On the Direction of Fibring Feature Logics with Concatenation Logics

N. Francez

Information Systems (INS)

INS-R9706 September 30, 1997

Report INS-R9706
ISSN 1386-3681

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

On the Direction of Fibring Feature Logics with Concatenation Logics

Nissim Francez
CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
and *OTS, Utrecht university*

On leave from the computer science dept., Technion-IIL, Haifa, Israel

ABSTRACT
A dual-fibring of a feature-logic and a concatenation-logic is proposed, in which syntactic categorial types "live in" feature terms, in contrast to current fibring, in which feature-terms "live in" types. The dual-fibring contains also arrow-introduction rules for hypothetical reasoning. It is used to explain some "privileged features" in HPSG and their non-unification manipulation.
1991 Computing Reviews Classification System: F.4.1, F.4.2, J.5
Keywords and Phrases: Feature-Logic, Lambek-calculus, unification, fibring, categorial-grammar, natural language

Note: Paper presented at the *Logical aspects of computational linguistics (LACL97)* conference, Nancy, France, 22-24/9/97.
Work carried out under project INS.3, PDP.

1. Introduction

Recent advances in the 'parsing as deduction' paradigm are advocating the combination of two logics, a "concatenation-logic" and a "feature-logic", as the adequate means of implementing this paradigm, in view of the advent of 'unification-based grammar-formalisms'. Two recent examples of this approach are [3] and [4]. The combination of the logics is called 'fibring' in [3], and we retain this name generically in this paper. Both papers consider the Lambek logic $\mathcal{L}$ [5] as the concatenation-logic, but differ in several respects regarding the way $\mathcal{L}$ should be fibred with a feature-logic:
1. Should the operators of $\mathcal{L}$ be applied directly to atomic elements of the feature-logic, or should the latter be embedded within atomic $\mathcal{L}$-types (as arguments)?
2. Should the feature-logic be based on unification or on subsumption?
3. Should the restrictions imposed by the feature-logic be localized (to derivation-steps), or kept globally?
However, one common implicit assumption in both approaches is, that in the fibring (in spite of its symmetric definition w.r.t. the fibred logics), the feature-logic should be embedded within the concatenation-logic, no matter how the above differences are to be resolved. Thus, the focus is on extended-types, in such a way that feature information "lives-within" types. In particular, this view induces a role of feature information as a *partitioning* of types into subtypes, or a refinement of the type-structure. For example, the basic type *np* (denoting noun-phrases) can be partitioned into[1] $np(num : sg)$ and $np(num : pl)$, denoting, respectively, singular and plural noun-phrases.

In this paper, a dual-fibring is proposed, that embeds the concatenation-logic within the feature-logic, by which categorial information is made to "live-within" feature-terms. It seems that this kind of dual-fibring fits better the practice of grammar writers, in particular in the HPSG framework [7], and explains the role of "privileged features" such as 'cat', 'subcat', 'dtrs' etc., a role stipulated in HPSG without any theoretical justification, except for the clear linguistic need and adequacy of its use. Often, such privileged features have values the manipulation of which exceeds unification, again, without an explicit justification. The latter phenomenon is mentioned in [8] (p. 295) as a general characteristic of many unification-based grammar-formalisms. The dual-fibring suggested here may constitute a common explanation and justification for many such "deviations" from unification, and explains the privileged features as arising from the interface-rules used in the fibring of two logics, while the deviation from unification in handling the values of these features reflects the proof-rules of the concatenation-logic. This view induces a dual

[1]In the introduction, the notation is supposed to be self-explanatory. It is properly introduced in the body of the paper.

$$\begin{array}{lll} T \longrightarrow & p & \text{(sorts, } p{\in}\mathcal{P}\text{)} \\ & |x & \text{(variables)} \\ & |f : T_1 & \text{(features, } f{\in}\mathcal{F}\text{)} \\ & |T_1{\wedge}T_2 \mid \neg T_1 & \text{(propositional} - \text{logic connectives)} \end{array}$$

Figure 1: The syntax of FL

$\langle\mathcal{A},g,d\rangle{\models}_{FL}p$ iff $d{\in}p^{\mathcal{A}}$
$\langle\mathcal{A},g,d\rangle{\models}_{FL}x$ iff $d = g[x]$
$\langle\mathcal{A},g,d\rangle{\models}_{FL}f : T_1$ iff there exists a d' s.t. $d' = f^{\mathcal{A}}(d)$ and $\langle\mathcal{A},g,d'\rangle{\models}_{FL}T_1$
$\langle\mathcal{A},g,d\rangle{\models}_{FL}T_1{\wedge}T_2$ iff both $\langle\mathcal{A},g,d\rangle{\models}_{FL}T_1$ and $\langle\mathcal{A},g,d\rangle{\models}_{FL}T_2$
$\langle\mathcal{A},g,d\rangle{\models}_{FL}\neg T_1$ iff $\langle\mathcal{A},g,d\rangle{\not\models}_{FL}T_1$

Figure 2: The semantics of FL

role of features and types; the types are now seen as partitioning feature-structures (or refining them) according to categorial information. For example, the class of feature-structures satisfying (in AVM-notation) $T = [vform : \ finite]$, denoting all entities with a finite verb-form feature, can be partitioned into $T[np \to s]$ and $T[np \to (s \leftarrow np)]$, denoting, respectively intransitive verbs (or object-saturated verb-phrases) and transitive verbs. Here $T[A]$ denotes an encoding of an extended-type (explained below) as a sub-feature-term within a feature-term T. The resulting logic has a lot in common with CUG ([2], [1]). However, it is, according to the classification of [6], a second-generation logic, due to the presence of *residuation-rules* for hypothetical reasoning, while CUG is classified as first-generation system in the absence of such rules. It still remains to be investigated what is the logic arising when neither of the entities "lives-within" the other, applying the fully-recursive definition of fibring as given in [3]. Many simplifications are assumed in the current paper. The uncovering of the fibred-logics structure of "full HPSG" remains an issue for further research.

2. The base-logics

The following definitions are basically extracted from [3].

The feature-term logic

The syntax of FL (over a signature $\langle\mathcal{F},\mathcal{P}\rangle$ of feature-names and node-predicate-names, respectively) is described in Figure 1. Terms of the feature-logic FL are interpreted over *feature-structures* $\mathcal{A} = \langle\mathcal{D}^{\mathcal{A}}, \{f^{\mathcal{A}}|f{\in}\mathcal{F}\}, \{p^{\mathcal{A}}|p{\in}\mathcal{P}\}\rangle$, where: $\mathcal{D}^{\mathcal{A}}$ is a non-empty set of *nodes*, for each $f{\in}\mathcal{F}$, $f^{\mathcal{A}}$ is a partial function on $\mathcal{D}^{\mathcal{A}}$, and for each $p{\in}\mathcal{P}$, $p^{\mathcal{A}}$ is a subset of $\mathcal{D}^{\mathcal{A}}$. The *satisfaction* of an FL term T over a feature structure $\mathcal{A}$ w.r.t. a variable assignment g (taking care of reentrancy), denoted by $\langle\mathcal{A},g,d\rangle{\models}_{FL}T$, is presented in Figure 2. A term T_2 is an FL-consequence of a term T_1, denoted by $T_1{\models}_{FL}T_2$, iff for all $\mathcal{A},g,d$: if $\langle\mathcal{A},g,d\rangle{\models}_{FL}T_1$, then $\langle\mathcal{A},g,d\rangle{\models}_{FL}T$.

The concatenation logic

As for the concatenation logic, we also consider $\mathcal{L}$, Lambek's basic logic, over a set $\mathcal{B}$ of *basic types*. The syntax of $\mathcal{L}$ is given in Figure 3. Terms of $\mathcal{L}$ (types) are interpreted over string structures (models) $(\mathcal{S}, .^{\mathcal{S}})$, consisting of a non-empty set $\mathcal{S}$ ("words"), and an interpretation that assigns to each basic category $b{\in}\mathcal{B}$ a non-empty set $b^{\mathcal{S}} \subseteq \mathcal{S}^+$ (strings over $\mathcal{S}$). The denotations of syntactic types are presented in Figure 4. The deductive calculus is defined over *(declarative) units* $U \rhd A$, where U keeps track of the resource managment over assumptions in proofs. The proof-rules are presented in Figure 5. The reader is referred to [3] for the definition of the fibred logic $\mathcal{L}(FL)$, the fibred structure over which this logic is interpreted, and a complete calculus for deriving valid declarative units.

$$\begin{array}{lll} A \longrightarrow & b & (b \in \mathcal{B},\ \text{basic types}) \\ & |C \rightarrow B & (\text{leftward type}) \\ & |B \leftarrow C & (\text{rightward type}) \end{array}$$

Figure 3: The syntax of $\mathcal{L}$

$$\begin{array}{l} [b]^{\mathcal{S}} = b^{\mathcal{S}} \\ [C \rightarrow B]^{\mathcal{S}} = \{\tau | \forall \tau' \in [C]^{\mathcal{S}} \tau' \tau \in [B]^{\mathcal{S}}\} \\ [B \leftarrow C]^{\mathcal{S}} = \{\tau | \forall \tau' \in [C]^{\mathcal{S}} \tau \tau' \in [B]^{\mathcal{S}}\} \end{array}$$

Figure 4: The denotation of $\mathcal{L}$-types

3. The dual-fibring logic

In this section, we show how to dual-fiber the concatenation-logic $\mathcal{L}$ into the feature-terms logic FL. The dual-fibred logic is referred to as FL($\mathcal{L}$), in analogy with the fibring $\mathcal{L}$(FL) in [3]. The corresponding calculus is also based on fibring satisfiability of feature-terms with validity of types, as in the usual fibrings. For the interface-rules of the dual-fibring, we assume that $\{cat, lsubcat, rsubcat, dtrs, hdtr, cdtr\} \cap \mathcal{F} = \emptyset$, and extend the $\mathcal{F}$-component of the signature of FL($\mathcal{L}$) by these feature-names. The first three features accomodate the directionality of $\mathcal{L}$-types similarly to the way *subcat* and *cat* act[2] in HPSG. Similarly, the last three encode the hierarchical structure of signs (strings), similarly to the use of the 'DTRS' (daughters), head-daughter and complement daughter in HPSG. As we are simplifying a lot here, the special role of heads is not fully reflected. The functor is the head, and the argument - the complement (in the right linear-order imposed by the directionality of the functor). In addition, we assume that $\mathcal{P} \cap \mathcal{B} = \emptyset$, and extend the $\mathcal{P}$-component of the signature with the elements of $\mathcal{B}$ as new sort-names.

For defining the syntax of FL($\mathcal{L}$), we use some auxiliary definitions. Wherever convenient, we use the AVM-notation for feature terms. First, *extended types* are defined by extending the arrow-operators to feature-terms encoding type on top of some other information. Let T be an FL-term and $b \in \mathcal{B}$.

$$EA \longrightarrow T[b] \mid T[EA_1 \rightarrow EA_2] \mid T[EA_2 \leftarrow EA_1].$$

With each extended-type EA we can naturally associate its underlying $\mathcal{L}$-type, "stripping " all its feature information, so to speak, denoted $\tau(EA)$, by letting

$$\tau(EA) = \begin{cases} b & EA = T[b] \\ (\tau(EA_1) \rightarrow \tau(EA_2)) & EA = (T[EA_1 \rightarrow EA_2]) \\ (\tau(EA_2) \leftarrow \tau(EA_1)) & EA = (T[EA_2 \leftarrow EA_1]) \end{cases}$$

We use a *type-extension* $T[EA]$ for denoting the extension of a feature-term T with an extended type EA, defined by $T[EA] =_{\text{df.}} T \sqcup \theta[EA]$, where $\theta[EA]$ is a function encoding[3] an extended-type as an FL($\mathcal{L}$)-term,

[2]We ignore here mutiple complements on the subcat list in HPSG, and consider a "binary version" of it. Also, we ignore here HPSG's multiple valency lists. A full exposition of HPSG as a fibred logic should deal with both issues.

[3]Similarly to the encoding in CUG ([2], [1]).

$$\begin{array}{ll} (\text{ax})\ A \rhd A & \\ (\rightarrow E)\ \frac{U_1 \rhd B,\ \ U_2 \rhd (B \rightarrow A)}{(U_1 U_2) \rhd A} & (\leftarrow E)\ \frac{U_2 \rhd (A \leftarrow B),\ \ U_1 \rhd B}{(U_2 U_1) \rhd A} \\ (\rightarrow I)\ \frac{(BU) \rhd A}{U \rhd (B \rightarrow A)} & (\leftarrow I)\ \frac{(UB) \rhd A}{U \rhd (A \leftarrow B)} \end{array}$$

Figure 5: The type calculus for $\mathcal{L}$

using the new feature-names and new type-sorts in the extended signature, defined as follows:

$$\theta[EA] =_{\text{df.}} \begin{cases} T \sqcup \begin{bmatrix} cat: & b \end{bmatrix} & EA = T[b] \\ T \sqcup \begin{bmatrix} cat: & \begin{bmatrix} cat: & \theta[EA_2] \\ lsubcat: & \theta[EA_1] \end{bmatrix} \end{bmatrix} & EA = (T[EA_1 \to EA_2]) \\ T \sqcup \begin{bmatrix} cat: & \begin{bmatrix} cat: & \theta[EA_2] \\ rsubcat: & \theta[EA_1] \end{bmatrix} \end{bmatrix} & EA = (T[EA_2 \leftarrow EA_1]) \end{cases}$$

By this construction, the categorial information "lives-within" feature-terms as desired, leaving the denotation of type-extended feature-terms to be a feature-structure. The dependence on the embedded categorial type is reflected in the definition of the validity of a declarative-unit as defined below, and in the preservation of this validity by the rules of the FL($\mathcal{L}$)-calculus presented in the sequel. This specific way of encoding types allows feature-information everywhere, not only in conjunction with basic types, or heads. The linguistic motivation for such encodings may be found in [2] and [1]. For example, assuming that $\{num, pers, vform, sform\} \subseteq \mathcal{F}$, $\{sg, 3rd, fin, affirm\} \subseteq \mathcal{P}$ and $\{np, s\} \subseteq \mathcal{B}$, and letting $T_1 = \begin{bmatrix} num: & sg \\ pers: & 3rd \end{bmatrix}$, $T_2 = \begin{bmatrix} sform: & affirm \end{bmatrix}$ and $T_3 = \begin{bmatrix} vform: & fin \end{bmatrix}$, we have:

$$T_3[T_1[np] \to T_2[s]] = \begin{bmatrix} cat: & \begin{bmatrix} cat: & \begin{bmatrix} cat: & s \\ sform: & affirm \end{bmatrix} \\ lsubcat: & \begin{bmatrix} cat: & np \\ pers: & 3rd \\ num: & sg \end{bmatrix} \end{bmatrix} \\ vform: & fin \end{bmatrix}$$

as an FL($\mathcal{L}$)-term. Note that it has feature information at all levels. Under this view, lexical entries assigned to words are extended-types.

Before embarking of the full presentation of FL($\mathcal{L}$), we make a small detour to explain more of HPSG's privileged features. One can make a distinction between two kinds of logics expressing grammars: *recognition-logics* and *parsing-logics*. In a recognition logic, derivability of a suitable term signals membership of a certain string in the language defined by the grammar encoded by the logic. However, no extra information about the member string is present. On the other hand, in a parsing-logic, the derived term expresses (in addition to membership in the language) some syntactic structure attributed to the member string by the grammar, e.g., a phrase-structure. This reflects the known distinction between weak generative power and strong generative power; usually, yhe latter is of more interest in computational linguistics.

As it turns out, the role of HPSG's features like 'DTRS' (daughters), 'HDTR' (head-daughter), 'CDTRS' (complement-daughters) and the like, is an encoding of the information needed for a parsing-logic. In defining a dual-fibring for that purpose, we use $\delta(T[EA], T_1[EA_1], T_2[EA_2])$ to denote the hierarchical embedding of two (type-extended) feature terms T_1, T_2 in (the type-extended) feature-term T, using the daughters features. Thus,

$$\delta(T[EA], T_1[EA_2 \leftarrow EA_1], T_2[EA_1]) =_{\text{df.}} T[EA] \sqcup \begin{bmatrix} dtrs: & \begin{bmatrix} hdtr: & T_1[EA_2 \leftarrow EA_1] \\ cdtr: & T_2[EA_1] \end{bmatrix} \end{bmatrix}$$

To maintain a simpler notation, we focus in the rest of this paper on a dual-fibring variant of FL($\mathcal{L}$) reflecting a recognition-logic only. The extension to a full parsing-logic is not hard. Thus, the syntax of FL($\mathcal{L}$) consists of type-extended feature terms $T[EA]$ as defined above.

We now turn to the denotation of FL($\mathcal{L}$)-terms and declarative-units. Both are interpreted in *dual-fibred* structures $\mathcal{M} = (\mathcal{S}, \mathcal{A}, g, \{R_b\}_{b \in \mathcal{B}})$, where $\mathcal{S}$ is a string-model, $\mathcal{A}$ is a feature-structure, g is a variable-assignment $\mathcal{V} \to \mathcal{D}^{\mathcal{A}}$, and $R_b \subseteq \mathcal{D}^{\mathcal{A}} \times b^{\mathcal{S}}$ is a famlilly (indexed by basic types) of fibring-relations[4], such that whenever $\mathcal{A}, g, d \models cat : b$ (for some $b \in \mathcal{B}$), there exists some string $\tau \in \mathcal{S}^+$ such that $d R_b \tau$.

The denotation $[T[EA]]^{\mathcal{M}}$ of an FL($\mathcal{L}$)-term $T[EA]$ has already been fixed via satisfaction in $\mathcal{A}$. To take the categorial type into account, we associate with every FL($\mathcal{L}$)-term $T[EA]$ a language $L[T[EA]]^{\mathcal{M}}$,

[4]Note the difference in the direction of the relation compared to the fibring-relations in [3].

$$\begin{array}{ll} L[T[b]]^{\mathcal{M}} =_{\text{df.}} & \{\tau \mid \exists d \in \mathcal{D}^{\mathcal{A}}(\mathcal{A}, g, d \models T \wedge cat : b \text{ and } dR_b\tau)\} \\ L[T[EA_1 \rightarrow EA_2]]^{\mathcal{M}} =_{\text{df.}} & \{\tau \mid \ \exists d \in \mathcal{D}^{\mathcal{A}}(A, g, d \models T \wedge cat : cat : EA_2 \wedge cat : lsubcat : EA_1 \\ & \quad \text{and } \forall \tau' \in L[EA_1]^{\mathcal{M}}.\tau'\tau \in L[EA_2]^{\mathcal{M}})\} \\ L[T[EA_2 \leftarrow EA_1]]^{\mathcal{M}} =_{\text{df.}} & \{\tau \mid \exists d \in \mathcal{D}^{\mathcal{A}}(A, g, d \models T \wedge cat : cat : EA_2 \wedge cat : rsubcat : EA_1 \\ & \quad \text{and } \forall \tau' \in L[EA_1]^{\mathcal{M}}.\tau\tau' \in L[EA_2]^{\mathcal{M}})\} \end{array}$$

Figure 6: The language associated with type-extended feature-terms

as presented in Figure 6. Thus, the "privileged" features are related to the denotations of $\mathcal{L}$-types in a fibred-structure. Note that by this definition we have for every FL($\mathcal{L}$)-term $T[EA]$ and $\mathcal{M}$

$$L[T[EA]]^{\mathcal{M}} \subseteq [\tau(T[EA])]^{\mathcal{M}}.$$

As for the denotation of the resources ("structured databases"), these are a natural extension of feature-structures called *multi-rooted structures* (MRSs), described[5] in detail in [9]. Basically, these are sequences of feature-structures with possible sharing (reentrancy) of substructures between elements of a sequence. We use sequences of FL($\mathcal{L}$)-terms (with possibly shared variables) to denote them. The definition of the language associated with a type-extended feature-term is naturally extended by letting

$$L[T_1[EA_1], T_2[EA_2]]^{\mathcal{M}} =_{\text{df.}} L[T_1[EA_1]]^{\mathcal{M}} \cdot L[T_2[EA_2]]^{\mathcal{M}}.$$

namely, the concatenation (in $\mathcal{S}^+$) of the respective languages. Note that by this definition, the inclusion of languages to the denotations of the underlying type is preserved, i.e.,

$$L[T_1[EA_1], T_2[EA_2]]^{\mathcal{M}} \subseteq [\tau(T_1[EA_1], T_2[EA_2])]^{\mathcal{M}}.$$

Finally, a declarative-unit $(U_1, ..., U_m) \rhd T[EA]$ is *valid* iff for all dual-fibred structures $\mathcal{M}$,

$$L[(U_1 ..., U_m)]^{\mathcal{M}} \subseteq L[T[EA]]^{\mathcal{M}}.$$

Following the HPSG convention, we represent also the lexical input words (and the generated concatenations thereof) as the value of yet another (new) feature, *phon*.

For the presentation of the fibred calculus, we use $\mathbf{u}(T_1, T_2)$ to denote that the (satisfiable) FL($\mathcal{L}$)-terms T_1, T_2 are *unifiable*, and $T_1 \sqcup T_2$ for the (satisfiable) outcome of their unification. The form of an axiom now becomes as follows.

$$\frac{T}{T[EA] \rhd T[EA]}$$

meaning that the type-extension of a satisfiable term with any extended type derives itself. The new form of the elimination rule $\leftarrow E$ now becomes:

$$(\leftarrow E) \quad \frac{U_1 \rhd T_1[T_{1.2}[EA_{1.2}] \leftarrow T_{1.1}[EA_{1.1}]],\ U_2 \rhd T_2[EA_{2.1}], \mathbf{u}(T_1, T_{1.2}), \mathbf{u}(T_{1.1}[EA_{1.1}], T_2[EA_{2.1}])}{(U_1, U_2) \rhd \sqcup(T_1, T_{1.2})[EA'_{1.2}]}$$

The primes indicate the feature-terms after the side-effect of unification. The unifiability requirement is a side-condition of the rule, and is placed as assumptions for notational convenience only. This rule reflects a simplified version of the subcategorization-principle, as well as the head-feature principle of HPSG.

In Figure 7 is a sample derivation of a highly simplified representation for Mary loves John, where the only syntactic restriction present is number-agreement between the subject and the verb. To save space, the feature names 'rsubcat' and 'lsubcat' are abbreviated to 'rsub' and 'lsub', respectively.

[5]Only rooted, connected, finite feature-structures are considered in [9], but all definitions extend naturally to the more relaxed definition employed here.

$$\dfrac{\begin{bmatrix} phon: & \text{Mary}\\ cat: & np \\ num: & sg\end{bmatrix} \quad \dfrac{\begin{bmatrix} phon: & \text{loves} \\ cat: & \begin{bmatrix} cat: & \begin{bmatrix} cat: & s \\ lsub: & \begin{bmatrix} cat: & np\\ num: & sg\end{bmatrix}\end{bmatrix} \\ rsub: & \begin{bmatrix} cat: & np\end{bmatrix}\end{bmatrix}\end{bmatrix} \quad \begin{bmatrix} phon: & \text{John} \\ cat: & np \\ num: & sg\end{bmatrix}}{(\leftarrow E)\quad \begin{bmatrix} phon: & \text{loves John}\\ cat: & \begin{bmatrix} cat: & s\\ lsub: & \begin{bmatrix} cat: & np\\ num: & sg\end{bmatrix}\end{bmatrix}\end{bmatrix}}}{(\rightarrow E)\quad \begin{bmatrix} phon: & \text{Mary loves John}\\ cat: & s\end{bmatrix}}$$

Figure 7: A derivation of Mary loves John

$$(\mathrm{Ax})\quad \frac{T}{T[EA] \rhd T[EA]}$$

$$(\leftarrow E)\quad \frac{U_1 \rhd T_1[T_{1.2}[EA_{1.2}] \leftarrow T_{1.1}[EA_{1.1}]],\ U_2 \rhd T_2[EA_{2.1}], \mathbf{u}(T_1,T_{1.2}), \mathbf{u}(T_{1.1}[EA_{1.1}],T_2[EA_{2.1}])}{(U_1,U_2) \rhd \sqcup(T_1,T_{1.2})[EA'_{1.2}]}$$

$$(\rightarrow E)\quad \frac{U_1 \rhd T_1[EA_{1.1}],\ U_2 \rhd T_2[T_{2.1}[EA_{2.1}] \rightarrow T_{2.2}[EA_{2.2}]],\ \mathbf{u}(T_2,T_{2.1}), \mathbf{u}(T_{2.1}[EA_{2.1}],T_1[EA_{1.1}])}{(U_1,U_2) \rhd \sqcup(T_2,T_{2.1})[EA'_{2.1}]}$$

$$(\leftarrow\ I)\quad \frac{(U,T_1[EA_1]) \rhd T_2[EA_2]}{U \rhd T_2[(EA_2 \leftarrow T_1[EA_1])]}$$

$$(\rightarrow\ I)\quad \frac{(T_1[EA_1]U) \rhd T_2[EA_2]}{U, \rhd T_2[(T_1[EA_1] \rightarrow EA_2)]}$$

Figure 8: The FL($\mathcal{L}$) calculus

Next, we turn to the new form of a residuation-rule. The feature percolations for such rules are not explicitly discussed in the literature. The main property of the new form of the residuation-rules is, that feature-information present in the assumption is preserved, to be made use of after the discharge of the assumption. The new form of the $(\leftarrow\ I)$ rule is as follows.

$$(\leftarrow\ I)\quad \frac{(U,T_1[EA_1]) \rhd T_2[EA_2]}{U \rhd T_2[(EA_2 \leftarrow T_1[EA_1])]}$$

To exemplify the particular feature percolation employed by the rule, we consider in the next section an example from Hebrew. The full calculus is presented in Figure 8. The following lemma ensures the soundness of the FL($\mathcal{L}$)-calculus, based on the soundness of the $\mathcal{L}$-calculus.

Lemma: (type-restriction)
If $U_i,\ i=1,n$ are satisfiable, and $\vdash_{\mathrm{FL}(\mathcal{L})}(U_1,...,U_m) \rhd T[EA]$, then $T[EA]$ is satisfiable, and furtheremore, $\vdash_{\mathcal{L}}(\tau(U_1),...,\tau(U_m)) \rhd \tau(T[EA])$.

Proof: By a simple inductive argument on the structure of proofs. Satisfiability is preserved by the proof-rules due to the unifiability tests. The inclusion of the associated languages in the denotations of the underlying types was observed before, and the proof-rule mimic the categorial manipulation of these languages.

4. An application of residuation

First, recall the derivation of John whom Mary likes slept in [3], using the residuation rule ($\leftarrow \ I$) of $\mathcal{L}$. The category assigned to whom is $(np \rightarrow np) \leftarrow (s \leftarrow np)$, which triggers the introduction of an assumption $[np]_1$ (with a null string value), later discharged by the residuation rule. We now show how the corresponding FL($\mathcal{L}$) rule can be used to solve elegantly a generalization of relativizing in English.

In Hebrew, many categories, including NPs, VPs and APs, are marked for gender, being either feminine or masculine. In particular, according to some syntactic analysis[6] of Hebrew, there are two relative pronouns of the two corresponding genders, ota (feminine) and oto (masculine), similarly to German and Russian, for example. One of the agreement rules in Hebrew calls for a gender agreement between a relative pronoun (in a relative clause), and the relativized NP (in addition to subject-verb gender agreement). Note that the first person singular pronoun any (i.e., I) is underspecified for gender, and agrees with both genders. Thus, we have[7]:

(1) hayeled oto ani ohev shar
namely
the boy$_m$ whom$_m$ I love$_m$ sings$_m$,
but
(2) (*) hayalda oto ani ohev shara
namely
(*) the girl$_f$ whom$_m$ I love$_m$ sings$_f$.
Similarly,
(3) hayalda ota ani ohev shara
namely
the girl$_f$ whom$_f$ I love$_m$ sings$_f$,
but
(4) (*) hayeld ota ani ohev shar
namely
(*) the boy$_m$ whom$_f$ I love$_m$ sings$_m$.

Actually, as there are also plural versions of these relative pronouns (otam for masculine-plural and otan for feminine-plural), we end up in four similar, though different, lexical entries. Let us ignore for this example all other featural distinctions as number, person, etc., as well as their agreement rules. One way of enforcing the gender agreement in relativization is to assign the following extended categories to oto and ota, using a gender feature *gen* with atomic values m (for masculine) and f (for feminine). This way, the specific gender expected by each relative pronoun is built-in in its lexical entry.

$$\begin{bmatrix} phon: & \textsf{oto} \\ cat: & \begin{bmatrix} cat: & \begin{bmatrix} cat: & \begin{bmatrix} cat: & np \\ gen: & [1] \end{bmatrix} \\ lsub: & \begin{bmatrix} cat: & np \\ gen & [1] \end{bmatrix} \end{bmatrix} \\ rsub: & \begin{bmatrix} cat: & s \\ rsub: & \begin{bmatrix} cat: & np \\ gen: & [1]m \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} phon: & \textsf{ota} \\ cat: & \begin{bmatrix} cat: & \begin{bmatrix} cat: & \begin{bmatrix} cat: & np \\ gen: & [1] \end{bmatrix} \\ lsub: & \begin{bmatrix} cat: & np \\ gen & [1] \end{bmatrix} \end{bmatrix} \\ rsub: & \begin{bmatrix} cat: & s \\ rsub: & \begin{bmatrix} cat: & np \\ gen: & [1]f \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The generated assumption now carries gender information, that has to be percolated when a residuation-rule is applied (otherwise this information disappeares with the discharged assumption). Figure 9 shows the main part of the derivation corresponding to hayeled oto ani ohev shar. The rest of the dertivation combines with the subject, matching the masculine gender, and then the relativized NP combines with the intransitive verb sings, again with the right gender agreement.

Note that the lexical value associated with the verbs ohev/ohevet right-subcategorizes for a (gender-underspecified) np, being ready to combine with noun-phrases of both genders. The gender of the generated assumption "dictates" the way the derivation proceeds. A similar derivation exists for (3),

[6]Other analyses regard these examples as having a phonologically-empty relative-pronoun, and a dislocated resumptive pronoun.

[7]To simplify, we ignore the fact that Hebrew is written from right to left.

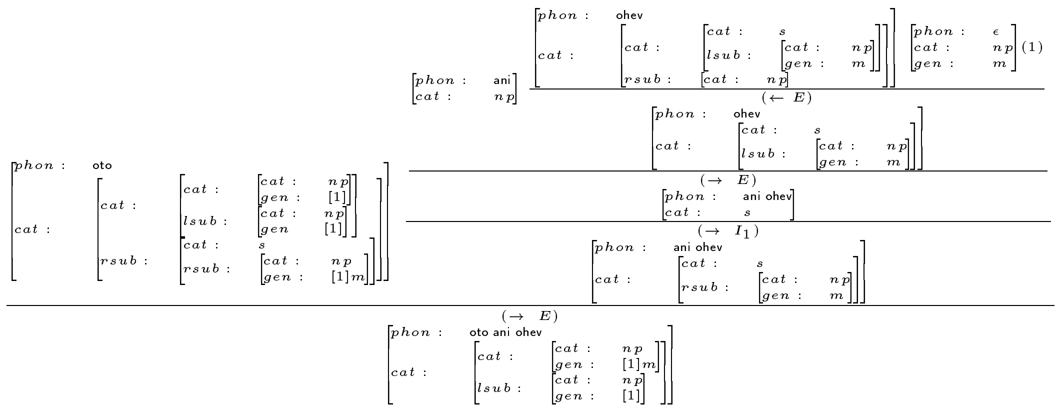

Figure 9: A derivation of oto ani ohev

where the generated assumption has $[gen: \quad f]$ as the extra-categorial initial feature-information, to match that assigned to ota. On the other hand, there is no way to generate assumptions that will cause the wrong gender matching with the relative pronoun, thus blocking (4) and (2). The structure of the above derivation can be viewed as a logical counterpart of a typical use of the SLASH feature in HPSG's treatment of object relativization. Similar use of hypothetical reasoning in other cases of long-distance dependencies suggests itself.

5. Acknowledgments

This research was carried out under a grant B 62-443 from Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO). I wish to thank NWO for their support. I thank Jan van Eijck and Michael Moortgat, as well as the OTS students in a seminar on the categories and features, for various discussions and an oppurtunity to present these ideas while in the making. I thank Gosse Bouma, Mark Johnson and Esther König-Baumer for various comments on earlier drafts. Ranni Nelken spotted an error in the Hebrew example in a previous draft.

References

1. Goose Bouma. Modifiers and specifiers in categorial unification grammar. *Linguistics*, 26:21 – 46, 1988.
2. Gosse Bouma. *Nonmonotonicity and Categorial Unification grammar.* PhD thesis, Groningen university, The Netherlands, 1988.
3. Jochem Dörre, Esther König, and Dov Gabbay. Fibred semantics for feature-based grammar logic. *J. of Logic, Language and Information*, 5, October, 1996.
4. Jochen Dörre and Suresh Manandhar. On constraint-based Lambek calculi. In Patrick Blackburn and Martin de Rijke, editors, *Specifying Syntactic Structures*, Studies in Logic, Language and Information (SiLLI). CSLI, Stanford, CA 94305, to appear. available from http://xxx.lanl.gov/cmp-lg/9508008.
5. Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, 65:154 – 170, 1958.
6. Michael Moortgat. Categorial type logics. In Johan van Benthem and Alice ter Muelen, editors, *Handbook for Logic and Language.* Elsevier, 1997.
7. Carl Pollard and Ivan Sag. *Head-Driven phrase structure grammar.* Chicago university press and CSLI publications, 1994.
8. Stephen G. Pulman. Unification encodings of grammatical notations. *Computational Linguistics*, 22:295 – 328, 1996.
9. Shuly Wintner and Nissim Francez. Parsing with typed feature structures. In *Proceedings of the 4th International workshop on parsing technologies (IWPT)*, pages 273–287, Prague, Czech republic, September, 1995.