Transfer Constructors

Josef van Genabith
Dublin City University
Computer Applications
Dublin 9, Ireland
+353+(0)1 704 5074
josef@compapp.dcu.ie

Anette Frank
Xerox Research Centre Europe
6, chemin de Maupertuis
F–38240 Meylan
+33+(0)4 76 61 50 37
Anette.Frank@xrce.xerox.com

Michael Dorna
IMS, Universität Stuttgart
Azenbergstr. 12
D–70174 Stuttgart
+49+(0)711 121 1367
michl@ims.uni-stuttgart.de

Proceedings of the LFG98 Conference
The University of Queensland, Brisbane
Miriam Butt and Tracy Holloway King (Editors)
1998
CSLI Publications
http://www-csli.stanford.edu/publications/

Transfer Constructors[1]

Abstract

We present a modular, lexicalized, reversible and ambiguity preserving approach to semantic based transfer on sets of linear logic meaning and transfer constructors. In many cases, transfer on sets of meaning constructors (rather than on derived disambiguated meaning assignments) obviates the need for spurious multiple transfer on disambiguations. We concentrate on adjuncts and embedded head switching phenomena.

1 Introduction

A set of meaning constructors induced by a σ projection of an f-structure [1] can be seen as an underspecified semantic representation [12]. Different scopes of quantifiers, modifiers, etc. are obtained via different deductions from such a set of meaning constructors. Transfer on sets of meaning constructors rather than on derived disambiguated meaning assignments holds the potential to avoid the complexity of multiple (often spurious) transfer on disambiguations.

To our knowledge the idea to formalize transfer rules in linear logic was first presented by Fujinami [5, 6] in a somewhat different setting (Verbmobil). In his case there is no distinction between meaning language and glue language. Here we apply linear logic based transfer in a LFG framework. Sections 2 and 3 introduce the approach, section 4 considers problematic cases of head switching. Section 5 concludes.

2 Meaning Constructors

Consider the following f-structure associated with the German sentence *Hans schwimmt*:

$$\begin{bmatrix} \text{SUBJ} & \left[\text{PRED} \quad \textsc{hans}\right] \boxed{2} \\ \text{PRED} & \textsc{schwimmen}\langle \uparrow \text{SUBJ} \rangle \end{bmatrix} \boxed{1}$$

In the linear logic based semantics of [1] the following meaning constructors are used:

$$\begin{array}{ll} hans: & \uparrow_\sigma \rightsquigarrow hans \\ schwimmen: & \forall X\ [(\uparrow \text{SUBJ})_\sigma \rightsquigarrow X \multimap \uparrow_\sigma \rightsquigarrow schwimmen(X)] \end{array}$$

Instantiated to the semantic projections of the f-structure nodes above we get a set of meaning constructors *Source*:

[1] We are grateful for helpful comments and suggestions by Mary Dalrymple, John Lamping, John Fry, Jürgen Wedekind, Hans Kamp, Ron Kaplan, Dick Crouch, Andy Way and our anonymous reviewers as well as the interest of the NLTT and IMS-ParGram groups at Xerox Parc, where earlier versions of this paper were presented. Mistakes and opinions expressed are our own. Part of this work was funded by the German Federal Ministry of Education, Science, Research and Technology (BMBF) in the framework of the Verbmobil project under grant 01 IV 701 N3.

$$Source = \left\{ \begin{array}{c} (f_2)_\sigma \rightsquigarrow hans \\ \forall X\ [(f_1\text{SUBJ})_\sigma \rightsquigarrow X \multimap (f_1)_\sigma \rightsquigarrow schwimmen(X)] \end{array} \right\}$$

Since $(f_1\text{SUBJ}) = f_2$, $Source \vdash (f_1)_\sigma \rightsquigarrow schwimmen(hans)$.

3 Transfer Constructors

3.1 A Simple Example

Next, consider the f-structure and the meaning assignments associated with the English sentence *Hans swims*:

$$\left[\begin{array}{l} \text{SUBJ}\ \left[\text{PRED HANS}\right]\boxed{2} \\ \text{PRED SWIM}\langle\uparrow \text{SUBJ}\rangle \end{array}\right]\boxed{1} \quad \left\{ \begin{array}{c} (f_2)_\sigma \rightsquigarrow hans \\ \forall X\ [(f_2)_\sigma \rightsquigarrow X \multimap (f_1)_\sigma \rightsquigarrow swim(X)] \end{array} \right\} \vdash (f_1)_\sigma \rightsquigarrow swim(hans)$$

Let the set of instantiated meaning constructors for the English sentence be referred to as *Target*. The task of transfer is to relate the two sets *Source* and *Target*. This can be achieved in terms of a set of linear logic "transfer constructors" *Trans*. In the forward direction, i.e. from *Source* to *Target*, each transfer constructor in *Trans* relates a source meaning constructor σ to a corresponding target meaning constructor τ: $\sigma \multimap \tau$. For the backward direction the $\multimap$ arrow is reversed (equivalently, its two arguments are swapped): $\tau \multimap \sigma$. For our simple example *Source* and *Target* are related by:

$$Trans = \left\{ \begin{array}{c} \forall F\ [F_\sigma \rightsquigarrow hans \multimap F_\sigma \rightsquigarrow hans] \\ \forall F\ [\forall X((F\ \text{SUBJ})_\sigma \rightsquigarrow X \multimap F_\sigma \rightsquigarrow schwimmen(X)) \\ \multimap \\ \forall X((F\ \text{SUBJ})_\sigma \rightsquigarrow X \multimap F_\sigma \rightsquigarrow swim(X))] \end{array} \right\}$$

Informally, below we will sometimes refer to particular transfer constructors as in $\{hans \multimap hans, schwimmen \multimap swim\}$. Note that in each case the transfer constructors have wide scope quantification over f-structure nodes. The f-structure nodes provide the scaffolding for how meaning constructors can be put together in derivations.[2] Hence, we need to carry this information over in transfer (otherwise constructors would not be connected in the target meaning constructor set).

The general idea is that the transfer constructor set *Trans* is used to rewrite the source meaning constructor set *Source* into a target meaning constructor set *Target*. Strictly speaking, we do not have separate source and target constructor sets (related by the rewriting) but continuously rewrite *Source* $\cup$ *Trans* by means of linear logic deductions.[3] Of course, the derived set *Target* should be the set of meaning constructors associated with the semantic projection of the target f-structure as obtained by independent analysis (parsing) of the target string. In the simple example discussed above this is indeed the case:

$$\begin{array}{lcll} Source \cup Trans & \vdash & Target & \vdash\ (f_1)_\sigma \rightsquigarrow swim(hans) \\ Target \cup Trans^{-1} & \vdash & Source & \vdash\ (f_1)_\sigma \rightsquigarrow swimmen(hans) \end{array}$$

[2] They specify combination possibilities in the linear logic deductions.

[3] However, we will use the locution "source" and "target" set whenever convenient.

where $Trans^{-1}$ is just like *Trans* except that $\multimap$ arrows relating source and target meaning constructors are reversed.

3.2 Generation

Target provides the input to generation. Generation proceeds as follows: for each of the $\tau_i \in$ *Target* generation retrieves exactly one matching lexical entry lex_i in the target grammar. Matching spreads instantiated f-structure nodes in τ_i to co-indexed f-structure nodes in the f-description parts φ_i in lexical entires lex_i. The conjunction $\bigwedge_{i=1}^{n} \varphi_i = \Phi$ over such φ_i yields an f-description Φ. Alternative matches may produce alternative Φ. Φ is then passed on to standard f-structure generation [9, 13] which produces the target string.

3.3 Ambiguity Preserving Transfer

Transfer on sets of source language meaning constructors rather than disambiguated meaning assignments obviates the need for multiple transfer on disambiguations if the ambiguity in question is preserved in both source and target. Examples are quantifier and operator scope, and adjuncts. Consider the following interactions between quantifier and adjunct scope in English and German: *Usually Hans suggests a restaurant* and *Normalerweise schlägt Hans ein Restaurant vor.* The corresponding f-structures are

$$\begin{bmatrix} \text{SUBJ} & \boxed{2}\left[\text{PRED} \quad \text{HANS}\right] \\ \text{PRED} & \text{SUGGEST}\langle \uparrow \text{SUBJ}, \uparrow \text{OBJ} \rangle \\ \text{OBJ} & \boxed{3}\begin{bmatrix} \text{PRED} & \text{RESTAURANT} \\ \text{SPEC} & \text{A} \end{bmatrix} \\ \text{ADJN} & \{\boxed{4}\left[\text{PRED} \quad \text{USUALLY}\right]\} \end{bmatrix}\boxed{1} \qquad \begin{bmatrix} \text{SUBJ} & \boxed{2}\left[\text{PRED} \quad \text{HANS}\right] \\ \text{PRED} & \text{VORSCHLAGEN}\langle \uparrow \text{SUBJ}, \uparrow \text{OBJ} \rangle \\ \text{OBJ} & \boxed{3}\begin{bmatrix} \text{PRED} & \text{RESTAURANT} \\ \text{SPEC} & \text{A} \end{bmatrix} \\ \text{ADJN} & \{\boxed{4}\left[\text{PRED} \quad \text{NORMALERWEISE}\right]\} \end{bmatrix}\boxed{1}$$

The meaning constructor for adjuncts (cf. [2]) involves inside-out functional uncertainty ($\text{ADJN} \ni \uparrow$) to access semantic material induced by the adjunct embedding f-structure:

$$usually: \quad \forall P\ [(\text{ADJN} \in \uparrow)_\sigma \rightsquigarrow P \multimap (\text{ADJN} \in \uparrow)_\sigma \rightsquigarrow usually(P)]$$

¿From the meaning constructors associated with the English example sentence

$$Source = \left\{ \begin{array}{ll} 1_e: & (f_2)_\sigma \rightsquigarrow hans \\ 2_e: & \forall X, Y\ [((f_2)_\sigma \rightsquigarrow X \otimes (f_3)_\sigma \rightsquigarrow Y) \multimap (f_1)_\sigma \rightsquigarrow suggest(X, Y)] \\ 3_e: & \forall X[(f_3)_\sigma \text{VAR} \rightsquigarrow X \multimap (f_3)_\sigma \text{RESTR} \rightsquigarrow restaurant(X)] \\ 4_e: & \forall R, S, Scope[(\forall X((f_3)_\sigma \text{VAR} \rightsquigarrow X \multimap (f_3)_\sigma \text{RESTR} \rightsquigarrow R(X)) \\ & \qquad \otimes \\ & \qquad \forall X((f_3)_\sigma \rightsquigarrow X \multimap Scope \rightsquigarrow S(X))) \\ & \qquad \multimap \rightsquigarrow exists(R, S)] \\ 5_e: & \forall P\ [(f_1)_\sigma \rightsquigarrow P \multimap (f_1)_\sigma \rightsquigarrow usually(P)] \end{array} \right\}$$

we deduce the two readings

$$\begin{array}{lll} \{1_e, 2_e, 3_e, 4_e, 5_e\} & \vdash & usually(exists(restaurant, \lambda x.suggest(hans, x))) \\ \{1_e, 2_e, 3_e, 4_e, 5_e\} & \vdash & exists(restaurant, \lambda x.usually(suggest(hans, x))) \end{array}$$

as required. The German target meaning constructors and deductions are completely analogous:

$$Target = \left\{ \begin{array}{ll} 1_g: & (f_2)_\sigma \rightsquigarrow hans \\ 2_g: & \forall X, Y\ [((f_2)_\sigma \rightsquigarrow X \otimes (f_3)_\sigma \rightsquigarrow Y) \multimap (f_1)_\sigma \rightsquigarrow vorschlagen(X,Y)] \\ 3_g: & \forall X[(f_3)_{\sigma\mathrm{VAR}} \rightsquigarrow X \multimap (f_3)_{\sigma\mathrm{RESTR}} \rightsquigarrow restaurant(X)] \\ 4_g: & \forall R, S, Scope[(\forall X((f_3)_{\sigma\mathrm{VAR}} \rightsquigarrow X \multimap (f_3)_{\sigma\mathrm{RESTR}} \rightsquigarrow R(X)) \\ & \qquad \otimes \\ & \qquad \forall X((f_3)_\sigma \rightsquigarrow X \multimap Scope \rightsquigarrow S(X))) \\ & \qquad \multimap Scope \rightsquigarrow exists(R,S)] \\ 5_g: & \forall P\ [(f_1)_\sigma \rightsquigarrow P \multimap (f_1)_\sigma \rightsquigarrow normalerweise(P)] \end{array} \right\}$$

It can be seen that the set *Trans* of transfer constructors

$$Trans = \left\{ \begin{array}{ll} 1_t: & \forall F\ [F_\sigma \rightsquigarrow hans \multimap F_\sigma \rightsquigarrow hans] \\ 2_t: & \forall F\ [\forall X, Y(((F\ \mathrm{SUBJ})_\sigma \rightsquigarrow X \otimes (F\ \mathrm{OBJ})_\sigma \rightsquigarrow Y) \multimap F_\sigma \rightsquigarrow suggest(X,Y)) \\ & \qquad \multimap \\ & \forall X, Y(((F\ \mathrm{SUBJ})_\sigma \rightsquigarrow X \otimes (F\ \mathrm{OBJ})_\sigma \rightsquigarrow Y) \multimap F_\sigma \rightsquigarrow vorschlagen(X,Y))] \\ 3_t: & \forall F\ [\forall X(F_{\sigma\mathrm{VAR}} \rightsquigarrow X \multimap F_{\sigma\mathrm{RESTR}} \rightsquigarrow restaurant(X)) \\ & \qquad \multimap \\ & \forall X(F_{\sigma\mathrm{VAR}} \rightsquigarrow X \multimap F_{\sigma\mathrm{RESTR}} \rightsquigarrow restaurant(X))] \\ 4_t: & \forall F(\forall R, S, Scope[(\forall X(F_{\sigma\mathrm{VAR}} \rightsquigarrow X \multimap F_{\sigma\mathrm{RESTR}} \rightsquigarrow R(X)) \\ & \qquad \otimes \forall X(F_\sigma \rightsquigarrow X \multimap Scope \rightsquigarrow S(X))) \\ & \qquad \multimap Scope \rightsquigarrow exists(R,S)] \\ & \qquad \multimap \\ & \forall R, S, Scope[(\forall X(F_{\sigma\mathrm{VAR}} \rightsquigarrow X \multimap F_{\sigma\mathrm{RESTR}} \rightsquigarrow R(X)) \\ & \qquad \otimes \forall X(F_\sigma \rightsquigarrow X \multimap Scope \rightsquigarrow S(X))) \\ & \qquad \multimap Scope \rightsquigarrow exists(R,S)]) \\ 5_t: & \forall F\ [\forall P\ (F_\sigma \rightsquigarrow P \multimap F_\sigma \rightsquigarrow usually(P)) \\ & \qquad \multimap \\ & \forall P\ (F_\sigma \rightsquigarrow P \multimap F_\sigma \rightsquigarrow normalerweise(P))] \end{array} \right\}$$

relates *Source* and *Target*: *Source* $\cup$ *Trans* $\vdash$ *Target*. Transfer on the set of source meaning constructors carries over the ambiguity intact and avoids multiple transfer on disambiguations.

Sections 3.1 and 3.3 have illustrated the basic idea. The reader may have noticed that as stated above, first, transfer constructors are massively redundant and second, there is nothing to guarantee that our deductions terminate in a set of target meaning constructors (rather then in disambiguated target meaning assignments). These issues are addressed below.

3.4 Transfer Architecture

The resource sensitivity of linear logic provides a natural setting to define transfer. In addition to providing a homogeneous framework for both the construction of and transfer on semantic representations in current LFG architectures we can exploit the resource sensitivity of linear logic to ensure that transfer exhaustively covers all source meaning contributions.

In order to be able to define transfer on underspecified representations we need to relate corresponding *sets* of source and target meaning constructors. If transfer is defined in terms of linear logic transfer constructors (and linear logic deductions) we need to ensure that transfer operating on a set of source meaning constructors and a set of transfer constructors terminates in a *set* of target meaning constructors. In other words, we need to prevent transfer deductions from delivering disambiguated target meaning assignments. As stated in the introductory sections above, there is nothing to prevent transfer deductions to terminate in completely disambiguated target meaning assignments. While this would not deliver incorrect results it would defeat the very idea of ambiguity preserving transfer on underspecified representations. Termination in sets of target meaning constructors can be ensured in a number of ways. Here we define a special *transfer* connective $\multimap_t$ and restrict application of $\multimap_t$ to linear implication elimination (linear modus ponens – here in a natural deduction style format [11]) as its only inference rule:

$$\frac{\Gamma \vdash_t \alpha \qquad \Delta \vdash_t \alpha \multimap_t \beta}{\Gamma\Delta \vdash_t \beta}$$

This together with the stipulations that (i) transfer deductions $\vdash_t$ may only be applications of $\multimap_t$ elimination (together with the required universal eliminations/introductions and α and β equivalences), (ii) transfer constructors have exactly one occurrence of $\multimap_t$ as its main connective and (iii) meaning constructors are not allowed to contain the $\multimap_t$ connective, ensures that transfer terminates in sets of target meaning constructors. In this scenario, transfer is a very simple and restricted operation.

There is a further issue to be dealt with here: so far we have simply assumed that in each of the transfer deductions we have at our disposal *exactly* the set of transfer constructors relevant to whatever is the current task at hand. This assumption was needed to insure that transfer deductions consume both the source meaning constructor set and the relevant transfer constructor set. Transfer is complete once the two "input" sets are consumed. This begs the question: how do we manage to obtain the set of transfer constructors relevant to the task at hand in the first place? For a realistic transfer scenario we will have to assume one large set *Trans* of transfer constructors associated with a given pair of source and target grammars. Furthermore, since, in all likelihood, transfer constructors may be used more than once we'll have to assume that all transfer constructores are prefixed by the "of course" operator !.[4] There are two possibilities to ensure termination in sets of target meaning constructors in this more global scenario. Given a local set of source meaning constructors *Source*, obtained from analysing a source sentence, and the global set of banged (!) transfer constructors *Trans*, associated with a pair of source and target grammars, we retrieve, at each stage of the transfer deductions, one source meaning constructor σ, perform $Trans \cup \{\sigma\} \vdash_t \tau$ and collect the result, a target meaning constructor τ in a new set *Target*. Transfer into sets of target meaning constructors is completed once *Source* is empty.

INPUT *Source, Trans*
OUTPUT *Target*

$Target := \emptyset$
REPEAT UNTIL $Source = \emptyset$
    BEGIN
    $Source := Source$ - σ
    $Trans \cup \{\sigma\} \vdash_t \tau$
    $Target := Target \cup \{\tau\}$
    END

[4] ! swiches off resource accounting.

An alternative is to simply tag transfer constructors to terminal strings (i.e. the words – and thereby the lexical entries – in the lexcion). In this way, relevant sets of transfer constructors tailored to the task at hand can be constructed automatically during analysis of the source input string. Given the constraints on $\multimap_t$ and $\vdash_t$ the derivation will terminate in a set *Target* consisting only of target meaning constructors.

3.5 A Shortcut

The transfer constructors we have presented so far are massively redundant. We identify two types of redundancy:

First, transfer constructors of the form $\sigma \multimap_t \sigma$ which relate identical source and target meaning constructors can be dropped. In fact, $\vdash \sigma \multimap \sigma$ is a simple tautology in linear logic.

The second type of redundancy occurs when the source and the target meaning constructors related by the transfer constructor have left-common prefixes π:

$$(\pi \multimap \sigma) \multimap_t (\pi \multimap \tau)$$

This transfer constructor matches a source meaning constructor $(\pi \multimap \sigma)$ to produce a target meaning constructor $(\pi \multimap \tau)$ under linear modus ponens. We can achieve exactly the same effect by reducing the transfer constructor to the more perspicuous

$$\sigma \multimap_t \tau$$

and combining this with the source meaning constructors $(\pi \multimap \sigma)$ under transitivity of linear implication (recall that $\multimap_t$ is $\multimap$ restricted to linear modus ponens and confined to single occurrences as main connective in transfer constructors):

$$\frac{\Gamma \vdash_t \pi \multimap \sigma \qquad \Delta \vdash_t \sigma \multimap_t \tau}{\Gamma\Delta \vdash_t \pi \multimap \tau}$$

In fact, $\vdash (\sigma \multimap \tau) \multimap ((\pi \multimap \sigma) \multimap (\pi \multimap \tau))$ is a tautology in linear logic.

Assuming that we restrict $\vdash_t$ to applications of transitivity of linear implication, we can now collapse the set of transfer constructors given in section 3.3 above relating the source and target meaning constructor sets corresponding to *Usually Hans suggests a restaurant* and *Normalerweise schlägt Hans ein Restaurant vor* to the much more readable and perspicuous:

$$Trans = \left\{ \begin{array}{c} \forall F, X, Y\ (F_\sigma \rightsquigarrow suggest(X,Y) \multimap_t F_\sigma \rightsquigarrow vorschlagen(X,Y)) \\ \forall F, P\ (F_\sigma \rightsquigarrow usually(P) \multimap_t F_\sigma \rightsquigarrow normalerweise(P)) \end{array} \right\}$$

As before, this set of transfer constructors relates underspecified representations (sets of source and target meaning constructors) thus obviating the need for multiple (and in this case spurious) transfer on disambiguated representations. Unless explicitly indicated, in the rest of the paper we will always give transfer constructors in the compact, non-redundant form. Note that the compact transfer constructors are still fully reversible: simply changing the direction of $\multimap_t$ in the set above provides ambiguity preserving transfer from German into English.

3.6 Argument Switching and other Standard Transfer Problems

Before concentrating on problems of structural mismatches with head switching, we briefly illustrate how this transfer approach deals with more unproblematic types of lexical transfer. One of the basic exercises in transfer is argument switching. A typical case is the translation of German *mißlingen* into French *râter*, as in *Das Photo ist Hans mißlungen – Hans a râté la photo* (*Hans messed up/ruined the photo*), with the two f-structures

$$\begin{bmatrix} \text{SUBJ} & \left[\text{PRED} \quad \text{PHOTO}\right] \boxed{2} \\ \text{PRED} & \text{MISSLINGEN}\langle\uparrow \text{OBJ2}, \uparrow \text{SUBJ}\rangle \\ \text{OBJ2} & \left[\text{PRED} \quad \text{HANS}\right] \boxed{3} \end{bmatrix} \boxed{1} \qquad \begin{bmatrix} \text{SUBJ} & \left[\text{PRED} \quad \text{HANS}\right] \boxed{3} \\ \text{PRED} & \text{RÂTER}\langle\uparrow \text{SUBJ}, \uparrow \text{OBJ}\rangle \\ \text{OBJ} & \left[\text{PRED} \quad \text{PHOTO}\right] \boxed{2} \end{bmatrix} \boxed{1}$$

In the more redundant formalization of section 3.3 the transfer constructor *mißlingen* $\multimap_t$ *râter* would have to rewrite the SUBJ of *mißlingen* into the OBJ of *râter*, and its secondary object OBJ2 into the SUBJ of *râter*. In our more sparse formalization the transfer constructor *mißlingen* $\multimap_t$ *râter* reduces to:

$$\forall F, X, Y \ (F_\sigma \rightsquigarrow misslingen(X, Y) \multimap_t F_\sigma \rightsquigarrow râter(X, Y))$$

¿From the set of instantiated meaning constructors and transfer constructors we derive the appropriate set of target meaning constructors:

$$\left\{ \begin{array}{c} (f_2)_\sigma \rightsquigarrow photo \\ (f_3)_\sigma \rightsquigarrow hans \\ \forall X \forall Y \ [((f_3)_\sigma \rightsquigarrow X \otimes (f_2)_\sigma \rightsquigarrow Y) \multimap (f_1)_\sigma \rightsquigarrow misslingen(X, Y)] \end{array} \right\}$$

$$\cup \left\{ \ misslingen \multimap_t râter \ \right\} \vdash_t \left\{ \begin{array}{ll} 1. & (f_2)_\sigma \rightsquigarrow photo \\ 2. & (f_3)_\sigma \rightsquigarrow hans \\ 3. & \forall X \forall Y \ [((f_3)_\sigma \rightsquigarrow X \otimes (f_2)_\sigma \rightsquigarrow Y) \multimap (f_1)_\sigma \rightsquigarrow râter(X, Y) \end{array} \right\}$$

The instantiated target meaning constructor (3.) correctly binds the meanings X and Y, which can be seen by comparison with the non-instantiated source and target versions, to be used in analysis and generation:

mißlingen: $\forall X \forall Y \ [((\uparrow \text{OBJ2})_\sigma \rightsquigarrow X \otimes (\uparrow \text{SUBJ})_\sigma \rightsquigarrow Y) \multimap \uparrow_\sigma \rightsquigarrow mißlingen(X, Y)]$
râter: $\forall X \forall Y \ [((\uparrow \text{SUBJ})_\sigma \rightsquigarrow X \otimes (\uparrow \text{OBJ})_\sigma \rightsquigarrow Y) \multimap \uparrow_\sigma \rightsquigarrow râter(X, Y)]$

In the compact formalization of transfer constructors the transfer of simple argument mismatches – where an argument is syntactically realized by distinct grammatical relations in source and target languages – goes in fact unnoticed. Since transfer constructors operate on instantiated meaning constructors, the argument bindings carried through in transfer are sufficient to allow for instantiation of the appropriate grammatical functions in the target structure for generation.

The resource sensitivity of linear logic also fares well with another type of lexical transfer, exemplified by the pair *commit suicide* and its French translation *se suicider*. Here the meaning of a transitive support verb construction must be rewritten into the meaning of an intransitive verb. This is captured by consuming the second argument of *commit* in the following transfer constructor:

$$\forall F, X, Y \ ((F \ \text{OBJ})_\sigma \rightsquigarrow suicide \otimes F_\sigma \rightsquigarrow commit(X, Y)) \multimap_t F_\sigma \rightsquigarrow se\_suicider(X))$$

Just as in the fully structure preserving examples of sections 3.1 and 3.3 the transfer constructors for lexical argument mismatches considered here are fully reversible.

4 Structural Mismatches in Transfer

4.1 Head Switching with Transfer Constructors

The English translation of *Hans schwimmt gerne* is *Hans likes swimming*. The sentences are associated with the following f-structures:

$$\begin{bmatrix} \text{SUBJ} & \left[\text{PRED} \quad \text{HANS}\right] \boxed{2} \\ \text{PRED} & \text{SCHWIMMEN}\langle \uparrow \text{SUBJ}\rangle \\ \text{ADJN} & \{\left[\text{PRED} \quad \text{GERNE}\right] \boxed{3}\} \end{bmatrix} \boxed{1} \qquad \begin{bmatrix} \text{SUBJ} & \left[\text{PRED} \quad \text{HANS}\right] \boxed{2} \\ \text{PRED} & \text{LIKE}\langle \uparrow \text{SUBJ}, \uparrow \text{XCOMP}\rangle \\ \text{XCOMP} & \begin{bmatrix} \text{SUBJ} & \left[\text{PRED} \quad \text{HANS}\right] \boxed{2} \\ \text{PRED} & \text{SWIM}\langle \uparrow \text{SUBJ}\rangle \end{bmatrix} \boxed{1} \end{bmatrix} \boxed{3}$$

Note that the German adverb *gerne* translates into a control verb construction *like* in English. The f-structures associated with the German and English sentences show structural misalignment (head switching): in the German case the adjunct f_3 is embedded inside the main f-structure f_1 with $\text{SCHWIMMEN}\langle \uparrow \text{SUBJ}\rangle$ as its governing PRED while in the English case the corresponding f-structure f_1 governed by $\text{SWIM}\langle \uparrow \text{SUBJ}\rangle$ is located inside the top f-structure f_3 governed by $\text{LIKE}\langle \uparrow \text{SUBJ}, \uparrow \text{XCOMP}\rangle$, the translation of the adjunct *gerne*. Embedded head switching cases are problematic for the LFG correspondence based transfer approach [7, 10]. Essentially the problem is that what used to be an embedding f-structure in the source (in our example f_1) turns out to be an embedded f-structure in the target representation. If, in addition, the embedding f-structure f_1 in the source was itself embedded inside another f-structure (say f_0) in the source then head switching in translation may destroy the link between its (i.e. f_1) translation and the translation of its embedding f-structure (f_0). From the f-structure associated with the German sentence we derive the set *Source*:

$$\left\{ \begin{array}{c} (f_2)_\sigma \rightsquigarrow hans \\ \forall X\ [(f_2)_\sigma \rightsquigarrow X \multimap (f_1)_\sigma \rightsquigarrow schwimmen(X)] \\ \forall P\ [(f_1)_\sigma \rightsquigarrow P \multimap (f_1)_\sigma \rightsquigarrow gerne(P)] \end{array} \right\} \vdash (f_1)_\sigma \rightsquigarrow gerne(schwimmen(hans))$$

The meaning constructor for *like*, to be used for independent analysis of the English sentence, is[5]

$$like: \quad \forall X, P\ [((\uparrow \text{SUBJ})_\sigma \rightsquigarrow X \otimes \forall Y\ ((\uparrow \text{XCOMP SUBJ})_\sigma \rightsquigarrow Y \multimap (\uparrow \text{XCOMP})_\sigma \rightsquigarrow P(Y))) \multimap \uparrow_\sigma \rightsquigarrow like(X, P(X))]$$

and analysis of the English sentence yields the set *Target*:

$$\left\{ \begin{array}{c} (f_2)_\sigma \rightsquigarrow hans \\ \forall X, P\ [((f_2)_\sigma \rightsquigarrow X \otimes \forall Y\ ((f_2)_\sigma \rightsquigarrow Y \multimap (f_1)_\sigma \rightsquigarrow P(Y))) \multimap (f_3)_\sigma \rightsquigarrow like(X, P(X))] \\ \forall X\ [(f_2)_\sigma \rightsquigarrow X \multimap (f_1)_\sigma \rightsquigarrow swim(X)] \end{array} \right\}$$

$$\vdash (f_3)_\sigma \rightsquigarrow like(hans, swim(hans))$$

Note that in our transfer constructors we refer to f-structures that project σ-structures (in fact we have wide scope universal quantification over such nodes). Hence if we are not careful we would expect to get problems with embedded head switching cases here as well. "Pure" semantic representations tend to avoid such problems since structural differences such as head switching would

[5] This constructor was suggested to us by Mary Dalrymple.

be expected to be ironed out on the level of semantic representation [8, 4]. Meaning constructors, however, relate syntactic and semantic representations. It is in this sense that they are not "pure".[6] On the other hand they naturally provide underspecified semantic representations and the possibility of ambiguity preserving transfer on such representations, an issue not addressed by earlier approaches such as [8].[7]

Since there are no left-common prefixes in the meaning constructors for *gerne* and *like*, the transfer constructor $gerne \multimap_t like$ consumes the entire meaning constructor for *gerne* and produces a full meaning constructor for *like*:

$$\begin{array}{c} \forall F\ [\forall P(F_\sigma \rightsquigarrow P \multimap F_\sigma \rightsquigarrow gerne(P)) \\ \multimap_t \\ \forall X, P\ ((F\ \textsc{subj})_\sigma \rightsquigarrow X \otimes \forall Y\ ((F\ \textsc{xcomp subj})_\sigma \rightsquigarrow Y \multimap F_\sigma \rightsquigarrow P(Y)) \multimap F_\sigma \rightsquigarrow like(X, P(X)))] \end{array}$$

Note crucially that the meaning constructor on the right-hand side of $\multimap_t$ rewrites a node F rather than accessing a node F XCOMP to match $P(Y)$. From the instantiated source meaning constructors together with the transfer constructors (and since $\uparrow$ SUBJ $=\uparrow$ XCOMP SUBJ) we deduce

$$Source \cup \left\{ \begin{array}{c} schwimmen \multimap_t swim \\ gerne \multimap_t like \end{array} \right\} \vdash_t$$

$$\left\{ \begin{array}{ll} 1. & (f_2)_\sigma \rightsquigarrow hans \\ 2. & \forall X\ [(f_2)_\sigma \rightsquigarrow X \multimap (f_1)_\sigma \rightsquigarrow swim(X)] \\ 3. & \forall X, P\ [((f_2)_\sigma \rightsquigarrow X \otimes \forall Y\ ((f_2)_\sigma \rightsquigarrow Y \multimap (f_1)_\sigma \rightsquigarrow P(Y))) \multimap (f_1)_\sigma \rightsquigarrow like(X, P(X))] \end{array} \right\} \vdash$$

$$(f_1)_\sigma \rightsquigarrow like(hans, swim(hans))$$

as required. Observe, however, that (3.) above is not exactly like the original meaning constructor for *like* employed in independent analysis of the target string *Hans likes swimming* (cf. p. 8 above). The difference is that the constructor obtained through the transfer operation "rewrites" a single node $(f_1)_\sigma$ rather than accessing a complement node $(f_1\ \textsc{xcomp})_\sigma$ to match against $P(Y)$.

4.2 Embedded Head Switching: Take 1

Next, we consider a complex instance of embedded head switching:

$$\begin{array}{rcl} \textit{Ede vermutet daß Hans gerne schwimmt} & \rightsquigarrow & vermuten(ede, gerne(schwimmen(hans))) \\ \textit{Ede suspects that Hans likes swimming} & \rightsquigarrow & suspect(ede, like(hans, swim(hans))) \end{array}$$

The corresponding f-structures are:

$$\begin{bmatrix} \textsc{subj} & \left[\textsc{pred}\ \textsc{ede}\right] 2 \\ \textsc{pred} & \textsc{vermuten}\langle\uparrow \textsc{subj}, \uparrow \textsc{comp}\rangle \\ \textsc{comp} & \begin{bmatrix} \textsc{subj} & \left[\textsc{pred}\ \textsc{hans}\right] 4 \\ \textsc{pred} & \textsc{schwimmen}\langle\uparrow \textsc{subj}\rangle \\ \textsc{adjn} & \{\left[\textsc{pred}\ \textsc{gerne}\right] 5\} \end{bmatrix} 3 \end{bmatrix} 1$$

$$\begin{bmatrix} \textsc{subj} & \left[\textsc{pred}\ \textsc{ede}\right] 2 \\ \textsc{pred} & \textsc{suspect}\langle\uparrow \textsc{subj}, \uparrow \textsc{comp}\rangle \\ \textsc{comp} & \begin{bmatrix} \textsc{subj} & \left[\textsc{pred}\ \textsc{hans}\right] 4 \\ \textsc{pred} & \textsc{like}\langle\uparrow \textsc{subj}, \uparrow \textsc{xcomp}\rangle \\ \textsc{xcomp} & \begin{bmatrix} \textsc{subj} & \left[\textsc{pred}\ \textsc{hans}\right] 4 \\ \textsc{pred} & \textsc{swim}\langle\uparrow \textsc{subj}\rangle \end{bmatrix} 5 \end{bmatrix} 3 \end{bmatrix} 1$$

[6] Expressions in the meaning representation language in meaning constructors, though are.

[7] The approach in [8] could, however, be combined with packed representation techniques to enable ambiguity preserving transfer.

The following meaning constructors are associated with *vermuten* and *suspect*:

$$\begin{array}{ll} \mathit{vermuten}: & \forall X, P\ [((\uparrow \textsc{subj})_\sigma \rightsquigarrow X \otimes (\uparrow \textsc{comp})_\sigma \rightsquigarrow P) \multimap \uparrow_\sigma \rightsquigarrow \mathit{vermuten}(X, P)] \\ \mathit{suspect}: & \forall X, P\ [((\uparrow \textsc{subj})_\sigma \rightsquigarrow X \otimes (\uparrow \textsc{comp})_\sigma \rightsquigarrow P) \multimap \uparrow_\sigma \rightsquigarrow \mathit{suspect}(X, P)] \end{array}$$

The corresponding transfer constructor is simply:

$$\forall F, X, P\ [F_\sigma \rightsquigarrow \mathit{vermuten}(X, P) \multimap_t F_\sigma \rightsquigarrow \mathit{suspect}(X, P)]$$

The semantic projection of the f-structure associated with the German sentence induces the following set of instantiated meaning constructors:

$$\mathit{Source} = \left\{ \begin{array}{ll} 1. & (f_2)_\sigma \rightsquigarrow \mathit{ede} \\ 2. & \forall X, P[((f_2)_\sigma \rightsquigarrow X \otimes (f_3)_\sigma \rightsquigarrow P) \multimap (f_1)_\sigma \rightsquigarrow \mathit{vermuten}(X, P)] \\ 3. & (f_4)_\sigma \rightsquigarrow \mathit{hans} \\ 4. & \forall X[(f_4)_\sigma \rightsquigarrow X \multimap (f_3)_\sigma \rightsquigarrow \mathit{schwimmen}(X)] \\ 5. & \forall P[(f_3)_\sigma \rightsquigarrow P \multimap (f_3)_\sigma \rightsquigarrow \mathit{gerne}(P)] \end{array} \right\}$$

$$\mathit{Source} \vdash (f_1)_\sigma \rightsquigarrow \mathit{vermuten}(\mathit{ede}, \mathit{gerne}(\mathit{schwimmen}(\mathit{hans})))$$

Furthermore, transfer yields

$$\mathit{Source} \cup \left\{ \begin{array}{c} \mathit{vermuten} \multimap_t \mathit{suspect} \\ \mathit{schwimmen} \multimap_t \mathit{swim} \\ \mathit{gerne} \multimap_t \mathit{like} \end{array} \right\} \vdash_t$$

$$\left\{ \begin{array}{ll} 1. & (f_2)_\sigma \rightsquigarrow \mathit{ede} \\ 2. & \forall X, P[((f_2)_\sigma \rightsquigarrow X \otimes (f_3)_\sigma \rightsquigarrow P) \multimap (f_1)_\sigma \rightsquigarrow \mathit{suspect}(X, P)] \\ 3. & (f_4)_\sigma \rightsquigarrow \mathit{hans} \\ 4. & \forall X[(f_4)_\sigma \rightsquigarrow X \multimap (f_3)_\sigma \rightsquigarrow \mathit{swim}(X)] \\ 5. & \forall X, P\ [((f_4)_\sigma \rightsquigarrow X \otimes \forall Y\ ((f_4)_\sigma \rightsquigarrow Y \multimap (f_3)_\sigma \rightsquigarrow P(Y))) \multimap (f_3)_\sigma \rightsquigarrow \mathit{like}(X, P(X))] \end{array} \right\} \vdash$$

$$(f_1)_\sigma \rightsquigarrow \mathit{suspect}(\mathit{ede}, \mathit{like}(\mathit{hans}, \mathit{swim}(\mathit{hans})))$$

as required. This set of target meaning constructors obtained via transfer is close to the set of meaning constructors obtained via independent analysis of the taget string. They differ, however, with respect to the constructor for *like*. The constructor obtained through transfer (5. above) rewrites node $(f_3)_\sigma$ rather than accessing a separate $(f_3\ \textsc{xcomp})_\sigma$ projection to match $P(Y)$.

4.3 Embedded Head Switching: Take 2

Let us briefly take stock: the linear logic based transfer approach provides ambiguity preserving transfer on underspecified representations and an elegant solution for embedded head switching cases. As it stands, however, for head switching cases it will not deliver exactly the set of target constructors obtained by independent analysis of the target string. This may cause problems for

target language generation from underspecified representations (sets of target meaning constructors). Recall that the linear logic based transfer solution to embedded head switching relies on node rewriting in a resource sensitive formalism. In order to enable generation as outlined in section 3.2, transfer should deliver exactly the set of meaning constructors as would be obtained by independent analysis of the target string. Alternatively, generation would have to proceed from fully disambiguated target meaning assignments. At the moment, generation from disambiguated target meaning assignments is not available to us yet and even if it were, it would defeat the idea of ambiguity preserving transfer.

However, it is possible to adapt a variant of a solution to embedded head switching in correspondence-based syntactic transfer approaches [10] to our present ambiguity preserving transfer scenario.

The basic idea is that in order to define transfer from a flat f-structure into an f-structure configuration involving an additional XCOMP layer, the transfer rule that maps the f-structure node f_3 under the COMP arc of *vermuten* in the German f-structure (cf. p. 9) into the target structure has to foresee that – in case of head switching – the arcs of this node have to map to a lower position in the target, namely the node under the path COMP XCOMP. One way to "foresee" such a possible deeper embedding is to "push down" the predicate-argument nucleus of verbs (here *schwimmen*) one or more levels down, via functional uncertainty over XCOMP. This is defined below, where we redefine the transfer constructor $schwimmen \multimap_t swim$:

$$\begin{array}{c} schwimmen \multimap_t swim: \\ \forall F, X\ (F_\sigma \rightsquigarrow schwimmen(X) \multimap_t (F\ \textsc{xcomp}^*)_\sigma \rightsquigarrow swim(X)) \end{array}$$

We also redefine the transfer constructor $gerne \multimap_t like$: we now rely on the fact that the predicate argument structure and the corresponding semantic projector associated with the translation of the proposition in the scope of the adjunct in the source will be "pushed down", in the target representation in terms of functional uncertainty (see $schwimmen \multimap_t swim$ above). The new constructor $gerne \multimap_t like$ is:

$$\begin{array}{c} gerne \multimap_t like: \\ \forall F\ [\forall P(F_\sigma \rightsquigarrow P \multimap F_\sigma \rightsquigarrow gerne(P)) \\ \multimap_t \\ \forall X, P\ [((F\ \textsc{subj})_\sigma \rightsquigarrow X \otimes \forall Y\ ((F\ \textsc{xcomp subj})_\sigma \rightsquigarrow Y \multimap (F\ \textsc{xcomp})_\sigma \rightsquigarrow P(Y))) \\ \multimap F_\sigma \rightsquigarrow like(X, P(X))] \end{array}$$

This transfer constructor consumes the entire source meaning constructor for *gerne* and produces the proper target meaning constructor for *likes* (as used in independent target analysis). From the set of source meaning constructors *Source* together with the set of transfer constructors *Trans* – this time with the updated $schwimmen \multimap_t swim$ and $gerne \multimap_t like$ – we derive

$$Source \cup \left\{ \begin{array}{c} vermuten \multimap_t suspect \\ schwimmen \multimap_t swim \\ gerne \multimap_t like \end{array} \right\} \vdash_t$$

$$\left\{\begin{array}{ll} 1. & (f_2)_\sigma \rightsquigarrow ede \\ 2. & \forall X, P\ [((f_2)_\sigma \rightsquigarrow X \otimes (f_3)_\sigma \rightsquigarrow P) \multimap (f_1)_\sigma \rightsquigarrow suspect(X, P)] \\ 3. & (f_4)_\sigma \rightsquigarrow hans \\ 4. & \forall X[(f_4)_\sigma \rightsquigarrow X \multimap (f_3\ \textsc{xcomp}*)_\sigma \rightsquigarrow swim(X)] \\ 5. & \forall X, P\ [((f_4)_\sigma \rightsquigarrow X \otimes \forall Y\ ((f_4)_\sigma \rightsquigarrow Y \multimap (f_3\ \textsc{xcomp})_\sigma \rightsquigarrow P(Y))) \\ & \qquad\qquad \multimap (f_3)_\sigma \rightsquigarrow like(X, P(X))] \end{array}\right\} \vdash$$

$$(f_1)_\sigma \rightsquigarrow suspect(ede, like(hans, swim(hans)))$$

where XCOMP* is resolved to XCOMP. Now the set of target meaning constructors derived via transfer does indeed correspond to the set of target meaning constructors obtained by independent analysis of the target string. In particular, the f-structure node instantiations of the target meaning constructor ensure that (i) the subject of *like* is the same as the subject of *swim* (sharing is also required by the control equation (↑ SUBJ) = (↑ XCOMP SUBJ) in the lexical entry for *likes*) and (ii) the meaning assignment produced by *swim* is available and retrieved at the XCOMP node required by *like*.

It may be presumed that the resulting set of target meaning constructors may not always correspond exactly to the set obtained from independent analysis of the target string: if the embedded target predicate requires complements other than the shared SUBJ would it not be the case that those complements be located and found at the "wrong" level of the target f-structure? Fortunately, this is not the case. Consider *Ede vermutet daß Hans gerne Chomsky liest* and its translation *Ede suspects that Hans likes reading Chomsky*. The corresponding f-structures are

$$\left[\begin{array}{ll} \text{SUBJ} & \left[\text{PRED}\ \textsc{ede}\right]\boxed{2} \\ \text{PRED} & \textsc{vermuten}\langle\uparrow \text{SUBJ}, \uparrow \text{COMP}\rangle \\ \text{COMP} & \left[\begin{array}{ll} \text{SUBJ} & \left[\text{PRED}\ \textsc{hans}\right]\boxed{4} \\ \text{PRED} & \textsc{lesen}\langle\uparrow \text{SUBJ}, \uparrow \text{OBJ}\rangle \\ \text{OBJ} & \left[\text{PRED}\ \textsc{chomsky}\right]\boxed{6} \\ \text{ADJN} & \{\left[\text{PRED}\ \textsc{gerne}\right]\boxed{5}\} \end{array}\right]\boxed{3} \end{array}\right]\boxed{1}$$

$$\left[\begin{array}{ll} \text{SUBJ} & \left[\text{PRED}\ \textsc{ede}\right]\boxed{2} \\ \text{PRED} & \textsc{suspect}\langle\uparrow \text{SUBJ}, \uparrow \text{COMP}\rangle \\ \text{COMP} & \left[\begin{array}{ll} \text{SUBJ} & \left[\text{PRED}\ \textsc{hans}\right]\boxed{4} \\ \text{PRED} & \textsc{like}\langle\uparrow \text{SUBJ}, \uparrow \text{XCOMP}\rangle \\ \text{XCOMP} & \left[\begin{array}{ll} \text{SUBJ} & \left[\text{PRED}\ \textsc{hans}\right]\boxed{4} \\ \text{PRED} & \textsc{read}\langle\uparrow \text{SUBJ}, \uparrow \text{OBJ}\rangle \\ \text{OBJ} & \left[\text{PRED}\ \textsc{chomsky}\right]\boxed{6} \end{array}\right]\boxed{5} \end{array}\right]\boxed{3} \end{array}\right]\boxed{1}$$

The source and target meaning constructors for simple transitive verbs are the standard

$$\begin{array}{ll} \textit{lesen}: & \forall X, Y\ [((\uparrow \text{SUBJ})_\sigma \rightsquigarrow X \otimes (\uparrow \text{OBJ})_\sigma \rightsquigarrow Y) \multimap \uparrow_\sigma \rightsquigarrow lesen(X, Y)] \\ \textit{read}: & \forall X, Y\ [((\uparrow \text{SUBJ})_\sigma \rightsquigarrow X \otimes (\uparrow \text{OBJ})_\sigma \rightsquigarrow Y) \multimap \uparrow_\sigma \rightsquigarrow read(X, Y)] \end{array}$$

With these the set of instantiated source meaning constructors is

$$Source = \left\{\begin{array}{ll} 1. & (f_2)_\sigma \rightsquigarrow ede \\ 2. & \forall X, P[((f_2)_\sigma \rightsquigarrow X \otimes (f_3)_\sigma \rightsquigarrow P) \multimap (f_1)_\sigma \rightsquigarrow vermuten(X, P)] \\ 3. & (f_4)_\sigma \rightsquigarrow hans \\ 4. & \forall X, Y\ [((f_4)_\sigma \rightsquigarrow X \otimes (f_6)_\sigma \rightsquigarrow Y) \multimap (f_3)_\sigma \rightsquigarrow lesen(X, Y)] \\ 5. & \forall P[(f_3)_\sigma \rightsquigarrow P \multimap (f_3)_\sigma \rightsquigarrow gerne(P)] \\ 6. & (f_6)_\sigma \rightsquigarrow chomsky \end{array}\right\}$$

$$Source \vdash (f_1)_\sigma \rightsquigarrow vermuten(ede, gerne(lesen(hans, chomsky)))$$

The transfer constructor $lesen\ \multimap_t\ read$ is

$$\forall F, X, Y\ [F_\sigma \rightsquigarrow lesen(X,Y) \multimap_t (F\ \textsc{xcomp}^*)_\sigma \rightsquigarrow read(X,Y)]$$

We derive

$$Source \cup \left\{ \begin{array}{c} vermuten \multimap_t suspect \\ lesen \multimap_t read \\ gerne \multimap_t like \end{array} \right\} \vdash_t$$

$$\left\{ \begin{array}{ll} 1. & (f_2)_\sigma \rightsquigarrow ede \\ 2. & \forall X, P[((f_2)_\sigma \rightsquigarrow X \otimes (f_3)_\sigma \rightsquigarrow P) \multimap (f_1)_\sigma \rightsquigarrow suspect(X,P)] \\ 3. & (f_4)_\sigma \rightsquigarrow hans \\ 4. & \forall X, Y\ [((f_4)_\sigma \rightsquigarrow X \otimes (f_6)_\sigma \rightsquigarrow Y) \multimap (f_3\ \textsc{xcomp}*)_\sigma \rightsquigarrow read(X,Y)] \\ 5. & \forall X, P\ [((f_4)_\sigma \rightsquigarrow X \otimes \forall Y\ ((f_4)_\sigma \rightsquigarrow Y \multimap (f_3\ \textsc{xcomp})_\sigma \rightsquigarrow P(Y))) \\ & \qquad \multimap (f_3)_\sigma \rightsquigarrow like(X, P(X))] \\ 6. & (f_6)_\sigma \rightsquigarrow chomsky \end{array} \right\} \vdash$$

$$(f_1)_\sigma \rightsquigarrow suspect(ede, like(hans, read(hans, chomsky)))$$

The set of target constructors obtained via transfer again corresponds to the set obtained by independent analysis of the target string.

Take 2 is made reversible as follows: first, swap the left- and right-hand sides of $\multimap_t$

$$\begin{array}{c} like \multimap_t gerne: \\ \forall F\ [\forall X, P\ ((F\ \textsc{subj})_\sigma \rightsquigarrow X \otimes \forall Y\ ((F\ \textsc{xcomp subj})_\sigma \rightsquigarrow Y \multimap (F\ \textsc{xcomp})_\sigma \rightsquigarrow P(Y)) \\ \multimap F_\sigma \rightsquigarrow like(X, P(X))) \\ \multimap_t \\ \forall P(F_\sigma \rightsquigarrow P \multimap F_\sigma \rightsquigarrow gerne(P))] \end{array}$$

second, redefine transfer constructors for main verbs, such that they admit functional uncertainty over XCOMP to both sides of $\multimap_t$:

$$\forall F, X, Y\ [(F\ \textsc{xcomp}^*)_\sigma \rightsquigarrow read(X,Y) \multimap_t (F\ \textsc{xcomp}^*)_\sigma \rightsquigarrow lesen(X,Y)]$$

Taking the second f-structure of p.12 as input for transfer, we obtain the set of meaning constructors:

$$Source = \left\{ \begin{array}{ll} 1. & (f_2)_\sigma \rightsquigarrow ede \\ 2. & \forall X, P\ [((f_2)_\sigma \rightsquigarrow X \otimes (f_3)_\sigma \rightsquigarrow P) \multimap (f_1)_\sigma \rightsquigarrow suspect(X,P)] \\ 3. & (f_4)_\sigma \rightsquigarrow hans \\ 4. & \forall X, Y\ [((f_4)_\sigma \rightsquigarrow X \otimes (f_6)_\sigma \rightsquigarrow Y) \multimap (f_5)_\sigma \rightsquigarrow read(X,Y)] \\ 5. & \forall X, P\ [((f_4)_\sigma \rightsquigarrow X \otimes \forall Y\ ((f_4)_\sigma \rightsquigarrow Y \multimap (f_3\ \textsc{xcomp})_\sigma \rightsquigarrow P(Y))) \\ & \qquad \multimap (f_3)_\sigma \rightsquigarrow like(X, P(X))] \\ 6. & (f_6)_\sigma \rightsquigarrow chomsky \end{array} \right\}$$

When applied to this set, the inverted transfer constructors derive the set of target meaning constructors required to generate the German f-structure. In this case the functional uncertainty over XCOMP will be resolved to ϵ to satisfy completeness and coherence conditions.

$$\left\{\begin{array}{ll} 1. & (f_2)_\sigma \rightsquigarrow ede \\ 2. & \forall X, P[((f_2)_\sigma \rightsquigarrow X \otimes (f_3)_\sigma \rightsquigarrow P) \multimap (f_1)_\sigma \rightsquigarrow vermuten(X, P)] \\ 3. & (f_4)_\sigma \rightsquigarrow hans \\ 4. & \forall X, Y\ [((f_4)_\sigma \rightsquigarrow X \otimes (f_6)_\sigma \rightsquigarrow Y) \multimap (f_3\ \text{XCOMP}*)_\sigma \rightsquigarrow lesen(X, Y)] \\ 5. & \forall P[(f_3)_\sigma \rightsquigarrow P \multimap (f_3)_\sigma \rightsquigarrow gerne(P)] \\ 6. & (f_6)_\sigma \rightsquigarrow chomsky \end{array}\right\}$$

5 Conclusion

In the present paper we have outlined an approach to semantic transfer based on linear logic transfer constructors operating on sets of linear logic meaning constructors. Transfer on sets of meaning constructors offers the advantage of ambiguity preserving transfer and in many cases obviates the need for multiple transfer on disambiguations. Linear logic transfer constructors blend well with the linear logic based semantic components of current LFG architectures [1]. Furthermore, the resource consciousness of transfer constructors allows a close modelling and provides a formalization of semantic transfer as developed and implemented in [3], a fact first noticed in [5].

Transfer as outlined in the present paper is reversible, modular and lexicalized. Lexical entries have corresponding and independently stated transfer constructors retrieved during transfer. A set of source meaning constructors and their corresponding transfer constructors together define transfer.

Transfer as outlined in the present paper comes with a well-formedness check on the resulting *Target* set representation. *Target* is well-formed iff it admits of target meaning assignments established via linear logic deductions consuming all target meaning constructors: $Target \vdash f_\sigma \rightsquigarrow M$.

Meaning constructors crucially refer to semantic projections induced by f-structure nodes to constrain composition possibilites during linear logic meaning assignment derivations. Ideally, transfer on sets of meaning constructors instantiated to f-structure nodes produces a target set of meaning constructors with f-structure node instantiations directly useful for target string generation. Since transfer constructors operate on instantiated meaning constructors, differences in syntactic function do not have to be transferred explicitly in many cases. Here, instantiation of the appropriate syntactic functions can be left to generation as long as the target meaning constructors match the meaning constructors used in analysis.

For instances of structural mismatches, in particular the notoriously difficult case of head switching, we have presented a way (Take 2) to define transfer constructors which produce a set of target meaning constructors with f-structure designators structurally corresponding to those obtained by independent analysis of the target sentence.

Meaning constructors effectively encode a syntax–semantics interface. Transfer as outlined in the present paper transfers syntax–semantics interfaces. Further research is directed at ambiguity preserving linear logic based transfer on "pure" semantic representations.

References

[1] M. Dalrymple, J. Lamping, F.C.N. Pereira, and V. Saraswat. A deductive account of quantification in lfg. In M. Kanazawa, C. Pinon, and H. de Swart, editors, *Quantifiers, Deduction and Context*, pages 33–57. CSLI Publications, No. 57, 1996.

[2] M. Dalrymple, J. Lamping, and V. Saraswat. Lfg semantics via constraints. In *Proceedings of the 6th Meeting of the European ACL (EACL)*, Utrecht, 1993.

[3] M. Dorna and M.C. Emele. Semantic-based Transfer. In *Proceedings of the 16th International Conference on Computational Linguistics (Coling'96)*, Copenhagen, Denmark, 1996.

[4] M. Dorna, A. Frank, J. van Genabith and M.C. Emele. Syntactic and Semantic Transfer with F-Structures. In *Proceedings of the 17th International Conference on Computational Linguistics (Coling'98)*, Montréal, Canada, 1998.

[5] T. Fujinami. Linear Logic for Translation with Ambiguity. ms., IMS, Stuttgart, Germany, 1996.

[6] T. Fujinami. A Decidable Logic for Transforming DRSs in Context. In *Proceedings of the 11th Amsterdam Colloquium*, pages 127–132, Amsterdam, Holland, 1997. ILLC, Department of Philosphy, University of Amsterdam.

[7] R. Kaplan, K. Netter, J. Wedekind, and A. Zaenen. Translation by Structural Correspondences. In *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics (EACL'89)*, pages 272–281, Manchester, UK, 1989.

[8] R. Kaplan and J. Wedekind. Restriction and Correspondence-based Translation. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics (EACL'93)*, pages 193–202, Utrecht, The Netherlands, 1993.

[9] D. Kohl. Generation from Under- and Overspecified Structures. In *Proceedings of the 14th International Conference on Computational Linguistics (Coling'92)*, pages 686–692, Nantes, France, 1992.

[10] L. Sadler and H.S. Thompson. Structural Non-correspondence in Translation. In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics (EACL'91)*, pages 293–298, Berlin, Germany, 1991.

[11] A.S. Troelstra. Lectures on Linear Logic. CSLI Lecture Notes, number 29, CSLI Publications, Stanford University, 1992.

[12] J. van Genabith and R. Crouch. How to Glue a Donkey to an f-Structure or Porting a Dynamic Meaning Representation Language into LFG's Linear Logic Based Glue Language Semantics. In *International Workshop for Computational Semantics, Tilburg, Proceedings*, pages 52–65, 1997.

[13] J. Wedekind. Generation as Structure Driven Derivation. In *Proceedings of the 12th International Conference on Computational Linguistics (Coling'88)*, pages 732–737, Budapest, Hungary, 1988.