Glue, Underspecification and Translation *

Dick Crouch, Anette Frank, Josef van Genabith
Xerox PARC, Xerox Research Centre Europe, Dublin City University

November 10, 2000

Abstract. This paper sketches how one can construct Underspecified Discourse Representation Structures (UDRSs (Reyle, 1993)) via glue semantics (Dalrymple *et al.*, 1999b). In most cases, UDRSs are constructed in linear time, analogously to the linear time construction of skeleton-modifier representations presented in (Gupta and Lamping, 1998). We show how this encoding can be used in ambiguity preserving, transfer-based machine translation, where it reduces problems with structural misalignment, such as head-switching problem.

Keywords: linear logic, glue languages, underspecification, machine translation

1. Introduction

The first part of this paper sketches how to construct Underspecified Discourse Representation Structures (UDRSs (Reyle, 1993)) via glue semantics (Dalrymple *et al.*, 1999b). The UDRSs produced bear a close similarity to the 'skeleton-modifier' glue representations described by (Gupta and Lamping, 1998), and like them can typically be constructed in linear time.

The second part of the paper considers structural misalignment problems in transfer-based approaches to machine translation, in particular head switching, and continues a line of investigation in (van Genabith *et al.*, 1998). UDRSs enjoy a very flat structure, minimizing the possibility of structural misalignments between source and target UDRSs. A proposal for ambiguity preserving transfer is discussed which involves transfer from source to target UDRSs, which would need to be followed by generation of glue-UDRS meaning constructors from the target UDRS, and then generation of word strings from the meaning constructors.

* Special thanks to Mary Dalrymple, Mark Hepple, Michael Ryan, Andy Way, the participants of the Linear Logic for NL II workshop, Birmingham, November 1998, and our anonymous referees for feedback and discussion.

2. Glue for UDRS

Glue semantics (Dalrymple *et al.*, 1999b) 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 (parsed) 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. As a glue logic, we will employ a slight extension of the implication-only fragment of propositional linear logic. This glue logic can be applied to assemble meanings expressed in higher-order intensional logic (Dalrymple *et al.*, 1997), as well as compositional Discourse Representation Theory (van Genabith and Crouch, 1999),(Muskens, 1996). In this paper, we show how propositional implicational linear logic can be used as a glue to assemble Underspecified Discourse Representation Structures.

2.1. An LFG Fragment

Although glue semantics is not necessarily restricted to Lexical Functional Grammar, we will use the following LFG fragment as the syntactic basis for all our examples. The syntax rules associate a CF-PSG grammar with f-structure representations in the standard fashion, where brackets indicate optionality, $\uparrow$ represents the functional attributes of the left hand side (mother) of the rule, and $\downarrow$ represents the functional attributes of the daughter (Kaplan and Bresnan, 1982):

$$\begin{array}{lcccccc} \text{S} & \rightarrow & \begin{array}{c}\text{NP}\\ \uparrow \text{SUBJ} = \downarrow\end{array} & \left(\begin{array}{c}\text{ADV}\\ \downarrow \in \uparrow \text{ADJN}\end{array}\right) & \begin{array}{c}\text{VP}\\ \uparrow = \downarrow\end{array} & \left(\begin{array}{c}\text{ADV}\\ \downarrow \in \uparrow \text{ADJN}\end{array}\right) \\ \text{NP} & \rightarrow & \begin{array}{c}\text{Det}\\ \uparrow = \downarrow\end{array} & \begin{array}{c}\text{N}\\ \uparrow = \downarrow\end{array} & & & \\ \text{VP} & \rightarrow & \begin{array}{c}\text{V}\\ \uparrow = \downarrow\end{array} & \left(\begin{array}{c}\text{NP}\\ \uparrow \text{OBJ} = \downarrow\end{array}\right) & \left(\begin{array}{c}\text{VP}\\ \uparrow \text{XCOMP} = \downarrow\end{array}\right) & \left(\begin{array}{c}\text{S}\\ \uparrow \text{COMP} = \downarrow\end{array}\right) & \end{array}$$

The functional annotations on the S $\rightarrow$ NP VP rule say that S and VP have the same attributes in the f-structure, and that their SUBJ attributes are those of the NP. The attributes of each (optional) adverb form members of the set-valued ADJN attribute.

Lexical entries include the following, and are shown with additional lexical meaning constructors, whose significance will be explained shortly.

cooks	V	$\uparrow$ PRED = cook$\langle\uparrow$ SUBJ$\rangle$ *cook* : $(\uparrow \text{SUBJ})_\sigma \multimap \uparrow_\sigma$
John	NP	$\uparrow$ PRED = John *john* : $\uparrow_\sigma$
certainly	ADV	$\uparrow$ PRED = certainly *certain* : $(\text{ADJN} \in \uparrow)_\sigma \multimap (\text{ADJN} \in \uparrow)_\sigma$

This fragment assigns the following functional structure to the sentence "*John certainly cooks*". Lexical meaning constructors, with their $\uparrow$ meta-variables instantiated by the parse to nodes in the f-structure, are shown alongside.

$$f:\begin{bmatrix}\text{PRED} & \text{cook}\ \langle\uparrow \text{SUBJ}\rangle' \\ \text{SUBJ} & g:\left[\text{PRED}\ \ \text{John}\right] \\ \text{ADJN} & \{h:\left[\text{PRED}\ \ \text{certainly}\right]\}\end{bmatrix} \qquad \begin{array}{rcl} \textit{john} & : & g_\sigma \\ \textit{cook} & : & g_\sigma \multimap f_\sigma \\ \textit{certain} & : & f_\sigma \multimap f_\sigma \end{array}$$

2.2. Review of Glue Semantics

Parsing with the LFG fragment above produces an f-structure and a set of instantiated meaning constructors. We now explain the significance of these constructors, and what needs to be done with them in order to build meanings.

Each constructor comprises a meaning language expression on the left of a colon, and a linear logic glue formula on the right.[1] The atomic propositions in the linear logic formulas (g_σ and f_σ) denote semantic resources, and correspond to the semantic (σ-) projections of f-structure nodes (g and f).

The instantiated constructor for the word "*John*", $john : g_\sigma$, pairs the meaning constant *john* with the resource g_σ. More succinctly, the meaning of node g_σ is *john*. The constructor for the word "*cooks*" pairs the 1-place predicate *cook* with the implication $g_\sigma \multimap f_\sigma$. The implication says that the meaning of g_σ must be consumed in order to produce the meaning of f_σ. The constructor for the adverb "*certainly*" has a glue formula $f_\sigma \multimap f_\sigma$, which consumes the meaning of f_σ in order to produce an updated meaning for f_σ. In what follows, we will often omit the σ subscript on semantic resources to reduce clutter.

[1] We are using the 'Curry-Howard' glue notation of (Dalrymple *et al.*, 1999a), rather than the older notation of (Dalrymple *et al.*, 1997) with its uninterpreted meaning assignment predicate $\rightsquigarrow$. The 'Curry-Howard' notation 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.

To build a meaning with these lexical constructors, we use the glue formulas as premises to a linear logic derivation, whose goal is to prove the single atomic proposition denoting the semantic resource corresponding to the sentence as a whole: in this case f. Put another way, the derivation must consume all the lexical semantic resources to produce a sentential semantic resource. For our current example, the successful and exhaustive consumption of lexical resources is achieved by means of the following (natural deduction) derivation:

$$\dfrac{\dfrac{g \qquad g \multimap f}{f}\ \multimap\varepsilon \qquad f \multimap f}{f}\ \multimap\varepsilon$$

Following the standard Curry-Howard isomorphism for linear logic, the rule of $\multimap$ elimination (or *modus ponens*) gives rise to functional application of the meaning terms. The derivation above thus automatically constructs a meaning term for f:

$$\dfrac{\dfrac{john : g \qquad cook : g \multimap f}{cook(john) : f}\ \multimap\varepsilon \qquad certain : f \multimap f}{certain(cook(john)) : f}\ \multimap\varepsilon$$

2.2.1. *Skeletons, Modifiers & Semantic Ambiguity*
Deduction in linear logic is known to be intractable. Even the propositional fragment which provides the scaffolding for the glue language approach is known to be NP complete (Kanovich, 1992). In practice, however, it has turned out that the glue analyses published in the literature are computationally feasible. The reason is that the meaning constructors employed in such analyses exhibit a simple and restricted combinatorial pattern. (Gupta and Lamping, 1998) make this precise in terms of a distinction between what they refer to as *modifier* and *skeleton* contributions within constructors. Skeleton contributions come in two forms: either a single *producer* of a certain semantic resource R (in which case the resource is marked as R^+), or a single *consumer* of the resource (in which case the resource is marked R^-). Modifier contributions are matched consumers and producers of a resource R within a given constructor. In the simplest case modifiers are instances of the identity implication $R^- \multimap R^+$.

In the example "*John certainly cooks*", the constructors fall into two classes: pure skeleton, and pure modifier.

$$\begin{array}{rcll} \mathit{john} & : & g^{+} & \text{skeleton} \\ \mathit{cook} & : & g^{-} \multimap f^{+} & \text{skeleton} \\ \mathit{certain} & : & f^{-} \multimap f^{+} & \text{modifier} \end{array}$$

The key empirical observation lying behind the tractability of the glue approach is the 'uniqueness of skeletal contributions'. Given a set of lexical meaning constructors for a sentence, in nearly all cases there will be exactly *one* positive and negative skeletal contribution for each resource.[2] This means that the skeletal contributions can be matched up in, on average, linear time.

The result of this linear pairing of skeleton contributions is a set of derivation trees: one skeleton tree representing the basic complement meaning of the sentence, and a set of modifier trees representing further adjunct meanings. Nearly all the combinatorial complexity of glue derivations lies in the different orders in which modifier trees can be interpolated into the skeleton. It is also these different orders of interpolation that lead to purely semantic ambiguity, such as quantifier scope ambiguity. Lamping and Gupta note that by stopping short at the point just before modifier interpolation, one efficiently obtains a meaning representation strongly resembling many underspecified representations, such as UDRS.

In the case of "*John certainly cooks*" we obtain one skeleton and one modifier tree:

$$\begin{array}{cc} \text{Skeleton} & \text{Modifier} \\ \dfrac{\mathit{john} : g \qquad \mathit{cook} : g \multimap f}{\mathit{cook}(\mathit{john}) : f} & \mathit{certain} : f \multimap f \end{array}$$

We can tell from this that the sentential modifier $f \multimap f$ must be interpolated at the single f conclusion in the skeleton derivation. Were there more than one $f \multimap f$ modifier, different meanings would be obtained by different orders of interpolation.

2.2.2. *Quantifier Scope & Horn Compilation*
A problem that has to be solved is that complex meaning constructors often contain a mixture of skeleton and modifier contributions that is hard to separate. A case in point is the kind of constructor one gets for quantified pronouns like

$$\mathit{everyone} : (g \multimap f) \multimap f$$

[2] Exceptions to this pattern include certain control and coordinate constructions.

where f occurs as a modifier contribution, while g is a skeleton contribution. Using Hepple's method of Horn clause compilation (Hepple, 1996), Lamping and Gupta are able to separate this into a pure skeleton and a pure modifier constructor:

$$\begin{array}{ll} \text{Skeleton} & x : g \\ \text{Modifier} & \lambda P.everyone(\lambda x.P) : f\{g\} \multimap f \end{array}$$

The idea is to excise the embedded antecedent g as a hypothesis, giving it some variable meaning x. The residual $f \multimap f$ implication is subject to the book-keeping restriction that the antecedent f must make use of the excised hypothesis g in its derivation. This dependency is recorded by enclosing the g in braces. In addition, the meaning term associated with the compiled out implication needs to ensure that it binds the variable x introduced by the hypothesis on which its antecedent depends.[3]

To see how this compilation method works, compare the full glue derivations for the sentence "*Everyone cooks*" with uncompiled

$$everyone : (g \multimap f) \multimap f, \quad cook : g \multimap f$$

$$\frac{everyone : (g \multimap f) \multimap f \quad cook : g \multimap f}{everyone(cook) : f}$$

and compiled meaning constrcutors:

$$\lambda P.everyone(\lambda x.P) : f\{g\} \multimap f, \quad x : g, \quad cook : g \multimap f$$

$$\frac{\lambda P.everyone(\lambda x.P) : f\{g\} \multimap f \qquad \dfrac{x : g \quad cook : g \multimap f}{cook(x) : f}}{everyone(\lambda x.cook(x)) : f}$$

A scope ambiguous sentence like "*Everyone cooks something*" yields the following f-structure

$$f : \begin{bmatrix} \text{PRED} & \text{cook } \langle \uparrow \text{SUBJ}, \text{OBJ} \rangle' \\ \text{SUBJ} & g : \begin{bmatrix} \text{PRED} & \text{person} \\ \text{SPEC} & \text{every} \end{bmatrix} \\ \text{OBJ} & h : \begin{bmatrix} \text{PRED} & \text{thing} \\ \text{SPEC} & \text{some} \end{bmatrix} \end{bmatrix}$$

[3] This involves a notion of 'improper binding' of free variables during beta-reduction; for details see (Hepple, 1996).

and the following skeleton and modifier trees when the premises are pre-compiled:

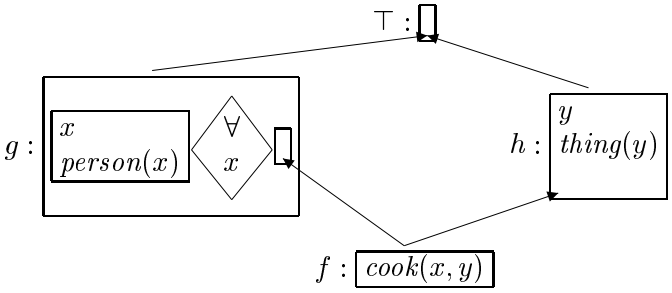
$$\lambda P.everyone(\lambda x.P) : f\{g\} \multimap f \qquad \lambda P.something(\lambda y.P) : f\{h\} \multimap f$$

$$\frac{x : g \quad y : h \quad cook : g \multimap h \multimap f}{cook(x)(y) : f}$$

The arrows between trees indicate dependencies induced by the hypotheses g and h. The two modifiers can be interpolated in either order.

2.3. Review of UDRS

The underspecified discourse representation structure for the scope ambiguous "*Everyone cooks something*" can be represented in a graphical form that is highly reminiscent of the skeleton-modifier glue derivations above:



The labels on the UDRSs bear a close correspondence to the semantic resources in the glue derivations, as implicitly noted in (van Genabith and Crouch, 1997). More interestingly, the dominance relations between UDRSs are related to the hypothetical dependencies in the glue derivations. In what follows, we will bring these correspondences into sharper focus.

The textual description of this UDRS consists of a set of labelled content constraints and a set of structural (subordination "$\leq$") constraints between labels:

$$\left\{ \begin{array}{c} g :: rstr(g)\ \forall x\ scp(g),\ rstr(g) :: x,\ rstr(g) :: person(x), \\ h :: y,\ h :: thing(y),\ f :: cook(x, y), \\ g \leq \top,\ h \leq \top,\ f \leq scp(g),\ f \leq h \end{array} \right\}$$

2.4. UDRS Glue Constructors

We now present lexical entries and meaning constructors, which in combination with the LFG fragment in section 2.1, will build up textual representations of UDRSs. First, some notational conventions to preserve distinctions between nodes in f-structures, σ nodes in semantic projections of f-structures, and UDRS labels. These conventions are necessary because there is often a 1-1-1 correspondence between f-structure nodes, σ-structure nodes and UDRS labels, which invites confusion.

- f-structure nodes
  These are either the meta-variable $\uparrow$,
  instantiated fstr-node values such as f, g or h, or
  f-structure path expressions e.g. $\uparrow$ SUBJ
- σ-structure nodes
  These are either fstr-nodes with a σ subscript, e.g. $\uparrow_\sigma$, f_σ, $(\uparrow \text{SUBJ})_\sigma$, or
  σ-structure path expressions, e.g. $\uparrow_\sigma$ RESTR, f_σVAR, f_σREF
- UDRS labels
  These are either σ-structure nodes with an l subscript, e.g. $\uparrow_{\sigma l}$, $(\uparrow_\sigma \text{RESTR})_l$, or
  the label function scp and $rstr$ applied to UDRS labels, e.g. $rstr(\uparrow_{\sigma l})$, $scp(\uparrow_{\sigma l})$

For nouns, we assume lexical entries like:

$$\begin{array}{ll} \textbf{man} & \text{N} \quad \uparrow \text{PRED} = \text{man} \\ & \lambda x.\{rstr(\uparrow_{\sigma l}) :: man(x)\} : \ \uparrow_\sigma \text{VAR} \multimap \uparrow_\sigma \text{RESTR} \end{array}$$

The glue formula, $\uparrow_\sigma$ VAR $\multimap$ $\uparrow_\sigma$ RESTR, is the one standardly assigned to nouns (Dalrymple *et al.*, 1999b). The meaning term sets up a parameterized set of UDRS conditions; specifically that the condition $man(x)$ is associated with the label $rstr(\uparrow_{\sigma l})$, where x has yet to be filled in with a discourse referent.

The entry for the determiner "*every*" is

$$\begin{array}{ll} \textbf{every} & \text{Det} \quad \uparrow \text{SPEC} = \text{every} \\ & \delta : \ \uparrow_\sigma \text{REF} \\ & \lambda P.P(\delta) \cup \{rstr(\uparrow_{\sigma l}) :: \delta,\ \uparrow_{\sigma l} :: rstr(\uparrow_{\sigma l}) \forall \delta scp(\uparrow_{\sigma l}), \\ & \qquad\qquad rstr(\uparrow_{\sigma l}) < \uparrow_{\sigma l},\ scp(\uparrow_{\sigma l}) < \uparrow_{\sigma l},\ \uparrow_{\sigma l} \leq \top\} \\ & \quad : (\uparrow_\sigma \text{VAR} \multimap \uparrow_\sigma \text{RESTR}) \multimap \uparrow_\sigma \end{array}$$

This gives rise to two separate constructors. The first just associates a (new) discourse referent δ with the $\uparrow_\sigma$ REF resource. The second consumes a noun resource, $(\uparrow_\sigma \text{ VAR} \multimap \uparrow_\sigma \text{ RESTR})$, with associated meaning P to produce a set of conditions for the resource $\uparrow_\sigma$. The set of conditions is obtained by applying the parameterized set of noun conditions, P, to the new discourse referent δ, and unioning this with some further conditions. These further conditions state that (a) the referent δ is introduced within the UDRS labelled $rstr(\uparrow_{\sigma l})$; (b) the UDRS labelled $\uparrow_{\sigma l}$ has a generalized quantifier condition linking its restriction and scope sub-UDRSs, (c) that these restriction and scope UDRSs are subordinate to $\uparrow_{\sigma l}$; and that the UDRS labelled $\uparrow_{\sigma l}$ is subordinate or identical to the topmost UDRS $\top$.

The entry for the determiner "*a*" has the same overall form as the one for "*every*", but conjoins different sets of conditions to those of its noun.

a DET $\uparrow$ SPEC = a
δ : $\uparrow_\sigma$ REF
$\lambda P.P(\delta) \cup \{\uparrow_{\sigma l} :: \delta,\ rstr(\uparrow_{\sigma l}) = \uparrow_{\sigma l},\ scp(\uparrow_{\sigma l}) = \uparrow_{\sigma l},\ \uparrow_{\sigma l} \leq \top\}$
: $(\uparrow_\sigma \text{ VAR} \multimap \uparrow_\sigma \text{ RESTR}) \multimap \uparrow_\sigma$

The conjoined conditions state that (a) the new referent δ is introduced within the UDRS labelled, $\uparrow_{\sigma l}$; (b) that the scope and the restriction of $\uparrow_{\sigma l}$ is $\uparrow_{\sigma l}$ itself; and (c) that $\uparrow_{\sigma l}$ is subordinate or identical to the topmost UDRS.

We can show the effects of combining determiners with nouns in the entries for quantified pronouns like "*everyone*" and "*something*"

everyone NP $\uparrow$ SPEC = every, $\uparrow$ PRED = person
δ : $\uparrow_\sigma$ REF
$\{rstr(\uparrow_{\sigma l}) :: \delta,\ rstr(\uparrow_{\sigma l}) :: man(\delta),$
$\uparrow_{\sigma l} :: rstr(\uparrow_{\sigma l}) \forall \delta scp(\uparrow_{\sigma l}),\ rstr(\uparrow_{\sigma l}) < \uparrow_{\sigma l},$
$scp(\uparrow_{\sigma l}) < \uparrow_{\sigma l},\ \uparrow_{\sigma l} \leq \top\}$: $\uparrow_\sigma$

something NP $\uparrow$ SPEC = some, $\uparrow$ PRED = thing
δ : $\uparrow_\sigma$ REF
$\{\uparrow_{\sigma l} :: \delta,\ \uparrow_{\sigma l} :: thing(\delta),$
$rstr(\uparrow_{\sigma l}) = \uparrow_{\sigma l},\ scp(\uparrow_{\sigma l}) = \uparrow_{\sigma l},\ \uparrow_{\sigma l} \leq \top\}$: $\uparrow_\sigma$

Finally, the meaning constructor associated with a transitive verb consumes the meanings associated with its SUBJ and OBJ resources, as well as its SUBJ REF and OBJ REF resources, and produces the meaning associated with the clause

cooks V $\uparrow \text{PRED} = \text{cook}\langle \uparrow \text{SUBJ}, \uparrow \text{OBJ} \rangle$
$\lambda X, Y, P, Q.\ \{\uparrow_{\sigma l} :: cook(X,Y)\ \ \uparrow_{\sigma l} \leq scp((\uparrow \text{SUBJ})_{\sigma l}),$
$\uparrow_{\sigma l} \leq scp((\uparrow \text{OBJ})_{\sigma l})\} \cup P \cup Q$
$(\uparrow \text{SUBJ})_\sigma \text{REF} \multimap (\uparrow \text{OBJ})_\sigma \text{REF} \multimap (\uparrow \text{SUBJ})_\sigma \multimap (\uparrow \text{OBJ})_\sigma \multimap \uparrow_\sigma$

The set of conditions associated with $\uparrow_\sigma$ is (a) the union of the subject and object conditions P and Q, plus (b) a condition relating the subject and object discourse referents X and Y, and (c) two subordination constraints relating the verbal UDRS to the scope UDRSs of the subject and object.

The earlier f-structure for "*Everyone cooks something*" now yields the instantiated constructors:

everyone $\delta\ :\ g_\sigma \text{REF}$
$\{rstr(g_{\sigma l}) :: \delta,\ rstr(g_{\sigma l}) :: man(\delta), rstr(g_{\sigma l}) < g_{\sigma l},$
$g_{\sigma l} :: rstr(g_{\sigma l}) \forall \delta scp(g_{\sigma l}),\ scp(g_{\sigma l}) < g_{\sigma l},\ g_{\sigma l} \leq \top\}\ :\ g_\sigma$

something $\gamma\ :\ h_\sigma \text{REF}$
$\{h_{\sigma l} :: \gamma,\ h_{\sigma l} :: thing(\gamma), rstr(h_{\sigma l}) = h_{\sigma l},$
$scp(h_{\sigma l}) = h_{\sigma l},\ h_{\sigma l} \leq \top\}\ :\ h_\sigma$

cooks $\lambda X, Y, P, Q.\ \{f_{\sigma l} :: cook(X,Y)\ f_{\sigma l} \leq scp(g_{\sigma l}),$
$f_{\sigma l} \leq scp(h_{\sigma l})\} \cup P \cup Q$
$:\ g_\sigma \text{REF} \multimap h_\sigma \text{REF} \multimap g_\sigma \multimap h_\sigma \multimap f_\sigma$

The glue derivation is trivial

$$\frac{g_\sigma \text{REF} \quad h_\sigma \text{REF} \quad g_\sigma \quad h_\sigma \quad g_\sigma \text{REF} \multimap h_\sigma \text{REF} \multimap g_\sigma \multimap h_\sigma \multimap f_\sigma}{f_\sigma}$$

Functional application of the meaning terms gives

$$\left\{ \begin{array}{c} g_{\sigma l} :: rstr(g_{\sigma l})\ \forall \delta\ scp(g_{\sigma l}),\ rstr(g_{\sigma l}) :: person(\delta),\ rstr(g_{\sigma l}) :: \delta \\ rstr(h_{\sigma l}) :: thing(\gamma),\ rstr(h_{\sigma l}) :: \gamma,\ f_{\sigma l} :: cook(\delta, \gamma), \\ g_{\sigma l} \leq \top,\ h_{\sigma l} \leq \top,\ f_{\sigma l} \leq scp(g_{\sigma l}),\ f_{\sigma l} \leq scp(h_{\sigma l}), \\ rstr(g_{\sigma l}) < g_{\sigma l},\ scp(g_{\sigma l}) < g_{\sigma l},\ rstr(h_{\sigma l}) = h_{\sigma l},\ scp(h_{\sigma l}) = h_{\sigma l} \end{array} \right\}$$

which, by a simple relabeling exercise and substitution of label equalities, can be seen to be a notational variant of the UDRS in our original example.

2.5. Underspecifying UDRS?

The UDRS-glue derivation above is purely skeletal, with exactly one positive and negative occurrence of each skeletal resource, and can thus be performed in linear time. The two scoping alternatives do not arise through different interpolation possibilities for modifier derivations. Instead, they arise from different ways of satisfying the label subordination constraints in the UDRS meaning language.

Horn compilation of non-UDRS constructors for quantified noun phrases shows them to be a mixture of skeleton and modifier. That is, quantified NPs are a form of *obligatory* modifier. The UDRS constructors exploit the obligatory, subcategorized nature of quantified NPs by making them purely skeletal, and building modifier scope variations into the representational scope underspecification of the meaning language. This permits efficient, skeleton-only derivations where quantified NPs are involved.

What happens with purely optional modifiers, like adjectives or adverbs? The UDRS constructors have no choice but to represent these as modifiers within the glue language. For instance, the lexical entry for a sentential adverb like "*certainly*" would be

$$\begin{aligned}\textbf{certainly}\ \ \textsc{adv}\ \ &\uparrow \textsc{pred} = \text{certainly}\\ &\lambda P.\ P \cup \{\uparrow_{\sigma l} :: certain((\textsc{adjn} \in \uparrow)_{\sigma l}),\\ &\qquad \uparrow_{\sigma l} > (\textsc{adjn} \in \uparrow)_{\sigma l}\}\\ &\quad :\ (\textsc{adjn} \in \uparrow)_{\sigma} \multimap (\textsc{adjn} \in \uparrow)_{\sigma}\end{aligned}$$

(In the sentence "*Everyone certainly cooks something*", this would lead to the constructor

$$\lambda P.\ P \cup \{k_{\sigma l} :: certain(f_{\sigma l}),\ k_{\sigma l} > f_{\sigma l}\} \quad : \quad f_{\sigma} \multimap f_{\sigma}$$

where k is the f-node of the adverb, and f of the sentence).

Skeleton-modifier glue derivations involving such constructors would lead to a skeleton derivation, plus modifier derivations to be interpolated into it, possibly in various orders. In the UDRS case, interpolating a modifier just amounts to taking the union of two sets of conditions: the order in which unions are taken does not affect the final meaning representations.

This is awkward. Glue derivations for UDRSs can lead to multiple derivations of a single UDRS. We would like there to be just one, to minimize combinatorial explosion. The reason for the multiple derivations is that glue constructors are geared towards accounting for scope variations in the glue language, by means of interpolating modifiers, while

UDRSs account for scope variation by representational scope subordination constraints within the meaning language. Skeleton-modifier glue derivations for UDRS construction thus *underspecify* an underspecified UDRS representation, in that all ways of fully specifying the UDRS in terms of modifier interpolations are equivalent.

In practice, this is not a problem. We stop short in the glue derivation at the same point as Lamping & Gupta, i.e. just before interpolating modifiers into the skeleton tree. But unlike Lamping & Gupta, we know that all ways of interpolating modifiers lead to equivalent UDRSs: and so we can pick just one. While computationally straightforward, this is logically suspect. The expedient of picking just one interpolation means that the derivation procedure is deliberately incomplete with respect to standard linear logic inference.

3. Ambiguity Preserving Transfer and Head-Switching

Semantic representations have been used in transfer based approaches to machine translation. In such a scenario, a source language semantic representation established by analysis of a source string is related to a target language representation by means of a transfer component. The target language representation is then handed to a generator which generates the target string. Often ambiguities present in the source language string can be carried over intact to the target string. Examples of such ambiguities include modifier attachment, quantifier and operator scope ambiguities. For example, notice that the quantifier scope ambiguity in our English example sentence carries over unchanged into the German translation: "*Jeder kocht etwas*". It would be inefficient to define transfer on disambiguated (semantic) representations only to find out later on by intersecting the strings produced from disambiguated target semantic representations that there exists a target string that covers the original readings of the source input string. A number of ambiguity preserving approaches to transfer based machine translation have been presented in the literature, e.g.: (Alshawi *et al.*, 1991; Emele and Dorna, 1998; Dorna *et al.*, 1998; van Genabith *et al.*, 1998) .

In (van Genabith *et al.*, 1998), the input to transfer is the set of source language glue constructors instantiated to the f-structure and σ-structure obtained from analysis of the source string. Such a set can be considered a (highly) underspecified semantic representation in the sense that it is an encoding of the linear logic derivations assigning disambiguated semantic representations to the source string (van Genabith and Crouch, 1999). Transfer consumes the source constructors to produce target constructors, in a way that is lexicalized and reversible.

Matching the instantiated target constructors against the target lexicon allows one to retrieve syntactic information about the f-structure attributes of the instantiated nodes, and construct a target f-structure. From this, target c-structures and strings may be generated. By transferring on meaning constructors, one can strive to ensure that the same range of scoping possibilities are open to the source and target.

The approach successfully deals with examples such as argument switching, and where several source constructors need to be consumed in order to produce a single target constructor (e.g "*commit suicide*" / "*se suicider*"). However, its treatment of structural misalignment problems, such as head switching, is not completely satisfactory.

3.1. Head Switching

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

$$\textit{Hans kocht gerne} \quad \leftrightarrow \quad \textit{Hans likes cooking}$$

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 (recall that f-structures are abstract syntactic representations):

$$f_1 : \begin{bmatrix} \text{SUBJ} & f_2 : \begin{bmatrix} \text{PRED} & \text{HANS} \end{bmatrix} \\ \text{PRED} & \text{KOCHEN}\langle \uparrow \text{SUBJ} \rangle \\ \text{ADJN} & \{ f_3 : \begin{bmatrix} \text{PRED} & \text{GERNE} \end{bmatrix} \} \end{bmatrix}$$

$$f_3 : \begin{bmatrix} \text{SUBJ} & f_2 : \begin{bmatrix} \text{PRED} & \text{HANS} \end{bmatrix} \\ \text{PRED} & \text{LIKE}\langle \uparrow \text{SUBJ}, \uparrow \text{XCOMP} \rangle \\ \text{XCOMP} & f_1 : \begin{bmatrix} \text{SUBJ} & f_2 : \begin{bmatrix} \text{PRED} & \text{HANS} \end{bmatrix} \\ \text{PRED} & \text{COOK}\langle \uparrow \text{SUBJ} \rangle \end{bmatrix} \end{bmatrix}$$

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

$$f_1 : \begin{bmatrix} \text{SUBJ} & f_2 : \begin{bmatrix} \text{PRED} & \text{EDE} \end{bmatrix} \\ \text{PRED} & \text{VERMUTEN}\langle \uparrow \text{SUBJ}, \uparrow \text{COMP} \rangle \\ \text{COMP} & f_3 : \begin{bmatrix} \text{SUBJ} & f_4 : \begin{bmatrix} \text{PRED} & \text{HANS} \end{bmatrix} \\ \text{PRED} & \text{KOCHEN}\langle \uparrow \text{SUBJ} \rangle \\ \text{ADJN} & \{ f_5 : \begin{bmatrix} \text{PRED} & \text{GERNE} \end{bmatrix} \} \end{bmatrix} \end{bmatrix}$$

$$f_1 : \begin{bmatrix} \text{SUBJ} & f_2 : \begin{bmatrix} \text{PRED} & \text{EDE} \end{bmatrix} \\ \text{PRED} & \text{ASSUME}\langle \uparrow \text{SUBJ}, \uparrow \text{COMP} \rangle \\ \text{COMP} & f_5 : \begin{bmatrix} \text{SUBJ} & f_4 : \begin{bmatrix} \text{PRED} & \text{HANS} \end{bmatrix} \\ \text{PRED} & \text{LIKE}\langle \uparrow \text{SUBJ}, \uparrow \text{XCOMP} \rangle \\ \text{XCOMP} & f_3 : \begin{bmatrix} \text{SUBJ} & f_4 : \begin{bmatrix} \text{PRED} & \text{HANS} \end{bmatrix} \\ \text{PRED} & \text{COOK}\langle \uparrow \text{SUBJ} \rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

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 "*suspect*". 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 "*suspect*" which expects f_3 is offered f_5, resulting in a disconnected f-structure.

It has been shown (Sadler and Thompson, 1991) that this kind of embedded head switching is problematic for purely correspondence based approaches to transfer on f-structures (Kaplan *et al.*, 1989). The same problem shows up for (van Genabith *et al.*, 1998). This is because transfer on instantiated glue constructors is not purely semantic transfer: in effect it transfers the syntax-semantics interface, where structural mismatches are still present.

3.2. Head-Switched Constructors

Instantiating meaning constructors against the f-structures of the English and German sentences illustrating embedded head switching would yield the following for "*likes*" and "*gerne*", and for "*assume*" and "*vermutet*"[4]

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

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

[4] We assume that "*gerne*" is a subject-oriented adverb.

Note how 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 introduces an extra level of structure, we might envisage a purely lexical transfer rule

$$\begin{array}{rl} \forall G, F. & \lambda P, x.gerne(x, P(x)) \ : (G \multimap F) \multimap (G \multimap F) \\ \Leftrightarrow & \\ & \lambda P, x.like(x, P(x)) \quad : (G \multimap F) \multimap (G \multimap new(F)) \end{array}$$

where G and F range over f/σ nodes, and $new(F)$ 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

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

In the absence of embedded head switching, this transfer rule works well. But just 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. It is very hard to see how this can be done solely on the basis of local, purely lexical transfers.

Two partial solutions are proposed in (van Genabith *et al.*, 1998). The first solution employs the transfer rule:

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

Applied to source constructor

$$\lambda P, x.gerne(x, P(x)) : (f_4 \multimap f_3) \multimap (f_4 \multimap f_3)$$

this yields the target constructor

$$\lambda P, x.like(x, P(x)) : (f_4 \multimap f_3) \multimap (f_4 \multimap f_3)$$

From this, and the other transferred target constructors, it is possible to derive the full range of meanings for the English sentence. However, the tranferred constructor does *not* match the lexical entry for "*likes*" in the monolingual English lexicon. One therefore cannot hope to generate the target f-structure directly, by matching transferred constructors against the target lexicon. Instead generation of target strings must proceed

by deriving all semantic interpretations, generating strings from fully specified semantics, and intersecting strings. However, when all the transferred constructors do match the lexicon, target generation can proceed directly from the target meaning constructors.

The second solution adapts an idea originally proposed in (Sadler and Thompson, 1991): predicate nuclei are pushed downwards via functional uncertainty constraints. It employs the transfer rule

$$\begin{array}{rl} \forall G, H, F. & \mathit{vermuten} : G \multimap ((H\,\text{XCOMP}^*) \multimap F) \\ \Leftrightarrow & \\ & \mathit{assume} \quad : G \multimap ((H\,\text{XCOMP}^*) \multimap F) \end{array}$$

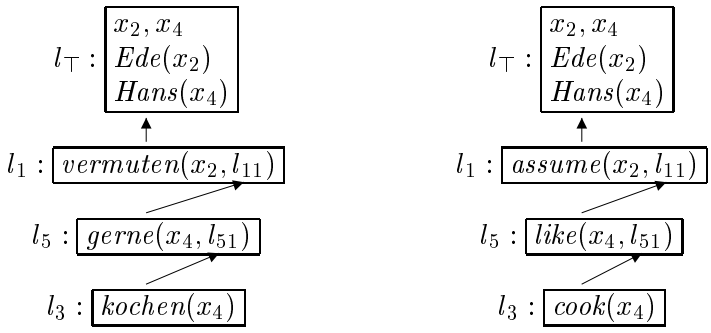
This allows complement verbs to look arbitrarily far down through XCOMPs, just in case head switching has introduced or removed structure. This approach supports ambiguity preserving transfer but at present it is not exactly clear how the additional non-determinism introduced by the functional uncertainty equations interacts with other components of the system. Nor is it entirely clear what this functional uncertainty of f-structure attributes means in the context of fully instantiated glue constructors.

3.3. UDRS Transfer

It is well known that structural misalignment problems of the kind discussed above can often be avoided by defining transfer on a more abstract level of representation such as semantic (rather then syntactic) levels of representation where the structural misalignment has disappeared (cf. (Kaplan and Wedekind, 1993)). While meaning constructors are predominantly semantic representations they do in fact encode an entire syntax—semantics interface between f-structures (and their σ-projections) and semantic representations.

The meaning representation assigned to a sentence in our UDRT based glue language semantics is both purely semantic and underspecified. Hence we can expect to be able to use this as the basis for ambiguity preserving transfer while side-stepping structural misalignment problems of the kind outlined above. To see this, consider the (simplified) UDRT representations for the complex embedded head switching case:[5]

[5] In order to derive these UDRSs, we extend our fragment by the (illustrative) lexical entries stated in the Appendix.



Ede vermutet daß Hans gerne kocht Ede assumes that Hans likes cooking

Notice that the semantic structures are completely aligned: the head switching problem has disappeared and transfer can easily be lexicalized.

The transfer rules required to do the head switching examples are stated below. We use the Prolog-style notation of (Dorna and Emele, 1996) where sets are represented as Prolog lists, and uppercase letters are logical variables instantiated during rule application.[6] The transfer rules are of the form `Source <-> Target`. They are bidirectional and in the forward direction map a `Source` set description into a `Target` set description, in the process removing the covering set `Source` from the input set. Transfer is complete once the input set is empty. The crucial transfer rules for our head switching examples are then simply the following:

```
      [ L:kochen(X) ] <-> [ L:cook(X) ]
[ L:vermuten(X,L1) ] <-> [ L:assume(X,L1) ]
   [ L:gerne(X,L1) ] <-> [ L:like(X,L1) ]
```

4. Discussion, Conclusion and Further Work

The first part of the paper sketched how to derive UDRT meanings for f-structures using glue semantics, extending the range of semantic formalisms to which the glue approach has been applied. In practice, glue for UDRT works out more simply than for traditional semantic representations. In particular, the treatment in UDRT of obligatory scope inducing items, like quantified NPs, removes the need to use

[6] The notation is inspired by the Prolog implementation of the transfer rule compiler and the transfer run time system.

Horn clause compilation to cleanly separate glue skeletons from modifiers. However, there is a theoretical twist. Glue semantics provides an account of scope ambiguity where alternate linear logic derivations are instrumental in spelling out the scope possibilities. But when applied to UDRT, the representation of quantifier scope ambiguity is relegated to the level of the meaning language, unburdening the glue language of one of its original functions. But for other types of modifier scope ambiguity, glue and UDRT compete to represent the possibilities.

For transfer-based machine translation, using an underspecified semantic meaning representation language (such as UDRT, Hole Semantics (Bos, 1995) or an underspecified version of CDRT) means that ambiguity preserving transfer can operate on pure (and underspecified) semantic representations. Unlike the otherwise related approach in (van Genabith *et al.*, 1998), this approach is often able to side-step problematic structural misalignment cases such as head switching in transfer. Finally, we showed how the semantic transfer machinery developed in (Dorna and Emele, 1996) integrates with the approach detailed in the present paper.

There are, of course, a number of important trade-offs made along the way. It is arguable that the role of the linear logic glue is somewhat trivialized. First, by pushing the representation of ambiguity into the meaning language, we no longer require a linear logic inference system to find the complete set of derivations possible from a set of premises. Second, in contrast to the transfer method of (van Genabith *et al.*, 1998), transfer here operates on pure semantic representations, and not glue meaning constructors. Moreover, generation from constructors with UDRT style meaning representations has yet to be developed.

Future work: In some respects, transfer on UDRSs represents a third partial solution to the problem of structural misalignment, to go along with the other two in (van Genabith *et al.*, 1998). However, investigations subsequent to the work done on this paper for IWCS-99 have suggested a way of using linear logic as a transfer formalism. This appears to operate successfully on purely local and lexically defined glue-constructor transfer rules, while rewriting structural misalignments. We hope to report on this elsewhere.

Appendix

Below we state the lexical entries needed for the extended fragment (Section 3.3) concerned with head switching:

Hans NP
$\uparrow$ PRED = Hans
δ : $\uparrow_\sigma$ REF
$\{\top :: \delta,\ \top :: hans(\delta)\}$

assumes V
$\uparrow$ PRED = assume$\langle\uparrow$SUBJ, $\uparrow$COMP$\rangle$
$\lambda X, P, Q.\ \{\uparrow_{\sigma l} :: assume(X, (\uparrow \text{COMP})_{\sigma l}),$
$\uparrow_{\sigma l} \leq scp((\uparrow \text{SUBJ})_{\sigma l}),\ (\uparrow \text{COMP})_{\sigma l} < \uparrow_{\sigma l}\}$
$\cup P \cup Q$
: $(\uparrow \text{SUBJ})_\sigma$REF $\multimap$ $(\uparrow \text{SUBJ})_\sigma \multimap (\uparrow \text{COMP})_\sigma \multimap \uparrow_\sigma$

likes V
$\uparrow$ PRED = like$\langle\uparrow$SUBJ, $\uparrow$XCOMP$\rangle$, $\uparrow$SUBJ =$\uparrow$XCOMP SUBJ
$\lambda X, Q.\ \{\uparrow_{\sigma l} :: like(X, (\uparrow \text{XCOMP})_{\sigma l}),\ \uparrow_{\sigma l} \leq (\uparrow \text{SUBJ})_{\sigma l},$
$(\uparrow \text{XCOMP})_{\sigma l} < \uparrow_{\sigma l}\} \cup Q(X)$
: $(\uparrow \text{SUBJ})_\sigma$REF $\multimap$ ($(\uparrow \text{SUBJ})_\sigma$REF $\multimap$ $(\uparrow \text{XCOMP})_\sigma$) $\multimap$ $\uparrow_\sigma$

gerne ADV
$\uparrow$ PRED = gerne
$\lambda X, Q.\ \{\uparrow_{\sigma l} :: gerne(X, (\text{ADJN} \in \uparrow)_{\sigma l}),$
$\uparrow_{\sigma l} \leq ((\text{ADJN} \in \uparrow)\text{SUBJ})_{\sigma l},\ (\text{ADJN} \in \uparrow)_{\sigma l} < \uparrow_{\sigma l}\}$
$\cup Q(X)$
: $((\text{ADJN} \in \uparrow)\text{SUBJ})_\sigma$REF $\multimap$ ($((\text{ADJN} \in \uparrow)\text{SUBJ})_\sigma$REF $\multimap$
$(\text{ADJN} \in \uparrow)_\sigma$) $\multimap$ $(\text{ADJN} \in \uparrow)_\sigma$

References

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.

Bos, J. 1995. Predicate logic unplugged. In *Proceedings of the 10th Amsterdam Colloquium*, University of Amsterdam, The Netherlands.

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, 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.

Dalrymple, M.; Lamping, J.; Pereira, F.C.N; and Saraswat, V. 1999b. Overview and introduction. In Dalrymple, M, editor 1999, *Semantics and Syntax in Lexical Functional Grammar.* MIT Press. 33–57.

Dorna, M. and Emele, M.C. 1996. Semantic-based transfer. In *COLING'96, Copenhagen, Denmark.* 316–321.

Dorna, M.; Frank, A.; van Genabith, J.; and Emele, M.C. 1998. Syntactic and semantic transfer with f-structures. In *COLING-ACL'98, Montreal, Canada*. 341–347.

Emele, M.C. and Dorna, M. 1998. Ambiguity preserving machine translation using packed representations. In *COLING-ACL'98, Montreal, Canada*. 365–371.

Gupta, V. and Lamping, J. 1998. Efficient linear logic meaning assembly. In *COLING-ACL'98, Montreal, Canada*. 464–470.

Hepple, M. 1996. A compilation-chart method for linear categorial deduction. In *COLING-96, Copenhagen, Denmark*. 537–542.

Kanovich, M. 1992. Horn programming in linear logic is NP-complete. In *Seventh Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press. 200–210.

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, R.M. and Wedekind, J. 1993. Restriction and correspondence-based translation. In Krauwer, S.; Moortgat, M.; and Tombe, Louisdes, editors 1993, *Sixth Conference of the European Chapter of the Association for Computational Linguistics — Proceedings of the Conference*. ACL. 193–202.

Kaplan, R.M.; Netter, K.; Wedekind, J.; and Zaenen, A. 1989. Translation by structural correspondence. In *EACL'89, Manchester, U.K.* ACL. 272–281.

Muskens, R. 1996. Combining montague semantics and discourse representation theory. *Linguistics and Philosophy* 19:143–186.

Reyle, U. 1993. Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics* 10:123–179.

Sadler, L. and Thompson, H. S. 1991. Structural non-correspondence in translation. In *EACL'91, Fifth Conference of the European Chapter of the Association for Computational Linguistics — Proceedings of the Conference, Berlin, Germany*. ACL. 293–298.

van Genabith, J. and Crouch, R. 1997. On interpreting f-structures as UDRSs. In *ACL-EACL-97, Madrid, Spain*. 402–409.

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

van Genabith, J. and Crouch, R. 1999. 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 (eds.) H. Bunt and R. Muskens, *Computing Meaning*, Volume 1. Kluwer Academic Publishers, Dordrecht, The Netherlands. 129–148.