Solving Headswitching Translation Cases in LFG-DOT

Andy Way

School of Computer Applications,
Dublin City University,
Dublin, Ireland.

Email: away@compapp.dcu.ie

Proceedings of the LFG01 Conference

University of Hong Kong, Hong Kong

Miriam Butt and Tracy Holloway King (Editors)

2001

CSLI Publications

http://csli-publications.stanford.edu/

Abstract

It has been shown that LFG-MT (Kaplan *et al.*, 1989) has difficulties with Headswitching data (Sadler *et al.*, 1989, 1990; Sadler & Thompson, 1991). We revisit these arguments in this paper. Despite attempts at solving these problematic constructions using approaches based on linear logic (Van Genabith *et al.*, 1998) and restriction (Kaplan & Wedekind, 1993), we point out further problems which are introduced.

We then show how LFG-DOP (Bod & Kaplan, 1998) can be extended to serve as a novel hybrid model for MT, LFG-DOT (Way, 1999, 2001), which promises to improve upon the DOT model of translation (Poutsma 1998, 2000) as well as LFG-MT. LFG-DOT improves the robustness of LFG-MT through the use of the LFG-DOP *Discard* operator, which produces generalized fragments by discarding certain f-structure features. LFG-DOT can, therefore, deal with ill-formed or previously unseen input where LFG-MT cannot. Finally, we demonstrate that LFG-DOT can cope with such translational phenomena which prove problematic for other LFG-based models of translation.

1 Headswitching in LFG-MT

Kaplan *et al.* (1989) illustrate their LFG-MT proposal with the well-known headswitching case *venir de X* ⟷*has just X-ed*, as in (1):

(1) The baby just fell ⟷Le bébé vient de tomber.

They propose to deal with such problems in two ways. The first of these is as in (2):

(2) *just*: $(\uparrow \text{PRED}) = \text{`just}\langle(\uparrow \text{ARG})\rangle\text{'}$, $(\tau\uparrow \text{PRED}) = \text{venir}$, $(\tau\uparrow \text{XCOMP}) = \tau(\uparrow \text{ARG})$

That is, the XCOMP function of *venir* (in (1), *de tomber*) corresponds to the ARG function of *just* (in (1), *the baby fell*), as shown by the respective source and target f-structures in (3) and (4):

(3)
$$\left[\begin{array}{ll} \text{PRED} & \text{`just}\langle[\text{fall}]\rangle\text{'} \\ \text{ARG} & \left[\begin{array}{ll} \text{SUBJ} & \left[\begin{array}{ll}\text{PRED} & \text{`baby'}\\ \text{SPEC} & \text{the}\end{array}\right] \\ \text{TENSE} & \text{PAST} \\ \text{PRED} & \text{`fall}\langle[\text{baby}]\rangle\text{'}\end{array}\right]\end{array}\right]$$

(4)
$$\left[\begin{array}{ll} \text{SUBJ} & \left[\begin{array}{ll}\text{PRED} & \text{`bébé'}\\ \text{SPEC} & \text{le}\end{array}\right] \\ \text{TENSE} & \text{PRES} \\ \text{PRED} & \text{`venir}\langle[\text{bébé}],[\text{tomber}]\rangle\text{'} \\ \text{XCOMP} & \left[\begin{array}{ll} \text{DE} & + \\ \text{SUBJ} & [\ \ ] \\ \text{PRED} & \text{`tomber}\langle[\text{bébé}]\rangle\text{'}\end{array}\right]\end{array}\right]$$

The second approach is where *just* is not treated as a head subcategorizing for an ARG, but as a 'normal' adverbial sentential modifier. Instead, headswitching occurs between source and target f-structures, as in (5):

$$(5)\quad \begin{array}{llll} \mathrm{S} \longrightarrow & \mathrm{NP} & \mathrm{ADVP} & \mathrm{VP} \\ & (\uparrow\mathrm{SUBJ})=\downarrow & (\uparrow\mathrm{SADJ})=\downarrow & \uparrow=\downarrow \\ & & (\tau\uparrow\mathrm{SADJ\ XCOMP}) = \tau\uparrow & \end{array}$$

$$\begin{array}{llllll} \textit{just}\text{: ADV,} & (\uparrow \mathrm{PRED}) = \mathrm{just} & & \textit{fall}\text{: V,} & (\uparrow \mathrm{PRED}) = \mathrm{fall} \\ & (\tau\uparrow \mathrm{PRED}) = \mathrm{venir} & & & (\tau\uparrow\mathrm{SUBJ}) = \tau(\uparrow\mathrm{SUBJ}) \\ & & & & (\tau\uparrow \mathrm{PRED}) = \mathrm{tomber} \end{array}$$

Here the τ annotation to ADVP states that the τ of the mother f-structure is the XCOMP of the τ of the SADJ slot. This set of equations (along with others of a more trivial nature) produces the f-structure (6):

$$(6)\quad \begin{bmatrix} \mathrm{SUBJ} & \begin{bmatrix} \mathrm{PRED} & \text{'baby'} \\ \mathrm{SPEC} & \text{the} \end{bmatrix} \\ \mathrm{TENSE} & \mathrm{PAST} \\ \mathrm{PRED} & \text{'fall}\langle\text{[baby]}\rangle\text{'} \\ \mathrm{SADJ} & \left\{ \begin{bmatrix} \mathrm{PRED} & \text{'just'} \end{bmatrix} \right\} \end{bmatrix}$$

1.1 Embedded Cases of Headswitching

However, Sadler *et al.* (1989, 1990) show that neither approach is able to deal elegantly and straightforwardly with more complex cases of headswitching, as in (7):

(7) I think that the baby just fell $\longleftrightarrow$ Je pense que le bébé vient de tomber.

In (7), the headswitching phenomenon takes place in the sentential COMP, rather than in the main clause, as in (1). Here the structure in (3) must be a COMP to a PRED in a higher f-structure. Hence, the normal f-description on embedded S nodes ($\uparrow$ COMP = $\downarrow$) must be optional, and instead the structure in (3) must be unified to the root f-structure as the value of its COMP node. This can be handled by the disjunction in (8):

$$(8)\quad \begin{array}{lllll} \mathrm{VP} & \longrightarrow & \mathrm{V} & \text{that} & \mathrm{S} \\ & & & & \{(\uparrow\mathrm{COMP})=\downarrow,\ (\uparrow\mathrm{COMP\ ARG})=\downarrow\} \end{array}$$

We require this disjunction on embedded S nodes to include ($\uparrow$COMP ARG)=$\downarrow$ just in case they contain such a headswitching construction, as f-structure (9) shows:

$$(9)\quad \begin{bmatrix} \mathrm{SUBJ} & \begin{bmatrix} \mathrm{PRED} & \text{'I'} \end{bmatrix} \\ \mathrm{PRED} & \text{'think}\langle\text{[i],[just]}\rangle\text{'} \\ \mathrm{COMP} & \begin{bmatrix} \mathrm{PRED} & \text{'just}\langle\text{[fall]}\rangle\text{'} \\ \mathrm{ARG} & \begin{bmatrix} \mathrm{SUBJ} & \begin{bmatrix} \mathrm{PRED} & \text{'baby'} \\ \mathrm{SPEC} & \text{the} \end{bmatrix} \\ \mathrm{TENSE} & \mathrm{PAST} \\ \mathrm{PRED} & \text{'fall}\langle\text{[baby]}\rangle\text{'} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Otherwise, structure (3) (rooted in *just*) is not connected to the higher COMP slot. Nevertheless, the solution proposed in (8) seems a little *ad hoc*, requiring a disjunction just in case the sentential COMP includes a headswitching case. We shall see in the next section that if such headswitching adverbs co-occur, then further disjuncts are required, unless these can be abbreviated by a functional uncertainty equation.

If we choose the second approach (5), where *just* is a sentential modifier, given that the headswitching is a τ operation, we require the lexical entry for *think* in (10):

(10) *think*: V, ($\tau \uparrow$ PRED) = penser, τ ($\uparrow$ SUBJ) = ($\tau \uparrow$ SUBJ), τ ($\uparrow$ COMP) = ($\tau \uparrow$ COMP)

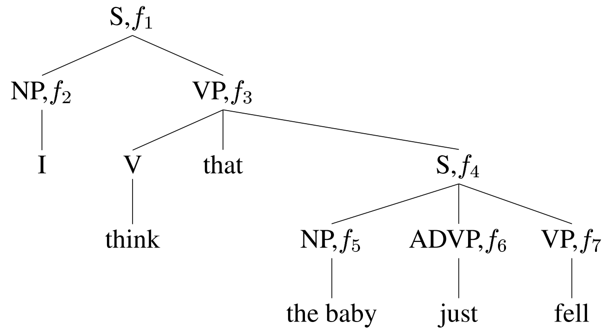
This specifies that τ of the mother f-structure's COMP slot is the COMP of the τ of the mother's f-structure. That is, both this argument, the COMP, and the SUBJ of *think* are to be translated straightforwardly. This is indeed the case in (11):

(11) I think that the baby fell $\longleftrightarrow$Je pense que le bébé est tombé.

However, when the COMP includes a headswitching case, as in (7), we end up with a doubly rooted target f-structure because of a clash between the regular τ equation in the lexical entry for *think*, (10), and the structural τ equation on the ADVP in the (5), which requires the τ of the same piece of f-structure to be the XCOMP of the τ of the SADJ slot. One piece of f-structure is required to fill two inconsistent slots. We will now illustrate this in detail.

The c- and f-structures for the source sentence in (7), *I think that the baby just fell*, are shown in (12):

(12)



$$f_1, f_3 \left[\begin{array}{ll} \text{SUBJ} & f_2\left[\begin{array}{ll}\text{PRED} & \text{‘I’}\end{array}\right] \\ \text{PRED} & \text{‘think}\langle\text{[i],[fall]}\rangle\text{’} \\ \text{COMP} & f_4, f_7 \left[\begin{array}{ll} \text{SUBJ} & f_5\left[\begin{array}{ll}\text{PRED} & \text{‘baby’} \\ \text{SPEC} & \text{the}\end{array}\right] \\ \text{TENSE} & \text{PAST} \\ \text{PRED} & \text{‘fall}\langle\text{[baby]}\rangle\text{’} \\ \text{SADJ} & \{f_6\left[\begin{array}{ll}\text{PRED} & \text{‘just’}\end{array}\right]\} \end{array}\right] \end{array}\right]$$

The set of τ equations obtained are those in (13):

(13) From the lexical entry for *think*, (10):
τf_3 PRED = *penser*
τf_3 SUBJ = $((\tau f_3)$ SUBJ)
τf_3 COMP = $((\tau f_3)$ COMP)

From the rules and entries in (5):
τf_6 PRED = *venir*
τf_7 PRED = *tomber*
$\tau f_4 = (\tau(f_4$ SADJ) XCOMP)

From straightforward equations for the NPs:
τf_2 PRED = *je*
τf_5 PRED = *bébé*
τf_5 SPEC = *le*

Now, from the c-structure in (12), we see that f_3's COMP is f_4, so the equation τf_3 COMP = $((\tau f_3)$ COMP) can be altered quite simply to $\tau f_4 = ((\tau f_3)$ COMP). By the same token, f_4's SADJ is f_6. We now have two equations which cannot both be solved with the result that a target f-structure is formed. The clash is shown in (14):

(14)

τ ($\uparrow$ COMP) = ($\tau \uparrow$ COMP)	$\tau f_4 = ((\tau f_3)$ COMP)
$\tau \uparrow = \tau$ ($\uparrow$ SADJ XCOMP)	$\tau f_4 = ((\tau f_6)$ XCOMP)

This results in the construction of the two partial target f-structures in (15):

(15)
$$\tau f_6 \left[\begin{array}{ll} \text{SUBJ} & \tau f_5\left[\ \right] \\ \text{PRED} & \text{‘venir}\langle\text{[bébé],[tomber]}\rangle\text{’} \\ \text{XCOMP} & \tau f_4 \left[\begin{array}{ll} \text{SUBJ} & \tau f_5\left[\begin{array}{ll}\text{PRED} & \text{‘bébé’} \\ \text{SPEC} & \text{le}\end{array}\right] \\ \text{PRED} & \text{‘tomber}\langle\text{[bébé]}\rangle\text{’} \end{array}\right] \end{array}\right] \qquad \tau f_3 \left[\begin{array}{ll} \text{SUBJ} & \tau f_2\left[\begin{array}{ll}\text{PRED} & \text{‘Je’}\end{array}\right] \\ \text{PRED} & \text{‘penser}\langle\text{[je],[...]}\rangle\text{’} \\ \text{COMP} & \tau f_4\left[\ \ \right] \end{array}\right]$$

That is, τf_4 is required to be both the XCOMP of *venir* and the COMP of *penser* simultaneously, a conflict that needs to be resolved if a proper target f-structure is to be produced. As was done in (8), we might disjoin the problematic τ equation in (10) with an equation for the ‘special’ case in (16):

(16) $(\tau \uparrow \text{COMP}) = \tau\ (\uparrow \text{COMP SADJ})$

That is, τ of the mother at the COMP slot is τ of the mother's COMP SADJ slot. This is clearly undesirable, since it must be specified for every embedding verb.

1.2 Scoping of Multiple Adverbs

In a similar fashion, Sadler *et al.* (1989, 1990) show that the two approaches of Kaplan *et al.* cannot deal straightforwardly with other complex cases of headswitching involving scoping of multiple adverbs, as in (17):

(17) a. Jan zwemt toevallig graag $\longrightarrow$

b. John happens to like to swim

c. *John likes to happen to swim

That is, despite the fact that (17c) is a grammatical string of English, it is not the translation of (17a)—(17b) is. (17c) is the translation of *Jan zwemt graag toevallig*. Given that the word order differs, we have a different scoping of the adverbs, resulting in a different translation. Let us assume the S rule in (5), and the lexical entries in (18):

(18) *graag*: ADV, $(\uparrow \text{PRED}) = \text{graag}$, $(\tau\uparrow \text{PRED}) = \text{like}$

zwemmen: V, $(\uparrow \text{PRED}) = \text{zwemmen}$, $(\tau\uparrow \text{SUBJ}) = \tau(\uparrow \text{SUBJ})$, $(\tau\uparrow \text{PRED}) = \text{swim}$

Let us also assume the somewhat simpler sentence (19):

(19) Jan zwemt graag ($\longrightarrow$John likes to swim)

Given the S rule in (5), and the lexical entries in (18), the f-structure for (19) is (20):

$$(20)\quad \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{`Jan'} \end{bmatrix} \\ \text{TENSE} & \text{PRES} \\ \text{PRED} & \text{`zwemmen}\langle[\text{Jan}]\rangle\text{'} \\ \text{SADJ} & \left\{ \begin{bmatrix} \text{PRED} & \text{`graag'} \end{bmatrix} \right\} \end{bmatrix}$$

This object is very similar to the f-structure in (6). The other possibility is that the lexical entry for *graag* is (21):

(21) *graag*: $(\uparrow \text{PRED}\ ) = \text{`graag}\langle(\uparrow \text{ARG})\rangle\text{'}$, $(\tau\uparrow \text{PRED}) = \text{like}$, $(\tau\uparrow \text{XCOMP}) = \tau(\uparrow \text{ARG})$

In this case, the f-structure in (22) would be built for the Dutch sentence in (19):

$$(22)\quad \begin{bmatrix} \text{PRED} & \text{`graag}\langle[\text{zwemmen}]\rangle\text{'} \\ \text{ARG} & \begin{bmatrix} \text{SUBJ} & \text{`Jan'} \\ \text{TENSE} & \text{PRES} \\ \text{PRED} & \text{`zwemmen}\langle[\text{Jan}]\rangle\text{'} \end{bmatrix} \end{bmatrix}$$

Being an adverb of the same type, *toevallig* can occur in similar contexts as *graag*, as in (23):

(23) Jan zwemt toevallig ($\longrightarrow$John happens to swim)

As seen in (17), such adverbs can co-occur. To maintain this approach and produce the translation (17b), we require the f-structure in (24):

(24)
$$\left[\begin{array}{ll} \text{PRED} & \text{`toevallig}\langle[\text{graag}]\rangle\text{'} \\ \text{ARG} & \left[\begin{array}{ll} \text{PRED} & \text{`graag}\langle[\text{zwemmen}]\rangle\text{'} \\ \text{ARG} & \left[\begin{array}{ll} \text{SUBJ} & \text{`Jan'} \\ \text{TENSE} & \text{PRES} \\ \text{PRED} & \text{`zwemmen}\langle[\text{Jan}]\rangle\text{'} \end{array}\right] \end{array}\right] \end{array}\right]$$

To associate the embedded S node with an f-structure which is the value of COMP ARG ARG of the mother S's associated f-structure, we would need to add ($\uparrow$ COMP ARG ARG) = $\downarrow$ to the disjunction on S in (8). We may be able to simplify these equations using functional uncertainty as in ($\uparrow$ COMP ARG*) = $\downarrow$. Nevertheless, it remains very unnatural to annotate embedded S symbols to allow for the possibility that they *may* contain (any number of) subcategorizing ADVs of this type. It should be clear that cases like (25) are problematic in the same way as (7)-(16):

(25) Ik denk dat Jan toevallig graag zwemt.

The alternative f-structure corresponding to (17a) is (26):

(26)
$$\left[\begin{array}{ll} \text{SUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{`Jan'} \\ \text{NUM} & \text{SG} \end{array}\right] \\ \text{PRED} & \text{`zwemmen}\langle(\uparrow\text{SUBJ})\rangle\text{'} \\ \text{TENSE} & \text{PRES} \\ \text{ADJUNCT} & \left\{\left[\begin{array}{ll} \text{PRED} & \text{`toevallig'} \end{array}\right], \left[\begin{array}{ll} \text{PRED} & \text{`graag'} \end{array}\right]\right\} \end{array}\right]$$

Given the original formulation of LFG, there is no way of producing the required embedding of *graag* ('likingly') under *toevallig* ('by chance'), and not vice versa, without resorting to tuning: under both approaches outlined, changing our c-structure assumptions to deal with a difficult translation case necessitates the abandonment of modularity.

In general, this approach requires the tuning of f-structures, which is arguably as problematic as producing a sufficiently abstract representation for simple transfer in other systems. For example, since German and Dutch both have such adverbs, as shown in (27), it is possible to treat them as 'normal' adverbs and still produce adequate translations:

(27) a. NL: toevallig $\longrightarrow$DE: zufällig

b. NL: graag $\longrightarrow$DE: gerne

Hence the danger exists of producing different source language f-structures according to the target language requirements: the cases where the adverbs are top-level PREDs (such as (21), for instance) is appropriate for translation from Dutch to English, just because this involves the switching of heads. The alternative lexical form for the same adverb, (18), is required for translation from Dutch to German, where there is no headswitching for this example.

$$\text{a.}\ \begin{bmatrix} \text{REL} & \text{‘toevallig’} \\ \text{ARG1} & \begin{bmatrix} \text{REL} & \text{‘graag’} \\ \text{ARG1} & \begin{bmatrix} \text{REL} & \text{‘zwemmen’} \\ \text{ARG1} & \text{‘Jan’} \end{bmatrix} \end{bmatrix} \end{bmatrix} \quad \text{b.}\ \begin{bmatrix} \text{REL} & \text{‘graag’} \\ \text{ARG1} & \begin{bmatrix} \text{REL} & \text{‘toevallig’} \\ \text{ARG1} & \begin{bmatrix} \text{REL} & \text{‘zwemmen’} \\ \text{ARG1} & \text{‘Jan’} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 1: S-structures derived using Restriction

1.3 Other Solutions

1.3.1 Restriction

Given cases with adverbial modifiers such as (17), Kaplan & Wedekind (1993) attempt to solve them by the introduction of the notion of *restriction*, which seeks to overcome problems in mapping between flat syntactic f-structures to hierarchical semantic ones. The intuition is that in such cases semantic units correspond to subsets of functional information, and restricting the f-structure (in other words, removing *graag* and *toevallig* in turn from the adjunct set in (26)) enables (26) to be associated with the alternative s-structures in Figure 1. Kaplan & Wedekind (*op cit.*, p.199) define the restriction of an f-structure by an element of an element's set-value, as in (28):

(28) If f is an f-structure and a is an attribute:

$$f \setminus \langle a\ g \rangle = \begin{cases} f \setminus a & \text{if } (f\ a) - \{g\} = \emptyset \\ f \setminus a \bigcup \{\langle a,\ (f\ a) - \{g\} \rangle\} & \text{otherwise} \end{cases}$$

That is, the restriction of an f-structure f by a particular member of an attribute a's set-value is the f-structure which results from deleting that member of the set, and also the attribute itself if an empty set results. We can illustrate how restriction works in (29), taking (26) as input:

(29)

$$f = \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{‘Jan’} \\ \text{NUM} & \text{SG} \end{bmatrix} \\ \text{PRED} & \text{‘zwemmen}\langle(\uparrow\text{SUBJ})\rangle\text{’} \\ \text{TENSE} & \text{PRES} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{‘toevallig’} \end{bmatrix}, \begin{bmatrix} \text{PRED} & \text{‘graag’} \end{bmatrix} \right\} \end{bmatrix} \qquad g = \begin{bmatrix} \text{PRED} & \text{‘toevallig’} \end{bmatrix}$$

$$f \setminus \langle g \rangle = \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{‘Jan’} \\ \text{NUM} & \text{SG} \end{bmatrix} \\ \text{PRED} & \text{‘zwemmen}\langle(\uparrow\text{SUBJ})\rangle\text{’} \\ \text{TENSE} & \text{PRES} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{‘graag’} \end{bmatrix} \right\} \end{bmatrix}$$

We are now in a position to describe the semantic correspondences for sentences containing adverbs, such as (17a), using the restriction operator. Let f be the f-structure (26), g the f-structure corresponding to *graag* and t the f-structure corresponding to *toevallig*. We give in (30) the constraints necessary to map (26) into the s-structure (b) in Figure 1, where *graag* has wide scope:

(30) a. $(\sigma f\ REL) = (\sigma g\ REL)$

b. $(\sigma f\ ARG1) = \sigma[f \setminus \langle ADJUNCT\ g\rangle]$

c. $(\sigma[f \setminus \langle ADJUNCT\ g\rangle]\ REL) = (\sigma t\ REL)$

d. $(\sigma[f \setminus \langle ADJUNCT\ g\rangle]\ ARG1)$
$= \sigma[f \setminus \langle ADJUNCT\ g\ t\rangle]$
$= \sigma[f \setminus ADJUNCT]$

(30a,b) describe the outermost REL and ARG1 configuration in Figure 1b, and (30c,d) describe the next level of embedding at s-structure. These constraints allow f-structure subsumption relations to be mapped into the desired hierarchical s-structures. However, we note that the number of such constraints will grow in proportion with the size of the set of adjuncts. Kaplan & Wedekind (*op cit.*, p.200) give a rule which generates codescription constraints, as in (31):

(31) For *f* an f-structure, $g \in (f\ ADJUNCT)$, and *g* a sentence adverb,
$\sigma f = \sigma g$, and
$(\sigma f\ ARG1) = \sigma[f \setminus \langle ADJUNCT\ g\rangle]$

(31) allows each element to be selected non-deterministically from an adjunct set to contribute to the relation for the s-structure of the enclosing f-structure. Furthermore, the s-structure corresponding to the f-structure minus the selected member becomes the ARG1 of that relation. Given that the *restriction* operation applies non-deterministically to all members of a set, we get, as here, a correct and some incorrect structures: with respect to sentence (17), s-structure Figure 1a is correct whilst Figure 1b is incorrect. Although this may be seen as an improvement on the codescription approach, where the production of the correct f-structure could not be guaranteed, it nevertheless leaves something to be desired in that human intervention is necessary to manually select the optimal structure from the (possibly large) set of candidate solutions.

1.3.2 Linear Logic

Some cases of headswitching, notably adjuncts and embedded headswitching phenomena, have been tackled using linear logic to formalize transfer rules (Van Genabith *et al.*, 1998). Nevertheless, when it comes to headswitching cases the linear logic approach encounters some problems. Van Genabith *et al.* use the *like* $\longleftrightarrow$ *gerne* example in the sentence pair *Hans schwimmt gerne* $\longleftrightarrow$ *Hans likes swimming*. Given the appropriate f-structures, the source set of meaning constructors in (32) is derived:

(32)

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

The meaning constructor for *like* is (33):

(33)

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

This final term, *like(X,P(X))*, indicates that X is the subject of *like*, and that somewhere in its second argument, X re-enters as the XCOMP SUBJ. The complete set of meaning constructors for *Hans likes swimming* is given in (34):

(34)

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

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

The transfer constructor *gerne* $\multimap_t$ *like* consumes the entire meaning constructor for *gerne* as there are no left-common prefixes in the meaning constructors, and the meaning constructor for *like* in (35) is derived:

(35)

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

Now the first problem with this approach can be seen. On the right-hand side of the $\multimap_t$, we observe that the meaning constructor has rewritten a node F rather than a node $F\ XCOMP$ to match $P(Y)$. Consequently, with the instantiated source meaning constructors and the transfer constructors, the equations in (36) are produced:

(36)

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

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

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

Comparison of the third target meaning constructor with (33) shows that the transfer operation has rewritten a single node $(f_1)_\sigma$ rather than accessing a complement node $(f_1\ XCOMP)_\sigma$ to match against $P(Y)$. Van Genabith *et al.* then give an example of embedded headswitching involving the *like* $\longleftrightarrow$ *gerne* case, and unsurprisingly the same fault is uncovered again.

Transfer should deliver exactly the set of meaning constructors as would be obtained by independent analysis of the target string. If it does not, target language generation from underspecified sets of target meaning constructors will not produce the required output, and the overall translation obtained will be wrong. Van Genabith *et al.* propose to rectify the problem by 'pushing down' the predicate-argument nucleus of verbs one (or more, as appropriate) levels, via functional uncertainty over XCOMP. Hence the transfer constructor *schwimmen* $\multimap_t$ *swim* gets amended to (37):

(37) *schwimmen* $\multimap_t$ *swim*:
$\forall F, X\ (F_\sigma \rightsquigarrow schwimmen(X) \multimap_t (F\ XCOMP{*})_\sigma \rightsquigarrow swim(X))$

Furthermore, the transfer constructor *gerne* $\multimap_t$ *like* is also redefined given the knowledge that the predicate-argument structure and the corresponding semantic projector associated with the translation of the proposition in the scope of the source adjunct is 'pushed down' via functional uncertainty in (37) on the target side. Given these amendments, the set of target meaning constructors produced is identical to those in (33), as required.

However, there are a number of problems with this solution. It is clear that *any* verb plus associated arguments can occur as the complement of *like*. Consequently, the amended transfer constructor in (37) will have repercussions for *every* verbal translation relation. That is, all transfer constructors relating verbs will need to include such an equation, *just in case* it ever occurs as an XCOMP in an infinitival phrase of another verb. Furthermore, the addition of how to translate verbs such as *schwimmen* $\longleftrightarrow$ *swim* in the event of *swim* appearing as the complement of *like* has nothing to do with the translation relation in question at all: the translation of *schwimmen* as *swim* is a case of simple transfer. Information about *swim* as an XCOMP has nothing to do with *swim*, or *schwimmen*, but is an artefact of the *like* $\longleftrightarrow$ *gerne* case. Consequently it should be removed from the context of *schwimmen* $\longleftrightarrow$ *swim* and relocated in its proper place. If the approach is subsequently unable to deal with headswitching examples, then so be it, but at least the basic translation relations are kept intact and untainted by *ad hoc* information which does not belong there.

2 LFG-DOT: a new Model of Translation

While LFG's τ equations are in the main able to link exactly those source-target elements which are translations of each other, leading to elegant translation models such as that of Kaplan *et al.* (1989), we have described a number of cases, in particular embedded headswitching examples, where this machinery, and others, are unable to cope.

Way (2001) proposes the use of LFG-DOP (Bod & Kaplan, 1998) as the basis for an innovative MT system. LFG-DOP combines the syntactic representations of LFG with the statistical language modelling of DOP (Bod, 1998) to create a new, more powerful hybrid model of language processing. LFG-DOP representations consist of LFG $\langle c, f \rangle$ pairs with a mapping ϕ between them. The *Root* and *Frontier* decomposition operations of DOP operate on CF-PSG trees only, so these operations are adapted in LFG-DOP using the notion of ϕ-accessibility to stipulate exactly which c-structure nodes are linked to which f-structure fragments, thereby maintaining the fundamentals of c- and f-structure correspondence. A third, new decomposition operation, *Discard*, is introduced in LFG-DOP by which generalized fragments produced by *Root* and *Frontier* are created by freely deleting any combination of attribute-value pairs from an f-structure except those that are ϕ-linked to some remaining c-structure node, or that are governed by the local predicate. Fragments are combined together in two stages: c-structures are combined by leftmost substitution, as in DOP, subject to the matching of their nodes. F-structures corresponding to these nodes are then recursively unified, and the resulting f-structures are subjected to the grammaticality checks of LFG. LFG-DOP probability models are based on relative frequency: Bod & Kaplan (1998) give different possible definitions of competition sets from which sample derivations are chosen.

Way (2001) presents four models of translation which use LFG-DOP as their language models, but which differ with respect to how translations are obtained. We shall present these briefly here, and comment on their ability to handle more complex headswitching examples such as (7).

2.1 LFG-DOT1: Translation via τ

Given a source language LFG-DOP treebank, the model builds a target f-structure f' from a source c-structure c and f-structure *f*, the mapping between them LFG-DOP-ϕ, and the LFG translation equations τ. From this target f-structure f', a target string is generated via a target language LFG-DOP model, as in (38):

(38)
$$\begin{tikzcd} c \arrow[r, "\text{LFG-DOP-}\phi"] & f \arrow[d, "\tau"] \\ c' & f' \arrow[l, "\text{LFG-DOP-}\phi'"] \end{tikzcd}$$

While LFG-DOT1 contains source and target LFG-DOP language models, thereby adding robustness to LFG via the *Discard* operation, it maintains the use of τ equations to drive the translation component. Unsurprisingly, therefore, the problems of LFG-MT when confronted with certain headswitching data are maintained in LFG-DOT1, so we do not consider this model further here.

2.2 LFG-DOT2: Translation via τ and γ

LFG-DOT1 relates languages just at the level of f-structure (via τ), and so fails in the same way as LFG-MT with the headswitching data presented in section 1. Way (2001) demonstrates that the DOT2 system of translation (Poutsma, 2000) is able to handle embedded headswitching cases correctly.[1] Accordingly, the γ relation, which links source and target subtree fragments in DOT, is introduced into the LFG-DOT2 translation model. This requires integrated bilingual LFG-DOP corpora, where each node n in a source c-structure tree c is related both to its corresponding f-structure fragment *f* (via LFG-DOP-ϕ) and its corresponding c-structure node n' in a target c-structure tree c' (via γ). In addition, each f-structure fragment s in a source f-structure *f* is related to its corresponding language fragment s' in a target f-structure f', via τ, as shown in (39):

(39)
$$\begin{tikzcd} c \arrow[r, "\text{LFG-DOP-}\phi"] \arrow[d, "\gamma"'] & f \arrow[d, "\tau"] \\ c' & f' \arrow[l, "\text{LFG-DOP-}\phi'"] \end{tikzcd}$$

That is, the translation component consists of an integration of the γ probabilities with the τ mapping. Way (2001) describes how these two translation sources might best be combined, but this need not concern us here. Ultimately, despite the fact that LFG-DOT2 is a richer model than LFG-DOT1, it too is rejected as it continues to maintain τ equations which are incapable of ensuring that the correct translation is obtained in all cases of headswitching.

The presence of the f-structure information is required in order to allow *Discard* to run and thereby make LFG-DOT more robust than LFG-MT. However, *Discard* can operate whether the f-structures are linked via τ or not, so it would appear that the τ operation itself is not needed.[2] The next two LFG-DOT models, therefore, omit the τ operation and use the γ relation to produce translations.

[1] There are two remaining caveats: (i) in dealing with ill-formed translation pairs such as ⟨*John swim*, *Jan zwemmen*⟩, DOT2 considers such pairs as 'grammatical with respect to the corpus'. In LFG-DOT, such pairs can be handled only by removing certain atrribute-value pairs in the respective f-structures via *Discard* in order to permit them to be unified. Such derivations are considered 'ungrammatical', in line with our intuitive notion of well-formedness; (ii) DOT2 cannot handle such cases fully compositionally. With respect to this latter point, we shall see that LFG-DOT3 suffers in the same way, but LFG-DOT4 avoids the problem of limited compositionality.

[2] We leave for future work the question as to whether this approach is fruitful for languages which differ significantly at the level of surface structure, e.g. English and Warlpiri. In such cases, perhaps an LFG-DOT1 or LFG-DOT2 model may be better to relate translational equivalents at the level of f-structure rather than c-structure.

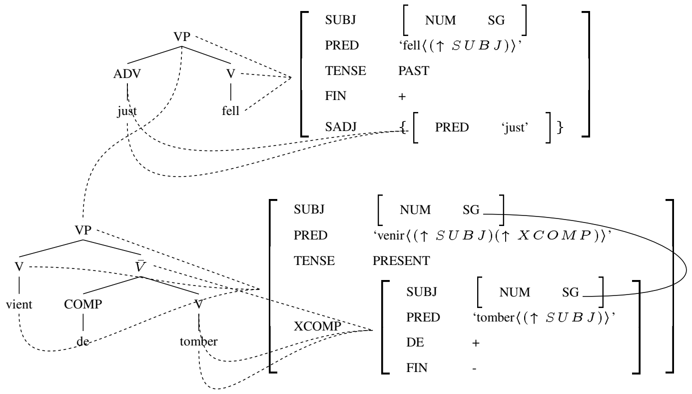


Figure 2: The *just* $\longleftrightarrow$ *venir de* case in LFG-DOT3

2.3 LFG-DOT Model 3: Translation via γ with Monolingual Filtering

The LFG-DOT3 translation model contains the DOT2 γ links between source and target c-structures, but with additional syntactic functional constraints which prevent the formation of certain ungrammatical structures (cf. note 1), thereby enabling correct translations to be output with appropriate probabilities. The f-structure information can be seen, therefore, as useful for monolingual disambiguation in both source and target sides. Ill-formed or unknown input is still processable by running *Discard* over the set of linked source and target $\langle c,$ LFG-DOP-$\phi, f\rangle$ fragments. The LFG-DOT3 architecture is shown in (40): [3]

(40)
$$\begin{array}{ccc} & \text{LFG-DOP-}\phi & \\ c & \longrightarrow & f \\ \gamma\ \downarrow & & \\ c' & \longrightarrow & f' \\ & \text{LFG-DOP-}\phi' & \end{array}$$

In order to exemplify how headswitching is dealt with in LFG-DOT3, let us consider the *just* $\longleftrightarrow$ *venir de* case (1). In terms of LFG-DOT3, the translation relation is shown in Figure 2. The γ link between semantically equivalent elements in the source and target c-structures can be seen on the VP nodes. *fell* is not considered to be semantically equivalent to *tomber* owing to their different FIN(ite) values, added to the fact that *fell* has a TENSE value whilst *tomber* does not. Hence this translation fragment can only be reused by substituting this pair with associated singular NP subjects at the appropriate nodes in an S-linked fragment. In this respect, as with DOT2 (and LFG-DOT2), this LFG-DOT3 model continues to suffer from limited compositionality. We address this concern further in section 2.4 dealing with the LFG-DOT4 model, which has an extra level of processing called 'Extended Transfer'. In all other respects, LFG-DOT3 and LFG-DOT4 are the same models, so while we end up rejecting LFG-DOT3 in favour of LFG-DOT4, much of the ensuing discussion is relevant to our final choice of model, LFG-DOT4.

2.3.1 LFG-DOT3 and Embedded Headswitching

Having shown how LFG-DOT3 copes with a headswitching problem, we shall now investigate whether LFG-DOT3 can handle translation examples which LFG-MT finds problematic. We showed that in example (7), the default τ equations in the entry for *think* (10) clash with those on the structural rule for ADVP (5).

[3]Way (2001) proves that LFG-DOT2 and LFG-DOT3 are different models by showing that while τ links can be inferred from ϕ and γ links in the general case, this is not possible when more complex translation data is examined.

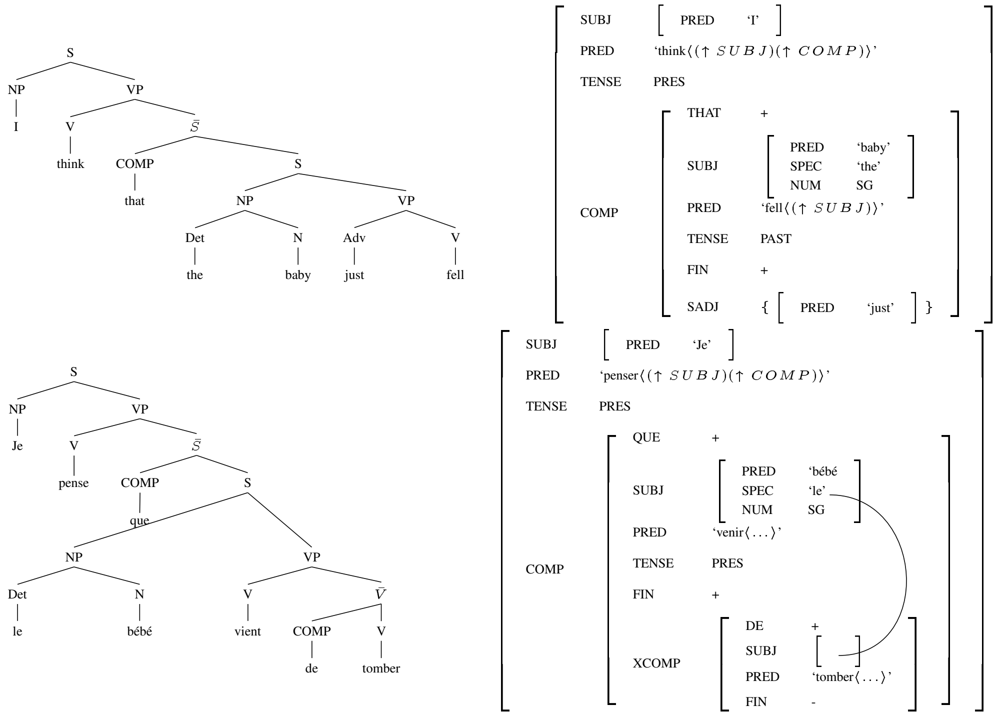
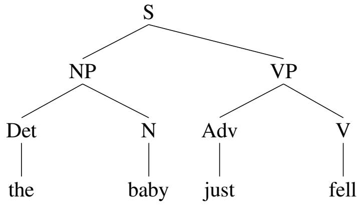


Figure 3: LFG-DOT3 representation for the embedded headswitching case *I think that the baby just fell* ⟷*Je pense que le bébé vient de tomber*

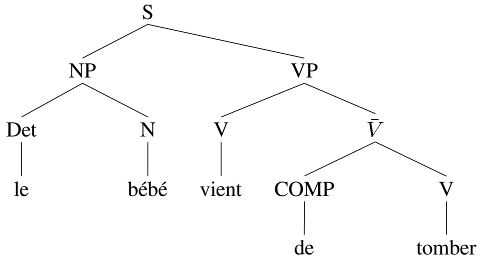
The LFG-DOT3 representations for the full trees in (7) are shown in Figure 3.[4] Source and target trees are linked at the topmost S, NP, V and VP levels, as well as at $\bar{S}$, COMP and embedded S levels. Given that each source fragment will be linked to its target counterpart with the same label, each ⟨*source, target*⟩ linked pair can be deleted to make the new linked fragment pair in (41):

[4]In this and some subsequent examples, ϕ and γ links are omitted for reasons of clarity.

(41)



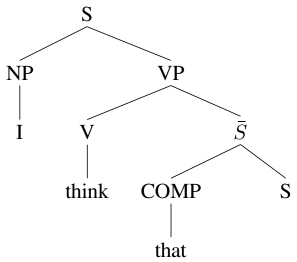
$$\left[\begin{array}{ll} \text{SUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{`baby'} \\ \text{SPEC} & \text{`the'} \\ \text{NUM} & \text{SG} \end{array}\right] \\ \text{PRED} & \text{`fell}\langle(\uparrow SUBJ)\rangle\text{'} \\ \text{TENSE} & \text{PAST} \\ \text{FIN} & + \\ \text{SADJ} & \left\{ \left[\begin{array}{ll} \text{PRED} & \text{`just'} \end{array}\right] \right\} \end{array}\right]$$



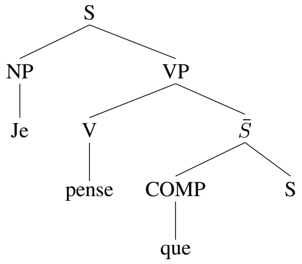
$$\left[\begin{array}{ll} \text{SUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{`bébé'} \\ \text{SPEC} & \text{`le'} \\ \text{NUM} & \text{SG} \end{array}\right] \\ \text{PRED} & \text{`venir}\langle(\uparrow SUBJ)(\uparrow XCOMP)\rangle\text{'} \\ \text{TENSE} & \text{PRES} \\ \text{FIN} & + \\ \text{XCOMP} & \left[\begin{array}{ll} \text{DE} & + \\ \text{SUBJ} & [\ ] \\ \text{PRED} & \text{`tomber}\langle(\uparrow SUBJ)\rangle\text{'} \\ \text{FIN} & - \end{array}\right] \end{array}\right]$$

The trees in (41) are linked at S, NP, VP, DET and N levels. Once these are deleted, the remaining fragments are linked at embedded VP level, exactly as in Figure 2. That is, embedded headswitching cases in LFG-DOT3 are dealt with in exactly the same manner as non-embedded headswitching cases. Given that fragments such as Figure 2 and (41) exist, such complex cases can also be dealt with compositionally, as (42) illustrates:

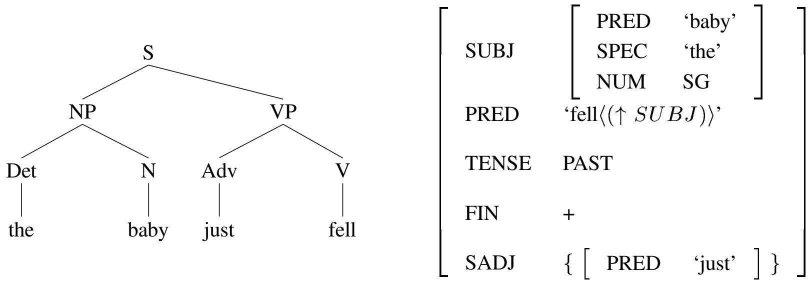
(42)

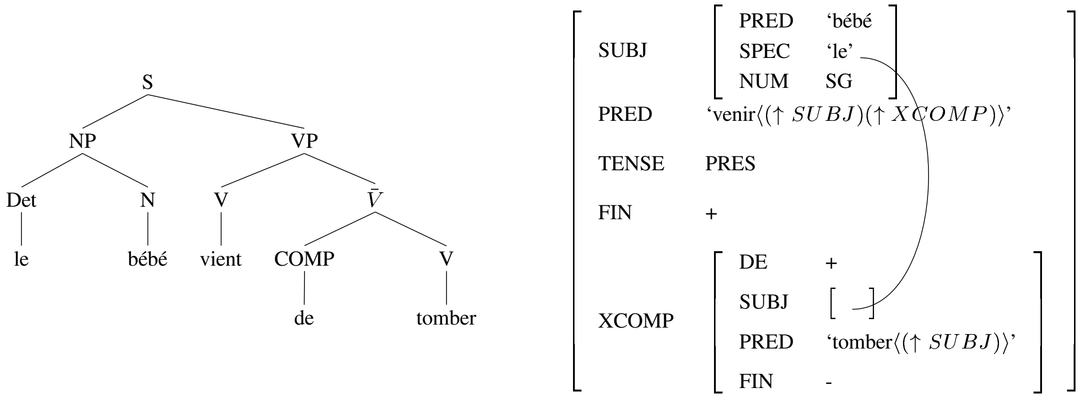


$$\left[\begin{array}{ll} \text{SUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{`I'} \end{array}\right] \\ \text{PRED} & \text{`think}\langle(\uparrow SUBJ)(\uparrow COMP)\rangle\text{'} \\ \text{TENSE} & \text{PRES} \\ \text{COMP} & \left[\begin{array}{ll} \text{THAT} & + \end{array}\right] \end{array}\right]$$



$$\left[\begin{array}{ll} \text{SUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{`Je'} \end{array}\right] \\ \text{PRED} & \text{`penser}\langle(\uparrow SUBJ)(\uparrow COMP)\rangle\text{'} \\ \text{TENSE} & \text{PRES} \\ \text{COMP} & \left[\begin{array}{ll} \text{QUE} & + \end{array}\right] \end{array}\right]$$



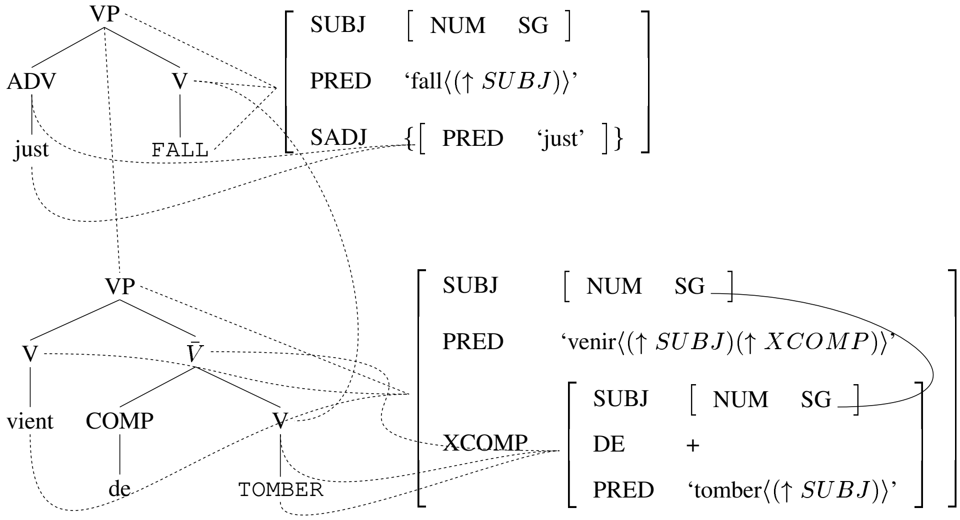


That is, the lower linked source and target sentence pair is substituted into the source and target S-nodes in the upper trees. At the same time, their f-structures are merged with the COMP f-structures of the source and target f-structures respectively. This is, of course, just one possible derivation of this translation. Others will be produced in the usual manner and their probabilities summed in order to derive probabilities for the translation as a whole with respect to the corpus.

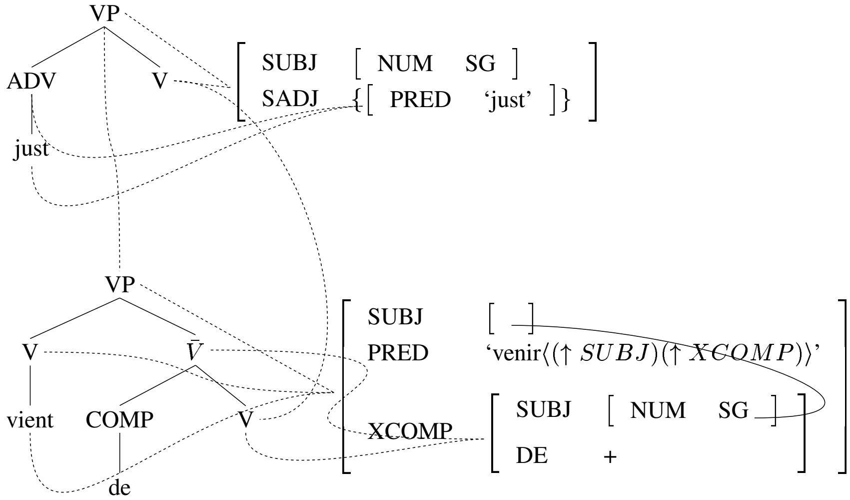
2.4 LFG-DOT4: Translation via γ and 'Extended Transfer'

In the previous section, we observed that the outstanding problem with Model 3 is its retention of the DOT2 problem of limited compositionality. Returning to the *just* $\longleftrightarrow$*venir de* headswitching case in Figure 2, we would like to be able to 'relax' some of the constraints in order to map $\langle fell, tomber\rangle$ to make these linked fragments more general, and hence more useful. In so doing, we would remove this problem of limited compositionality.

In LFG-DOT4, the basic translation relation is expressed by γ, as with LFG-DOT3. In LFG-DOT4, however, there is a second application of *Discard*, by which 'lemmatized' forms are arrived at on which 'extended transfer' can be performed. *Discard* relaxes constraints in order to produce a set of generalized fragments with the potential to deal with ill-formed or unknown input. Once the `TENSE` and `FIN` features have been relaxed on the lowest verbs in both fragments in Figure 2, they can be regarded as translationally equivalent. Given this, $\langle fell, tomber\rangle$ are linked and lemmatized, as in (43):

(43) 

Now that ⟨FALL, TOMBER⟩ are linked, they can be deleted to produce the generalized form of the translation relation, namely (44):

(44) 

If fragment pairs such as (44) prove subsequently to be of use in combining with other fragments, any resultant translation will be marked as ungrammatical with respect to the corpus, given that *Discard* was used in its derivation. Nevertheless, even if we restrict the impact of *Discard* on the probability space (cf. Way, 1999, 2001; Bod, 2000), in order to ensure that translations obtained via *Root* and *Frontier* are preferred over those derived via *Discard*, such translations will receive *some* probability, whereas the semi-compositional variants from which they were derived may not be able to produce *any* translation.

3 Contributions and Further Work

We described the two ways in which the original LFG model of translation (Kaplan *et al.*, 1989) attempted to cope with headswitching data. We summarized previous accounts (Sadler *et al.*, 1989, 1990) which showed that LFG-MT cannot cope with certain cases of headswitching. Other attempts at solving these cases using restriction (Kaplan & Wedekind, 1993) and linear logic (Van Genabith *et al.*, 1998) introduce further problems.

We then presented a number of new hybrid models of translation based on LFG-DOP. The first, LFG-DOT1, uses LFG-DOP for the source and target language models, but imports the τ equations from LFG-MT as the translation relation. Accordingly, therefore, it fails to cope with embedded headswitching and other complex data in the same way as the original LFG-MT model. Given this, we augmented LFG-DOT1 with the γ function from DOT2 to give an improved model of translation. Nevertheless, given that τ equations fail to derive the correct translation in all cases, subsequent LFG-DOT models omit the τ function and rely wholly on γ to express the translation relation.

LFG-DOT3 uses f-structure information purely for monolingual filtering. The presence of this functional information prevents the formation of certain ill-formed structures which can be produced in DOT2. LFG-DOT3, therefore, has a notion of grammaticality which is missing from DOT2. Importantly, this can be used to guide the probability models in the manner required. We showed that LFG-DOT3 copes with cases of embedded headswitching in exactly the same way as non-embedded headswitching examples. However, like DOT2, it suffers from the problem of limited compositionality, so that in some cases the minimal statement of the translation relation is impossible.

LFG-DOT4 adds an 'Extended Transfer' phase to LFG-DOT3 by producing lemmatized forms using a second application of *Discard*. This extension overcomes the problem of limited compositionality, enabling the statement of the translation relation in an intuitive, concise fashion.

More generally, we have shown:

1. that LFG-DOP can serve as a model of translation;
2. that contrary to previous perception, basic statements of translation relations can be stated at the level of c-structure, rather than at 'deeper' levels of linguistic analysis, provided that these trees are accompanied by syntactic (f-structure) information which acts as a monolingual filter on the structures produced.

As to future work, the models presented here need to be tested more thoroughly on large-scale LFG-DOT corpora, and the work of Frank *et al*. (2001, this volume) on producing such resources automatically seems promising in this regard. The work described here and in (Way, 2001) uses as its evaluation metric the ability to cope with 'hard' translation cases such as embedded headswitching. Nevertheless, as well as examining such complex translation phenomena, we need to investigate how well our models deal with simpler translation data, such as *Fido barks* sentences. Different probability models will also be evaluated (cf. Bonnema *et al.*, 2000), as will the possibility of pruning the search space, by cutting down the number of fragments produced (cf. Sima'an, 1999) in order to improve the efficiency of the models proposed.

References

[1] Bod, R. (1998): *Beyond Grammar: An Experience-Based Theory of Language*, CSLI Publications, Stanford, California.

[2] Bod, R. (2000): 'An Empirical Evaluation of LFG-DOP', in COLING: *Proceedings of the 19th International Conference on Computational Linguistics*, Saarbrücken, Nancy and Luxembourg.

[3] Bod, R. & R. Kaplan (1998): "A Probabilistic Corpus-Driven Model for Lexical-Functional Analysis", in COLING: Proceedings of the 17th International Conference on Computational Linguistics & 36th Conference of the Association for Computational Linguistics, Montreal, Canada, 1:145–151.

[4] Bonnema, R., P. Buying and R. Scha (2000): 'Parse Tree Probability in Data-Oriented Parsing', in *Proceedings of PACLING*, Mexico City.

[5] Frank, A., J. van Genabith and A. Way (2001): 'Treebank vs. X-BAR based Automatic F-Structure Annotation', in *Proceedings of 6th International Conference on Lexical Functional Grammar (LFG-2001)*, Hong Kong.

[6] Kaplan, R., K. Netter, J. Wedekind & A. Zaenen (1989): "Translation by Structural Correspondences", in Fourth Conference of the European Chapter of the Association for Computational Linguistics, Manchester, pp.272–281.

[7] Kaplan, R. and J. Wedekind (1993): 'Restriction and Correspondence-based Translation', in *Sixth Conference of the European Chapter of the Association for Computational Linguistics*, Utrecht, The Netherlands, pp.193–202.

[8] Poutsma, A. (1998): "Data-Oriented Translation", in Ninth Conference of Computational Linguistics In the Netherlands, Leuven, Belgium.

[9] Poutsma, A. (2000): *Data-Oriented Translation: Using the Data-Oriented Parsing framework for Machine Translation*, MSc thesis, University of Amsterdam, The Netherlands.

[10] Sadler, L., I. Crookston, D. Arnold and A. Way (1990) 'LFG and Translation', in *Third Conference on Theoretical and Methodological Issues in MT*, University of Texas, Austin, pp.121–130.

[11] Sadler, L., I. Crookston and A. Way (1989) 'Co-description, projection, and 'difficult' translation', Working Papers in Language Processing 8, Department of Language and Linguistics, University of Essex, Colchester.

[12] Sadler, L., and H. Thompson (1991): "Structural Non-correspondence in Translation", in *Fifth European Conference on Computational Linguistics*, Berlin, Germany, pp.293–298.

[13] Sima'an, K. (1999): *Learning Efficient Disambiguation*, PhD Thesis, University of Utrecht, The Netherlands.

[14] Van Genabith, J., A. Frank and M. Dorna (1998): 'Transfer Constructors', in *Proceedings of LFG-98*, Brisbane, Australia, pp.190–205.

[15] Way, A. (1999): "A hybrid architechture for robust MT using LFG-DOP", *Journal of Experimental and Theoretical Artificial Intelligence* **11**:441–471.

[16] Way, A. (2001): *LFG-DOT: A Hybrid Architechture for Robust MT*, PhD thesis, the University of Essex, Colchester, UK.