Lexical Resource Semantics: From Theory to Implementation

Gerald Penn
University of Toronto

Frank Richter
Universität Tübingen

Proceedings of the 11th International Conference on
Head-Driven Phrase Structure Grammar

Center for Computational Linguistics, Katholieke Universiteit Leuven

Stefan Müller (Editor)

2004

CSLI Publications

pages 423–443

http://csli-publications.stanford.edu/HPSG/2004

Penn, Gerald, & Richter, Frank. 2004. Lexical Resource Semantics: From Theory to Implementation. In Müller, Stefan (Ed.), *Proceedings of the 11th International Conference on Head-Driven Phrase Structure Grammar, Center for Computational Linguistics, Katholieke Universiteit Leuven*, 423–443. Stanford, CA: CSLI Publications.

Abstract

This paper summarizes the architecture of Lexical Resource Semantics (LRS). It demonstrates how to encode the language of two-sorted theory (Ty2; Gallin, 1975) in typed feature logic (TFL), and then presents a formal constraint language that can be used to extend conventional description logics for TFL to make direct reference to Ty2 terms. A reduction of this extension to Constraint Handling Rules (CHR; Frühwirth and Abdennadher, 1997) for the purposes of implementation is also presented.

1 Introduction

Lexical Resource Semantics (LRS) has already been used in analyses of various syntactic and semantic phenomena on paper,[1] but until now it did not have a computational implementation, in part because standard typed feature logic (TFL) is so ill-suited to the job of serving as the formal basis of a computational language for describing semantics. All is far from lost, however; it turns out to be relatively simple to extend a TFL-based description language to incorporate the primitives required, which we believe will have application to computational semantics extending well beyond LRS. Those primitives are also described here.

Implementations of computational semantics can be accomplished in TFL — Minimal Recursion Semantics (MRS; Copestake et al., 2003) stands as one particularly well-known example of this. Even in MRS, however, structure that encodes embedding constraints (the so-called *qeq* constraints) must be represented alongside the basic components of the semantic terms being constructed, and several necessary "bookkeeping" principles to address free variables, acyclicity etc. must either be stated in the grammar alongside the real principles that are the subject of linguistic investigation, or (as is conventional in MRS) relegated to an extragrammatical resolution procedure that exists outside TFL altogether. TFL's own semantic type system also does not provide semantic typing beyond $e \rightarrow t$, and so the richer typing required by all non-trivial theories of semantical form must either be structurally encoded into the object language or forgotten entirely. Indeed, no MRS-based grammar to our knowledge avails itself of any true semantic typing beyond *animate*, *time*, *event* and other $e \rightarrow t$ sorts that are syntactically convenient for the English Resource Grammar.

This is not to say that LRS is merely an alternative to MRS. In some respects, they are simply incomparable. MRS also has no model-theoretic interpretation, serving instead as a sort of front-end for "the real semantics" that is deemed to be

[†]We are greatly indebted to Manfred Sailer for his co-development of LRS, and the insightful feedback he has provided, arising from his ongoing grammar development work with this implementation.

[1]These include: negative concord in Polish, sentential (interrogative) semantics in German, the scope of adjuncts in Dutch, the (past) tense system of Afrikaans, and negation in Welsh. See Richter and Sailer (2004) and the references cited therein.

too impractical for the rapid development of large grammars. Parts of the present proposal are probably better thought of as an alternative to the Constraint Language for Lambda Structures (CLLS; Egg et al., 2001), a constraint language over lambda-term trees with linguistically motivated constraints. CLLS's description language, however, has taken shape around a very orthodox view of the syntax-semantics interface as a set of translation rules that augment phrase structure. In our view, *pace* Egg and Erk (2002), making this suitable to HPSG requires more than using typed feature structures in place of atomic categories. Many semantic principles, just as many syntactic principles, are better expressed as universally quantified constraints, and a semantical description language must provide the primitives necessary to accommodate this. CLLS also takes the very traditional view that semantic composition proceeds entirely through beta-reduction. In CLLS, this view brings a certain amount of explicit overhead into the grammar too, in the form of explicit links between lambdas and bound variables. Like MRS and many other underspecified approaches, however, we have been forced to abandon it in recognition of the abundance of concord, discontinuity and proper naming effects from natural language with which it seems irreconcilable.

Section 2 introduces the semantic intuitions behind LRS and the principles that institutionalize them. Section 3 provides further justification (in brief) through some examples of difficult logical form constraints that they enable us to express. Section 4 then presents a constraint language that directly extends standard models of typed feature logic to incorporate Ty2 terms, and shows how to straightforwardly implement this extension using Constraint Handling Rules (CHR; Frühwirth and Abdennadher, 1997) on top of the TRALE system (Penn, 2004).

2 LRS: Fundamental Principles

Although LRS was originally conceived of as a framework-dependent improvement on Flexible Montague Grammar (Hendriks, 1993) implemented within HPSG, it has moved beyond a reconstruction of the Montagovian tradition within TFL. With its combination of techniques derived from model theoretic semantics in the Montagovian tradition, Logical Form (LF) semantics in the generative tradition (von Stechow, 1993), and underspecified processing in computational semantics, LRS now merges insights from several linguistic traditions into a very expressive but computationally feasible framework for natural language semantics.

The architecture of LRS envisages underspecified processing as mediated on the syntactic side by TFL descriptions, and on the semantic side by expressions from a term description language which comprises the necessary devices for scope underspecification as developed in computational semantics. The guiding assumptions behind LRS are that: (a) all semantic and syntactic idiosyncrasies are lexical (including construction type idiosyncrasies), and (b) there is no non-functional semantic contribution from outside of the lexicon. LRS distinguishes between lexical semantics and compositional semantics. Lexical semantics remains under the

CONTENT attribute. CONTENT values are subject to theories of linking, of semantic selection and of HPSG's traditional BINDING THEORY. Compositional semantics, on the other hand, is located in the value of a new attribute LF of signs, and is thus not visible to syntactic and semantic selection by heads. An interface theory which links certain components of the local content to certain parts of the compositional semantics allows for some amount of interaction, such as the lexical selection of the semantic variables of arguments by syntactic heads.

In what follows, we discuss only compositional semantics.[2] In Section 2.1 we show how to encode the language of two-sorted theory (Ty2; Gallin, 1975) in TFL. This encoding then serves to illuminate the connection between LRS and HPSG in Section 2.2, in which we discuss the constraints which constitute the semantic composition mechanism of LRS. In our computational implementation of LRS (Section 4), the encoding part of the theory disappears entirely and is replaced by providing the terms of Ty2 as first class citizens in the denotation of an appropriately extended description language. The semantic composition mechanism will remain effectively unchanged, however.

2.1 Specification of Ty2

The purpose of this section is to demonstrate how Ty2 can be encoded in a particular version of TFL, Relational Speciate Re-entrant Language (RSRL). Readers not interested in the technical details might skip this section and proceed with Section 2.2.

A specification of Ty2 needs an appropriate signature and a set of constraints which denotes models whose objects correspond to the natural numbers (used as indices of variables and non-logical constants), the types, and the well-formed expressions of Ty2. The signature, Σ_{Ty2}, is shown in Figure 1. It must be part of any signature of a grammar specification in TFL using Ty2 for semantic representations. The sorts, attributes, and relation symbols in Σ_{Ty2} will be explained together with the principles which enforce the well-formedness of the Ty2 expressions in grammar models. They are shown in (1).

(1) a. The NATURAL NUMBERS PRINCIPLE:

$$integer \rightarrow \exists x\, {}^{x}[zero]$$

b. The COMPLEX TERM PRINCIPLES:

$$application \rightarrow \begin{bmatrix} \text{TYPE } \boxed{2} \\ \text{FUNCTOR TYPE } \begin{bmatrix} \text{IN } \boxed{1} \\ \text{OUT } \boxed{2} \end{bmatrix} \\ \text{ARG TYPE } \boxed{1} \end{bmatrix} \qquad abstraction \rightarrow \begin{bmatrix} \text{TYPE } \begin{bmatrix} \text{IN } \boxed{1} \\ \text{OUT } \boxed{2} \end{bmatrix} \\ \text{VAR TYPE } \boxed{1} \\ \text{ARG TYPE } \boxed{2} \end{bmatrix}$$

[2] See (Sailer, 2004) for a discussion and empirical motivation of the architecture of local semantics in LRS.

$$\textit{equation} \rightarrow \begin{bmatrix} \text{TYPE} & \textit{truth} \\ \text{ARG1 TYPE} & \boxed{1} \\ \text{ARG2 TYPE} & \boxed{1} \end{bmatrix} \qquad \textit{negation} \rightarrow \begin{bmatrix} \text{TYPE} & \textit{truth} \\ \text{ARG TYPE} & \textit{truth} \end{bmatrix}$$

$$\textit{l-const} \rightarrow \begin{bmatrix} \text{TYPE} & \textit{truth} \\ \text{ARG1 TYPE} & \textit{truth} \\ \text{ARG2 TYPE} & \textit{truth} \end{bmatrix} \qquad \textit{quantifiers} \rightarrow \begin{bmatrix} \text{TYPE} & \textit{truth} \\ \text{SCOPE TYPE} & \textit{truth} \end{bmatrix}$$

c. The TY2 NON-CYCLICITY PRINCIPLE:[3]

$$\textit{ty2} \rightarrow \forall\boxed{1}\left(\left(\bigvee\left\{[\alpha\ \boxed{1}]\ \middle|\ \alpha \in \mathcal{A}_{Ty2}\right\}\right) \rightarrow \neg\,\texttt{ty2-component}(:, \boxed{1})\right)$$

d. The TY2 FINITENESS PRINCIPLE:

$$\textit{ty2} \rightarrow \exists\boxed{1}\forall\boxed{2}\,(\texttt{ty2-component}(\boxed{2}, :) \rightarrow \texttt{member}(\boxed{2}, \boxed{1}[\textit{chain}]))$$

e. The TY2 IDENTITY PRINCIPLE:

$$\textit{ty2} \rightarrow \forall\boxed{1}\forall\boxed{2}\,(\texttt{copy}(\boxed{1}, \boxed{2}) \rightarrow \boxed{1} = \boxed{2})$$

f. The TY2-COMPONENT PRINCIPLE:

$$\forall\boxed{1}\forall\boxed{2}\left(\begin{array}{l} \texttt{ty2-component}(\boxed{1}, \boxed{2}) \leftrightarrow \\ \left(\begin{array}{l} \boxed{1} = \boxed{2} \vee \\ \bigvee\left\{\exists\boxed{3}\left(\begin{array}{l} \boxed{2}[\alpha\ \boxed{3}] \wedge \\ \texttt{ty2-component}(\boxed{1}, \boxed{3}) \end{array}\right)\ \middle|\ \alpha \in \mathcal{A}_{Ty2}\right\} \end{array}\right) \end{array}\right)$$

g. The COPY PRINCIPLE:

$$\forall\boxed{1}\forall\boxed{2}\left(\begin{array}{l} \texttt{copy}(\boxed{1}, \boxed{2}) \leftrightarrow \\ \left(\begin{array}{l} \bigvee\left\{\boxed{1}[\sigma]\ \wedge\ \boxed{2}[\sigma]\ \middle|\ \sigma \in \mathcal{S}_{Ty2}\right\} \wedge \\ \bigwedge\left\{\forall\boxed{3}\left(\begin{array}{l} \boxed{1}[\alpha\ \boxed{3}]\ \rightarrow \\ \exists\boxed{4}\ (\boxed{2}[\alpha\ \boxed{4}]\ \wedge\ \texttt{copy}(\boxed{3}, \boxed{4})) \end{array}\right)\ \middle|\ \alpha \in \mathcal{A}_{Ty2}\right\} \end{array}\right) \end{array}\right)$$

h. The SUBTERM PRINCIPLE:

$$\forall\boxed{1}\forall\boxed{2}\left(\texttt{subterm}(\boxed{1}, \boxed{2}) \leftrightarrow \left(\begin{array}{l} \boxed{1}[\textit{me}] \wedge\ \boxed{2}[\textit{me}] \wedge \\ \texttt{ty2-component}(\boxed{1}, \boxed{2}) \end{array}\right)\right)$$

The meaningful expressions of Ty2 are simple or complex expressions in the denotation of the sort *me*. Objects in the denotation of *me* have an attribute TYPE, whose value indicates the type of the expression. If it is a simple expression (a *variable* or a non-logical *constant*), it is indexed by a natural number, which is the value of the attribute NUM-INDEX.

The NATURAL NUMBERS PRINCIPLE, (1a), guarantees the correspondence of objects in the denotation of *integer* to the natural numbers. An *integer* configuration in models of Γ_{Ty2} is either a *zero* entity or a *non-zero* entity on which a term consisting of a finite sequence of PRE attributes is defined whose interpretation on the *non-zero* entity yields an entity of sort *zero*. The number of PRE attributes in

[3]The symbol $\mathcal{A}_{Ty2}$ denotes the set of attributes of the signature Σ_{Ty2}. Similarly, $\mathcal{S}_{Ty2}$ in (1g) denotes the set of maximally specific sorts of Σ_{Ty2}.

ty2

- *me* TYPE *type*
  - *variable* NUM-INDEX *integer*
  - *constant* NUM-INDEX *integer*
  - *application* FUNCTOR *me*; ARG *me*
  - *abstraction* VAR *variable*; ARG *me*
  - *equation* ARG1 *me*; ARG2 *me*
  - *negation* ARG *me*
  - *l-const* ARG1 *me*; ARG2 *me*
    - *disjunction*
    - *conjunction*
    - *implication*
    - *bi-implication*
  - *quantifiers* VAR *variable*; SCOPE *me*
    - *universal*
    - *existential*
- *type*
  - *atomic-type*
    - *entity*
    - *truth*
    - *w-index*
  - *complex-type* IN *type*; OUT *type*
- *integer*
  - *zero*
  - *non-zero* PRE *integer*

```
Relations
append/3
copy/2
member/2
subterm/2
ty2-component/2
```

Figure 1: The signature Σ_{Ty2} for a grammar of Ty2 expressions

this term corresponds to the natural number represented by the configuration under the *non-zero* entity.

The six COMPLEX TERM PRINCIPLES, (1b), are responsible for the proper typing of complex Ty2 expressions. These are the *application* of a functor to an argument ($\left(\alpha_{\langle\tau',\tau\rangle}(\beta_{\tau'})\right)_{\tau}$), lambda *abstractions* ($(\lambda v_{n,\tau'}.\alpha_{\tau})_{\langle\tau',\tau\rangle}$), *equations* ($(\alpha_{\tau} = \beta_{\tau})_{t}$), negated expressions ($(\neg\alpha_{t})_{t}$), expressions formed from two meaningful expressions by conjoining them with a logical connective (e.g., $(\alpha_{t} \wedge \beta_{t})_{t}$), and quantificational expressions (e.g., $(\exists v_{n,\tau}\alpha_{t})_{t}$). In models of the TFL grammar the correct typing of the meaningful expressions, indicated in the examples given in parentheses with the type t (for *truth*) and the meta-variable τ, is guaranteed by the COMPLEX TERM PRINCIPLES.

The remaining principles fall in two groups: the task of the principles (1c)–(1e) is to guarantee the well-formedness of the *ty2* configurations in grammar models in the sense that all all *ty2* configurations correspond to Ty2 expressions (or natural numbers and types); the remaining three principles, (1f)–(1h), determine the meaning of relation symbols which are needed either in the preceding three principles or in Section 2.2 in the composition principles of LRS. According to (1f) the relation `ty2-component` holds between each pair of Ty2 objects $\boxed{1}$ and $\boxed{2}$ such that either $\boxed{1}$ and $\boxed{2}$ are identical or $\boxed{1}$ is a component of $\boxed{2}$ (i.e., $\boxed{1}$ can be reached by starting from $\boxed{2}$ and following a finite sequence of attributes). With (1g), two Ty2 objects $\boxed{1}$ and $\boxed{2}$ in an expression are in the `copy` relation iff the configurations of objects under them are isomorphically configured: they all have the same attributes and corresponding attribute values of the same sorts. The `subterm` relation, determined by (1h), will be particularly important in Section 2.2. It holds between each pair of *me* objects $\boxed{1}$ and $\boxed{2}$ iff $\boxed{1}$ is a subterm of $\boxed{2}$. For perspicuity we will use an infix notation below and write '$\boxed{1} \lhd \boxed{2}$' for `subterm`$(\boxed{1}, \boxed{2})$. The `append` and `member` relation symbols, which also belong to the signature Σ_{Ty2} in Figure 1, receive their usual interpretation. We omit the principles defining their intended meaning.

The TY2 NON-CYCLICITY PRINCIPLE, (1c), excludes the possibility of cyclic term configurations. Cyclic terms (and types) are terms which contain themselves as components. Since it is not clear what cyclic configurations of this kind should correspond to in two-sorted type theory, they have to be excluded from our models. The TY2 FINITENESS PRINCIPLE, (1d), uses the finiteness of *chains* in RSRL to enforce the finiteness of *ty2* configurations. The last principle, the TY2 IDENTITY PRINCIPLE in (1e), enforces a kind of extensionality in our models of Ty2 expressions. It requires that any two isomorphic subconfigurations in a *ty2* configuration be actually identical. For example, if the first variable of type s, $v_{s,0}$, occurs more than once in a Ty2 expression, its corresponding model (as determined by our constraints) will contain exactly one configuration of objects representing $v_{s,0}$.[4]

[4]For an extensive discussion and concrete examples of models, see (Richter, 2004). Sailer (2003) proves that the RSRL specification of Ty2 sketched here is correct.

2.2 Semantic Composition

For the purpose of illuminating the connection between LRS and HPSG in a familiar way, one can think of LRS in terms of a simple TFL specification. The signature of the RSRL encoding of an LRS grammar contains the following attributes, sorts and appropriateness specifications:

(2) The sort *lrs* (LF value of signs)

lrs	EX(TERNAL-)CONT(ENT)	*me*
	IN(TERNAL-)CONT(ENT)	*me*
	PARTS	*list(me)*

From the previous section we know that the objects in the denotation of the sort *me* are the elements of the set of well-formed expressions of Ty2. The crucial difference between systems such as Flexible Montague Grammar and LRS is that the former employs the lambda calculus with (intensional) functional application and beta-reduction for semantic composition. Semantic composition in LRS builds on a tripartite distinction between internal content, external content and the semantic contribution(s) of a sign to the overall semantic representation of an utterance. While external and internal content are substantive concepts, the representation format of PARTS as a list of *me*s is an artifact of the LRS encoding in TFL. Rather than thinking of PARTS values as lists of expressions, it is more accurate to view them as the specification of those nodes of the term graph of a Ty2 expression which are contributed to the meaning of the natural language expression by the given sign. Only the topmost node of each element on PARTS counts as being contributed.

The internal content of a sign is the scopally lowest semantic contribution of the semantic head of the sign. Its membership in PARTS characterizes it as a necessary contribution of meaning to each syntactic head.

(3) The INCONT PRINCIPLE (IContP):
In each *lrs*, the INCONT value is an element of the PARTS list and a component of the EXCONT value.

$$lrs \rightarrow \left(\begin{bmatrix} \text{EXCONT} & \boxed{1} \\ \text{INCONT} & \boxed{2} \\ \text{PARTS} & \boxed{3} \end{bmatrix} \wedge \texttt{member}(\boxed{2}, \boxed{3}) \wedge \boxed{2} \lhd \boxed{1} \right)$$

The external content of a sign is the meaning contribution of its maximal projection to the meaning of the overall expression. When a sign enters into a syntactic construction as a non-head, its external content must have been contributed in the completed syntactic domain:

(4) The EXCONT PRINCIPLE (EContP):

a. In every phrase, the EXCONT value of the non-head daughter is an element of the non-head daughter's PARTS list.

$$phrase \rightarrow \left(\left[\text{NH-DTR LF} \begin{bmatrix} \text{EXCONT} & \boxed{1} \\ \text{PARTS} & \boxed{2} \end{bmatrix} \right] \wedge \texttt{member}(\boxed{1}, \boxed{2}) \right)$$

b. In every utterance, every subexpression of the EXCONT value of the utterance is an element of its PARTS list, and every element of the utterance's PARTS list is a subexpression of the EXCONT value.

$$\textit{u-sign} \rightarrow \left(\begin{array}{l} \forall\boxed{1}\,\forall\boxed{2}\,\forall\boxed{3}\,\forall\boxed{4} \\ \left(\left[\text{LF} \left[\begin{array}{l} \text{EXCONT}\ \boxed{1} \\ \text{PARTS}\ \ \boxed{2} \end{array} \right] \right] \wedge\ \boxed{3} \lhd \boxed{1} \wedge\ \texttt{member}(\boxed{4}, \boxed{2}) \right) \rightarrow \\ (\texttt{member}(\boxed{3}, \boxed{2}) \wedge\ \boxed{4} \lhd \boxed{1}) \end{array} \right)$$

The external content value of an utterance is its logical form in the traditional sense. According to the second clause of the EContP, which is a kind of closure principle, the logical form comprises all and only the meaning contributions of the lexical elements in the utterance.

The LRS PROJECTION PRINCIPLE makes the internal and external content locally accessible throughout head projections, and it guarantees that the meaning contributions of all subsigns of a sign will be collected.

(5) LRS PROJECTION PRINCIPLE:

In each *headed-phrase*,

a. the EXCONT value of the head and the mother are identical,

$$\textit{phrase} \rightarrow \left[\begin{array}{l} \text{LF EXCONT}\ \boxed{1} \\ \text{H-DTR LF EXCONT}\ \boxed{1} \end{array} \right]$$

b. the INCONT value of the head and the mother are identical,[5]

$$\textit{phrase} \rightarrow \left[\begin{array}{l} \text{LF INCONT}\ \boxed{1} \\ \text{H-DTR LF INCONT}\ \boxed{1} \end{array} \right]$$

c. the PARTS value contains all and only the elements of the PARTS values of the daughters.

$$\textit{phrase} \rightarrow \left(\left[\begin{array}{l} \text{LF PARTS}\ \ \boxed{1} \\ \text{H-DTR LF PARTS}\ \ \boxed{2} \\ \text{NH-DTR LF PARTS}\ \boxed{3} \end{array} \right] \wedge\ \texttt{append}(\boxed{2}, \boxed{3}, \boxed{1}) \right)$$

The SEMANTICS PRINCIPLE (SP) specifies restrictions on how to combine the meaning contributions of different types of syntactic and semantic daughters. For each kind of meaning composition which introduces subterm restrictions, the SP specifies a clause. The primary task of these clauses is to state mutual embedding constraints between the terms of each syntactic daughter. If the relative embedding of the meaning contributions is not fixed deterministically, we achieve a descriptive underspecification of readings:

(6) SEMANTICS PRINCIPLE (SP):

[5] We take the noun to be the head of a quantified NP.

1. if the non-head is a quantifier then its INCONT value is of the form $Qx[\rho \circ \nu]$, the INCONT value of the head is a component of ρ, and the INCONT value of the non-head daughter is identical with the EXCONT value of the head daughter,

$$\left[\text{NH-DTR SS LOC CAT HEAD } \textit{det}\right] \rightarrow$$

$$\left(\begin{bmatrix} \text{H-DTR LF} & \begin{bmatrix}\text{EXCONT } \boxed{1}\\ \text{INCONT } \boxed{2}\end{bmatrix} \\ \text{NH-DTR LF } \boxed{1} & \begin{bmatrix}\text{INCONT} & \begin{bmatrix}\textit{quantifiers} & \\ \text{SCOPE} & \begin{bmatrix}\textit{l-const}\\ \text{ARG1 } \boxed{3}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix} \wedge\ \boxed{2} \lhd \boxed{3}\right)$$

2. if the non-head is a quantified NP with an EXCONT value of the form $Qx[\rho \circ \nu]$, then the INCONT value of the head is a component of ν,

$$\forall\boxed{1}\left(\begin{bmatrix}\text{NH-DTR} & \begin{bmatrix}\text{SS LOC CAT} & \begin{bmatrix}\text{HEAD} & \textit{noun}\\ \text{SUBCAT} & \langle\rangle\end{bmatrix}\\ \text{LF EXCONT} & \begin{bmatrix}\textit{quantifiers}\\ \text{ARG2 } \boxed{1}\end{bmatrix}\end{bmatrix}\end{bmatrix} \rightarrow \exists\boxed{2}\left(\begin{bmatrix}\text{H-DTR LF INCONT } \boxed{2}\end{bmatrix} \wedge\ \boxed{2} \lhd \boxed{1}\right)\right)$$

3. [clauses for adverbial modifiers, markers, fillers, ...]

2.3 An Example

With the LRS principles of Section 2.2 we can now analyze sentences with quantifier scope ambiguities such as *Every student reads a book*. In a first approximation, we would like to assign the two readings in (7b) and (7c) to this sentence:

(7) a. Every student reads a book.

b. $\lambda w.\exists e \exists y\,(\mathsf{book}'(w,y)\ \wedge\ \forall x\,(\mathsf{student}'(w,x) \rightarrow \mathsf{read}'(w,e,x,y)))$

c. $\lambda w.\exists e \forall x\,(\mathsf{student}'(w,x)\ \rightarrow\ \exists y\,(\mathsf{book}'(w,y) \wedge \mathsf{read}'(w,e,x,y)))$

In the first reading, (7b), the existential quantifier of *a book* takes wide scope over the universal quantifier of *every student*. There is one particular book which every student reads. In (7c) the scope relation of the two quantifiers is reversed. Every student reads a book, but it is not necessarily the same book. We write w for the first variable of type s and use intuitive names for non-logical constants, such as $\mathsf{student}'$, book' and read'. The variable e is a Davidsonian event variable.

Figure 2 illustrates how an LRS grammar of English licenses the two readings of sentence (7a). Because of the perspicuity of the computational description language of LRS, we do not use RSRL descriptions of *lrs* objects at the nodes of the syntactic tree in the figure. Instead we use the description language of the LRS implementation language to be introduced in Section 4. In this language *lrs* descriptions are notated as Ty2 expressions augmented with a small inventory of

additional symbols. The figure depicts the LRS specifications of the lexical entries of an implemented grammar and the information about the *lrs* of each phrase which can be derived according to the LRS principles of the previous section.

In the implementation language $\hat{}\,\alpha$ means that the (possibly augmented) Ty2 expression α is the EXCONT value of the sign's *lrs*. The INCONT value is notated between curly brackets, $\{\beta\}$. Square brackets signal a `subterm` relationship. $\alpha[\beta]$ (or, equivalently, $\alpha : [\beta]$) means that β is a subterm of α. We write $[\beta]$ if we do not know anything about an expression except that β is a subterm of it.

At the two NP nodes, Clause (1) of the SEMANTICS PRINCIPLE requires the internal content of the head (`student(w,X)` and `book(w,Y)`) to be in the restrictor of the universal and the existential quantifier, respectively. At the VP and S nodes, it is Clause (2) that brings the internal content of the verbal head (`read(w,e,X,Y)`) in the scope of both quantifiers.

As an effect of the EContP, we know the external contents of the non-heads in all phrases: At the NP nodes, the external content of the determiners is identical to their internal contents, as these are the only elements being contributed that contain the internal contents as their subexpression. Being contributed is the same as being on the PARTS list in the RSRL specification. In the implementation, every (sub-) expression mentioned counts as being contributed, unless explicitly marked otherwise. At the VP node (and analogously at the S node), the external content of the NP *some book* must be identical to that of the determiner: this is the only element being contributed by the NP that satisfies the condition expressed in the lexical entry of *book*, i.e., that the external content be a quantifier that binds the variable y. Note that the lexical entry of *book* has a meta-variable, `Y`, in the argument slot of type *entity*. The local selection mechanism, not depicted here, is responsible for identifying the variable y, contributed by the existential quantifier, with `Y`.

At the S node, the second clause of the EContP applies, i.e., the expressions being contributed specify exactly the expressions which can be used in the resulting logical form. There are exactly two Ty2 expressions which are compatible with the contributions and structural requirements of the lexical entries and that also fulfill the subexpression requirements given by the SEMANTICS PRINCIPLE. Either the universal quantifier of *every student* is in the nuclear scope of the existential quantifier of *a book*, or *vice versa*.

3 LF Constraints

The linguistic rationale behind the architecture of LRS is evidenced by the smooth integration of: (I) "typical" LF constraints such as quantifier island constraints, (II) a straightforward and novel account of negative concord (or multiple exponents) phenomena, and (III) a treatment of traditionally problematic LF discontinuities, thus integrating insights from the generative literature on the syntax-semantics interface in terms of Logical Forms. LRS reanalyzes these insights in terms of the additional expressive flexibility provided by a truly constraint-based grammar frame-

Figure 2: The sentence *Every student reads a book*

work. With a systematic account of typical LF constraints, LRS goes beyond the data analyzed in alternative frameworks used in HPSG such as MRS or UDRT.

For reasons of space, we will give only a very brief and abstract overview of the types of LF constraints for which LRS is designed, and we do not discuss concrete examples in this paper. Quantifiers islands (type I constraints) are discussed in (Sailer, 2003, pp. 58–61). A comprehensive empirical motivation for the "multiple exponent" analysis of negative concord in Polish (type II constraints) and an explicit LRS analysis thereof are provided in (Richter and Sailer, 2004, pp. 115–126). Examples of LF discontinuities (type III constraints) and their LRS analysis are discussed in (Richter and Sailer, 2004, pp. 126–131) and in (Richter, 2004, Chapter 6).

A typical LF constraint of type I concerns *quantifier islands*: A universal quantifier may not take scope outside of the clause in which it is realized overtly. As a constraint this is usually regarded as odd, since existential quantifiers in embedded contexts may outscope opaque matrix predicates, producing so-called *de-re* readings. LRS can state restrictions on the scope of quantifiers naturally. They do not differ from any ordinary syntactic restriction in HPSG. Another source of type I LF constraint is statements that postulate that *no quantifier (of a certain type) may intervene* in the logical form between two given logical operators. Conversely, we might require that between a negation operator and some constant only certain quantifiers may intervene.

Type II constraints take advantage of HPSG's concept of token identity for a novel description of puzzling facts such as negative concord in Romance languages or in Polish. The NEG CRITERION restates a principle of Haegeman and Zanuttini (1991) from a new perspective: For every verb, if there is a negation in the EXCONT value of the verb that has scope over the verb's INCONT value, then that negation must be an element of the verb's PARTS list. In other words, if there is a negation with scope over the verb in the verbal projection, the verb itself must also contribute the very same negation. Similarly, the NEGATION COMPLEXITY CONSTRAINT for Polish expresses an insight in terms of LRS which Corblin and Tovena (2001) found to hold for many languages: For each sign, there may be at most one negation that is a component of the EXCONT value and has the INCONT value as its component. This expresses a (language-dependent) upper bound on the number of negations taking scope over each other and over the main verb of sentences. It also relies on the possibility that, in negative concord languages, negations contributed by different lexical items might be identified with each other in the semantic representation. A third principle which builds on the fact that the same meaning component might be contributed by several lexical elements is the WH-CRITERION (for German, Richter and Sailer, 2001, p. 291): In every clause, if the EXCONT value is of the form $\lambda p.\phi$, then the EXCONT value of the clause must be contributed by the topicalized sign (again rephrasing a well-known principle from the literature). Similar "multiple exponent" effects were found in the LRS analysis of tense in Afrikaans.

LF discontinuities (III) are a lexical phenomenon: A lexical element might

make meaning contributions to a sentence that must be realized discontinuously in the logical form of the overall expression to which they belong. The intervening meaning components are unpredictable and can not be stated in a finite list. Analyses of these phenomena are typically provided by underspecification formalisms which allow for decomposing the logical contributions of lexical items and leaving slots for inserting other pieces of representations.

4 Formal Specification

A formal specification of the core principles of LRS requires a term description language for Ty2 with an is-component-of relation ('α is a subterm of β'), metavariables, which, for us, are the variables of the TFL, and a way of attributing semantic "contribution" by lexical items. It also requires a set of axioms for well-formed expressions of Ty2 (which we have presented in Section 2.1), as well as four HPSG principles for IContP, EContP, LRS PROJECTION PRINCIPLE and SP themselves (Section 2.2). Note the absence of beta-reduction from the formalism.

This core of the LRS architecture allows one to assign to each sentence a logical representation with a model-theoretic interpretation; it uses descriptive underspecification to assign to each well-formed utterance one or more fully specified logical form(s) as its meaning representation. If an utterance is n-ways ambiguous, the denotation of the grammar will contain n models of it which differ at least in their meaning representation. In keeping with the tradition of logical form semantics, however, the semantical components of these utterances are modelled by Ty2 *terms*, not the entities of Ty2's models themselves; thus, no extra expressive power beyond TFL's model theory is actually required.

Formally, we augment TFL's model theory with four additional partial functions:

$$\begin{aligned}&\mathrm{sem} : U \to Ty2\\ &\mathrm{incont} : U \to Ty2\\ &\mathrm{excont} : U \to Ty2\\ &\mathrm{contrib} : Ty2 \to \mathcal{P}(U)\end{aligned}$$

where U is the universe of the TFL model. If *sem*, *incont* and *excont* were features in the signature of TFL, they would be interpreted by functions mapping U to U. These functions, however, allow us to refer instead to a separate collection of entities that model the structure of Ty2 terms. *Sem* is the principal means of access to this collection, potentially associating any entity in the model with a Ty2 term, i.e., a semantics. In practice, this association probably only occurs with signs, and *sem* replaces the LF attribute. Any entity can additionally have *incont* and *excont* values, which in practice are employed in accordance with the intuitions of LRS. Among other things, this means that these values, where they exist, will typically be subterms of the term that *sem* refers to. *Contrib* conversely attributes every Ty2 term in the image of *sem* to the entities that contributed it.

4.1 Constraint Language

Once this model is in place, we need a syntax to refer to it. Augmenting the standard sorts-and-features signature of an HPSG description language, we add to it a collection of semantic type declarations, such as shown in Figure 3. These decla-

```
semtype [t,f]: t.
semtype [student,book]:  (s->e->t).
semtype read:  (s->e->e->e->t).
semtype [every,some]:  (e->t->t->t).
semtype w: var(s).
semtype q:  findom [every,some].
semtype [a,e,x,y,z]:  var(e).
semtype lambda:  (var(A)->B->(A->B)).
```

Figure 3: An example `semtype` signature.

rations declare the semantic constants and semantic variables that can then be used in our Ty2 terms. Our description language does not stipulate the basic types of the semantic type system (above, `t`, `e`, and `s`), but it does allow for functional closure. Notice that even `lambda` is just another constant, although it has a polymorphic functional type. There is no reason to distinguish it because beta-reduction has no distinguished role in this semantics — if it were desired, it would need to be encoded as a relation or phrase-structure rule just as any other transformation. The `var/1` operator distinguishes semantic variables from semantic constants. This distinction is important because, although there is no beta-reduction, there is still alpha-renaming within variable scope, which we define to be the same as the scope of TFL variables in descriptions. Constants are unique and never renamed. In the example above, `q` is a finite domain variable — an instance of it stands for one of either `every` or `some`.

Having enhanced the signature, we are then in a position to enhance a TFL description language with extra LRS descriptions. Given a countably infinite collection of meta-variables (V), the LRS descriptions (χ) are the smallest set containing:

- the semantic constants of the signature,
- the semantic variables of the signature,
- applications, $f(\chi_1, \ldots, \chi_n)$, where
  - $\text{semtype}(f) = \rho_1 \to \ldots \to \rho_n \to \tau$,
  - $\text{semtype}(\chi_i) = \rho_i$, all $1 \leq i \leq n$, and
  - τ can be any type (functional or not),[6]

[6]Thus, the case of $n = 0$ is already covered by including the semantic constants.

- meta-variable binding: $V{:}\chi$,
- subterm binding: $\chi_0 : [\chi_1, \ldots, \chi_n]$,
- subterm chain: $V_{top} \sim V_{bottom}$,
- *incont* binding: $\{\chi\}$,
- *excont* binding: $\hat{}\chi$,
- contribution: $\chi/{+}V$,
- unique contribution: χ/V,
- negative contribution: $\chi/\sim V$, and
- implication: $\backslash/S : \chi_S \longrightarrow \chi$.

Because these are included in the closure of the TFL description language, they can be conjoined and disjoined with conventional TFL descriptions, and they can also be applied at feature paths. In the interpretation, however, while TFL descriptions constrain the choice of element $u \in U$ directly, LRS descriptions mostly constrain our choice of $\mathrm{sem}(u) \in Ty2$. *Incont* and *excont* binding instead constrain our choice of $\mathrm{incont}(u) \in Ty2$ and $\mathrm{excont}(u) \in Ty2$, and the contribution constraints constrain our choice of the elements of $\mathrm{contrib}(\mathrm{sem}(u)) \subseteq \mathcal{P}(U)$.

Subterm binding, $\chi_0 : [\chi_1, \ldots, \chi_n]$, says that $\chi_1, \ldots \chi_n$ are all subterms of χ_0. Meta-variables establish the equality of subterms within an LRS description, within a larger TFL description (which may refer to the semantic term of more than one feature path's value), or across the scope of description variables in a single construct (such as sharing the semantics of the mother and head daughter of a phrase-structure rule). A subterm chain constrains a subterm from both ends: it must fall along the chain from V_{top} to V_{bottom}.

What descriptions do not need to explicate, crucially, are all of the well-formedness properties entailed by our interpretation of these description primitives. Mathematically, our models will already be limited to those that observe the necessary well-formedness properties, and computationally, LRS descriptions are closed under a fixed set of algebraic rules that enforce them, as given below. These rules can be extended, in fact, to allow for universally quantified implicational constraints over semantic terms, much as HPSG principles appear in TFL. The implication, $\backslash/S : \chi_S \longrightarrow \chi$, states that for every subterm S of the term being described, if S is described by χ_S, then the χ holds of the term described. In keeping with TRALE's interpretation of implicational constraints in TFL, the antecedents of these semantic implications are interpreted using subsumption rather than classical negation.

4.2 Description Language Integration

LRS descriptions are identified within ALE descriptions by their embedding within a `@lrs/1` macro that provides the necessary glue to CHR. As a simple example, consider the following expression of clause (1) of the LRS PROJECTION PRINCIPLE above in TRALE syntax:

```
phrase *> (daughters:hdtr:lf: @lrs([^Alpha]),
           lf: @lrs([^Alpha])).
```

The meta-variable `Alpha` is bound in the consequent of this universally quantified principle to both the *excont* of the head daughter and the *excont* of the mother, thus equating them. The square brackets are necessary because this *excont* must only be a subterm of the semantics of the head daughter and the semantics of the mother, and not necessarily identical.

The IContP above is expressed as:

```
sign *> lf: @lrs([^E:[{I}]]).
```

The meta-variable `I` is identified as the *incont* of the sign's LF value by the curly braces of the *incont* binding primitive. This is a subterm (inner square brackets) of `E`, which is identified as its *excont* (caret), and as a subterm (outer square brackets) of its *sem* value. Unlike the TFL presentation, PARTS lists and other structural overhead are not required in our typed feature structures because meta-variables dually refer to both a term and the collection of all of its subterms.

4.3 CHR Implementation

In our Prolog implementation of LRS within the TRALE system, all LRS descriptions are compiled into constraints of a Constraint Handling Rules (CHR) handler, and their well-formedness properties are implemented as the constraint handling rules themselves. The primitive constraints they are compiled into are:

- `node(N,ArgTypes,ResType)`: node `N` has argument types `ArgTypes` with result type `ResType`.
- `literal(N,Lit,Arity)`: node `N` is labelled by literal `Lit` with arity `Arity`.
- `findom(N,Lits)`: node `N` is labelled by one of the literals in `Lits`.
- `ist(N,M,A)`: node `N` is the `A`th argument of node `M`.
- `st(N,M)`: node `N` is a subterm of node `M`.
- `excont(FS,N)`: the excont of feature structure `FS` is `N`.
- `incont(FS,N)`: the incont of feature structure `FS` is `N`.

- `contrib(FS,N)`: feature structure `FS` contributed `N`.
- `uniquecontrib(FS,N)`: feature structure `FS` uniquely contributed `N`.
- `nocontrib(FS,N)`: feature structure `FS` did not contribute `N`.

The nodes referred to here are nodes of the typed term graphs that represent the logical forms that we are assembling. In addition to these primitives, the transitive closure of `st/2`, called `ststar/2` is also computed on-line.

CHR rules consist of propagators ($\Longrightarrow$) that detect the presence of a combination of constraints (left-hand side) in a global constraint store, and in that presence, execute Prolog goals (right-hand side) that typically add more constraints to the store. Detection, as in TRALE, amounts to suspending until subsumption holds. Simplification rules ($\Leftrightarrow$) additionally remove left-hand-side constraints designated by appearing to the right of the \. If no \ is provided, then all left-hand-side constraints are removed. Right-hand-side goals can also be guarded (|) — if the guard fails, then the goal is not executed.

In CHR then, the following algebraic rules are used to enforce well-formedness:

- literal/arity consistency
  `literal(N,Lit1,Arity1) \ literal(N,Lit2,Arity2)`
  $\Leftrightarrow$ `Lit1 = Lit2, Arity1 = Arity2.`
- literal extensionality
  `literal(N,F,A), literal(M,F,A)`
  $\Longrightarrow$ `ext_args(A,N,M) | N=M.`
- constants
  `literal(N,_,0) \ st(M,N)` $\Leftrightarrow$ `M = N.`
- finite domains
  `findom(N,Lits), literal(N,Lit,_)`
  $\Longrightarrow$ `member(Lit,Lits).`
- immediate subterm irreflexivity
  `ist(N,N,_)` $\Longrightarrow$ `fail.`
- immediate subterm uniqueness
  `ist(M1,N,A) \ ist(M2,N,A)` $\Leftrightarrow$ `M1 = M2.`
- subterm reflexivity
  `st(N,N)` $\Leftrightarrow$ `true.`
- subterm idempotence
  `st(M,N) \ st(M,N)` $\Leftrightarrow$ `true.`
- subterm subsumption
  `ist(M,N,_) \ st(M,N)` $\Leftrightarrow$ `true.`

- subterm antisymmetry
  `st(M,N), st(N,M) ⇔ M = N.`
- subterm upward antisymmetry
  `ist(M,N,_), st(N,M) ⇔ M = N.`
- type consistency
  `node(N,ATypes1,RType1) \ node(N,ATypes2,RType2)`
  `⇔ RType1 = RType2, ATypes1 = ATypes2.`
- literal well-typing
  `node(N,ATypes,_,_), literal(N,_,A)`
  `⟹ length(ATypes,A).`
- immediate subterm well-typing
  `node(M,_,MResType), node(N,NArgTypes,_), ist(M,N,A)`
  `⟹ nth(A,NArgTypes,MResType).`
- incont and excont functionhood
  `incont(X,N) \ incont(X,M) ⇔ N=M.`
  `excont(X,N) \ excont(X,M) ⇔ N=M.`
- unique contribution injectivity
  `uniquecontrib(FS1,N) \ uniquecontrib(FS2,N)`
  `⇔ FS1=FS2.`
- unique contribution subsumption
  `uniquecontrib(FS1,N) \ contrib(FS2,N) ⇔ FS1=FS2.`
- negative contribution idempotence
  `nocontrib(X,N) \ nocontrib(X,N) ⇔ true.`
- negative contribution negativity
  `nocontrib(FS,N), contrib(FS,N) ⇔ fail.`

`ext_args/3` is a guard that checks the arguments of `N` and `M` for equality.

This collection of rules is complete in the sense that any inconsistency that may exist with respect to our standard of well-formedness in Ty2 will be detected. They are not complete in the sense that the result of simplification under these rules will be minimal in any useful sense, such as having a minimal number of distinct nodes in the resulting term graph, or a minimal number of (non-immediate) subterm arcs. The quest for a combination of propagators and search that would establish minimality efficiently (in practice, at least) remains to be pursued.

Turning to semantic implication, every instance of an implication description is compiled into its own CHR primitive constraint. This constraint occurs on the left-hand side of exactly one new propagator, which is charged with enforcing the implication. To require that universal quantifiers not outscope clausal boundaries, for example, we may require of clauses that:

```
\/X:every(_,_,_) --> [^[X]]).
```

Let us call this implication instance `i`. We introduce a new primitive `i/1`, which will be applied to semantic terms for which implication `i` is asserted to hold. We then add one new propagator to enforce the implication:

```
i(LF),ststar(X,LF),literal(X,every,3)
     ==> node(Ex,_,_),st(Ex,LF),excont(FS,Ex),st(X,Ex).
```

Here, `LF` will be bound to the term to which `i` applies, `X`, to the universally quantified variable `X` stated in the implication source code, and `FS`, to the feature structure with which `LF` is the associated semantical term.

5 Conclusion

By separating linguistic representations and principles from structural well-formedness and computational considerations, we aspire to do a better job of both. This separation can be achieved by expanding the description language with a set of primitives that intuitively capture the requirements of semantical theories (LRS and otherwise) that manipulate logical forms as typed term graphs, not by using TFL at any aesthetic cost.

The extension presented here captures the basic primitives found in MRS and CLLS, the exceptions being the parallelism and anaphoric binding constraints primitively expressed by CLLS. Future empirical study is required before extending the language in this direction, in our opinion, because much of what falls under the rubrics of ellipsis and binding is not purely a semantic phenomenon.

A more interesting direction in which to extend the present work is towards a semantics which does not stop at logical form. Semantics involves meaning, and meaning is only distinguishable in the presence of inference. Although we have presented the semantic implication of our description language as a tool for conventional linguistic constraints or for perhaps extending a very common-sensical core of well-formedness conditions, they are equally well applicable to inference more broadly construed. This inference would be grounded in presuppositional content and shared background knowledge as much as it would be in the syntactic structure of typed lambda terms.

References

Copestake, Ann, Flickinger, Dan, Pollard, Carl and Sag, Ivan A. 2003. Minimal Recursion Semantics: An Introduction. Journal submission, November 2003.

Corblin, Francis and Tovena, Lucia M. 2001. On the Multiple Expression of Negation in Romance. In Yves D'Hulst, Johan Rooryck and Jan Schroten (eds.), *Romance Languages and Linguistic Theory 1999*, pages 87–115, John Benjamins, Amsterdam.

Egg, Markus and Erk, Katrin. 2002. A Compositional Account of VP Ellipsis. In Frank Van Eynde, Lars Hellan and Dorothee Beermann (eds.), *Proceedings of the 8th International Conference on Head-Driven Phrase Structure Grammar*, pages 162–179, Stanford: CSLI Publications.

Egg, Markus, Koller, Alexander and Niehren, Joachim. 2001. The Constraint Language for Lambda Structures. *Journal of Logic, Language and Information* 10(4), 457–485.

Frühwirth, T. and Abdennadher, S. 1997. *Constraint-Programmierung*. Springer.

Gallin, Daniel. 1975. *Intensional and Higher-Order Modal Logic*. North-Holland, Amsterdam.

Haegeman, Liliane and Zanuttini, Raffaella. 1991. Negative Heads and the Neg Criterion. *The Linguistic Review* 8, 233–251.

Hendriks, Herman. 1993. *Studied Flexibility*. ILLC Dissertation Series 1995-5, Institute for Logic, Language and Computation, Amsterdam.

Penn, G. 2004. Balancing Clarity and Efficiency in Typed Feature Logic through Delaying. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 240–247.

Richter, Frank. 2004. Foundations of Lexical Resource Semantics. Habilitation Thesis. Seminar für Sprachwissenschaft, Universität Tübingen.

Richter, Frank and Sailer, Manfred. 2001. On the Left Periphery of German finite Sentences. In W. Detmar Meurers and Tibor Kiss (eds.), *Constraint-Based Approaches to Germanic Syntax*, pages 257–300, Stanford: CSLI Publications.

Richter, Frank and Sailer, Manfred. 2004. Basic Concepts of Lexical Resource Semantics. In Arnold Beckmann and Norbert Preining (eds.), *European Summer School in Logic, Language and Information 2003 – Course Material I*, volume 5 of *Collegium Logicum*, pages 87–143, Kurt Gödel Society Wien.

Sailer, Manfred. 2003. Combinatorial Semantics and Idiomatic Expressions in Head-Driven Phrase Structure Grammar. Phil. Dissertation (2000). Arbeitspapiere des SFB 340. 161, Eberhard-Karls-Universität Tübingen.

Sailer, Manfred. 2004. Local Semantics in Head-Driven Phrase Structure Grammar. In Olivier Bonami and Patricia Cabredo Hofherr (eds.), *Empirical Issues in Syntax and Semantics*, volume 5, pages 197–214, URL: http://www.cssp.cnrs.fr/eiss5/.

von Stechow, Arnim. 1993. Die Aufgaben der Syntax. In Joachim Jacobs, Arnim von Stechow, Wolfgang Sternefeld and Theo Vennemann (eds.), *Syntax. Ein internationales Handbuch zeitgenössischer Forschung*, volume 1, pages 1–88, Walter de Gruyter.