Linear Logic Based Transfer and Structural Misalignment

Dick Crouch
Xerox PARC
USA
crouch@parc.xerox.com

Anette Frank
DFKI
Germany
frank@dfki.de

Josef van Genabith
Dublin City University
Eire
josef@compapp.dcu.ie

Abstract

[van Genabith *et al.*, 1998] described an approach to ambiguity preserving machine translation, where transfer takes place on the glue language meaning constructors of [Dalrymple *et al.*, 1996]. Unfortunately, it did not deal with structural misalignment problems, such as embedded head switching, in a fully satisfactory way. This paper proposes the use of a fragment of linear logic as a transfer formalism, and shows how it provides a more general and satisfactory solution to the difficulties encountered by [van Genabith *et al.*, 1998].

1 Introduction

In machine translation, ambiguities in the source language often carry across to the target language. These include syntactic ambiguities, such as some prepositional phrase attachments, (*John saw the man with a telescope / Jean a vu l'homme avec un téléscope*) or semantic ambiguities such as quantifier scope (*Every student answered a question / Jeder Student beantwortete eine Frage*). Rather than mechanically trying to pick a single intended interpretation of the source utterance, more accurate translation is likely if the full range of ambiguity can be preserved, leaving it to the human interpreter to resolve the ambiguity in the target. In cases like the above, a single sentence preserves all the ambiguities; in others, ambiguity preservation may necessitate generating a (hopefully small) range of alternatives.

Proposals for ambiguity preserving translation typically involve transferring an underspecified semantic representation of the source sentence to an underspecified representation of the target, and from it generating target sentences, e.g. [Alshawi *et al.*, 1991, Emele and Dorna 1998]. A variant of this approach was proposed by [van Genabith *et al.*, 1998] (henceforth GFD), where transfer takes place on lexical meaning constructors of the kind used in glue semantics [Dalrymple *et al.*, 1996]. As GFD point out, these lexical meaning constructors provide a form of underspecified semantic representation, allowing one to determine when transfer preserves semantic ambiguity. Transfer at the level of glue constructors also has other advantages. It allows for a highly lexicalized, reversible, and semi-automatable definition of transfer rules by comparing lexical entries from two monolingual lexicons. Since meaning constructors actually provide an encoding of the syntax-semantics interface, generation of target sentences is more direct than it would be from a purely semantic representation.

Precisely because glue meaning constructors encode the syntax-semantics interface, transfer at this level faces problems of structural misalignment, familiar from purely syntax-based approaches to transfer [Kaplan et al. 1989]. One of the most notorious cases of this is (embedded) head switching, two treatments of which are discussed by GFD, neither of them fully satisfactory.

This paper provides a more satisfactory account of structural misalignment. As with GFD the source sentence is parsed, and a set of instantiated lexical meaning constructors obtained, to which transfer rules are applied. However, the result of application is not a set of target meaning constructors. Instead it is a set of *transfer* constructors; a linear logic derivation consumes these to produce a set of target meaning constructors, from which the target sentence is generated. The resource-sensitive nature of the transfer derivation allows problematic cases of structural misalignment to be dealt with locally and lexically. Moreover, transfer derivations are structurally similar to glue derivations: techniques for efficient glue derivation, e.g. [Gupta and Lamping 1998], can be exported directly to transfer derivations.

2 Glue Semantics and Transfer

2.1 Glue Semantics

Glue semantics embodies a notion of 'interpretation as deduction' closely related to the 'parsing as deduction' paradigm of categorial grammar. A glue logic is used to deductively piece together the meanings of words and phrases in a (syntactically analysed) sentence, to assemble the meaning of the sentence as a whole. The meaning logic, used to represent the meanings of words and phrases, is quite distinct from the glue logic used to assemble those meanings.

Following [Dalrymple *et al.*, 1999a], we use a minor extension of the implication-only fragment of propositional linear logic as the glue logic, and a 'vanilla' logic of generalised quantifiers as the meaning language. We also adopt their 'Curry-Howard' formulation of glue semantics, where meaning language expressions are treated as terms labelling glue logic formulas. This replaces the older notation of [Dalrymple *et al.*, 1996], with its uninterpreted meaning assignment predicate $\leadsto$. This has the distinct advantages of (i) completely separating the glue and meaning logics, and (ii) removing the need to use higher-order unification in glue derivations.

Although glue semantics is not necessarily restricted to Lexical Functional Grammar [Kaplan and Bresnan, 1982], we will employ LFG as our syntactic base. We illustrate glue semantics by means of the simple example "*Hans cooks.*" Assume the following two lexical entries

$$\begin{array}{lll} \text{cooks} & \text{V} & \uparrow \text{PRED} = \text{cook}\langle\uparrow\text{SUBJ}\rangle \\ & & \mathit{cook} : (\uparrow\text{SUBJ})_\sigma \multimap \ \uparrow_\sigma \\[1ex] \text{Hans} & \text{NP} & \uparrow \text{PRED} = \text{Hans} \\ & & \mathit{hans} : \uparrow_\sigma \end{array}$$

The $\uparrow$ meta-variables refer to the nodes in f(unctional)-structure onto which the lexical items project in a given parse. The glue constructors, shown on the second line of each entry, refer to semantic (σ) projections of these f-structure nodes: these correspond to resources that consume and produce meanings. The constructor for "*Hans*" pairs the meaning term *hans* with the resource $\uparrow_\sigma$. The constructor for the intransitive verb "*cooks*" pairs the one-place meaning predicate *cook* with the implication $(\uparrow \text{SUBJ})_\sigma \multimap \ \uparrow_\sigma$. The implication says that the meaning of the verb's subject, $(\uparrow\text{SUBJ})_\sigma$ must be consumed in order to produce the meaning of the clause headed by the verb, $\uparrow_\sigma$.

Assume a grammar that, with this lexicon, derives the following f-structure for the example sentence, where f and g are arbitrary labels used to name the f-structure nodes. In doing so, the parse instantiates the $\uparrow$ meta-variables in the glue constructors to give the instantiated constructors shown alongside:

$$f : \begin{bmatrix} \text{PRED} & \text{cook } \langle\uparrow\text{SUBJ}\rangle \\ \text{SUBJ} & g : \begin{bmatrix} \text{PRED} & \text{Hans} \end{bmatrix} \end{bmatrix} \qquad \begin{array}{lcl} \mathit{cook} & : & g_\sigma \multimap f_\sigma \\ \mathit{hans} & : & g_\sigma \end{array}$$

Here, f_σ and g_σ correspond to f-structure nodes, but denote semantic resources.

The instantiated meaning constructors form the premises to a *glue derivation.* The goal of a glue derivation is to consume all the lexically obtained premises to prove that there is a single semantic resource corresponding to the outermost f-structure node producing a meaning.

Ignoring the meaning terms for the moment, in our example there are two lexical premises, g_σ and $g_\sigma \multimap f_\sigma$, and we need to prove f_σ. A simple derivation suffices:

$$\frac{g_\sigma \multimap f_\sigma \quad g_\sigma}{f_\sigma} \multimap_{\mathcal{E}}$$

The Curry-Howard isomorphism links the natural deduction rule of implication elimination ($\multimap_{\mathcal{E}}$ or *modus ponens*) with the functional application of the proof/meaning terms of the two premises. (Implication introduction gives rise to λ-abstraction.) The derivation above consequently automatically constructs the meaning term *cook*(*hans*) for the sentence, as follows

$$\frac{\mathit{cook} : g_\sigma \multimap f_\sigma \quad \mathit{hans} : g_\sigma}{\mathit{cook}(\mathit{hans}) : f_\sigma} \multimap_{\mathcal{E}}$$

This is, of course, a very simple illustrative example. However, in all more complex cases a propositional linear logic derivation builds the scaffolding on which meaning terms are combined by means of functional application or λ-abstraction, as dictated by the proof rules used.

In many cases, though not in the example above, distinct glue derivations, constructing distinct meaning terms, can be obtained from a single set of glue premises. These multiple derivations account for non-syntactic ambiguities like quantifier scope, as we will see later.

2.2 Generation from Instantiated Constructors

Starting just with the instantiated meaning constructors and the lexicon, it is possible to reconstruct the f-structure of our example sentence. Using the meaning terms as indices into the lexicon, we can retrieve the entries for "*Hans*" and "*cooks*". Comparing the instantiated and uninstantiated constructors

$$\mathit{cook} : g_\sigma \multimap f_\sigma \qquad\qquad \mathit{cook} : (\uparrow \textsc{subj})_\sigma \multimap \ \uparrow_\sigma$$

we can see that node g is the SUBJ of node f. Moreover, by looking at the feature equations in the entry for "*cooks*", namely

$$\uparrow \textsc{pred} = \text{cook}\langle \uparrow \textsc{subj} \rangle$$

we can determine what the PRED of f is. Likewise, by matching the instantiated constructors $hans : g_\sigma$ against the uninstantiated entry for "*Hans*", we can determine the PRED of f's subject (i.e. g). This gives us enough information to reconstruct the original f-structure. And from this, we generate the original sentence.

2.3 Basic Transfer on Glue Constructors

Suppose we have a German lexicon including the following two entries

kocht	V	$\uparrow$ PRED = kochen$\langle\uparrow$SUBJ$\rangle$	$kochen : (\uparrow \textsc{subj})_\sigma \multimap \uparrow_\sigma$
Hans	NP	$\uparrow$ PRED = Hans	$hans : \uparrow_\sigma$

and a grammar that derives the following f-structure for the sentence "*Hans kocht*" (*Hans cooks*), with instantiated meaning constructors shown alongside:

$$f : \begin{bmatrix} \textsc{pred} & \text{kochen } \langle\uparrow\textsc{subj}\rangle \\ \textsc{subj} & g : \begin{bmatrix} \textsc{pred} & \text{Hans} \end{bmatrix} \end{bmatrix} \qquad \begin{array}{rcl} kochen & : & g_\sigma \multimap f_\sigma \\ hans & : & g_\sigma \end{array}$$

By the previous section, given the instantiated constructors and the German lexicon, we could generate the German f-structure and hence the German sentence.

Starting from the previously mentioned instantiated source (English) constructors — $cook : g_\sigma \multimap f_\sigma$ and $hans : g_\sigma$ — the following transfer rules yield the required instantiated target (German) constructors

$$\begin{array}{llcl} \forall G, F & cook : G \multimap F & \Leftrightarrow & kochen : G \multimap F \\ \forall G & hans : G & \Leftrightarrow & hans : G \end{array}$$

from which generation of the target sentence can proceed.

GFD make a number of points about this transfer scheme. First, the transfer rules are in many cases derivable from a simple comparison of paired lexical entries, and much of this can be done automatically. Second, neither the instantiated constructors nor the transfer rules make reference to f-structure attributes such as SUBJ or OBJ. Information about these attributes is only obtained by matching instantiated constructors against mono-lingual lexical entries. GFD exploit this to deal with argument switching, as in *Das Photo ist Hans mißlungen – Hans a râté la photo* (*Hans messed up/ruined the photo*), where grammatical roles get switched. Third, in cases where the source and target constructors are isomorphic, the range of possible glue derivations is preserved, thus preserving semantic ambiguity. This is illustrated by showing how scope ambiguities can be preserved in transfer.

3 Head Switching

Head switching is exemplified by the English — German translation pair:

Hans kocht gerne $\leftrightarrow$ *Hans likes cooking*

The German attitudinal adjunct *gerne* is translated in English as a control construction involving the verb *like*. Syntactically *like* is the head of the English sentence (the sentence is the maximal projection of *like*) whereas *gerne* is an adverbial subconstituent of the German sentence. These differences are manifest in the corresponding f-structures:

$$f_1 : \begin{bmatrix} \textsc{pred} & \text{kochen}\langle f_2\rangle \\ \textsc{subj} & f_2 : \begin{bmatrix} \textsc{pred} & \text{hans} \end{bmatrix} \\ \textsc{adjn} & \{ f_3 : \begin{bmatrix} \textsc{pred} & \text{gerne} \end{bmatrix} \} \end{bmatrix} \quad f_3 : \begin{bmatrix} \textsc{pred} & \text{like}\langle f_2, f_1\rangle \\ \textsc{subj} & f_2 : \begin{bmatrix} \textsc{pred} & \text{hans} \end{bmatrix} \\ \textsc{xcomp} & f_1 : \begin{bmatrix} \textsc{pred} & \text{cook}\langle f_2\rangle \\ \textsc{subj} & f_2 : \begin{bmatrix} \textsc{pred} & \text{hans} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Note that in translation from, say, the German to the English f-structure, the translation of the embedded adjunct f-structure f_3 turns out to be embedding the translation of the rest of the source f-structure f_1 in target. Transfer on f-structure representations has to involve a complex inside-out folding operation. Worse still is where a head switching case is embedded inside another structure as in

Ede vermutet daß Hans gerne kocht $\leftrightarrow$ *Ede assumes that Hans likes cooking*

$$f_1 : \begin{bmatrix} \textsc{pred} & \text{vermuten}\langle f_2, f_3\rangle \\ \textsc{subj} & f_2 : \begin{bmatrix} \textsc{pred} & \text{ede} \end{bmatrix} \\ \textsc{comp} & f_3 : \begin{bmatrix} \textsc{pred} & \text{kochen}\langle f_4\rangle \\ \textsc{subj} & f_4 : \begin{bmatrix} \textsc{pred} & \text{hans} \end{bmatrix} \\ \textsc{adjn} & \{ f_5 : \begin{bmatrix} \textsc{pred} & \text{gerne} \end{bmatrix} \} \end{bmatrix} \end{bmatrix} \quad f_1 : \begin{bmatrix} \textsc{pred} & \text{assume}\langle f_2, f_5\rangle \\ \textsc{subj} & f_2 : \begin{bmatrix} \textsc{pred} & \text{ede} \end{bmatrix} \\ \textsc{comp} & f_5 : \begin{bmatrix} \textsc{pred} & \text{like}\langle f_4, f_3\rangle \\ \textsc{subj} & f_4 : \begin{bmatrix} \textsc{pred} & \text{hans} \end{bmatrix} \\ \textsc{xcomp} & f_3 : \begin{bmatrix} \textsc{subj} & f_4 : \\ \textsc{pred} & \text{cook}\langle f_4\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Consider again the translation from German into English (the other direction is analogous). Here *vermuten* expects an f-structure f_3 as its complement and so would its translation *assume*. Now, during translation we have a head switching operation in the complement between f_5 and f_3 (the translation of the embedded source f_5 turns out to be embedding in target) and *assume* which expects f_3 is offered f_5, resulting in a disconnected f-structure.

3.1 Head-Switched Meaning Constructors

The following are instantiated meaning constructors for the German sentence (σ subscripts omitted to avoid clutter):

$$\begin{array}{rl}
\mathit{ede}: & f_2 \\
\mathit{vermuten}: & f_2 \multimap (f_3 \multimap f_1) \\
\mathit{hans}: & f_4 \\
\mathit{kochen}: & f_4 \multimap f_3 \\
\lambda P, x.\ \mathit{gerne}(x, P(x)): & (f_4 \multimap f_3) \multimap (f_4 \multimap f_3)
\end{array}$$

These lead to the following glue derivation

$$\dfrac{\dfrac{f_2 \multimap (f_3 \multimap f_1) \qquad f_2}{f_3 \multimap f_1} \qquad \dfrac{\dfrac{(f_4 \multimap f_3) \multimap (f_4 \multimap f_3) \qquad f_4 \multimap f_3}{f_4 \multimap f_3} \qquad f_4}{f_3}}{\mathit{vermuten}(\mathit{ede}, \mathit{gerne}(\mathit{hans}, \mathit{kochen}(\mathit{hans}))) : f_1}$$

with the final meaning term shown.

The instantiated English constructors for "*Ede*", "*Hans*" and "*cooking*" differ only from the German constructors in their meaning terms. But note the differences between the constructors for *likes–gerne* and *assumes–vermutet*:

$$\begin{array}{rl}
\lambda P, x.\mathit{like}(x, P(x)) & : (f_4 \multimap f_3) \multimap (f_4 \multimap f_5) \\
\lambda P, x.\mathit{gerne}(x, P(x)) & : (f_4 \multimap f_3) \multimap (f_4 \multimap \underline{f_3})
\end{array}$$

$$\begin{array}{rl}
\mathit{assume} & : f_2 \multimap (f_5 \multimap f_1) \\
\mathit{vermuten} & : f_2 \multimap (\underline{f_3} \multimap f_1)
\end{array}$$

The node f_5 in the English constructors replaces the underlined occurrences of f_3 in the German constructors.

Since the *gerne–likes* translation clearly needs to introduce an extra level of structure, we might envisage a purely lexical transfer rule

$$\begin{array}{rll}
\forall G, F. & \lambda P, x.\mathit{gerne}(x, P(x)) & : (G \multimap F) \multimap (G \multimap F) \\
\Leftrightarrow & \lambda P, x.\mathit{like}(x, P(x)) & : (G \multimap F) \multimap (G \multimap \mathit{New})
\end{array}$$

where G and F range over matched structures, and New denotes the additional node introduced by the English control construction.

The problem with this is that a similar, purely lexical transfer rule for *vermuten–assume* would most naturally be

$$\begin{array}{rll}
\forall G, H, F. & \mathit{vermuten} & : G \multimap (H \multimap F) \\
\Leftrightarrow & \mathit{assume} & : G \multimap (\underline{H} \multimap F)
\end{array}$$

In the absence of an embedded head switching, this transfer rule works well. But in the case where the complement of *vermuten* induces head switching on transfer, we need to replace the underlined occurrence of H by the newly introduced head switched node. How to do this solely on the basis of local, purely lexical transfer is described in the next section.

4 Linear Logic Transfer Constructors

To summarize the embedded head switching problem from the last section: translating "*gerne*" to "*likes*" involves wrapping an extra layer of structure, f_5, around f_3. The constructor that was originally expecting to consume f_3, obtained by translating "*vermutet*" to "*assumes*", has to be told to consume f_5 instead of f_3. We would like this change to be communicated while only using local, purely lexical transfer rules.

Another way of describing what happens is that the *gerne–likes* transfer associates a new 'topmost' structure with f_3. In the German sentence, f_3 is its own topmost structure, which we represent by the assertion $T(f_3, f_3)$. The *gerne–likes* transfer updates this assertion with $T(f_3, f_5)$. The meaning constructor for "*assumes*" needs to consume the topmost structure associated with f_3, whatever that structure happens to be.

The association of a topmost structure with a node does not take place within glue *meaning* constructors — the association simply does not make any sense there. Instead, we will make these associations within linear logic based *transfer* constructors. In order to keep the transfer logic distinct from the glue logic, we will use $\multimap_{\mathcal{T}}$ and $\otimes_{\mathcal{T}}$ to refer to the connectives of the transfer logic.

The basic transfer architecture is this. A set of lexically defined transfer rules map instantiated source meaning constructors onto transfer constructors. The transfer constructors are premises to a *transfer derivation*. By analogy to glue derivations, the goal of a transfer derivation is to prove a single assertion about the topmost structure associated with outermost source f-structure node. A consequence of deriving this will be to produce a set of instantiated target meaning constructors, from which generation of the target sentence can proceed.

4.1 A Transfer Derivation

Section 4.3 describes the transfer rules mapping source meaning constructors onto transfer constructors. In this section, we merely state what the transfer constructors are for our German–English embedded head switching example, and show how the transfer derivation proceeds.

Recall the German source meaning constructors (meaning terms slightly simplified, and numbered for ease of reference):

$$\begin{array}{lrl}
1. & \mathit{ede}: & f_2 \\
2. & \mathit{hans}: & f_4 \\
3. & \mathit{kochen}: & f_4 \multimap f_3 \\
4. & \mathit{vermuten}: & f_2 \multimap f_3 \multimap f_1 \\
5. & \mathit{gerne}: & (f_4 \multimap f_3) \multimap (f_4 \multimap f_3)
\end{array}$$

From the source meaning constructors and the transfer mapping rules we obtain the following transfer constructors:

$$\begin{array}{ll}
1. & T(f_2, f_2) \quad \otimes_{\mathcal{T}} \mathit{ede} : f_2 \\
2. & T(f_4, f_4) \quad \otimes_{\mathcal{T}} \mathit{hans} : f_4 \\
3.\ \forall X. & [T(f_4, X) \multimap_{\mathcal{T}} T(f_3, f_3)] \quad \otimes_{\mathcal{T}} \mathit{cook} : X \multimap f_3 \\
4.\ \forall X, Y. & [T(f_2, X) \multimap_{\mathcal{T}} (T(f_3, Y) \multimap_{\mathcal{T}} T(f_1, f_1))] \\
 & \quad \otimes_{\mathcal{T}} \mathit{assume} : X \multimap (Y \multimap f_1) \\
5.\ \forall X, Y. & [(T(f_4, X) \multimap_{\mathcal{T}} T(f_3, Y)) \multimap_{\mathcal{T}} (T(f_4, X) \multimap_{\mathcal{T}} T(f_3, \mathit{new}))] \\
 & \quad \otimes_{\mathcal{T}} \mathit{like} : (X \multimap Y) \multimap (X \multimap \mathit{new})
\end{array}$$

Each transfer constructor is a conjunction of two formulas: a transfer formula that consumes and produces topmost node assertions, and a glue formula giving a target meaning constructor. For example, transfer constructor (1) says that f_2 is its own topmost node, and produces the meaning constructor *ede* : f_2. Transfer constructor (3) consumes an assertion about the topmost node of f_4 to produce an assertion that f_3 is its own topmost node. It also produces the meaning constructor *cook* : $X \multimap f_3$, where X is whatever topmost node was associated with f_4. Constructor (5) is the crucial one, but is best understood after looking at the transfer derivation. Note how, in all cases, the transfer formula replicates exactly the structure of the target glue formula.

The transfer derivation from premises 1–5 proceeds as follows (meaning terms in glue constructors omitted, and glue constructors in smaller font). First combine premises (3) and (5)

$$\begin{array}{ll}
3 & T(f_4, X) \multimap_{\mathcal{T}} T(f_3, f_3) \quad \otimes_{\mathcal{T}} X \multimap f_3 \\
5 & (T(f_4, X) \multimap_{\mathcal{T}} T(f_3, Y)) \multimap_{\mathcal{T}} (T(f_4, X) \multimap_{\mathcal{T}} T(f_3, \mathit{new})) \\
 & \quad \otimes_{\mathcal{T}} (X \multimap Y) \multimap (X \multimap \mathit{new}) \\
\rhd & T(f_4, X) \multimap_{\mathcal{T}} T(f_3, \mathit{new}) \quad \otimes_{\mathcal{T}} X \multimap f_3 \otimes_{\mathcal{T}} (X \multimap f_3) \multimap (X \multimap \mathit{new})
\end{array}$$

This associates with f_3 a new topmost node, *new*, provided that we can find the topmost node of f_4. The value *new* is instantiated in one of the meaning constructors. Premise (2) produces f_4 as its own topmost node, allowing us to conclude:

$$\begin{array}{ll} & T(f_4, X) \multimap_{\mathcal{T}} T(f_3, new) \quad \scriptstyle \otimes_{\mathcal{T}} X \multimap f_3 \otimes_{\mathcal{T}} (X \multimap f_3) \multimap (X \multimap new) \\ 2 & T(f_4, f_4) \quad \scriptstyle \otimes_{\mathcal{T}} f_4 \\ \triangleright & T(f_3, new) \quad \scriptstyle \otimes_{\mathcal{T}} f_4 \multimap f_3 \otimes_{\mathcal{T}} (f_4 \multimap f_3) \multimap (f_4 \multimap new) \otimes_{\mathcal{T}} f_4 \end{array}$$

That is, *new* is now asserted to be the topmost node of f_3. This assertion combines with premise (4), corresponding to the word *assumes*, (and premise 1). *Assumes* consumes whatever the topmost node of f_3 is: in this case *new* rather than f_3. Hence

$$\begin{array}{ll} & T(f_3, new) \quad \scriptstyle \otimes_{\mathcal{T}} f_4 \multimap f_3 \otimes_{\mathcal{T}} (f_4 \multimap f_3) \multimap (f_4 \multimap new) \otimes_{\mathcal{T}} f_4 \\ 1 & T(f_2, f_2) \quad \scriptstyle \otimes_{\mathcal{T}} f_2 \\ 4 & T(f_2, X) \multimap_{\mathcal{T}} (T(f_3, Y) \multimap_{\mathcal{T}} T(f_1, f_1)) \quad \scriptstyle \otimes_{\mathcal{T}} X \multimap (Y \multimap f_1) \\ \triangleright & T(f_1, f_1) \\ & \quad \scriptstyle \otimes_{\mathcal{T}} f_4 \multimap f_3 \otimes_{\mathcal{T}} (f_4 \multimap f_3) \multimap (f_4 \multimap new) \otimes_{\mathcal{T}} f_4 \otimes_{\mathcal{T}} f_2 \otimes_{\mathcal{T}} f_2 \multimap (new \multimap f_1) \end{array}$$

This consumes all the transfer constructors, results in a single assertion that f_1 is its own topmost structure, and produces the desired set of target meaning constructors. No other derivation consuming all the premises and producing a single $T(f_1, \_)$ assertion is possible. Note how the last step of the derivation instantiates the variable Y to the value *new* in the meaning constructor for *assumes*, communicating the changes brought about by the head switch in the first step of the derivation.

4.2 Multiple Head Switching

This approach generalizes straightforwardly to cases of multiple head switching, e.g. "*Hans schliesslich kocht gerne*" where the adverb *schliesslich* is analogous to *gerne*, and translates into the English control verb *ends up*. The sentence can translate either as *Hans ends up liking cooking* or as *Hans likes ending up cooking.* This ambiguity corresponds to an adverb scope ambiguity in German, and is reflected in transfer by the availability of two transfer derivations.

From the German source constructors (meaning terms simplified)

1	*hans* :	f_2
2	*kochen* :	$f_2 \multimap f_1$
3	*gerne* :	$(f_2 \multimap f_1) \multimap (f_2 \multimap f_1)$
4	*schliesslich* :	$(f_2 \multimap f_1) \multimap (f_2 \multimap f_1)$

it is evident that the two adverbials (3) and (4) are of the same type, and can permute in either order around the *kochen* constructor (2). Assuming similar transfer rules for *schliesslich* and *gerne*, the transfer constructors will be (meaning constructors omitted)

1 $T(f_2, f_2)$
2 $T(f_2, X) \multimap_T T(f_1, f_1)$
3 $(T(f_2, X) \multimap_T T(f_1, Y)) \multimap_T (T(f_2, X) \multimap_T T(f_1, new_1))$
4 $(T(f_2, X) \multimap_T T(f_1, Y)) \multimap_T (T(f_2, X) \multimap_T T(f_1, new_2))$

It is likewise evident that the transfer constructors (3) and (4) can permute in either order around (2). If (3) and (2) are combined first (the *ends up liking* translation) the top of f_1 is first updated to new_1, and then by (4) to new_2. If (4) and (2) are combined first (the *likes ending up* translation), the top of f_1 is first updated to new_2 and then to new_1.

This is a case where ambiguity preservation necessitates the generation of two target sentences. Because transfer derivations mirror ambiguities in the glue derivations, we succeed in detecting the two sentences required.

4.3 Deriving Transfer Rules

Obtaining transfer rules from aligned monolingual lexicons proceeds along the same lines as for GFD. The hard part is to recognise the parallel semantic resources in the source and target constructors. In many cases this can be done either through recognition of parallel f-structure attributes in source and target, or balancing up occurrences of distinct resources on either side. Hard cases, or where it is clear that there is not complete parallelism (as in head switching) can be passed to human rule writers. As an example, in comparing the entries

$$\begin{array}{ll} \textit{vermuten}: & (\uparrow\textsc{subj}) \multimap (\uparrow\textsc{xcomp}) \multimap \uparrow \\ \textit{assume}: & (\uparrow\textsc{subj}) \multimap (\uparrow\textsc{xcomp}) \multimap \uparrow \end{array}$$

it is easy to identify $\uparrow$, $(\uparrow\ \textsc{subj})$ and $(\uparrow\ \textsc{xcomp})$ as parallel resources in source and target. The source side of the transfer rule is given by the source meaning constructor with variables in place of the parallel resources. The resulting transfer constructor is obtained by making two copies of the target constructor, again with variables in place of parallel resources. We strip the meaning term off the first copy to form the basis of the transfer formula, giving as an intermediate stage

$$\begin{array}{rl} \forall F, G, H & \textit{vermuten} : G \multimap (H \multimap F) \\ \Rightarrow & [G \multimap (H \multimap F)] \otimes_T \textit{assume} : G \multimap (H \multimap F) \end{array}$$

We now identify the rightmost consequent variable in the transfer formula, in this case F. We replace this by the predication $T(F,F)$. All other variables are associated a unique topmost variables, e.g. $T(G,X)$, and the variables in the transfer formula replaced by these predications. Variables in the meaning constructors are replaced by their associated topmost variables. The associated topmost variables are universally quantified with scope over the whole transfer constructor. Thus we finally obtain the transfer rule:

$$\begin{array}{ll} \forall F,G,H & \mathit{vermuten} : G \multimap (H \multimap F) \\ \quad\Rightarrow & \forall X,Y.\ [T(G,X) \multimap (T(H,Y) \multimap T(F,F))] \\ & \qquad \otimes_{\mathcal{T}}\ \mathit{assume} : X \multimap (Y \multimap F) \end{array}$$

This way of constructing transfer rules ensures that transfer formulas exactly mirror target glue formulas. As a result, transfer derivations mirror glue derivations

4.4 Quantifiers

One exception to this exact correspondence between transfer and glue formulas occurs in the case of quantifier meanings. A quantified pronoun like "*everyone*" illustrates the standard glue treatment of quantifiers, and is given a meaning constructor

$$\mathit{everyone} : (\uparrow_\sigma \multimap \mathcal{S}) \multimap \mathcal{S}$$

where $\mathcal{S}$ is a variable that can range over atomic semantic resources (the scope of the quantifier). The formula $(\uparrow_\sigma \multimap \mathcal{S}) \multimap \mathcal{S}$ is just a type raised version of the atomic formula $\uparrow_\sigma$. The transfer formula in the constructed rule is taken from the lower-type formula. Thus, for example

$$\begin{array}{ll} \forall G & \mathit{jeder} : (G \multimap \mathcal{S}) \multimap \mathcal{S} \\ \Rightarrow & T(G,G) \quad \otimes_{\mathcal{T}}\ \mathit{everyone} : (G \multimap \mathcal{S}') \multimap \mathcal{S}' \end{array}$$

Assuming a similar transfer rule for "*etwas*" (*something*), the transfer constructors obtained from the sentence "*Jeder sah etwas*" (*everyone saw something*) would be

1	$T(g,g)$	$\mathit{everyone}{:}(g \multimap \mathcal{S}') \multimap \mathcal{S}'$
2	$T(h,h)$	$\mathit{something}{:}(h \multimap \mathcal{S}) \multimap \mathcal{S}$
3	$T(g,X) \multimap (T(h,Y) \multimap T(f,f))$	$\mathit{see}{:}X \multimap (Y \multimap f)$

Here, there is just one transfer derivation, instantiating X to g and Y to h, despite the possibility of two distinct target glue derivations

4.5 The Nature of Transfer Derivations

As previously noted, transfer constructors parallel target glue constructors, so that transfer derivations parallel target glue derivations. This has a number of consequences. First, the existence of a transfer derivation guarantees the existence of a target glue derivation; we can be sure that we translate only into semantically interpretable sentences.

Second, techniques developed for efficient glue derivation (such as the skeleton-modifier approach of [Gupta and Lamping 1998]) can be applied directly to transfer derivations; there is sharing of technology.

Third, as observed in connection with multiple head-switching, different transfer derivations can lead to distinct sets of target constructors. This arises in cases where there is no one target sentence that captures the full range of meanings open to the source sentence; ambiguity preservation necessitates the generation of multiple target sentences. Given the close connection between glue and transfer derivations, we can have some confidence that the correct ambiguities are being preserved.

However, in some cases it is formally possible to have multiple transfer derivations all leading to the same set of target constructors. This parallels what often happens in glue derivations where, e.g., distinct ways of scoping existentially quantified NPs all lead to logically equivalent meanings. (Note, though, that the type-lowered transfer constructors for quantified NPs actually eliminate spurious transfer derivations arising from quantifier scope ambiguities). Techniques for efficiently detecting and removing such equivalent glue derivations can fortunately also be applied to transfer derivations.

5 Conclusions

This paper presented a resource-sensitive approach to transfer. A source sentence is parsed, and a set of instantiated lexical meaning constructors is obtained. Transfer rules rewrite the source meaning constructors to a set of transfer constructors. A linear logic derivation consumes the transfer constructors to produce a set of instantiated target meaning constructors, from which a target sentence can be generated. The resource-sensitive nature of the transfer derivation allows problematic cases of structural misalignment to be dealt with smoothly and locally. In most cases, the transfer rules can be derived semi-automatically from aligned mono-lingual source and target lexicons. Cases where ambiguity preservation can only be achieved by multiple target translations are readily accommodated. Techniques developed for efficient linear logic derivations in the context of glue semantics apply directly to efficient transfer derivations.

Using linear logic for transfer is also suggested by [Fujinami 1999], but not applied to structural mismatch. The treatment of head-switching bears some relation to unpublished work of Martin Emele's, though it is not clear that his use of 'internal' and 'external' variables extends to cases of multiple head-switching. Although applied to transfer at the level of glue language meaning constructors, we would hope that our linear logic based transfer scheme could be extended to deal with structural mismatches at other levels of representation.

Finally the resource sensitive nature of the transfer derivations allows for the possibility that some target lexical glue constructors get consumed in transfer. This might apply, for example, in translating the two word English expression "*commit suicide*" into the French verb "*se suicider*": the transfer constructor for *commit–se suicider* can be set up so as to consume the results of transferring the noun "*suicide*". Examples such as this also often lead to a specificity ordering over transfer rules. It is an interesting question whether this kind of specificity ordering can receive a direct and explicit encoding in a linear logic based transfer scheme.

References

[Alshawi *et al.*, 1991] Alshawi, H.; Carter, D.; Gambäck, B.; and Rayner, M. 1991. Translation by quasi logical form transfer. In *Proceedings 29th Annual Meeting of the Association for Computational Linguistics (ACL'91).* 161–168.

[Dalrymple *et al.*, 1996] Dalrymple, M.; Lamping, J.; Pereira, F.C.N; and Saraswat, V. 1996. Quantification, anaphora, and intensionality. *Journal of Logic, Language and Information* 6(3) 219–273. Reprinted in Dalrymple, M, editor 1999, *Semantics and Syntax in Lexical Functional Grammar.* MIT Press.

[Dalrymple *et al.*, 1999a] Dalrymple, M.; Gupta, V.; Lamping, J.; and Saraswat, V. 1999a. Relating resource-based semantics to categorial semantics. In Dalrymple, M, editor 1999, *Semantics and Syntax in Lexical Functional Grammar.* MIT Press. 261–280.

[Emele and Dorna 1998] M.C. Emele and M. Dorna. Ambiguity preserving machine translation using packed representations. In *Proceedings of COLING-ACL'98*, Montréal, Canada. pp. 365–371, 1998.

[Fujinami 1999] T. Fujinami. A Decidable Logic for Speech Translation. Dagstuhl Workshop on Linear Logic and its Applications, August 1999.

[van Genabith *et al.*, 1998] Genabith, J. van; Frank, A.; and M., Dorna 1998. Transfer constructors. In Butt, M. and King, T. H., editors 1998, *Proceedings LFG'98, Brisbane, Australia.* CSLI Publications, http://www-csli.stanford.edu/publications/. 190–205.

[Gupta and Lamping 1998] V. Gupta and J. Lamping. Efficient Linear Logic Meaning Assembly. In *Proceedings of COLING-ACL'98*, Montréal, Québec, Canada. pp. 464–470, 1998.

[Kaplan and Bresnan, 1982] Kaplan, R.M. and Bresnan, J. 1982. Lexical functional grammar. In Bresnan, J., editor 1982, *The mental representation of grammatical relations.* MIT Press, Cambridge Mass. 173–281.

[Kaplan et al. 1989] R. Kaplan, K. Netter, J. Wedekind, and A. Zaenen. Translation by Structural Correspondences. In *Proceedings of 4th EACL)*, pages 272–281, Manchester, UK, 1989.

[Kaplan and Wedekind 1993] R. Kaplan and J. Wedekind. Restriction and Correspondence-based Translation. In *Proceedings of 6th EACL*, pages 193–202, Utrecht, The Netherlands, 1993.