user: The moment of evaluation is not determined by the formalism, but instead can be choosen freely by the formalism user.

- Linkable extensions. The proposal mentioned before restricts the monotonicity and mathematical handling of the formalism. Thus the formalism has to be neutral with respect to the extensions: whenever they are not used, there are no drawbacks in running time or other respects.

- Uniquely identification. Every feature structure constructed by the formalism should have its own, unique label just like the surrogate for object-oriented databases. This enables the adaption of reason-maintenance-techniques for data of the formalism.

4.2 A formalism for transfer

As we mentioned above there are two relations that are essentially needed when describing contrastive knowledge for transfer: Identity and recursive transfer. Identity is included in most feature-based formalisms as the notion of coreference. Recursive transfer can be formulated as a relation within a

[2]The assumptions stated here are part of the result of *Amtrup and Weber 1994*

[3]cf. *Emele and Zajac 1991*

formalism under the assumption that relations are supported; otherwise one could use a separate feature for the specification of subsequent transfer. This is the direction of research we will pursue here.

We will take the formalism described in *Euler 1992a* as a starting point. It is a formalism for typed feature structures with appropriateness and disjunction. It supports multiple inheritance, but has no means to classify a feature structure according to its maximal type. There are no notions of functions or relations, but since a very fine-grained control structure is used the formalism can easily be extended to handle such devices. We have extended the formalism to formulate functions within feature structures. The data type of a feature structure was extended with a special attribute FUNCTIONS that takes a list of function call descriptions. This feature is handled carefully during a unification of feature terms in order to preserve the function calls. Thus when unifying two feature terms A and B, the resulting feature term C is the unification of the two except for the attribute FUNCTIONS. The FUNCTIONS-value of C is simply the concatenation of the FUNCTIONS-values of A and B at all levels — the functional attribute is guaranteed to be on the topmost level of a feature structure. At any desired point the user can call a method for feature terms that evaluates one or all of the function calls. When evaluating the last call for a term, the attribute vanishes.

A function call consists of a list of feature terms with a length of at least two. The first term in the list which has to be of type *string* denotes the function to be called. This function is an arbitrary lisp function. The remaining terms except the last one provide the arguments to the function call. Usually they are provided by coreference. Finally the last term in the list denotes the result of the function call. In most cases this is a coreference denoting the place where the result should be inserted. An example for a function call is given in fig. 4.1 that shows a simple grammar rule in a phrase structure style.

While this extension is sufficient for the incorporation of rules of recursive transfer we establish a separate attribute to state transfer equations. Thus the formulation is a little bit clearer. To be more explicit, a transfer rule is a feature structure of type *t-rule* that has three attributes with the following properties:

- SOURCE contains the source language description.
- TARGET contains the target language description. Coreferences may mediate between the two to denote token identity.

$$\begin{bmatrix} \textit{rule} & \\ \text{RULE} & \begin{bmatrix} \langle \\ \quad \begin{bmatrix} \textit{s} \\ \text{PHON} \quad \boxed{11}=[\,] \\ \text{SYN} \quad \langle\ \boxed{2}\ \boxed{3}\ \rangle \\ \text{FUNCTIONS} \quad \langle\ \langle\ \text{“append-phon”}\ \boxed{12}\ \boxed{13}\ \boxed{11}\ \rangle\ \rangle \end{bmatrix} \\ \quad \boxed{2}=\begin{bmatrix} \textit{np} \\ \text{PHON} \quad \boxed{12}=[\,] \end{bmatrix} \\ \quad \boxed{3}=\begin{bmatrix} \textit{vp} \\ \text{PHON} \quad \boxed{13}=[\,] \end{bmatrix} \\ \rangle \end{bmatrix} \end{bmatrix}$$

Figure 4.1: A phrase structure grammar rule with functions

- SUBSEQ is a list of subsequent transfer equations. They trigger the recursive call of the transfer mechanism. Each equation in turn is a feature structure of type *st-rule* that has two attributes SOURCE and TARGET that denote the respective parts of the equation. The values of these attributes normally are coreferences into the respective language description parts.

Fig. 4.2 contains a transfer rule to enable head shift as is exemplified in the contrast:

Hans	*schwimmt*	*gerne*
John	swims	likingly

“John likes to swim.”

The german adverb *gerne* becomes the head verb of the English translation.

The approach presented here is similar to some of the models outlined in chapter 3, it closely resembles the one of Beskow. We don't take into consideration his remarks regarding the establishment of a hierarchy of rules in order to control the number of applicable rules. However, we will follow his suggestion for the possible adaption of techniques from chart-parsing within the next chapter.

$$
\begin{bmatrix}
\textit{t-rule} & \\
\textsc{source} & \begin{bmatrix} \textit{vp} & \\ \textsc{syn} & \left\langle \boxed{11}\!=\!\boxed{v}\ \boxed{12}\!=\!\begin{bmatrix}\textit{adv}\end{bmatrix} \right\rangle \end{bmatrix} \\
\textsc{target} & \begin{bmatrix} \textit{vp} & \\ \textsc{phon} & \boxed{31}\!=\![\,] \\ \textsc{syn} & \left\langle \boxed{21}\!=\!\boxed{v}\ \boxed{22}\!=\!\boxed{v} \right\rangle \\ \textsc{functions} & \left\langle \left\langle \text{“append-tphon”}\ \boxed{22} \begin{bmatrix} \textit{sign} & \\ \textsc{phon} & \langle \text{“to”} \rangle \end{bmatrix} \boxed{21}\ \boxed{31} \right\rangle \right\rangle \end{bmatrix} \\
\textsc{subseq} & \left\langle \begin{bmatrix} \textsc{source} & \boxed{11} \\ \textsc{target} & \boxed{21} \end{bmatrix} \begin{bmatrix} \textsc{source} & \boxed{12} \\ \textsc{target} & \boxed{22} \end{bmatrix} \right\rangle
\end{bmatrix}
$$

Figure 4.2: A transfer rule for head switching

Chapter 5

Charts: Adaptions for transfer

The notion of a *chart* was introduced by *Kay 1973* mainly for parsing. The starting point was an observation that certain partial analyses could be used several times during the analysis of an utterance. The mechanism to store the partial results in a *wellformed substring table* was extended to be able to establish a bookkeeping of incompletely analyzed constituents, too. Normally incomplete items on a chart are related to phrase structure rules with right sides which have not yet been fully processed. Since then charts have been very well established and widely used within natural language processing.

An essential property of chart parsing mechanisms supporting their success is their inherent flexibility. Without a central data structure in which the results of the analysis can be stored and without the possibility to specify the control structure elsewhere the system designer has to construct analysis algorithms that hide processing and search strategies deeply within code. Such algorithms can not easily be compared or altered. The use of a chart that forms the central data structure and holds all complete and incomplete partial results and the corresponding agenda that implements any search strategy enables the designer to specify an analysis algorithm that abstracts from the actual strategies for processing and search — Kay calls this an *algorithm schema* (cf. *Kay 1980*). The separation of *What* from *How* allows for a pragmatic insight into the computational processes concerning the analysis.

The monotonicity of the chart that allows the computation of results in a basically unordered way and the inherent possibility of a coarse-grained parallelization (cf. *Amtrup 1992*) have similar relevance.

Switching to the problem of transfer it must be noted that similar things hold. A partial analysis — i.e. a partial translation — could in fact be used many times during transfer. Especially when attempting incremental transfer there is no single interpretation that has to be transferred but rather there are several competing analyses and one can not tell in advance which of them is finally the best one. So there should be a mechanism like a wellformed substring table that stores partial results. The extension into the direction of a transfer chart is straightforward: incomplete analyses correspond to partially transferred items that have at least one open transfer equation.

Some of the constraints that architecture places on transfer can be motivated within a chart-processing paradigm. Synchronicity in time could be realized through functions for access of data items in the chart that restrict themselves to a definite time interval. This is in contrast to the advocated monotonicity of chart processing, but the relaxation of this property has been proposed elsewhere, e.g. for correction of input[1]. Incrementality is a property of processing and search strategies. Both can be formulated to operate on items first that belong to an earlier interval in order to guarantee an incremental left-to-right processing.

The integration of interactive communication between a transfer component and other parts of an interpretation system can be stated in two ways:

- One could formulate communication methods directly within the transfer algorithms.
- One could set up *communication tasks* which are handled via an agenda and which are treated analogously to other tasks.

Both kinds of approaches may benefit from a chart-based paradigm. Especially the establishment of separate tasks for communication enables the transfer algorithms to suspend execution of transfer statements while waiting for results from other components. The transfer process could initiate an interactive request and put the task at hand onto the agenda again to resume operation in case the answer arrives.

It is unknown whether the integration of a chart into transfer algorithms simplifies the formulation of a multi-level transfer or not. But as the relevance of this extension is equally unknown at the moment the benefits or drawbacks can not be estimated. Parallelization finally is an extension that to a great extent emerges directly from the chart-based paradigm. The simplest starting point for parallel transfer with a chart is to divide the tasks at hand onto

[1] *Wirén 1992*

different processors. Again the consequences of such an approach can not be estimated.

5.1 Construction of a transfer chart

What is a transfer chart? In the simpliest case, which is assumed here, it is an extension of an analysis chart that could have been used for parsing. This renders a very close connection between parsing or any other kind of analysis and transfer; it has to be reflected upon with modularity and integration in mind. We take a transfer chart as annotation to the analysis chart: Every inactive chart egde is the root of a tree of transfer edges. Thus, translations are directly derived from analyzed syntactic constituents. Every node at the first level of the rooted tree is the result of the application of a transfer rule to the given constituent. There are as many branches from the root of the tree as there are applicable transfer rules. If the transfer rules contain recursive transfer equations subsequent levels of the tree are possible. A daughter node within a tree is created by solving exactly one transfer equation from the mother node information. Overall there exists a two-dimensional structure of the combined analysis- and transfer chart:

- One temporal dimension that describes the progress of time within the input signal, and
- One dimension in the number of unsolved transfer equations that describe the progress while constructing target language equivalents.

Within the annotated trees a criterion of local maximality (cf. *Satta and Stock 1989*) can be formulated. The leaves of the trees are normally the most interesting nodes because most equations have already been solved here. Thus the criterion can be extended in a twodimensional way to only take into consideration locally maximal transfer edges with attached locally maximal analysis edges. The desired state is the reduction of the tree of transfer edges to a linear list. This would correspond to the principle of *specificity* (cf. *Beskow 1993, p. 54*).

The components and properties of a transfer edge are as follows: Every transfer edge belongs to an analysis edge. Important data like start and end point, rating of the described constituent as well as the original feature graph are located here. The originating transfer rule that leads to the construction of this edge has to be recorded as well as the set of transfer equations still to be solved. Equally important are the immediately incorporated partial translations that were used during the construction. Activity of a transfer edge is measured with respect to the open equations: a transfer edge is called *active* if there are such equations and otherwise called *inactive*. Similar to analysis one can say that active transfer edges describe incomplete translations of a constituent while active analysis edges describe incomplete recognized constituents.

Some aspects are different within a transfer chart, of course. For example, to deal with transfer, the proposal of new categories falls apart. Inactive incoming analysis edges play that role. The fundamental rule of chart parsing has a counterpart, too. The combination of two transfer edges makes sense, if one of them (call it the big one) contains an unsolved transfer equation, whose source language part can be unified with the source part of the other edge (call it the small one). If this can be done, a second unification with the target part has to occur; the newly constructed edge can be inserted into the chart. The third operation to be modified is the proposal of new edges that works differently.

5.2 An example: Translation of "montering"

The following section demonstrates the terminology and functioning of a transfer chart with a small example, the translation of the Swedish nominal phrase MONTERING into the English version THE MOUNTING.

The description of MONTERING consists mainly of two edges, namely the preterminal lexical edge (1) that stems from the lexicon and the inactive edge (2) that constitutes the nominal phrase.

$$(1) \qquad \begin{bmatrix} \textsc{phon} & \langle \text{ montering } \rangle \end{bmatrix}$$

$$(2) \qquad \begin{bmatrix} \textit{np} & \\ \textsc{phon} & \langle \text{ montering } \rangle \\ \textsc{num} & \text{SING} \\ \textsc{def} & \text{DEF} \\ \textsc{head} & \begin{bmatrix} \textsc{phon} & \langle \text{ montering } \rangle \end{bmatrix} \end{bmatrix}$$

The necessary repertoire of transfer rules includes the lexical relation of MONTERING (3) and the general rule for Swedish definite nominal phrases

(4) that have to be augmented with a determiner THE in English.

(3)
$$\begin{bmatrix} \text{SOURCE} & \begin{bmatrix} \text{PHON} & \langle \text{ montering } \rangle \end{bmatrix} \\ \text{TARGET} & \begin{bmatrix} \text{PHON} & \langle \text{ mounting } \rangle \end{bmatrix} \end{bmatrix}$$

(4)
$$\begin{bmatrix} \textit{t-rule} & \\ \text{SOURCE} & \begin{bmatrix} \textit{np} & \\ \text{DEF} & \text{DEF} \\ \text{NUM} & \boxed{1}\begin{bmatrix} \; \end{bmatrix} \\ \text{HEAD} & \boxed{2}\begin{bmatrix} \; \end{bmatrix} \end{bmatrix} \\ \text{TARGET} & \begin{bmatrix} \textit{np} & \\ \text{PHON} & \boxed{4} \\ \text{DEF} & \text{DEF} \\ \text{NUM} & \boxed{1} \\ \text{HEAD} & \boxed{3}\begin{bmatrix} \; \end{bmatrix} \\ \text{FUNCTIONS} & \langle \, \langle \text{ ``append-tphon''} \begin{bmatrix} \textit{sign} & \\ \text{PHON} & \langle \text{ ``the''} \rangle \end{bmatrix} \boxed{3} \; \boxed{4} \rangle \, \rangle \end{bmatrix} \\ \text{SUBSEQ} & \left\langle \begin{bmatrix} \textit{st-rule} & \\ \text{SOURCE} & \boxed{2} \\ \text{TARGET} & \boxed{3} \end{bmatrix} \right\rangle \end{bmatrix}$$

During analysis the montering-edge (1) comes up first as an inactive edge. All applicable transfer rules are applied. There is only one rule applicable, i.e. only one rule has a source part subsuming the chart edge information. This transfer rule (3) is applied by unifying its content with the source language description, yielding the transfer edge (5) below. This one is inactive since it does not have any open transfer equations. Thus there are no actions necessary besides entry into the transfer chart.

(5)
$$\begin{bmatrix} \text{SOURCE} & \begin{bmatrix} \text{PHON} & \langle \text{ montering } \rangle \end{bmatrix} \\ \text{TARGET} & \begin{bmatrix} \text{LEX} & \langle \text{ mounting } \rangle \end{bmatrix} \end{bmatrix}$$

Next, the Swedish NP (2) is entered into the analysis chart. Transfer rule (4) can be applied to it and results in the transfer edge (6). This one is active since it contains an open equation.

$$
(6)\qquad \begin{bmatrix} \textit{t-rule} & \\ \text{SOURCE} & \begin{bmatrix} \textit{np} & \\ \text{DEF} & \text{DEF} \\ \text{NUM} & \boxed{1}\ \text{SING} \\ \text{HEAD} & \boxed{2}\begin{bmatrix}\text{PHON} & \langle \text{ montering } \rangle\end{bmatrix}\end{bmatrix} \\ \text{TARGET} & \begin{bmatrix} \textit{np} & \\ \text{PHON} & \boxed{4} \\ \text{DEF} & \text{DEF} \\ \text{NUM} & \boxed{1} \\ \text{HEAD} & \boxed{3}\begin{bmatrix} \ \end{bmatrix} \\ \text{FUNCTIONS} & \left\langle \left\langle \text{“append-tphon”} \begin{bmatrix} \textit{sign} & \\ \text{PHON} & \langle \text{“the”} \rangle \end{bmatrix} \boxed{3}\ \boxed{4} \right\rangle \right\rangle \end{bmatrix} \\ \text{SUBSEQ} & \left\langle \begin{bmatrix} \textit{st-rule} & \\ \text{SOURCE} & \boxed{2} \\ \text{TARGET} & \boxed{3}\end{bmatrix}\right\rangle \end{bmatrix}
$$

After inserting (6) into the transfer chart the fundamental rule is applied. This is the combination of active and inactive edges. Here it results in a search of inactive transfer edges that belong to the same interval of the analysis chart as (6) and that have a source language part that subsumes the source part of (6). Edge (5) ist such a candidate.

Now the source language and target language part of both transfer edges have to be unified. The result of this combined unification is (7), an inactive transfer edge that can directly be inserted into the chart. The target language translation is extracted under the path TARGET.

(7)
$$\begin{bmatrix} \textit{t-rule} & \\ \text{SOURCE} & \begin{bmatrix} \textit{np} & \\ \text{DEF} & \text{DEF} \\ \text{NUM} & \boxed{1}\ \text{SING} \\ \text{HEAD} & \boxed{2} \begin{bmatrix} \text{PHON} & \langle \text{ montering } \rangle \end{bmatrix} \end{bmatrix} \\ \text{TARGET} & \begin{bmatrix} \textit{np} & \\ \text{PHON} & \langle \text{ the mounting } \rangle \\ \text{DEF} & \text{DEF} \\ \text{NUM} & \boxed{1} \\ \text{HEAD} & \boxed{3} \begin{bmatrix} \text{PHON} & \langle \text{ mounting } \rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Chapter 6

Any-time transfer: What is a transfer quantum?

This chapter deals with the current focus on *any-time algorithms* for speech processing. We try to outline the possible nature of an any-time transfer and speculate about the means to implement such a mechanism.

Russel and Zilberstein 1991 define anytime algorithms as *algorithms whose quality of results degrades gracefully as computation time decreases*. The monotonic growing output quality of such a procedure is described with a probabilistic performance profile which is a function of time. Anytime algorithms are classified according to the moment at which the perspective computation time must be known. *Interruptible algorithms* deliver a solution even when interrupted without prior warning whereas *contract algorithms* have to be informed about the computing time at hand in advance in order to produce a result. Roughly, for any contract algorithm one can construct an interruptible algorithm that produces results of equal quality in fourfold time (cf. *Russel and Zilberstein 1991*).

Anytime algorithms are highly desirable for advanced speech processing systems (cf. *Wahlster 1993*, *Menzel 1994*). Only fast system responses with a minimal delay are useful. Taking into consideration the requirements drawn upon the architecture of an experimental interpreting system and specifically upon the transfer stage of such a device, the application of anytime algorithms should be another central point besides time synchronicity and incrementality. Unfortunately the more tractable class of contract algorithms falls apart due to inherent contradictions. Contract algorithms by definition have to be equipped with the available processing time in advance. This is problematic within real-world speech applications. The users' tolerance regarding processing time is limited. Pauses between utterances within conversation of more than a few seconds are hardly tolerable. Even if the user of a speech translation system is specially instructed to be more patient regular pauses of much more than ten seconds seem to be unacceptable.[1]

This results in the separation of the processing course into two phases:

- The first phase is the time span in which the source language text is produced that has to be translated. During this phase the time available simply can not be estimated.[2]
- The second part of processing starts as soon as the utterance is completed. From this point on, the computation time is very restricted since the system has to produce an interpretation of the users' utterance before the attention is reduced.

The earliest point at which a bound can be given to the algorithm is the space between the two phases. We suppose that length of phase one mentioned above is at least of the same order of magnitude as phase two; often, phase one will be significantly longer. Thus, only algorithms that do not know in advance how much computing time is available can be used.

6.1 Strategies for Transfer

How should a useful strategy for transfer be formulated? The easiest way would be to first translate word by word like in the early stages of machine translation. After obtaining at least a rough translation of the whole utterance, one could start more complex operations like transferring structure built by other modules. So we have two goals for the transfer stage, namely

1. to produce a *complete* translation of what was said and

[1]An interesting possibility to further enlarge the time available for interpreting is to "mumble" during work (cf. *Karlgren 1994*). Sometimes uttering about the stage of processing and the contents currently working on may increase the patience of users what results in a larger amount of processing time available. An interpreting system could at some points try to start a target language sentence or could utter fragments of the content to draw the recipients attention on the interpretation.

[2]A simple way to provide a perspective of the computing time would be to take the notion of time-synchronicity very serious and demand that the processing of any part of an utterance is computed using at most the time it occupies.

2. to construct a *better* translation if there is enough time at hand.

This results in the establishment of two notions:

- quality of translated items and
- amount of work needed to augment the translation of a particular part of an utterance.

The first notion is needed to choose between competing alternatives of transfer. The optimal solution possible has to be presented upon request (i.e. a few seconds after the utterance has ended). The second notion steers the selection of *agenda tasks* that are carried out; it is used as an ordering constraint. This function decides which transfer rule has to be applied first if there are more than one applicable for a given configuration.

The quality measure associated with each transfer rule should thus show to what extent the translation could be augmented by applying it. Statistical methods are not a good choice to estimate this value because they favor common translations. By attaching a frequency measure to every transfer rule the system would hardly encounter models for more specific translations that are nevertheless needed for some collocations. The complexity of a transfer rule could be a better guidance. We assume a more complex rule to be more specific and to apply to fewer source language configurations. This allows the conclusion that it describes a specifically desired translation (*Beskow 1993*). So simply the size of a transfer rule could be a first starting point for a quality measure for transfer rules. It is unknown how to predict he amount of work needed to apply a transfer rule. The size of the input edges for a rule influences this amount. Furthermore, if complex inferences are drawn during the application of transfer rules, their impact can not be estimated.

Now that we have given an outline what to do first during transfer we have to look at the units of work that are done before a solution could be presented. There are several possibilities:

- Since the system works incrementally, one could try to integrate each incoming hypothesis completely before sending output to the next component. The problem arising is the existence of right context and different sentence schemata in different languages.

- Every task on an agenda for transfer could be seen as a transfer quantum. Every task either creates a new transfer edge from an analysis edge and a transfer rule or augments a transfer edge by solving an open transfer equation. It could be useful to feed output with these speculations.

- In case of a transfer rule being completely applied with all equations solved the transfer stage could send the translation to an output (or generation) component. Thus the interface between transfer and following stages consists of completely transferred source language constituents. With respect to bandwidth and content we assume this to be the best of the three presented alternatives. It also resembles the interface between syntax and transfer described earlier: Inactive analysis chart edges represent completely analyzed source language constituents.

Assuming that transfer edges with all equations solved are presented as output we can divide the data structures for transfer into three parts from an architectonic point of view:

- The representation of input consists of the inactive edges resulting from syntactic analysis of the source language utterance.

- The representation of ongoing work is done through the trees of transfer edges attached to each analysis edge. Here incorporation of previous work and compositional transfer takes place.

- The output that consists of the inactive transfer chart edges is the last representational stage. Each of these edges describes the complete translation of a given source language constituent.

To meet any-time constraints we have to be able to provide an output at every desired time. Thus we now need a function that computes from a set of complete transfer edges a coverage of the source language utterance while maximizing rating. The translation can not be output as the realization of the best-rated edge spanning the whole utterance simply because such a complete translation may not exist due to an interruption. Thus, we have to take into consideration all parts of translation that were produced so far. The output representation forms a graph (chart) of its own; the output function has to compute a path from the start point to the end point with a maximum rating — a classical graph search problem that has to be done incrementally every time a new inactive transfer edge arrives.

One remaining problem is the numbering of the chart: The order within an analysis chart poses no difficulties: You could take either time stamps that belong to the utterance (measured in frames or whatever) or simply count the nodes from left to right. The order clearly reflects the linear odering of time that is a property of the source language utterance. Just the opposite is the case with the transfer chart and the resulting chart on the target language side: Certainly the ordering will be a different one but it can not be predicted in advance. So the only possible strategy here is to use the same ordering as on the source side and to adapt generation algorithms accordingly.

6.2 Combination of any-time components

Russel and Zilberstein 1991 show optimal methods for combining any-time algorithms into a more complex system. Their approach is not easily extensible to handle the case of a interpreting system. The probability function that describes the quality of a component as function of time sometimes can not be given (cf. *Menzel 1994*). Moreover the amount of time available is not known in advance (see above). Thus one can not simply combine any-time algorithms for analysis, transfer and generation and come up with an any-time system for automatic machine interpretation.

The whole system has to be interruptible or at least has to be able to respect the timing properties stated above. This implies that results are propagated through the system as early as possible. Every component must not only operate like an interruptible system but has to produce output at every time on the fly. Every partial result that can be valuable for the following components has to be sent out. No component may wait until an interruption is done by the user or an overall scheduling monitor. There has to be a constant flow of results through the system. The only part that is able to show interruptible behaviour is the final speech-realizing subsystem. This one waits for a short period after the source language utterance and then starts to realize the target language equivalent as then available. This is clearly an any-time fashion.

To summarize, the properties of an interpretation system and its components should be the following:

- The system as a whole has to show any-time behaviour. It should be interruptible or at least be able to produce a result only seconds after the end of an utterance.

- Every component works incremental and produces a constant flow of results that it feeds into the subsequent subsystems.
- The final component waits until the tolerance pause ends and then produces a result which is the best available at that time.

Chapter 7

Conclusion

The overall behaviour of an automatic interpretation system has strong consequences for the architecture and construction of the parts of such a system. The transfer stage has to meet certain criteria because it is part of an interpretation system. We mentioned some of them, especially

- Incrementality, the ability to start working as soon as data is available
- Interactivity to eliminate ambiguity as soon as possible through cooperative work that is carried out together with other components
- The ability to sort transfer rule applications according to their presumed quality augmentation

We characterized current models of unification-based transfer and proposed a formalism variant to be used within our architectural experiments. Transfer rules formulated with this formalism are used for chart-based transfer, a mechanism that adapts well-known notions and strategies from chart-based parsing. These techniques allow the incremental operation of the transfer stage which is highly desirable in our perspective.

Further work will focus on three lines of research:

- What impact do strategies for chart-based transfer have upon the power of a transfer module and upon the overall power of an interpretation system?
- Can any time-behaviour of an interpretation system be achieved by constantly augmenting output of several system stages?

- What kind of interaction is crucial for transfer and how can it be modeled?

Bibliography

Amtrup and Weber 1994: Amtrup, Jan W. and Weber, Hans. „Architektur und Formalismus“. Interne Notiz, January 1994.

Amtrup 1992: Amtrup, Jan W. „Parallele Strukturanalyse Natürlicher Sprache mit Transputern“. ASL-TR 44-92/UHH, Univ. of Hamburg, 1992.

Beskow 1993: Beskow, Björn. „Unification Based Transfer: Multilingual Support for Translation and Writing“. Draft, Uppsala University, Uppsala, Sweden, February 1993.

Briscoe 1987: Briscoe. *Modelling Human Speech Comprehension: A Computational Approach.* Wiley, 1987.

Carlson and Vilkuna 1990: Carlson, Lauri and Vilkuna, Maria. „Independent Transfer Using Graph Unification“. In: *Proc. of the 13th COLING*, pages 3/60–3/63, Helsinki, Finnland, 1990.

Dorr 1993: Dorr, Bonnie Jean. *Machine Translation: A View from the Lexicon.* MIT Press, 1993.

Eberle et al. 1992: Eberle, Kurt, Kasper, Walter, and Rohrer, Christian. „Contextual Constraints for MT“. In: *Proc. of the 4th Int. Conf. on Theoretical and Methodological Issues in Machine Translation*, pages 213–224, Montreal, June 1992.

Emele and Zajac 1990: Emele, Martin E. and Zajac, Rémi. „Typed Unification Grammars“. In: *Proc. of the 13th COLING*, pages 293–298, Helsinki, Finnland, 1990.

Emele and Zajac 1991: Emele, Martin and Zajac, Rémi. „A Fixed-Point Semantics for Feature Type Systems“. Technical report, Univ. of Stuttgart, 1991.

Euler 1992a: Euler, Lutz. „The ASL Feature Term Formalism“. Technical report, Univ. of Hamburg, December 1992.

Euler 1992b: Euler, Lutz. „Die Programmierschnittstelle des ASL-Attributterm-Formalismus“. Technical Report ASL-Memo-48-92/UHH, Univ. of Hamburg, April 1992.

Euler 1992c: Euler, Lutz. „Syntax des ASL-Attributterm-Formalismus“. Technical Report ASL-Memo-45-92/UHH, Univ. of Hamburg, April 1992.

Görz 1993: Görz, Günther. „Kognitiv orientierte Architekturen für die Sprachverarbeitung“. Technical Report ASL-TR-39-92, Universität Erlangen-Nürnberg, February 1993.

Hauenschild and Prahl 1994: Hauenschild, Christa and Prahl, Birte. „Konzept Translationsprobleme - Translationsstrategien“. Technical report, Univ. of Hildesheim, 1994.

Hauenschild 1985: Hauenschild, Christa. „KIT/NASEV oder die Problematik des Transfers bei der Maschinellen Übersetzung“. KIT Report 29, Technische Universität Berlin, Berlin, November 1985.

Kaplan and Bresnan 1982: Kaplan, Ronald M. and Bresnan, Joan. „Lexical-Functional Grammar: A Formal System for Grammatical Representation“. In: Bresnan, Joan, ed. , *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, Ma., 1982.

Kaplan et al. 1989: Kaplan, R., Netter, K., Wedekind, J., and Zaenen, A. „Translation by Structural Correspondence“. In: *Proc. of the 4 th EACL*, Manchester, UK, 1989.

Karlgren 1994: Karlgren, Jussi. „Mumbling — User-Driven Cooperative Interaction“. Technical report, SICS, January 1994.

Kay 1973: Kay, Martin. „The MIND System“. In: Rustin, R., ed. , *Natural Language Processing*, pages 155–188. Algorithmic Press, New York, 1973.

Kay 1980: Kay, Martin. „Algorithmic Schemata and Data Structures in Syntactic Processing“. Technical Report CSL-80-12, Xerox Palo Alto Research Center, Palo Alto, 1980.

Kay 1984: Kay, Martin. „FUG: A Formalism for Machine Translation". In: *Proc. of the 10 th COLING*, pages 75–78, Stanford, CA, 1984.

Menzel 1994: Menzel, Wolfgang. „Parsing of Spoken Language under Time Constraints". submitted to ECAI 1994, 1994.

Morimoto et al. 1992: Morimoto, Tsuyoshi, Suzuki, Masami, Takezawa, Tosiyuki, Kikui, Genichiro, Nagata, Masaaki, and Tomokiyo, Mutsuko. „A Spoken Language Translation System: SL-TRANS2". In: *Proc. of the 14 th COLING*, pages 1048–1052, Nantes, France, 1992.

Noord 1990: Noord, Gertjan van. „Reversible Unification Based Machine Translation". In: *Proc. of the 13 th COLING*, pages 299–304, Helsinki, Finnland, 1990.

Pyka 1991: Pyka, Claudius. „Deterministische, inkrementelle und zeitsynchrone Verarbeitung und die Architektur von ASL-Nord". Technical Report ASL-TR-22-91/UHH, Universität Hamburg, Hamburg, December 1991.

Russel and Zilberstein 1991: Russel, Stuart J. and Zilberstein, Shlomo. „Composing Real-Time Systems". In: *Proc. of the 12 th IJCAI*, pages 212–217, August 1991.

Satta and Stock 1989: Satta, G. and Stock, Oliviero. „Formal Properties and Implementation of Bidirectional Charts". In: *Proc. International Joint Conference on Artificial Intelligence*, pages 1480–1485, Detroit, Mich., 1989.

Schütz 1988: Schütz, (Ed.), Jörg. „Workshop Semantik und Transfer. WP 6/88". Eurotra-d working papers, IAI, Saarbrücken, 1988.

Wahlster 1993: Wahlster, Wolfgang. „Translation of Face-to-Face-Dialogs". In: *Proc. MT Summit IV*, pages 127–135, 1993.

Wirén 1992: Wirén, Mats. *Studies in Incremental Natural-Language Analysis.* PhD thesis, Linköping University, 1992.

Zajac 1992: Zajac, Rémy. „Inheritance and Constraint-Based Grammar Formalisms". *Computational Linguistics*, 18(2):159–182, 1992.