Foundations of Lexical Resource Semantics

Frank Richter
Seminar für Sprachwissenschaft
Abteilung Computerlinguistik
Eberhard-Karls Universität Tübingen

Habilitationsschrift

2004

Contents

Chapter 4

Logical Languages as Object Languages of RSRL Grammars

In this chapter I will introduce the type-theoretic logical language which forms the basis of the combination of the model-theoretic syntactic framework with model-theoretic semantics. The integration will proceed in two steps. In Section 4.1 I will present an RSRL grammar, Γ_{Ty2} and show that its exhaustive models are good representations of the intended type-theoretic language. In Section 4.2 the grammar Γ_{Ty2} will be combined with the normal form HPSG grammars from Part I of this thesis. In this way I will obtain a very general characterization of HPSG grammars which can employ the given type-theoretic language for assigning meaning to the unembedded sign configurations in their (minimal) exhaustive models. It will become clear that the method which is applied to integrate our particular type-theoretic language with HPSG grammars can also be used to integrate other classes of logical languages with HPSG grammars.

The type-theoretic language I choose is the language of two-sorted type theory, Ty2. Ty2 was introduced by [Gallin, 1975]. It is a generalization of Montague's Intensional Logic, and this has certain technical advantages [Zimmermann, 1989]. The two logical representations for the two readings which I will assign to the sentence (29a) illustrate Ty2:

(29) a. Every student reads some book.

b. $\forall x_{se}[\mathsf{student}'_{s\langle et\rangle}(@, x_{se}(@)) \rightarrow \exists y_{se}[\mathsf{book}'_{s\langle et\rangle}(@, y_{se}(@)) \wedge \mathsf{read}'_{s\langle e\langle et\rangle\rangle}(@, x_{se}(@), y_{se}(@))]]$

c. $\exists y_{se}[\mathsf{book}'_{s\langle et\rangle}(@, y_{se}(@)) \wedge \forall x_{se}[\mathsf{student}'_{s\langle et\rangle}(@, x_{se}(@)) \rightarrow \mathsf{read}'_{s\langle e\langle et\rangle\rangle}(@, x_{se}(@), y_{se}(@))]]$

The most salient difference between the logical expressions in Ty2 and the expressions of Intensional Logic is the syntactic occurrence of a world variable. By convention, I write @ for the first variable of type s, where s is the type for world indices.[1] The typing of the non-logical constants follows the system in [Sailer, 2003], which is a translation into Ty2

[1] With the convention of choosing the first variable of type s as the unbound world variable in Ty2 representations for the meaning of sentences I follow [Groenendijk and Stokhof, 1982].

of the type system in [Hendriks, 1993]. A more thorough analysis of the sentence (29a) might also include event variables in the tradition of [Davidson, 1980]. More details about the formulae will be explained when I introduce Ty2 in Section 4.1.

Ty2 is chosen for theoretical as well as for pragmatic reasons. Although some linguists argue that the weak intensionality of the system is insufficient for the description of natural language semantics, this system is still very widespread among formal semanticists in linguistics. For this reason analyses couched in Ty2 will be understood by most semanticists. Its close relationship to Intensional Logic makes Ty2 analyses compatible with Montague Grammar. Moreover, Ty2 is the type-theory behind the theory of *Transparent Logical Form*, an important research direction in the tradition of Chomskian syntactic theory [von Stechow, 1993]. Hence the use of Ty2 guarantees maximal compatibility of the semantic analyses with alternative frameworks and functions as a bridge to Montague Grammar. Analyses can draw on the experience with the system and with many analytical suggestions which have been formulated in other linguistic frameworks. Investigations of the new framework of *normal form HPSG grammars with Ty2* can thus take advantage of a rich research tradition. Last but not least I hope that the choice of Ty2 makes semantic research in HPSG more attractive to formal semanticists.

A combination of Ty2 with HPSG was first suggested in a study of negative concord and sentential negation in French in combination with a collocation theory by [Richter and Sailer, 1999a]. Most of the formal results concerning the RSRL grammar of Ty2 come from [Sailer, 2003]. However, in contrast to earlier work, the present study systematically separates the integration of Ty2 in HPSG from the possible systems of semantic compositions which can be defined based on the general architecture of a normal form grammar with Ty2.

4.1 Two-sorted Type Theory in RSRL

In this section I will lay the foundations for the integration of a truth-conditional semantics in the tradition of the semantics for English given in [Montague, 1974b] with normal form grammars. For this purpose, I will first introduce the syntax of two-sorted type theory. The syntax of the formal language is then the target for our RSRL theory of two-sorted type theory. I will present an RSRL grammar Γ_{Ty2} which has an exhaustive model which contains Ty2. The aptness of the grammar Γ_{Ty2} for our purposes will be shown with results from [Sailer, 2003].

My presentation follows [Richter, 2004a, Section 5.4.1] with a number of modifications assimilating the definitions syntactically to the version of Ty2 in [Sailer, 2003], since I will rely on the results in that work concerning Γ_{Ty2}. However, my generalized treatment of logical constants stays closer to the exposition of Ty2 in [Zimmermann, 1989] than to Sailer's. Sailer's encoding of constants in Γ_{Ty2} presupposes a finite set of logical constants. My view on type theory, the relationship between the typed lambda calculus and logical languages and their function within various linguistic research programs in model-theoretic semantics is inspired by the very rewarding overview in [Hamm, 1999].

Assume that e, s and t are distinct objects, none of which is an ordered pair. The set of types, $Types$, is defined as the smallest set which contains these three objects and is closed under pairs:

Definition 29 *$Types$ is the smallest set such that*

$e \in Types$,

$t \in Types$,

$s \in Types$,

for each $\tau_1 \in Types$, for each $\tau_2 \in Types$, $\langle \tau_1, \tau_2 \rangle \in Types$.

The type e will be used for entities, t for the truth values True and False, and s for possible worlds.

In the following definitions, I write $\mathbb{N}_0$ for the set of positive integers and zero.

Definition 30 *Var is the smallest set such that for each $\tau \in Types$, for each $n \in \mathbb{N}_0$, $v_{\tau,n} \in Var$.*

Definition 31 *$Const$ is the smallest set such that for each $\tau \in Types$, for each $n \in \mathbb{N}_0$, $const_{\tau,n} \in Const$.*

Variables and constants are typed. With the DEFINITIONS 30 and 31 we obtain a denumerably infinite supply of variables and constants for every type. Var_τ is the set of variables of type τ. Analogously, $Const_\tau$ is the set of constants of type τ.

The *meaningful expressions of Ty2* are defined with all the logical connectives which are typically needed in discussions of basic linguistic examples.

Definition 32 *The **meaningful expressions of Ty2** are the smallest family $(\mathrm{Ty2}_\tau)_{\tau \in Types}$ such that*

for each $\tau \in Types$, $Var_\tau \cup Const_\tau \subset \mathrm{Ty2}_\tau$,

for each $\tau \in Types$, for each $\tau' \in Types$,

if $\alpha_{\langle \tau', \tau \rangle} \in \mathrm{Ty2}_{\langle \tau', \tau \rangle}$ and $\beta_{\tau'} \in \mathrm{Ty2}_{\tau'}$, then $\left(\alpha_{\langle \tau', \tau \rangle}(\beta_{\tau'})\right)_\tau \in \mathrm{Ty2}_\tau$,

for each $\tau \in Types$, for each $\tau' \in Types$, for each $n \in \mathbb{N}_0$, for each $v_{n,\tau'} \in Var_{\tau'}$, for each $\alpha_\tau \in \mathrm{Ty2}_\tau$,

$(\lambda v_{n,\tau'}.\alpha_\tau)_{\langle \tau', \tau \rangle} \in \mathrm{Ty2}_{\langle \tau', \tau \rangle}$,

for each $\tau \in Types$, for each $\alpha_\tau \in \mathrm{Ty2}_\tau$, for each $\beta_\tau \in \mathrm{Ty2}_\tau$,

$(\alpha_\tau = \beta_\tau)_t \in \mathrm{Ty2}_t$,

for each $\alpha_t \in \mathrm{Ty2}_t$,

$(\neg\alpha_t)_t \in \mathrm{Ty2}_t$,

for each $\alpha_t \in \mathrm{Ty2}_t$, *for each* $\beta_t \in \mathrm{Ty2}_t$,

$(\alpha_t \wedge \beta_t)_t \in \mathrm{Ty2}_t$, *and* *(analogously for* $\vee, \rightarrow, \leftrightarrow$*)*

for each $\tau \in Types$, *for each* $n \in \mathbb{N}_0$, *for each* $v_{n,\tau} \in Var_\tau$, *for each* $\alpha_t \in \mathrm{Ty2}_t$,

$(\exists v_{n,\tau}\alpha_t)_t \in \mathrm{Ty2}_t$. *(and analogously for* $\forall$*)*

Standard definitions of the expressions of Ty2 typically only include variables and constants as atomic expressions and *applications*, $\left(\alpha_{\langle\tau',\tau\rangle}(\beta_{\tau'})\right)_\tau$, *abstractions*, $(\lambda v_{n,\tau'}.\alpha_\tau)_{\langle\tau',\tau\rangle}$, and *equations*, $(\alpha_\tau = \beta_\tau)_t$ as complex expressions. They proceed with ontologies D_τ for all types τ, in which D_t is the set comprising True and False, D_e is a set of entities, D_s is a set of world indices and the complex types $\langle\tau_1, \tau_2\rangle$ induce functions, $D_{\langle\tau_1,\tau_2\rangle}$, with the domain D_{τ_1} and the range D_{τ_2}. Models of Ty2 are pairs of the family of ontologies D_τ and an interpretation function I which assigns an element of the appropriate set $D_{\tau'}$ to each non-logical constant of type τ' from $Const_{\tau'}$. A variable assignment function g maps each element of $Var_{\tau'}$ to an entity in $D_{\tau'}$. Using these constructs, the non-logical constants and variables can be interpreted with I and g, equations are interpreted as equality in the domain of interpreting structures, and applications and abstraction receive their standard interpretation in the given domain of entities. I omit the details here, since none of this will be needed in my foundational grammar-theoretic investigations, which only concern the syntactic side of Ty2. A thorough exposition of the topic can be found in [Gallin, 1975].[2]

Equipped with models of the meaningful expressions of Ty2 it is possible to demonstrate that applications, abstractions and equations are sufficient to define the logical connectives and the quantifiers which I already included in DEFINITION32.[3] For this reason, they are often omitted in the basic syntax to facilitate the definition of the semantics.

For the notation of logical expressions I will follow a few abbreviatory conventions to minimize the size of the expressions and to make them more readable. Compare the expressions in (30), corresponding to the two readings of the sentence in (30a), to their more explicit renderings in (29):

(30) a. Every student reads some book.

b. $\forall x_{se}[\mathsf{student}'_{@}(x_{@}) \rightarrow \exists y_{se}[\mathsf{book}'_{@}(y_{@}) \wedge \mathsf{read}'_{@}(x_{@}, y_{@})]]$

[2]For an introductory and more general overview, the reader might want to consult the comprehensive textbook [Hindley and Seldin, 1986], which contains an introduction to the lambda calculus and to combinatory logic, including models of the typed and untyped lambda calculus and a discussion of different type theories.

[3]A summary and very careful explanation for each one can be found in [Sailer, 2003, pp. 40–41].

c. $\exists y_{se}[\mathsf{book}'_{@}(y_{@}) \wedge \forall x_{se}[\mathsf{student}'_{@}(x_{@}) \rightarrow \mathsf{read}'_{@}(x_{@}, y_{@})]]$

The reserved world variable $v_{s,0}$, notated as @, is written as a subscript to its functors for the sake of perspicuity. The type subscripts of the constants, variables and complex expressions are often omitted, unless they are important for the discussion or to help read the expression. The notation of complex types is simplified by omitting commata and unnecessary brackets. For example, $\langle s, e\rangle$ becomes se, and the type of the constant $\mathsf{student}'$, $\langle s, \langle e, t\rangle\rangle$, is written $s\langle et\rangle$. Following the usual conventions, I also prefer a suggestive symbolization of the non-logical constants over neutral symbols with number subscripts. For example, I might thus write $\mathsf{student}'_{s\langle et\rangle}$ (or even simply $\mathsf{student}'$) for the second constant of type $\langle s, \langle e, t\rangle\rangle$, more strictly written in the form $c_{s\langle et\rangle,1}$.[4]

When we now combine RSRL and Ty2, we have to be careful with the symbols which occur in both classes of languages, such as the universal and existential quantifiers and the logical connectives. Fortunately, no harm can be done, since the context of an expression will always disambiguate which logical language each symbol belongs to. As soon as we have seen examples, it will become clear how the meta-language (RSRL) can always safely be distinguished from the object language (Ty2).

DEFINITION 32 gives us the object language which will be specified by the grammar of Ty2, Γ_{Ty2}. For our purposes, we first need a signature Σ_{Ty2} which provides a class of interpretations with an appropriate ontology for a specification of Ty2 expressions. Σ_{Ty2} is shown in Figure 4.1.

The sort hierarchy of Σ_{Ty2} distinguishes three major classes of entities. The sorts in whose denotation we will find them are immediate subsorts of the top sort, *ty2*. For the intended standard exhaustive model of the grammar, the following properties are expected. In the denotation of *integer* we will find the elements of $\mathbb{N}_0$, *type* will denote the elements of $Types$, and *me* the set of meaningful expressions of Ty2. The sort *integer* is partitioned into *zero* and *non-zero*, with the attribute PRE(DECESSOR) appropriate for *non-zero*. The immediate subsorts of *type* distinguish between the atomic types (*entity, truth* and *w-index*, corresponding to e, t and s respectively) and *complex-types*. Entities of sort *complex-type* have an IN and an OUT attribute. The two attributes have values of sort *type*. The sort hierarchy and appropriateness specification under *type* will generate a space of configurations which stand in an obvious relationship to the type system of Ty2.

The expressions of Ty2 are typed, which is reflected by the fact that the *type* valued attribute TYPE is appropriate for all subsorts of the sort *me*. Variables (*variable*) and constants (*constant*) are also indexed by an element of $\mathbb{N}_0$, which is located under the attribute NUM-INDEX. Entities of sort *application* consist of two meaningful expressions, a functor under the attribute FUNCTOR and an argument under the attribute ARG. Similarly, an abstraction needs a variable under the attribute VAR and a meaningful expression under the attribute ARG. Equations as well as logical negation, the binary logical connectives and the existential and universal quantifiers are similarly provided in the ontology of Σ_{Ty2} with species with the necessary number of attributes and the attribute values required to mirror DEFINITION 32.

[4]The first constant of type $\langle s, \langle e, t\rangle\rangle$ is $c_{s\langle et\rangle,0}$.

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`
`member/2`
`ty2-component/2`
`copy/2`

Figure 4.1: The signature Σ_{Ty2} for a grammar of Ty2 expressions

The three relation symbols `member`, `ty2-component` and `copy` will be needed in the theory of Ty2 expressions to guarantee that each configuration under a *ty2* entity is well-formed. I will write $\langle \mathcal{G}_{Ty2}, \sqsubseteq_{Ty2}, \mathcal{S}_{Ty2}, \mathcal{A}_{Ty2}, \mathcal{F}_{Ty2}, \mathcal{R}_{Ty2}, \mathcal{AR}_{Ty2} \rangle$ for the septuple Σ_{Ty2} and I will use these symbols whenever I need to refer to a constituent of Σ_{Ty2}.

Before I start explaining the grammar of the meaningful expressions of Ty2, it is necessary to become familiar with their representations in Σ_{Ty2} models. The choices which we face in this context are reminiscent of the choices we faced in the discussion of the intensionality and (weak) extensionality of lists in Chapter 2. A first possibility is shown in Figure 4.2.

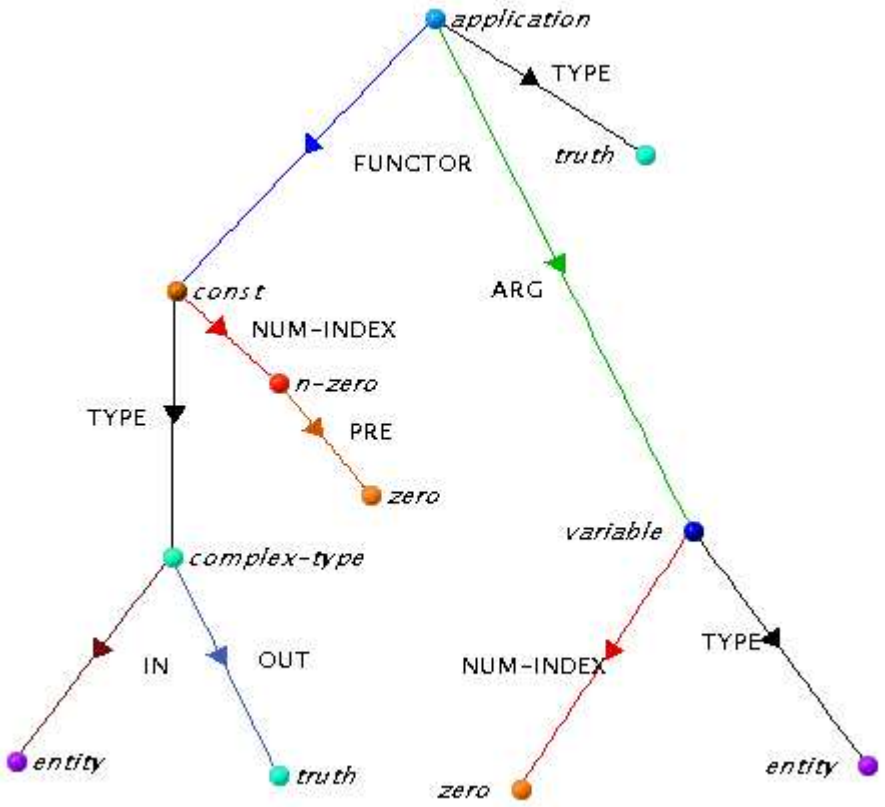

Figure 4.2: A tree-shaped representation of $c_{et,1}(v_{e,0})$

Figure 4.2 shows a configuration of entities under an *application* entity which corresponds to the Ty2 expression $c_{et,1}(v_{e,0})$. An important design feature in this representation is the decision to treat the entities in the configuration as intensional entities. For example, there are two entities of sort *truth* and two entities of sort *entity* in this configuration. Since these entities are atomic, the trivial configurations under each pair of *entity* and *truth* entities are isomorphic. Figure 4.2 illustrates that the resulting representation can be nicely depicted as a tree and is very readable. In the latter respect it corresponds to the

intensional representation of lists in Chapter 2. Note that in order to avoid ambiguities in the possible representation of terms, the principles of the grammar of Ty2 must enforce the non-identity of entities throughout all possible *ty2* configurations if we choose this representation. If it is not enforced, there could be a second configuration for the expression $c_{et,1}(v_{e,0})$ in models of the grammar which is just like the one depicted in Figure 4.2, except that there is only one *truth* entity to which the TYPE arc originating from the *application* entity and the OUT arc originating from the *complex-type* entity point. This is, of course, just one arbitrary example for many ambiguities which would arise.

An alternative design decision is illustrated in Figure 4.3, which shows another conceivable configuration corresponding to the same expression, $c_{et,1}(v_{e,0})$.

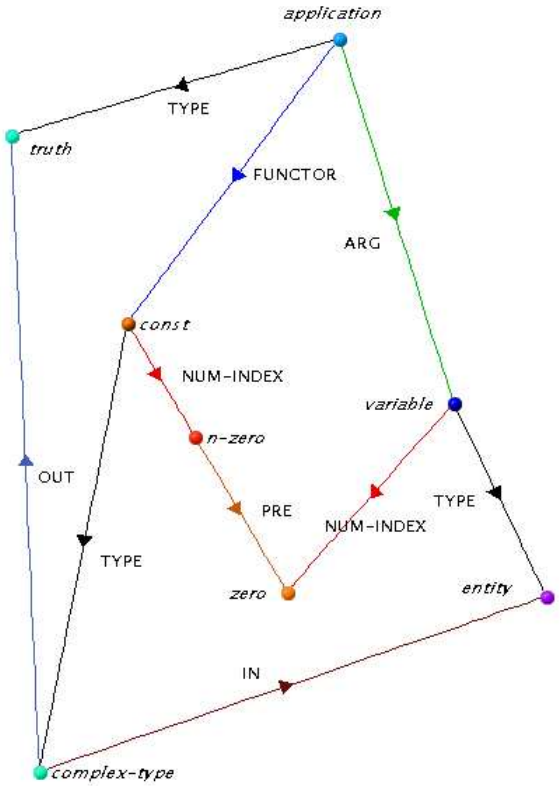

Figure 4.3: A minimized representation of $c_{et,1}(v_{e,0})$

The configuration in Figure 4.3 follows a minimality principle. It is minimal in the sense that each possible configuration under any entity in it occurs at most once. The idea behind this strategy corresponds to the weak extensionality of *elist* configurations in Chapter 2, generalized to complex term representations. The special case of atomic entities is the easiest case for illustrating the weak extensionality in terms. As a consequence of weak extensionality, each atomic entity may only occur once in a connected configuration: there is only one *truth* entity, one *zero* entity and one entity of sort *entity*. Depictions

of this kind of configurations are harder to read, but their minimized size is technically very attractive, in particular if we consider the use of relations over *ty2* configurations in grammatical principles.

To appreciate the compactness of this representation in the general case, consider a slightly more complex expression. If we extend the expression in our example by lambda abstraction to the new expression $(\lambda v_{e,0}.c_{et,1}(v_{e,0}))_{e,t}$, the size of the configuration increases by only one entity: there is an additional *abstraction* entity, which is the topmost entity in the configuration. Three arcs originate from it: a TYPE arc, pointing to the *complex-type* entity; a VAR arc, pointing to the *variable* entity; and an ARG arc, pointing to the *application* entity. The weakly extensional configuration corresponding to the Ty2 expression $\lambda v_{e,0}.c_{et,1}(v_{e,0})$ is depicted in Figure 4.4.

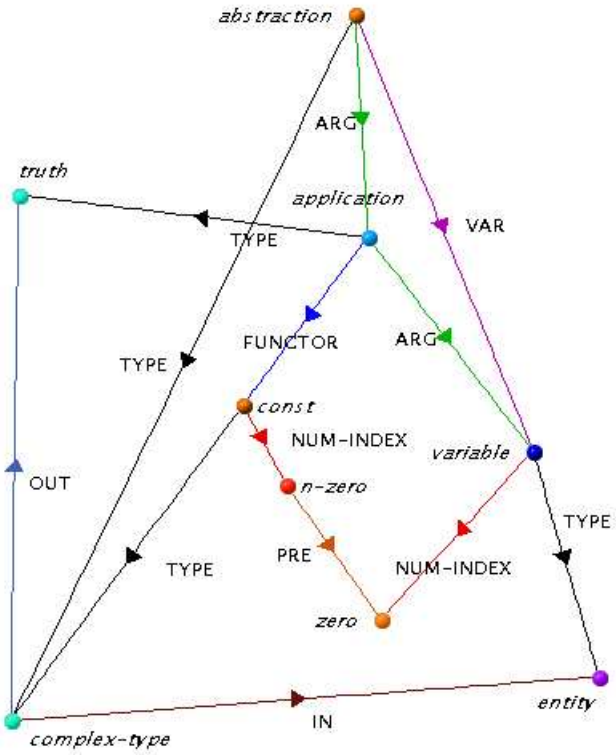

Figure 4.4: A minimized representation of $\lambda v_{e,0}.c_{et,1}(v_{e,0})$

Because of the technical advantages of weakly extensional term representations, the structure of *ty2* configurations illustrated in Figure 4.3 and Figure 4.4 is the one which I will choose. The grammar of Ty2 will thus not be modeled by the representation illustrated in Figure 4.2. The proofs which will show that the grammar successfully encodes the syntax of Ty2 will rely on the weak extensionality of the *ty2* configurations.

While the configurations of semantic representations in grammar models are the crucial level of representation for model-theoretic semantics in HPSG, it is also important to keep

the description level in mind. After all, grammatical principles will be formulated on the description level, and matters of notation are important for the convenience of a grammar framework. From this perspective, the Σ_{Ty2} formulae for the description of Ty2 expression look quite awkward at first, as the example in (31) readily demonstrates.

(31) A description that denotes the term $\lambda x_e.\mathsf{const}_{et}(x_e)$:

$$\left[\begin{array}{l} \textit{abstraction} \\ \textsc{type}\ \boxed{0}\left[\begin{array}{l}\textit{c-type}\\ \textsc{in}\ \ \boxed{1}\ \textit{entity}\\ \textsc{out}\ \boxed{2}\ \textit{truth}\end{array}\right] \\ \textsc{var}\ \ \boxed{3}\left[\begin{array}{l}\textit{var}\\ \textsc{type}\ \boxed{1}\end{array}\right] \\ \textsc{arg}\ \left[\begin{array}{l}\textit{application}\\ \textsc{func}\ \left[\begin{array}{l}\textit{constant}\\ \textsc{num-index pre}\ \textit{zero}\\ \textsc{type}\ \boxed{0}\left[\begin{array}{l}\textit{c-type}\\ \textsc{in}\ \ \boxed{1}\\ \textsc{out}\ \boxed{2}\end{array}\right]\end{array}\right] \\ \textsc{arg}\ \ \boxed{3}\end{array}\right]\end{array}\right]$$

The *abstraction* configuration in Figure 4.4 is in the denotation of the description in (31). Note that all tags in (31) could be omitted, and the *abstraction* configuration would still be in its denotation.

Fortunately it will be possible to define a much more perspicuous notation for the description of representations of *ty2* configurations. We will be able to show that in HPSG grammars with Ty2 we may simply write a Ty2 expression instead of its description, due to the properties of the grammar of Ty2. To see why this is the case, I will now introduce Γ_{Ty2} and study its properties.

All necessary restrictions on *types* are already given by the signature Σ_{Ty2}. In order to characterize the elements of the set $\mathbb{N}_0$, we only need a simple Natural Numbers Principle which prevents infinite configurations in the denotation of *integer*:

(32) The Natural Numbers Principle:

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

An *integer* configuration in models of Γ_{Ty2} consists of 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 this term corresponds to the natural number represented by the configuration under the *non-zero* entity.

In contrast to the simple *integer* configurations, configurations under *me* and *types* entities require a complex theory lest we permit configurations in models which do not correspond to expressions of Ty2. The Ty2 principles must enforce basic well-formedness conditions such as the proper typing of complex expressions and the finiteness of each

expression. They must also prevent oddities which become possible by the encoding of terms and complex types as configurations of entities. An oddity which can occur unless explicitly prevented is expressions or types which are a component of themselves because a Σ_{Ty2} term defined on their topmost entity has this entity as its value. Such a cyclic term is, of course, not a possible term of Ty2 according to DEFINITION 32. Finally, the principles of Ty2 must enforce the decision that isomorphic configurations in a *ty2* configuration must be identical.

The principles which impose the necessary type restrictions on the constituent expressions of logical operators and on the resulting expression are straightforward. All necessary restrictions are summarized in the COMPLEX TERM PRINCIPLES in (33).

(33) COMPLEX TERM PRINCIPLES:

a. *application* $\rightarrow$ $\begin{bmatrix} \text{TYPE } \boxed{2} \\ \text{FUNCTOR TYPE } \begin{bmatrix} \text{IN } \boxed{1} \\ \text{OUT } \boxed{2} \end{bmatrix} \\ \text{ARG TYPE } \boxed{1} \end{bmatrix}$

b. *abstraction* $\rightarrow$ $\begin{bmatrix} \text{TYPE } \begin{bmatrix} \text{IN } \boxed{1} \\ \text{OUT } \boxed{2} \end{bmatrix} \\ \text{VAR TYPE } \boxed{1} \\ \text{ARG TYPE } \boxed{2} \end{bmatrix}$

c. *equation* $\rightarrow$ $\begin{bmatrix} \text{TYPE } \textit{truth} \\ \text{ARG1 TYPE } \boxed{1} \\ \text{ARG2 TYPE } \boxed{1} \end{bmatrix}$

d. *negation* $\rightarrow$ $\begin{bmatrix} \text{TYPE } \textit{truth} \\ \text{ARG TYPE } \textit{truth} \end{bmatrix}$

e. *l-const* $\rightarrow$ $\begin{bmatrix} \text{TYPE } \textit{truth} \\ \text{ARG1 TYPE } \textit{truth} \\ \text{ARG2 TYPE } \textit{truth} \end{bmatrix}$

f. *quantifiers* $\rightarrow$ $\begin{bmatrix} \text{TYPE } \textit{truth} \\ \text{SCOPE TYPE } \textit{truth} \end{bmatrix}$

In an application the type of the overall expression is the OUT value type of the functor, and the IN value type of the functor is the type of the argument (33a). In abstractions the type of the variable is the IN type of the overall expression, and the OUT type of the overall expression equals the type of the argument (33b). Equations are of type *truth*, because they are either true or false, and the two arguments of the equation must be of the same type (33c). Negations are of type *truth*, and the argument of a negation must also be of type *truth* (33d). While it is necessary to list all the previous cases separately, the potential benefits of using a description logic for the specification of a logical object language first become apparent with the binary logical connectives. Using the common immediate supersort of *disjunction*, *conjunction*, *implication* and *bi-implication*, one principle is enough for

specifying that both arguments and the overall expression are of type *truth* in all four cases (33e). Similarly, all quantificational expressions are of type *truth*, and so is the expression in the scope of the quantifiers (33f).

The remaining three principles of θ_{Ty2} are more complex than the ones we have seen so far. They rely on auxiliary relations to express the desired shape of *ty2* configurations in models of Γ_{Ty2}. I will now present them in turn together with the relation principles which guarantee the necessary meaning of the relation symbols in Γ_{Ty2} models.

The first principle, the Ty2 Non-Cyclicity Principle, will exclude cyclic terms from models. Cyclic terms are terms which contain themselves as a component. A *ty2* configuration constitutes a cyclic term if there is an entity u in it such that there is a Σ_{Ty2} term of the form '$:\alpha_1 \ldots \alpha_n$', $1 \leq n$, defined on it which yields u as its value. Cyclic terms in this sense will be excluded because it is not clear what they should correspond to in the domain of Ty2 expressions.

To express the Ty2 Non-Cyclicity Principle, I define a binary relation `ty2-component` between each *ty2* entity in a Γ_{Ty2} model and its components:

(34) The Ty2-component Principle:[5]

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

According to (34), x is in the `ty2-component` relation with y just in case x equals y or there is an attribute α defined on y which leads to an entity $\boxed{1}$, and x and $\boxed{1}$ are in the `ty2-component` relation. In other words, the first element, x, in each pair in the relation is a component of the second, y.

The Ty2 Non-Cyclicity Principle requires for each *ty2* entity u in a Γ_{Ty2} model that there be no attribute value $\boxed{1}$ for any attribute α defined on u such that u is a component of $\boxed{1}$. In short, u is not a component of any of its attribute values:

(35) The Ty2 Non-Cyclicity Principle:[6]

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

The `ty2-component` relation is also used in the Ty2 Finiteness Principle which guarantees that all configurations which represent Ty2 expressions are finite. It is clear that a configuration is finite in case it has finitely many components. Therefore, we want to

[5] The relation `ty2-component` is obviously a close relative of the general `component` relation which was used in the u-sign Component Condition of Section 2.2.3 in Part I to give substance to the requirement that each entity be a component of a *u-sign* entity. The difference will become important below: `component` is defined for all entities in normal form grammars; `ty2-component` is a relation between entities which belong to *ty2* configurations.

[6] The expression `ty2-component`$(:, \boxed{1})$ is a legitimate abbreviation of a formula in the description language due to the Relation Argument Convention (Convention 8, page 141), which permits the use of the reserved symbol ':' in the argument slots of relational expressions.

say that each *ty2* entity in a Γ_{Ty2} model has finitely many components. Besides a relation which gives us access to all components of an entity, we then need a construct which ensures that the set of components is finite.

A convenient construct of RSRL for this purpose are chains, because chains are finite by definition.[7] If there is a chain which contains all components of an entity, it follows that the set of components of that entity is finite. All that is left to do is to require that there be a chain with all its components for each *ty2* entity in each Γ_{Ty2} model.

First of all, we need a `member` relation between entities and chains of entities which allows us to state that an entity is a member of a chain. The CHAIN MEMBER PRINCIPLE in (36) is, of course, just a slight variant of the MEMBER PRINCIPLE for a `member` relation between entities and lists of entities employed throughout Part I of this study.

(36) A CHAIN MEMBER PRINCIPLE for Ty2:

$$\forall\boxed{1}\forall\boxed{2}\left(\texttt{member}(\boxed{1},\boxed{2}) \leftrightarrow \left(\begin{array}{l}\boxed{2} \,\|\, \boxed{1} \,|\, \mathit{chain} \,\| \vee \\ \exists\boxed{3}(\boxed{2} \,\|\mathit{ty2}\, |\, \boxed{3} \,\| \wedge\ \texttt{member}(\boxed{1},\boxed{3}))\end{array}\right)\right)$$

An entity $\boxed{1}$ is a member of chain $\boxed{2}$ just in case it is the first element of the *chain*, or it is in the membership relation with the tail of the chain. The notation of the CHAIN MEMBER PRINCIPLE uses the Chain Convention (CONVENTION 6, page 140) for a simplified notation in the descriptions of chains.

The CHAIN MEMBER PRINCIPLE defines the meaning of `member` in a very idiosyncratic way. This is not a problem in the grammar Γ_{Ty2}, in which `member` is only used in the TY2 FINITENESS PRINCIPLE. However, the membership principle will have to be generalized when we characterize the general integration of Γ_{Ty2} with normal form HPSG grammars in Section 4.2, since grammarians will want to use the `member` relation as a relation between entities and lists of entities as well. For the purposes of Γ_{Ty2} the CHAIN MEMBER PRINCIPLE is sufficient.

The finiteness of Ty2 expressions is now very simple to express. We require that for each *ty2* entity u in a Γ_{Ty2} model, there be a chain, $\boxed{1}$, such that each component of u is a member of the chain $\boxed{1}$. Note that for each finite configuration there will be infinitely many chains which fulfill this requirement, because any element may occur any (finite) number of times on the chain. For the present purposes, however, it is sufficient to require the existence of an appropriate chain, uniqueness is not necessary.

(37) The TY2 FINITENESS PRINCIPLE:

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

The last principle of Ty2 for the well-formedness of *ty2* configurations is the TY2 IDENTITY PRINCIPLE, (39). With the TY2 IDENTITY PRINCIPLE maximal token identity in *ty2* expressions is guaranteed in order to obtain a unique shape of each configuration which represents a given Ty2 expression. There are no isomorphic copies of any subconfiguration in any *ty2* configuration.

[7] See page 22 for a brief discussion of chains in the RSRL formalism.

As for the previous principles, an auxiliary relation is necessary to formulate this last principle. The binary copy relation is a relation between entities u_1 and u_2 such that the configurations under u_1 and u_2 are isomorphic. The relevant conditions are formulated in the COPY PRINCIPLE.

(38) The COPY PRINCIPLE:

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

y is a copy of x if x and y have the same species (second line), and if an attribute α with value $\boxed{1}$ is defined on x, then α is also defined on y, and $\boxed{1}$ and the value of α on y, $\boxed{2}$, are in the copy relation (third line). Note that the copy relation is formulated with respect to the sorts and attributes of Σ_{Ty2}. This restriction will remain in place when we integrate Γ_{Ty2} with normal form grammars in the next section. The copy relation will then remain a relation which only considers those entities in *ty2* configurations which belong to the Ty2 signature. Although unembedded sign entities will become components of *ty2* entities under the EMBEDDED attribute of normal form grammars, they will not enter into the copy relation.

In the TY2 IDENTITY PRINCIPLE, we require that all pairs of entities which stand in the copy relation to each other be identical. There are no isomorphic copies of configurations in representations of Ty2 terms:

(39) The TY2 IDENTITY PRINCIPLE:

$$\mathit{ty2} \rightarrow \forall \boxed{1} \forall \boxed{2} \left(\mathtt{copy}(\boxed{1}, \boxed{2}) \rightarrow \boxed{1} = \boxed{2} \right)$$

With the presentation and discussion of the principles of Ty2 completed, I can now be precise about the terminology which I have used throughout this section and say what the RSRL grammar of Ty2, Γ_{Ty2}, is. Let θ_{Ty2} be the set of Σ_{Ty2} descriptions shown in (32)–(39). Then I call the grammar $\Gamma_{Ty2} = \langle \Sigma_{Ty2}, \theta_{Ty2} \rangle$ the *grammar of Ty2*.

[Sailer, 2003] proves the properties of the relationship between exhaustive Γ_{Ty2} models and expressions of Ty2 which we will rely on throughout the rest of this study. First of all, there is a special exhaustive Γ_{Ty2} model whose universe consists of the natural numbers (including zero), the set of types of Ty2 and the meaningful expressions of Ty2:

Proposition 10 [Sailer, 2003, p. 117] *There is an exhaustive* Γ_{Ty2} *model* $\mathsf{I}_{Ty2} = \langle \mathsf{U}_{Ty2}, \mathsf{S}_{Ty2}, \mathsf{A}_{Ty2}, \mathsf{R}_{Ty2} \rangle$ *such that*

$$\mathsf{U}_{Ty2} = \mathbb{N}_0 \cup \mathit{Types} \cup \bigcup_{\tau \in \mathit{Types}} \mathrm{Ty2}_{\tau}.$$

Sailer proves PROPOSITION 10 by constructing a Γ_{Ty2} model with the required universe and proving that it is an exhaustive Γ_{Ty2} model.[8] PROPOSITION 10 assures us that Γ_{Ty2} is on the right track. Given Sailer's result and the properties of exhaustive models, we know that all exhaustive Γ_{Ty2} models may only contain configurations which are descriptively indistinguishable from Sailer's standard model, except for the number of isomorphic copies of configurations corresponding to the numbers, types, and Ty2 expressions. In any case, there will always be at least one configuration in each exhaustive model which corresponds to any given element in $\mathbb{N}_0 \cup Types \cup \bigcup_{\tau \in Types} \mathrm{Ty2}_\tau$. Each exhaustive Γ_{Ty2} model functions as a model of Ty2.

The remaining results are important for the practical use of Γ_{Ty2} in grammar writing. First of all, descriptions of expressions of Ty2 are not very convenient to work with. Even if we assume for the moment that—as we will show—each Ty2 expression can be described with a description in $\mathbb{AVM}_0^{\Sigma_{Ty2}}$, we know already from the example in (31) that these descriptions can be very cumbersome and are certainly hard to read. Is there a way to simplify the notation to make grammars more perspicuous? The result reported in LEMMA 5 will eventually give us the desired simplification by allowing us to work with the standard notation of Ty2 expressions in grammatical descriptions just as if they were $\mathbb{AVM}_0^{\Sigma_{Ty2}}$ descriptions. It gives us a function which maps each Ty2 expression ϕ to a Σ_{Ty2} AVM description which denotes an indiscernibility class of entities in any exhaustive Γ_{Ty2} model. We will see in a moment that it in fact denotes the indiscernibility class to which ϕ belongs.

LEMMA 5 is an important step toward the crucial result in PROPOSITION 11. For an encoding of Ty2 in HPSG grammars to be useful, it is necessary that a linguist can refer to any expression of Ty2 in grammatical descriptions. If this were not possible, it could happen that certain lexical entries or principles could not be written, because of a descriptive gap in the description language relative to the expressions of Ty2 in the denotation of Γ_{Ty2}. PROPOSITION 11 guarantees that there are no descriptive gaps in $\mathbb{AVM}_0^{\Sigma_{Ty2}}$ relative to the expressions of Ty2. For any expression there is a way to refer to it (and to nothing else).

But the situation is even better. Sailer's function, symbolized as '*', from numbers, types and Ty2 expressions to Σ_{Ty2} descriptions, produces an element of $\mathbb{AVM}_0^{\Sigma_{Ty2}}$ for each natural number, type and expression of Ty2 such that the description denotes this natural number, type or expression of Ty2 in the standard exhaustive Γ_{Ty2} model.[9] With LEMMA 5

[8] Sailer's proof initially only considers a definition of the syntax of Ty2 without negation, the binary logical connectives and quantifiers, since these can be defined on the basis of application, abstraction and equation alone, as I have already pointed out above. However, Sailer later shows that a syntactic extension of Γ_{Ty2} which includes these constructs does not change the results [Sailer, 2003, Section 3.5]. Hence it is justified to refer to Sailer's result in connection with our definition of the meaningful expression of Ty2 and the properties of Γ_{Ty2}.

[9] My characterization of Sailer's function '*' is actually not entirely accurate, since he uses the RSRL formalism of [Richter, 2004a] instead of the new version of RSRL which I proposed in Part I. However, THEOREM 3 (page 149) tells us that the old and the new version of RSRL are equivalent. Whatever can be done with the descriptions of one can be done with the descriptions of the other. We can thus rest assured that '*' can be modified to produce appropriate Σ_{Ty2} AVM descriptions instead of Σ_{Ty2} descriptions.

we can see the relevance of this result. For any number, type or expression of Ty2, the description produced by '*' denotes the elements of the equivalence class of configurations in each exhaustive Γ_{Ty2} model which contains the entities which are indistinguishable from the number, type or expression of Ty2 which '*' started with. The practical value of this result is very high. If we want to refer to the expressions in grammatical descriptions, we may simply use the standard symbols for a Ty2 expression, natural number or type instead of a complicated AVM formula describing it.. We can do so, because we know that an appropriate description exists and can be determined by the function '*'.

Let us now take a look at some of the mathematical details. The function '*' from $\left(\mathbb{N}_0 \cup Types \cup \bigcup\limits_{\tau \in Types} \mathrm{Ty2}_\tau\right)$ to $\mathbb{AVM}_0^{\Sigma_{Ty2}}$ establishes a connection between the natural numbers, the set of types and the set of Ty2 expressions to Σ_{Ty2} AVM descriptions which we need for descriptive convenience in grammars. The function assigns a Σ_{Ty2} AVM description to each element in its domain. [Sailer, 2003, p. 128] points out that the finiteness, acyclicity and the maximal token identities which Γ_{Ty2} requires in the configurations of exhaustive Γ_{Ty2} models are crucial for allowing a definition of the function '*' which delivers the result cited in Lemma 5. Sailer writes '*' in postfix notation, and I follow his convention.

According to Lemma 5, the description which '*' assigns to each natural number, type and Ty2 expression denotes an equivalence class of entities in each Γ_{Ty2} model.

Lemma 5 [Sailer, 2003, p. 132] *For each exhaustive Γ_{Ty2} model $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $\phi \in \left(\mathbb{N}_0 \cup Types \cup \bigcup\limits_{\tau \in Types} \mathrm{Ty2}_\tau\right)$, for each $u_1 \in \mathsf{U}$, for each $u_2 \in \mathsf{U}$,*

if $u_1 \in \Delta_{\mathsf{I}}(\phi^*)$ and $u_2 \in \Delta_{\mathsf{I}}(\phi^*)$,
then $\langle u_1, \mathsf{I} \rangle$ and $\langle u_2, \mathsf{I} \rangle$ are congruent.

To understand Proposition 11, we must first introduce an auxiliary function, SR_{I}. For each exhaustive Γ_{Ty2} model $\langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, $SR_{\langle \mathsf{U},\mathsf{S},\mathsf{A},\mathsf{R} \rangle}$ is a function from the powerset of U to $\left(\mathbb{N}_0 \cup Types \cup \bigcup\limits_{\tau \in Types} \mathrm{Ty2}_\tau\right)$ which assigns to each indiscernibility class of objects $[u]$ in U the natural number, type or expression of Ty2 to which the objects in the class $[u]$ correspond. Proposition 11 guarantees that there is a formula in $\mathbb{AVM}_0^{\Sigma_{Ty2}}$ which can pick out precisely the equivalence class of objects $[u]$ in U which corresponds to any given Ty2 expression ϕ.

Proposition 11 [Sailer, 2003, p. 127] *For each exhaustive Γ_{Ty2} model $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $\tau \in Types$, for each $\phi \in \mathrm{Ty2}_\tau$, for some $\kappa \in \mathbb{AVM}_0^{\Sigma_{Ty2}}$,*

$$\Delta_{\mathsf{I}}(\kappa) = \left\{ u \in \mathsf{U} \,\middle|\, SR_{\mathsf{I}}([u]) = \phi \right\}.$$

The complete picture we obtain from this is as follows: For each natural number, type or expression of Ty2, the function '*' hands us a Σ_{Ty2} AVM description which singles out

the indiscernibility class of entities $[u]$ in each exhaustive Γ_{Ty2} model $\mathfrak{I}$ which the function $SR_{\mathfrak{I}}$ maps to the number, type or expression we started from. In other words, there is a syntactic equivalence between the indiscernibility classes of entities in each exhaustive Γ_{Ty2} model and the corresponding elements in the sets of natural numbers, types and expressions of Ty2.

Since we can exchange equivalence classes of descriptively indiscernible configurations in exhaustive Γ_{Ty2} models and their corresponding elements in the sets of natural numbers, types and Ty2 expressions syntactically, we can assign them the same meaning in models of Ty2. This is the way in which [Sailer, 2003, Section 3.3] defines the meaning of the *me* entities in Γ_{Ty2} models. Instead of the entire equivalence classes, we may also take the members of the indiscernibility classes of *me* entities in exhaustive Γ_{Ty2} models as good syntactic representations of Ty2 expressions and define the semantics of Ty2 for them. No matter which way one prefers to look at it, Γ_{Ty2} is obviously an adequate specification of Ty2.

It is also clear from the results of this section that the method which was used to encode the syntax of Ty2 as configurations of entities in exhaustive models of an RSRL grammar could be applied to any logic with a recursively defined formal language with expressions of finite length and finitely many logical operators and connectives. It opens the door to integrating any model-theoretic semantics of natural languages using this kind of formal language with normal form HPSG grammars. How this integration proceeds will be shown in the next section for the grammar of Ty2, Γ_{Ty2}.

4.2 Two-sorted Type Theory in Normal Form HPSG Grammars

In the previous section we specified an RSRL grammar of connected configurations in exhaustive models which were—or corresponded to—the expressions of the languages of two-sorted type theory. So far this has only been a logical specification of the syntax of a formal language with no immediate significance for grammars of natural languages. The linguistically interesting step comes when we combine Γ_{Ty2} with normal form HPSG grammars in order to symbolize the truth-conditional semantics which we intend to assign to linguistic expressions in accordance with a linguistic theory about their meaning. As we will see, there are different conceivable ways to use Ty2 for this purpose in HPSG. In the present section, we will specify the general architecture which all of them have in common.

Assume that $\Gamma = \langle \Sigma, \theta \rangle$ is an arbitrary normal form HPSG grammar in the sense of Chapter 2 with $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$. Further assume that Σ is a signature with lists as described on page 139 above, and the sort hierarchy, $\langle \mathcal{G}, \sqsubseteq \rangle$, has a top sort subsuming all other sorts. I will refer to the top sort with the symbol *top*. We will now characterize in general terms *normal form HPSG grammars with Ty2*, or simply *HPSG grammars with Ty2*.

First of all, recall that normal form signatures obey the restrictions of Section 2.2.3.

Normal form signatures include sort symbols and a hierarchy for unembedded signs, and the attribute EMBEDDED is required to be appropriate to all sorts in the sort hierarchy. In addition the signature of *HPSG grammars with Ty2* shall incorporate the Ty2 signature $\Sigma_{Ty2} = \langle \mathcal{G}_{Ty2}, \sqsubseteq_{Ty2}, \mathcal{S}_{Ty2}, \mathcal{A}_{Ty2}, \mathcal{F}_{Ty2}, \mathcal{R}_{Ty2}, \mathcal{AR}_{Ty2} \rangle$ of Figure 4.1. This incorporation must obey the following rules. The sort *top* immediately subsumes the top sort *ty2* of $\langle \mathcal{G}_{Ty2}, \sqsubseteq_{Ty2} \rangle$, and no other sort in the hierarchy which is not in $\mathcal{G}_{Ty2}$ subsumes *ty2* or any subsorts of it. The species in $\mathcal{S}_{Ty2}$ are species in $\mathcal{S}$. Moreover, with the exception of the sort *me*, which we will turn to below, no sort in $\mathcal{G}_{Ty2}$ is appropriate for any attribute in $\mathcal{A}$ except as specified by $\mathcal{F}_{Ty2}$. Finally, no additional attribute is appropriate to any of the sorts in $\mathcal{G}_{Ty2}$ besides the attribute EMBEDDED and the attributes specified appropriate to the sorts in $\mathcal{G}_{Ty2}$ by $\mathcal{F}_{Ty2}$.

Informally we may characterize the incorporation of Σ_{Ty2} into a normal form signature Σ as the insertion of a signature module Σ_{Ty2} which exhibits only a minimal interaction with the rest of the signature. The sort hierarchy of the grammar of Ty2 is inserted under the top element in the sort hierarchy of the HPSG grammar with Ty2, and there is no interaction between the Ty2 part of the sort hierarchy and the other parts of it in terms of the appropriateness function, except for the connection established by the attribute EMBEDDED and the sort *me*. The attribute EMBEDDED will serve as the anchor of the Ty2 terms in the unembedded sign to which they will belong, and entities of sort *me* will be reachable by the interpretation of some Σ term defined on signs. In typical HPSG grammars in the tradition of the grammar of English in [Pollard and Sag, 1994], it is the attribute CONTENT for which we will declare the sort *me* appropriate. Terms of Ty2 then replace the uninterpreted semantic representations of traditional HPSG grammars as CONTENT values. This property of an HPSG grammar with Ty2 deserves to be recorded as a separate statement:

(40) For each HPSG grammar $\langle \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle, \theta \rangle$ with Ty2, for some sort $\sigma \in (\mathcal{G} \backslash \mathcal{G}_{Ty2})$, for some attribute $\alpha \in (\mathcal{A} \backslash \mathcal{A}_{Ty2})$,

$$\mathcal{F}(\sigma, \alpha) = me.$$

The theory θ of an HPSG grammar with Ty2, $\langle \Sigma, \theta \rangle$, must include the RSRL theory of the natural numbers, of the type system and of the meaningful expressions of Ty2 which we formulated in the theory θ_{Ty2}. Essentially this means that I will require that $\theta_{Ty2} \subset \theta$. However, there is one minor modification in the theory θ_{Ty2} when it is combined with a normal form grammar with lists. The TY2 FINITENESS PRINCIPLE in the theory of meaningful expressions refers to a binary `member` relation. The MEMBER PRINCIPLE of θ_{Ty2} defines `member` as a relation between entities and chains. HPSG grammars with lists use the same `member` relation simultaneously as a relation between entities and lists. To accommodate this twofold usage of the `member` relation, the version of the MEMBER PRINCIPLE in (41g) below uses the Tape Convention (CONVENTION 7, page 141) for defining it between entities and tapes. A tape is either a chain or a list.[10]

[10]Some grammars additionally use `member` as a relation between entities and sets. The grammar of

The theory of the Ty2 module in HPSG grammars with Ty2 is summarized in (41). (41a) is the theory of natural numbers. The principles (41b)–(41e) guarantee that, in models of the grammar, all configurations of entities under an entity which are in the denotation of the sort *type* and *me* are well-formed types and expressions of Ty2. (41f)–(41h) are three relation principles for the three relations necessary in the theory of well-formed types and Ty2 terms, `ty2-component, member` and `copy`.

(41) a. The Natural Numbers Principle:

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

b. The Complex Term Principles:

$$\mathit{application} \rightarrow \begin{bmatrix} \text{TYPE} & \boxed{2} \\ \text{FUNCTOR TYPE} & \begin{bmatrix} \text{IN} & \boxed{1} \\ \text{OUT} & \boxed{2} \end{bmatrix} \\ \text{ARG TYPE} & \boxed{1} \end{bmatrix}$$

$$\mathit{abstraction} \rightarrow \begin{bmatrix} \text{TYPE} & \begin{bmatrix} \text{IN} & \boxed{1} \\ \text{OUT} & \boxed{2} \end{bmatrix} \\ \text{VAR TYPE} & \boxed{1} \\ \text{ARG TYPE} & \boxed{2} \end{bmatrix}$$

$$\mathit{equation} \rightarrow \begin{bmatrix} \text{TYPE} & \mathit{truth} \\ \text{ARG1 TYPE} & \boxed{1} \\ \text{ARG2 TYPE} & \boxed{1} \end{bmatrix}$$

$$\mathit{negation} \rightarrow \begin{bmatrix} \text{TYPE} & \mathit{truth} \\ \text{ARG TYPE} & \mathit{truth} \end{bmatrix}$$

$$\mathit{l\text{-}const} \rightarrow \begin{bmatrix} \text{TYPE} & \mathit{truth} \\ \text{ARG1 TYPE} & \mathit{truth} \\ \text{ARG2 TYPE} & \mathit{truth} \end{bmatrix}$$

$$\mathit{quantifiers} \rightarrow \begin{bmatrix} \text{TYPE} & \mathit{truth} \\ \text{SCOPE TYPE} & \mathit{truth} \end{bmatrix}$$

c. The Ty2 Non-Cyclicity Principle:

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

d. The Ty2 Finiteness Principle:

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

e. The Ty2 Identity Principle:

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

English in [Pollard and Sag, 1994] is a prominent example for this practice. It requires a Member Principle which defines `member` as a binary relation between entities and tapes or sets. In those grammars (41g) must be replaced by a Member Principle such as the one presented in [Richter, 2004a, p. 283].

f. The Ty2-component Principle:

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

g. The Member Principle for grammars with lists:

$$\forall \boxed{1} \forall \boxed{2} \left(\texttt{member}(\boxed{1}, \boxed{2}) \leftrightarrow \left(\begin{array}{l} \boxed{2} \ll \boxed{1} \mid \mathit{chain} \gg \vee \\ \exists \boxed{3} (\boxed{2} \ll \mathit{top} \mid \boxed{3} \gg \wedge \texttt{member}(\boxed{1}, \boxed{3})) \end{array} \right) \right)$$

h. The Copy Principle:

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

Note that the Ty2 Non-Cyclicity Principle, Ty2 Finiteness Principle and the Ty2 Identity Principle are blind to the fact that the expressions of Ty2 are now embedded in *u-sign* configurations in grammar models. The `ty2-component` relation and the `copy` relation take into account only attributes and sorts in $\mathcal{A}_{Ty2}$ and $\mathcal{S}_{Ty2}$. This means that non-cyclicity is only enforced for the expressions of Ty2 and not for the unembedded sign in which they occur. Analogously, it is only the terms that must be finite, not the complete u-sign configuration. Finally, maximal token identity is only required in the *ty2* configurations within unembedded sign configurations. It does not affect the other parts of the configurations.

HPSG grammars with Ty2 are no longer neutral with respect to certain fundamental assumptions in a theory of natural language semantics. With the choice of Ty2 we adopt the position that expressions of natural languages can be assigned a truth-conditional semantics in the tradition of Montague Semantics. At the same time, however, the first potential differences to Montague's semiotic program [Montague, 1974c] appear. According to the theory of normal form grammars developed in Part I, the empirical subject of linguistic research are unembedded sign configurations. Unembedded sign configurations comprise tectogrammatical structure, the local syntactic structure of signs and phonological structure. Words have morphological structure. With the Ty2 component of signs we add meaning to the properties of signs. In a given context of use, unembedded signs have a truth-conditional denotation which is given by expressions of Ty2 which are part of them.

Crucially, this general architecture does not say anything yet about the relationship between the syntactic structure of unembedded signs and their semantic structure. Disregarding for a moment the possibility that the general architecture permits several meaning-defining Ty2 expressions in a given sign, there is no guarantee that there is a homomorphism from syntactic structure to semantic structure. In particular, there might be an arbitrary number of unembedded sign configurations in a minimal exhaustive model of a grammar which have isomorphic syntactic structures and differ only with respect to their Ty2 terms. In other words, although they have the same syntactic structure, they have distinct denotations in the same contexts. The idea of regarding meaning specifications as a property

of unembedded sign configurations is clearly different from the position which Montague adopts in his theory of universal grammar [Montague, 1974c]. In the architecture of universal grammar the interpretation of a syntactic expression of a natural language is obtained directly and does not depend on logical representations, which only occur for convenience as optional translations of syntactic expressions.

Alternatively, one might adopt a different perspective on unembedded sign configurations and say that unembedded sign configurations as a whole are the syntactic level of representation. In this case, one would view the terms of Ty2 in unembedded signs as part of their syntactic form, in analogy to logical form semantics in the GB tradition. In Montagovian terms, unembedded signs with terms of Ty2 then constitute a disambiguated syntactic structure. The terms of Ty2 are not the result of a translation of the syntax into a (dispensable) logical language. The interpretation of the disambiguated unembedded signs is given by the Ty2 expressions which are a component of them. While this seems to be a position which is defendable on technical grounds, it is not in the spirit of Montague's conception of universal grammar, in which a categorial syntax is mapped by a homomorphic function to the structures in the interpreting domain and, optionally, to the algebra of a logical representation language such as Intensional Logic or Ty2.

What these conceptual differences mean for actual grammars is a different question. For theoretical reasons one might decide to specify an HPSG grammar with Ty2 in such a way that the Montagovian relationship between syntax and semantics is respected. One would then write grammars with a disambiguated syntax of syntactic structures, and there would be no pair of unembedded sign configurations with isomorphic syntax in a minimal exhaustive model with distinct Ty2 terms. This would preserve the translation from the syntactic base to the intermediary logical terms of Ty2. Whether or not this is a preferable strategy can ultimately only be determined by the success or failure of the resulting grammars relative to their predictions in the empirical domain of natural languages. It is one of the virtues of HPSG to provide a framework which is at the same time explicit and expressive enough to develop precise grammars with competing architectures and to test concrete alternative theories of the relationship between syntactic and semantic structures and alternative theories of semantic composition.

In the next two chapters, we will see two significantly different ways of defining semantic composition in the present framework. The first is inspired by the Flexible Montague Grammar of [Hendriks, 1993] and is thus a reconstruction of a theory which stands directly in the tradition of Montague Grammar. The properties of the resulting system, LF-Ty2, motivate the revisions in the mechanisms of semantic composition which lead to LRS in Chapter 6. In contrast to LF-Ty2, LRS exploits the specific descriptive means of a model-theoretic grammar framework in an effort to develop an empirically and conceptually more satisfactory architecture of semantics.

Chapter 5

Lexicalized Flexible Ty2

Based on the *normal form HPSG grammars with Ty2* introduced in the previous chapter, I will now explain a first general framework which combines an HPSG syntax with a model-theoretic semantics and can assign meaning to an infinite collection of utterances. The framework is called *Lexicalized Flexible Ty2* (LF-Ty2) and was first presented at an early stage of its development in [Richter and Sailer, 1999a] in an analysis of sentential negation and negative concord in French. With the present version of LF-Ty2 I will largely follow [Sailer, 2003] with only minor revisions. Sailer developed earlier versions of LF-Ty2 into a framework which translated large portions of Flexible Montague Grammar (FMG) of [Hendriks, 1993] faithfully into an HPSG grammar. Sailer's version of LF-Ty2 pairs semantic representations with expressions of a fragment of English in such a way that they correspond directly to the predictions of FMG. Sailer explores several varieties of the LF-Ty2 system, each one adopting conceptually slightly different design options for the technical realization of the semantic composition mechanisms of FMG in HPSG. I will choose a version of LF-Ty2 which I consider well suited for a brief and informal presentation of the system. My choice should not be taken as an indication of my preferences among the various possible realizations of LF-Ty2.[1] The crucial aspect of the present discussion of LF-Ty2 is not the precise specification of the relevant composition structures; the crucial aspect is the investigation of a classical Montague Semantics as part of a model-theoretic syntactic grammar framework. In particular, I am interested in the relationship between syntax and semantics which emerges when the ontology of abstract syntactic and semantic linguistic entities is observed from the perspective of model-theoretic syntax.

FMG uses Intensional Logic (IL) as the translation language of the syntactic algebra. It differs from Montague Grammar in not postulating the same semantic type for the translation of every expression in a given lexical category. The strategy of choosing a uniform semantic type for the translations of all words in each syntactic category in Montague Grammar leads to the strategy of *generalizing to the worst case*. This term describes the

[1] Quite to the contrary, I am very sympathetic with Sailer's chain encoding of Ty2 expressions, which eschews a reification of beta reduction and alpha conversion in components of utterances [Sailer, 2003, Section 4.2.2]. However, an explanation of this inspired exercise in logic programming in RSRL would take us much too far afield. The reader is encouraged to read the original work.

fact that for each syntactic category C a type must be chosen for the basic translation of each word of category C which is adequate for those members of category C which require the highest typing. The higher types of some members of the syntactic category are necessary to obtain the right combinatorial properties of these elements to describe the full range of their combinatorial potential. It has been observed in this context that the higher types of one category often also infect other categories with which they need to combine. The strategy of generalizing to the worst case is thus very visible in the typing of the basic translations of classical Montague Grammars. To quote a prominent example (in form of the Ty2 counterpart of the IL term originally used), this strategy leads to a situation in which a proper name such as *Uther* is assigned the translation $\lambda P_{s((se)t)}.P_{@}(\lambda @.\mathsf{uther}_e)$ instead of the much simpler translation uther_e, because noun phrases are uniformly analyzed as generalized quantifiers, and proper names must thus receive a translation of the type of a generalized quantifier. The term $\lambda P_{s((se)t)}.P_{@}(\lambda @.\mathsf{uther}_e)$ has the type of a generalized quantifier, whereas the simple and more intuitive term uther_e does not.

In FMG, each expression is assigned a basic translation of a minimal type. Proper names such as *Uther* receive a basic translation of type e such as the non-logical constant uther_e. In order to obtain all necessary combinatorial possibilities in semantics, FMG introduces the new mechanism of *type shifting*. By means of the application of a small set of type shifting rules, the type of all expressions in each family of expressions of type τ can be raised or lowered, and we obtain systematically related expressions of a higher or lower type τ'. This means that each word is associated with a basic translation and with an infinite family of systematically related expressions which are obtained through the iterated application of type shifting rules to the basic translation. Starting from this collection of terms associated with words, the term associated with each phrase is obtained by intensional functional application of the term of one syntactic daughter to the term of another syntactic daughter. Whether intensional functional application with two given terms is possible is determined, of course, by the types of the two constituents. The resulting combinatorial system stands in the tradition of the type-driven approach first advocated by [Klein and Sag, 1985] in the context of Generalized Phrase Structure Grammar (GPSG). In GPSG it was introduced to avoid the alternative solution of a much more complicated individual pairing of syntactic structures licensed by general ID rules with semantic translation rules. In a very graphic formulation Hendriks calls the principle guiding the combinatorics of functional application in type-driven systems the "survival of the fitting".

The type-driven architecture of FMG is particularly well suited for an integration with a linguistic syntax which does not mirror semantic ambiguities in its syntactic tree structures. In this sense, FMG is reminiscent of the strategy of [Cooper, 1975], which introduced a storage mechanism in a transformational syntax to treat scope ambiguities without the syntactic ambiguities necessary in Montague's technique of *quantifying in* the generalized quantifiers. However, FMG does not even need a storage mechanism, since it captures the ambiguities through type shifting of logical expressions. The sparseness of the adopted syntactic tree structures facilitates the integration of the compositional semantic system of FMG with the syntactically motivated phrase structure-like tectogrammatical structures of normal form HPSG grammars with Ty2. The conformity of FMG with the syntactic

tectogrammatical base of HPSG grammars was originally the main motivation for choosing FMG as the first system for a model-theoretic semantics in HPSG.

There are a few notable differences between FMG and its HPSG relative, the system LF-Ty2. The most obvious difference is the choice of the logical language. In contrast to FMG, which uses IL, LF-Ty2 uses Ty2. Since the translation of expressions from IL into Ty2 is straightforward, this is unproblematic. Secondly, LF-Ty2 imposes a restriction on the type shifting mechanism. Whereas FMG allows type shifting to apply to basic translations and to expressions derived by intensional functional application, LF-Ty2 restricts type shifting to basic translations. This means that type shifting occurs only in words or at the lexical level in LF-Ty2. A result from [Hendriks, 1993] guarantees that this is not a restriction on the logical expressions which can be derived by the system. The reason for the restriction on type shifting in LF-Ty2 is that unrestricted type shifting at the level of phrases would introduce spurious ambiguities in the unembedded sign configurations predicted by the grammars. We would obtain unembedded sign configurations with identical syntactic structures and semantic interpretation which would differ structurally in their relational structures, in which the application of type shifting is represented. There would be no possible empirical criterion to distinguish between these non-isomorphic configurations. Hence they are undesirable for methodological reasons. Finally, the fragment of LF-Ty2 presented below does not incorporate all the type shifting rules of Hendriks. However, there is no principled reason for omitting some of them, and LF-Ty2 could be extended to capture larger portions of FMG.

In this chapter I will precede as follows. In Section 5.1 I will introduce the mechanisms of semantic composition in LF-Ty2. They are directly taken over from FMG. They consist of basic translations of the words in the fragment and of type shifting rules which can freely apply to the basic translations and to the expressions resulting from intensional functional application of one basic or derived expression to another. Intensional functional application is the general mechanism of semantic composition throughout. The system will be illustrated without reference to a syntactic component, since the syntax will later be provided by normal form grammars. The syntax will come into play in Section 5.2, in which I present an RSRL encoding of the composition mechanism and combine the HPSG counterpart of the mechanisms of Section 5.1 with the syntax of a normal form grammar with Ty2. This will give us the opportunity to inspect a Montagovian Semantics in HPSG. The final section of this chapter, Section 5.3, is devoted to an investigation of the properties of LF-Ty2. I will highlight some of the encoding techniques and discuss the structure and properties of natural languages which emerge from studying the unembedded sign configurations in exhaustive models of LF-Ty2 grammars. The relationship of unembedded signs to model-theoretic semantic interpretation will give rise to an investigation of their relationship to Montague's architecture of universal grammar and of the function which the lambda calculus plays in the two systems. I will outline the reasons why the lambda calculus might not be an optimal technique of semantic composition in the framework of normal form grammars with Ty2. In addition to the conceptual and technical arguments, there are even more important empirical arguments for abandoning the mechanisms of semantic composition of LF-Ty2. In addition to the grammar-theoretic considerations in

this chapter I will cite data from Polish and from German in Chapter 6 which suggest in combination with certain assumptions about syntax inherent to the HPSG framework that a more adequate description of the data can be achieved by exploring composition mechanisms familiar from model-theoretic syntax.

My focus in this chapter will be on a conceptual level. I am ultimately interested in the question of which kind of architecture of semantic composition best fits the ontological assumptions underlying the framework of normal form grammars about the unembedded sign configurations which represent their predictions. What are the consequences of the mechanisms for semantic composition of LF-Ty2 for the structure of unembedded sign configurations? The meta-theory of normal form grammars postulates that unembedded sign configurations embody the empirical predictions of grammars. Structurally different unembedded sign configurations in a minimal exhaustive model of a grammar typically reflect different predictions of the grammar. How well does the architecture of semantic composition in LF-Ty2 respect this postulate? What do the results entail for how we can interpret the meaning of LF-Ty2 grammars as predictions about the empirical domain of utterances of a natural language? How satisfactory are the minimal exhaustive models of LF-Ty2 grammars in this respect? In light of the answers to these questions, I will then investigate what kinds of techniques LF-Ty2 uses to specify the meaning of signs, and how they compare to the specification techniques we generally find in constraint-based grammar frameworks.

The main contribution of this chapter will thus be the critical discussion of the architecture of LF-Ty2 in Section 5.3. Following up on the inclusion of Ty2 in HPSG grammars in Chapter 4, I will investigate a successful technique of compositional semantics which I import into HPSG in order to observe the behavior of the resulting system in the new, model-theoretic syntactic environment. Independent of the results of this investigation, LF-Ty2 marks an important improvement on our previous normal form grammars and on HPSG grammars with uninterpreted semantic representations. LF-Ty2 grammars have a distinguished attribute value in each sign which is a Ty2 expression. The model-theoretic interpretation of this expression is the meaning of the sign. In this way each unembedded sign receives a precise meaning specification which can be subjected to empirical tests. Nothing like this can be done with uninterpreted syntactic representations.

Throughout the discussion I will presuppose familiarity with Montague Semantics and the mathematical concepts it uses. References to the literature will serve as guides to those basic definitions which are not repeated here. For our purposes it will not be necessary to illustrate all details of LF-Ty2. Complex relation principles will occasionally be skipped, since their exact form is of no consequence for the overall architecture of the system. All of these omissions will be duly indicated.

5.1 Semantic Composition

In this section I will present the basic architecture of semantic composition in Hendriks's Lexicalized Flexible Montague Grammar to the extent in which I need it for integrating an

adaptation of FMG into normal form grammars with Ty2 in Section 5.2. The presentation presupposes familiarity with the basics of the typed lambda calculus such as beta reduction and alpha conversion. Beta reduction is also known as beta contraction or lambda conversion. Alpha conversion is sometimes called the renaming of bound variables.[2] Extensive examples with complete computations for all constructions which I will discuss can be found in [Sailer, 2003, pp. 46–56].[3]

The architecture of semantic composition comprises the following three main features:

- There is a basic translation for every word in the grammar.
- A small number of type shifting rules can freely and iteratively apply to the basic translation of words.[4]
- The logical form of each phrase is obtained by intensional functional application of the logical forms of its daughters.

For simplicity I assume that all structures are binary branching. Intensional functional application means that the functor remains as given by the daughter which contributes the functor, and the argument expression is obtained by lambda abstracting over the distinguished world variable, @. The expression contributed by the second daughter is the argument of the abstraction. To give a concrete example, assume the semantic functor daughter contributes the term α and the semantic argument daughter contributes the term β. Then the intensional functional application of α to β is $\alpha(\lambda@.\beta)$. This means that the type of the functor is generally of the form $\langle\langle s, \tau_2\rangle, \tau_1\rangle$, with τ_2 the type of the semantic argument daughter.

In (42) I enumerate the basic translations of a few words in a small fragment of English. There are intransitive verbs, (42a); different kinds of transitive verbs, (42b)–(42c); count nouns, (42d)–(42e); proper names, (42f); quantifiers, (42g)–(42h); and the complementizer *that*, (42i).

(42) a. $walks \rightsquigarrow \lambda x_{se}.\mathsf{walk}'_{@}(x_{@})$

b. $reads \rightsquigarrow \lambda y_{se}\lambda x_{se}.\mathsf{read}'_{@}(x_{@}, y_{@})$

c. $believes \rightsquigarrow \lambda p_{st}\lambda x_{se}.\mathsf{believe}'_{@}(x_{@}, p)$

d. $book \rightsquigarrow \lambda y_{se}.\mathsf{book}'_{@}(y_{@})$

e. $student \rightsquigarrow \lambda y_{se}.\mathsf{student}'_{@}(y_{@})$

[2] All definitions which I will use can be found in [Hindley and Seldin, 1986, pp. 1–7]. Here they are specialized to the typed lambda calculus.

[3] A few errors in Sailer's Figure 1.12 [Sailer, 2003, p. 50] are corrected in the corresponding Figure 5.2 below. Many thanks are due to Manfred for giving me the tex sources of his dissertation so I could copy the layout of his trees for my figures in this section.

[4] Recall that this is a restriction LF-Ty2 introduces without loss of generality, as known from a result by [Hendriks, 1993, p. 126].

f. $mary \rightsquigarrow \mathsf{mary}_e$

g. $some \rightsquigarrow \lambda P_{s((se)t)} \lambda Q_{s((se)t)}.\exists x_{se}[P_@(x) \wedge Q_@(x)]$

h. $every \rightsquigarrow \lambda P_{s((se)t)} \lambda Q_{s((se)t)}.\forall y_{se}[P_@(y) \rightarrow Q_@(y)]$

i. $that \rightsquigarrow \lambda p_{st}.p_@$

On the basis of the basic translations, we can already derive very simple sentences. Figure 5.1 shows how the reading of the familiar sentence *Uther walks* is obtained from intensional functional application of the basic translation of *walks* to the basic translation of *Uther*. The tree in the figure can be read as a tectogrammatical tree whose nodes are annotated with the logical expressions indicating the meaning of the corresponding signs. The category symbols NP, V, S, as well as VP in later figures, give an intuitive description of the appropriate syntactic categories of the signs at the nodes. In addition, I will follow [Sailer, 2003] in recording term manipulations at each node. In Figure 5.1 there is only one. At the sentence node, S, the top expression is derived from the term indicating the intensional functional application by a sequence of beta reductions. This sequence of term manipulations is indicated by the arrow symbol followed by a lambda, '$\uparrow \lambda$'. Another possible term manipulation indicated by the same symbols is alpha conversion.

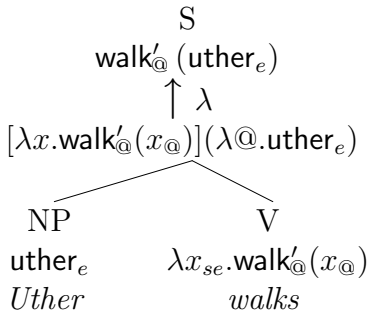

Figure 5.1: The expression *Uther walks*

Figure 5.1 also illustrates how the meaning of a phrase is derived by type-driven functional application rather than by an individual translation rule for each syntactic rule as in [Montague, 1974b]. The only possible functional application is the one in the figure. The term uther_e cannot apply to $\lambda @ \lambda x.\mathsf{walk}'_@(x_@)$ because of a type mismatch: an expression of type e cannot be the functor of an expression of type $\langle s, \langle \langle se \rangle, t \rangle \rangle$.

In order to capture sentences with transitive verbs and two quantificational expressions, we need to introduce a type shifting rule.

The argument raising rule, AR, in Definition 33 is cited after [Sailer, 2003, p. 147]. It is actually a set of rules, because each rule AR_i allows type raising of the ith argument which undergoes the rule:

Definition 33 *For each $i \in \mathbb{N}$, AR_i is a relation between two expressions α and β such that*

if α is of type $(a_1(\ldots((sa_i)(\ldots(a_nb)\ldots))))$,
then β is an expression of the form
$\lambda x_{a_1,1}\ldots\lambda X_{s((s((sa_i)b))b),i}\ldots\lambda x_{a_n,n}.X(@)(\lambda@\lambda x_{sa_i,i}.\alpha(x_1)\ldots(x_i)\ldots(x_n))$.

Argument raising relates two expressions of different types, which means that it is not meaning preserving. Intuitively, it raises the type of the argument of a functor so that the functor can combine with an argument which would otherwise be of a higher type than the argument slot permits. How this works can best be understood by considering examples. Figures 5.2 (page 192) and 5.3 (page 193) illustrate the derivation of the two readings of the sentence *Every student reads some book*. The two readings differ with respect to the scope of the two quantifiers. Both readings involve argument raising.

Figure 5.2 illustrates the reading in which the universal quantifier takes scope over the existential quantifier. I call this reading the $\forall\exists$ reading. For each student, there is a potentially different book such that the student reads it. As can be seen in Figure 5.2, the reading can be derived by the application of argument raising to the first argument (λy) of the basic translation of verb *reads*.[5] The symbol '↑ AR1' indicates application of argument raising to the first argument. There are also lambda conversions involved in this step to reduce the expression resulting from argument raising to the equivalent minimal expression. These conversion steps are never indicated separately in the figures when a type shifting rule is applied.

In the second reading of the sentence, depicted in Figure 5.3, the existential quantifier takes scope over the universal quantifier ($\exists\forall$ reading). There is one particular book such that every student reads it. To derive this reading, it is necessary to apply argument raising to the second argument of the basic translation of *reads* first (application to the logical subject), immediately followed by an application of argument raising to the first argument (the logical object). As can be seen in the figure, this yields an expression of a type which allows first intensional functional application to the expression specifying the meaning of the quantifier *some book* and then intensional functional application of the resulting expression to the expression specifying the meaning of the subject, *every student*.

The second type shifting rule is value raising, cited here again from [Sailer, 2003, p. 144] in a version compatible with Ty2:

Definition 34 *For each type $\tau \in Types$, VR_τ is a relation between two expressions α and β such that*

if α is of a type $a_1(\ldots(a_nb)\ldots)$,
then β is an expression $\lambda x_{a_1,1}\ldots\lambda x_{a_n,n}\lambda u_{s((sb)\tau)}.u(@)(\lambda@.\alpha(x_1)\ldots(x_n))$.

[5]Note that the logical object of the verb is its first semantic argument, and its logical subject is its second argument.

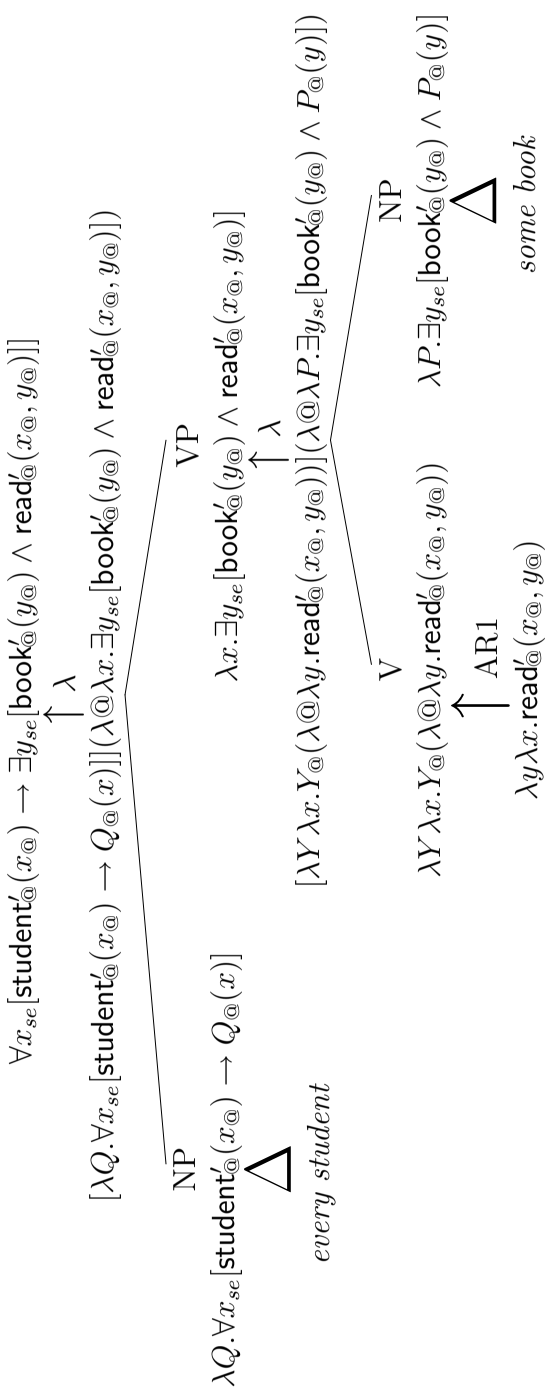

Figure 5.2: The ∀∃ reading of the sentence *Every student reads some book*

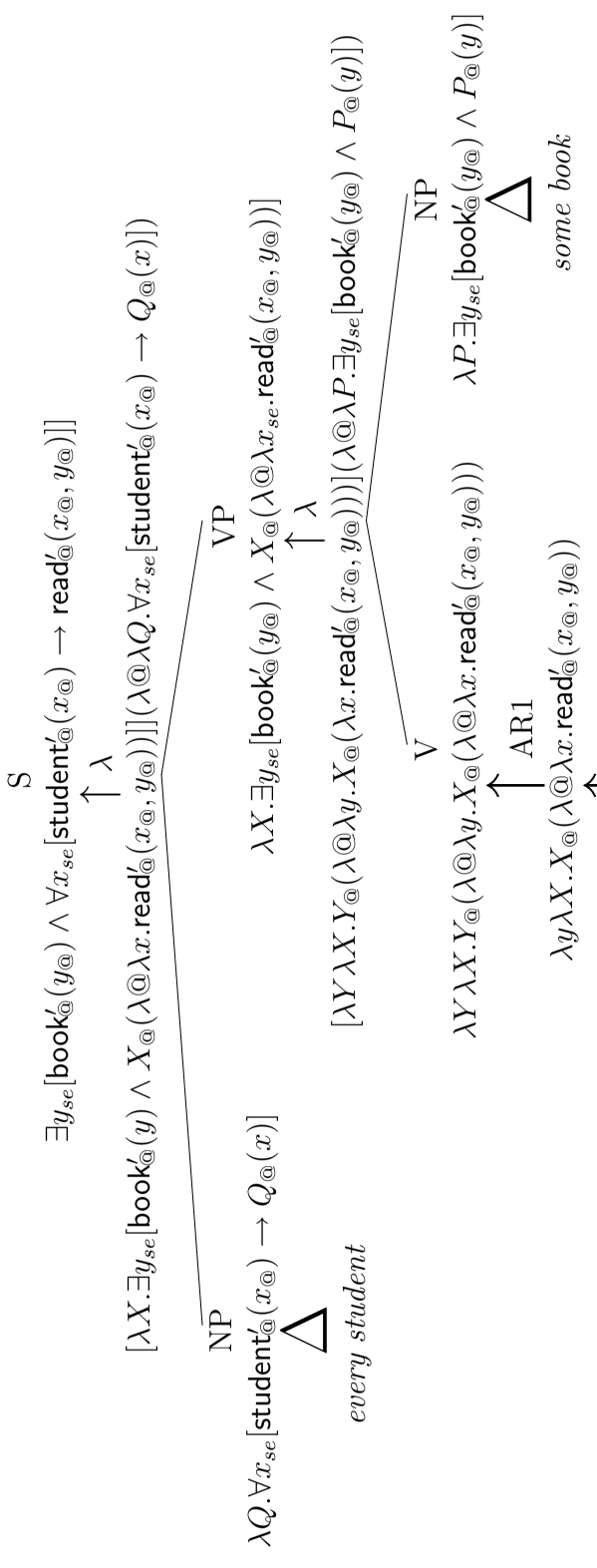

Figure 5.3: The $\exists\forall$ reading of the sentence *Every student reads some book*

Value raising is again a name for a collection of relations. Whereas argument raising raises the type of an argument of a logical functor, value raising adds an argument to an expression. The simplest example can be given for non-logical constants of type e. The constant uther_e becomes $\lambda P_{s((se)t)}.P_@(\lambda@.\mathsf{uther}_e)$ by value raising with VR_t. The second expression is, of course, the Ty2 counterpart to Montague's translation of proper names as generalized quantifiers. Consequently the application of value raising to the basic translation of *Uther* gives us an expression which allows an alternative derivation of the meaning of the sentence *Uther walks*. It is depicted in Figure 5.4.

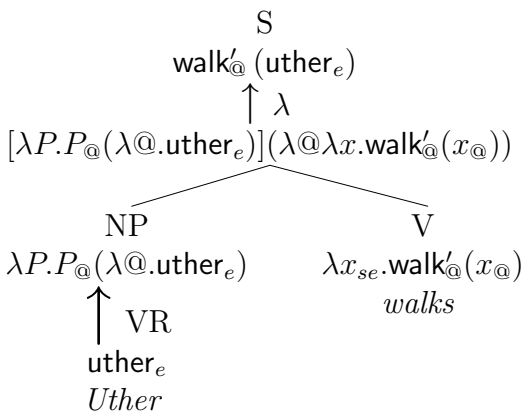

Figure 5.4: Illustration of value raising with the sentence *Uther walks*

After value raising, indicated in the figure by the symbol '↑ VR', the raised meaning contribution of *Uther* is the functor of the intension of the basic translation of *walks*. Subsequent lambda conversion leads to the expected reading of the utterance *Uther walks*.

Value raising is, of course, not necessary in order to obtain this reading for *Uther walks*, as we know from Figure 5.1. With the two derivations of one reading of an utterance we observe a general property of FMG and of its HPSG counterpart LF-Ty2 of the next section. Through the (iterated) application of type shifting rules it is possible to derive identical readings in infinitely many ways. In particular, argument raising of a functor can be countered by corresponding argument raising of its arguments. I will have to say more about this below.

In (43) I list a number of constructions which can be analyzed with the basic translations and the two type shifting rules introduced so far. In addition to constructions whose derivations were exemplified in figures above, the list also contains a new kind of construction involving *de re*/*de dicto* ambiguities with quantifiers embedded under intensional predicates in (43d).

(43) a. Uther walks.

$\mathsf{walk}'_@(\mathsf{uther}_e)$

b. Every student walks.

$\forall x_{se}[\mathsf{student}'_{@}(x_{@}) \rightarrow \mathsf{walk}'(x_{@})]$

c. Every student reads some book.

$\forall\exists$-reading:
$\forall x_{se}[\mathsf{student}'_{@}(x_{@}) \rightarrow \exists y_{se}[\mathsf{book}'_{@}(x_{@}) \wedge \mathsf{read}'_{@}(x_{@}, y_{@})]]$

$\exists\forall$-reading:
$\exists y_{se}[\mathsf{book}'_{@}(x_{@}) \wedge \forall x_{se}[\mathsf{student}'_{@}(x_{@}) \rightarrow \mathsf{read}'_{@}(x_{@}, y_{@})]]$

d. Every student believes that some president sucks.

de dicto reading:
$\forall x_{se}[\mathsf{student}'_{@}(x_{@}) \rightarrow \mathsf{believe}'_{@}(x_{@}, \lambda@.\exists y_{se}[\mathsf{president}'_{@}(y_{@}) \wedge \mathsf{suck}'_{@}(y_{@})])]$

$\forall\exists$ *de re* reading:
$\forall x_{se}[\mathsf{student}'_{@}(x_{@}) \rightarrow \exists y_{se}[\mathsf{president}'_{@}(y_{@}) \wedge \mathsf{believe}'(x_{@}, \lambda@.\mathsf{suck}'_{@}(y_{@}))]]$

$\exists\forall$ *de re* reading:
$\exists y_{se}[\mathsf{president}'_{@}(y_{@}) \wedge \forall x_{se}[\mathsf{student}'_{@}(x_{@}) \rightarrow \mathsf{believe}'(x_{@}, \lambda@.\mathsf{suck}'_{@}(y_{@}))]]$

In the *de dicto* reading of *Every student believes that some president sucks* it is not guaranteed that a president exists at all. The students might hold beliefs about an entity which does not even exist in their world. In the $\forall\exists$ *de re* reading, the quantifier of the embedded clause takes scope over the clause. In this reading, there exists at least one president, but the beliefs held by the students might be about different presidents. Every student might believe that an actual but possibly different president sucks. In the $\exists\forall$ *de re* reading, finally, we are in the peculiar situation in which one single president is such that every student believes that he sucks, although there might be other presidents in the same world and at least some of the students might believe that they suck as well. All of these readings can be obtained by appropriate applications of value raising and of argument raising to the basic translations of the words in the sentence. Since the details of the derivations are of no importance for our discussion, they are omitted here.[6]

Before we turn to an integration of the mechanisms above with normal form grammars with Ty2, let me briefly return to the observation that the same reading of utterances can be derived in several different ways through different applications of type shifting rules. Given the theoretical framework in which Hendriks works, this is not a problem for him as long as the system is capable of deriving all empirically correct readings. There are no empirical claims about different derivations. However, one might at first think that there is a potential computational problem. As [Bouma, 1994] has shown, this is not the case. Assuming that the computational system only tries to compute readings of type t for an input expression, Bouma uses results by Hendriks to design an algorithm which reduces

[6]Corresponding utterances are analyzed in [Sailer, 2003, pp. 53–56] and [Hendriks, 1993, Section 5.1]. A third argument shifting rule, argument lowering, which Hendriks introduces together with argument raising and value raising, is omitted here [Hendriks, 1993, p. 75].

the infinite number of possible derivations to a finite subset. With the exception of one construction, the algorithm even succeeds in considering only one derivation per sentence. I conclude that the system proves to be tractable from a computational point of view.

One final result about FMG should be mentioned before we proceed, because it is not obvious considering the fact that the system of FMG is capable of analyzing scope ambiguities with and without opaque predicates without requiring distinct syntactic tree structures for the different readings. [Hendriks, 1993, Chapter 2] proves that FMG is strictly compositional in the algebraic sense if type shifting is recorded in some appropriate way in the syntactic category structure.

5.2 Lexicalized Flexible Ty2

When I now discuss LF-Ty2, I presuppose that we are working in the class of normal form grammars with Ty2. These normal form grammars comprise, among many other things, the specification of a tectogrammatical structure on the basis of daughters attributes of phrases as described in Section 2.1.1. The tectogrammatical structure is important for us, because it provides the syntactic combinatorics to which semantic composition corresponds.

In this section I will make additional assumptions to simplify the exposition of LF-Ty2. Two sign-valued tectogrammatical daughters attributes, H-DTR for the syntactic head daughter and NH-DTR for the syntactic non-head daughter, are appropriate to phrases. This means that all syntactic structures induce a binary branching tectogrammatical tree with these two daughters attributes. These assumptions are not necessary for LF-Ty2 in general. A binary branching structure makes the specification of intensional functional application simpler, and adopting specific names and the given appropriateness function for the daughters attributes allows me to formulate concrete examples.

I will also adopt convenient notational conventions for Ty2 expressions in AVM descriptions. Following the usual logical conventions I will use names for variables in the same way in which I use names for non-logical constants. I will write x and y for the first and second variable of type se, $v_{se,0}$ and $v_{se,1}$, P and Q for the first and second variable of type $s((se)t)$, $v_{s((se)t),0}$ and $v_{s((se)t),1}$, etc. Occasionally I will add the type subscripts to the variable names to remind the reader of the type conventions which I follow.

The basic architectural requirements of LF-Ty2 are obvious. We need sorts, attributes, appropriateness specifications and relations which provide enough and the right kind of structure in interpretations to specify lambda conversion, including alpha conversion, and the type shifting mechanisms for argument raising and value raising. Furthermore, we need a SEMANTICS PRINCIPLE for phrases which demands that the value of an appropriate attribute in phrases be the intensional functional application of the Ty2 expressions in two attribute values of their tectogrammatical daughters.

(44) Requirements for LF-Ty2:

a. Lexical entries appropriate for specifying basic translations and derived translations

b. A SEMANTICS PRINCIPLE which ensures that the value of a particular attribute in phrases is the intensional functional application of corresponding attribute values of their tectogrammatical daughters

c. A type shifting mechanism for argument raising and value raising

d. A mechanism for lambda conversion, including alpha conversion

The goal of LF-Ty2 can be seen in the description in (45). The idea is that LF-Ty2 must be specified in such a way that the grammar predicts the described value of the path LF EXPR of the unembedded phrase *Uther walks* on the basis of the lexical specification of the two words in it. The lexical specifications must be such that they predict the LF EXPR values of the words described in (45) as their basic translation. A specification of these specific expressions in the lexical entries of the two words is trivial, since we know from the results of Chapter 4 that any Ty2 expression can be singled out with an exact description. The task of the specification of LF-Ty2, however, is more demanding. Together with the basic translation for each word in the grammar, the specification must foresee the possibility of type shifting and lambda conversion. For this reason the signature of LF-Ty2 must provide sufficiently rich structures to permit type shifting and lambda conversion, including alpha conversion.

(45)
$$\begin{bmatrix} \textit{u-phrase} \\ \text{PHON } \langle \textit{uther, walks} \rangle \\ \text{SYNSEM LOC CAT } \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{LF EXPR } \mathsf{walk}'_{@}(\mathsf{uther}_e) \\ \text{NH-DTR } \begin{bmatrix} \textit{e-word} \\ \text{PHON } \langle \textit{uther} \rangle \\ \text{SYNSEM } \boxed{1} \begin{bmatrix} \text{LOC CAT } \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \end{bmatrix} \\ \text{LF EXPR } \mathsf{uther}_e \end{bmatrix} \\ \text{H-DTR } \begin{bmatrix} \textit{e-word} \\ \text{PHON } \langle \textit{walks} \rangle \\ \text{SYNSEM LOC CAT } \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{SUBCAT} & \langle \boxed{1} \rangle \end{bmatrix} \\ \text{LF EXPR } \lambda x_{se}.\mathsf{walk}'_{@}(x_{@}) \end{bmatrix} \end{bmatrix}$$

Example (45) already reveals the first decisions regarding the signature of LF-Ty2. In contrast to [Sailer, 2003], which locates semantic composition under Pollard and Sag's CONTENT attribute, I introduce a new attribute LOGICAL-FORM, usually to be written as LF, at all signs. LF does not have Ty2 expressions as its value. Ty2 expressions are located under one more attribute called EXPRESSION, abbreviated as EXPR. As we will see, the reason for this architecture is that LF values are complex configurations which might

comprise configurations which encode beta conversion steps and alpha conversion steps. These conversions lead to the expressions which occur as LF EXPR values.

The signature Σ_{LF-Ty2} contains a signature Σ_{Ty2} and the sorts, sort hierarchy, attributes, appropriateness specifications and relation symbols which are illustrated in (46). The only exceptions are the sort *top*, which is not obligatory, and the sort *sign*, for which there might be a different symbol in a legitimate LF-Ty2 signature.

(46) *top*

sign LOGICAL-FORM *complex-lf*

complex-lf EXPRESSION *me*
REDUCTION *reduction*

reduction TERM *me*
AUX *me*

no-reduction

β-contraction REC *reduction*

change-bound-variable REC *reduction*

```
Relations
argument-raising/2
ar-aux/4
value-raising/2
shifting/2
intensional-functional-application/3
subterm/2
free-variable/2
replace/4
replace1/4
```

Apart from the abbreviations which I have already introduced, I will abbreviate the attribute REDUCTION as RED. There is no further interaction through appropriateness between the new sorts in the sort hierarchy above and other sorts in normal form grammars with Ty2. The new species only occur as labels of entities in configurations under LF values.[7]

The LF EXPR value specifies the truth-conditional meaning of each sign in an unembedded sign configuration of LF-Ty2 grammar models. The LF EXPR value of an unembedded sign yields its meaning when the sign is interpreted in a context. Entities of sort *complex-lf* occur as values of LF at signs. Besides the attribute EXPR, a second attribute called RED(UCTION) is appropriate to *complex-lf*. The *reduction* configurations under RED determine beta reduction and alpha conversion steps which might occur between the Ty2 expression under RED AUX and the Ty2 term in EXPR. This means that the RED AUX and the EXPR values are either the same Ty2 expression, or the expression in EXPR is obtained from the expression in RED AUX by finitely many conversion steps. Beta conversion and

[7]In this remark I ignore, of course, the effect of the EMBEDDED attribute, which is appropriate to all new sorts. If we take the effect of the EMBEDDED attribute on LF-Ty2 models into account, each entity has as its components all entities in the entire unembedded sign configuration to which it belongs.

alpha conversion cannot be defined directly as relations between Ty2 expressions which occur as attribute values in a configuration. The reason for this is that a term which is related to another term by beta reduction or alpha conversion does not contain the other term as a component configuration, or *vice versa*. The conversion steps must, therefore, be represented as separate configurations under attributes.[8]

The signature Σ_{LF-Ty2} introduces nine new relation symbols. The relations `argument-raising`, `ar-aux` and `value-raising` will be used to specify the type shifting relationship between Ty2 expressions which occur as attribute values.[9] The `shifting` relation will specify a closure over argument raising and value raising. The relation `intensional-functional-application` between three entities will specify the intensional functional application mechanism in the SEMANTICS PRINCIPLE. The remaining four relations, finally, will serve to encode lambda conversion and alpha conversion. Besides their function in the specification of lambda conversion, the `subterm` relation between Ty2 expressions and the `free-variable` relation are very valuable relations for principles at the syntax-semantics interface. Moreover, the `subterm` relation will be fundamental in the LRS framework in Chapter 6.

The basic translations of words are specified in the lexicon of LF-Ty2 grammars. In HPSG grammars the term *lexicon* normally refers to the set of disjuncts in the consequent of the WORD PRINCIPLE. These disjuncts are descriptions of words which occur in the grammar. The lexical entries in (47) illustrate what they look like in an LF-Ty2 grammar. For illustration I assume a normal form signature with Ty2 which closely follows the signature of Pollard and Sag's grammar of English in those parts of the signature which are not related to semantic representations. The symbols 'NP' and 'DetP' are convenient abbreviations for descriptions of appropriate *synsem* entities on SUBCAT lists.

The lexical entries in (47) correspond to some of the basic translations in (42), and it should be obvious how the remaining basic translations must be specified. In each lexical entry there is a description of the basic translation. The entity which is assigned to the variable [1] is the basic translation. However, what we find as value of LF RED AUX might be a type shifted variant [2] of the expression [1], since [1] and [2] stand in the `shifting` relation. This specification works in the intended way because the expression [1] is a component configuration of each type shifted expression [2] which can be derived from it. The expression [2] is specified as the LF RED AUX value, not as the LF EXPR value, which contains the final reading of the word. The reason is that the expression [2] might not be fully beta reduced. The principles which govern the shape of *complex-lf* and *reduction* entities in models of LF-Ty2 grammars will guarantee that the LF EXPR value of each sign is the term which is obtained by fully beta reducing the term in LF RED AUX.

The interaction of these components of the theory of LF-Ty2 achieves the desired effect. The LF EXPR value of each word is either the Ty2 expression directly specified in the lexical

[8] To avoid these additional configurations under attributes, [Sailer, 2003, Section 4.2.2] proposes an encoding of lambda conversion and alpha conversion in chain arguments of conversion relations.

[9] Note that this can only be done with relations because a type-shifted expression always contains the 'original' expression as its proper subpart. The reason that this is hardly visible in my examples is that I maximally beta reduce the type-shifted expression in examples to make them more readable.

entry (its basic translation), or a fully beta reduced type shifted variant thereof. We will take a closer look at the theories of type shifting and lambda conversion below, without going into all details.

(47) a.
$$\begin{bmatrix} \textit{word} \\ \text{PHON} \ \langle \textit{reads} \rangle \\ \text{SYNS LOC CAT} \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{SUBCAT} & \langle \text{NP, NP} \rangle \\ \text{MARKING} & \textit{unmarked} \end{bmatrix} \\ \text{LF RED AUX} \ \boxed{2} \end{bmatrix} \begin{array}{l} \wedge \boxed{1}[\lambda y_{se} \lambda x_{se}.\mathsf{read}'_{@}(x_{@}, y_{@})] \\ \wedge \ \mathtt{shifting}(\boxed{1}, \boxed{2}) \end{array}$$

b.
$$\begin{bmatrix} \textit{word} \\ \text{PHON} \ \langle \textit{walks} \rangle \\ \text{SYNS LOC CAT} \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{SUBCAT} & \langle \text{NP} \rangle \\ \text{MARKING} & \textit{unmarked} \end{bmatrix} \\ \text{LF RED AUX} \ \boxed{2} \end{bmatrix} \begin{array}{l} \wedge \boxed{1}[\lambda x_{se}.\mathsf{walk}'_{@}(x_{@})] \\ \wedge \ \mathtt{shifting}(\boxed{1}, \boxed{2}) \end{array}$$

c.
$$\begin{bmatrix} \textit{word} \\ \text{PHON} \ \langle \textit{student} \rangle \\ \text{SYNS LOC CAT} \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle \text{DetP} \rangle \\ \text{MARKING} & \textit{unmarked} \end{bmatrix} \\ \text{LF RED AUX} \ \boxed{2} \end{bmatrix} \begin{array}{l} \wedge \boxed{1}[\lambda x_{se}.\mathsf{student}'_{@}(x_{@})] \\ \wedge \ \mathtt{shifting}(\boxed{1}, \boxed{2}) \end{array}$$

d.
$$\begin{bmatrix} \textit{word} \\ \text{PHON} \ \langle \textit{book} \rangle \\ \text{SYNS LOC CAT} \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle \text{DetP} \rangle \\ \text{MARKING} & \textit{unmarked} \end{bmatrix} \\ \text{LF RED AUX} \ \boxed{2} \end{bmatrix} \begin{array}{l} \wedge \boxed{1}[\lambda x_{se}.\mathsf{book}'_{@}(x_{@})] \\ \wedge \ \mathtt{shifting}(\boxed{1}, \boxed{2}) \end{array}$$

e.
$$\begin{bmatrix} \textit{word} \\ \text{PHON} \ \langle \textit{mary} \rangle \\ \text{SYNS LOC CAT} \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle \rangle \\ \text{MARKING} & \textit{unmarked} \end{bmatrix} \\ \text{LF RED AUX} \ \boxed{2} \end{bmatrix} \begin{array}{l} \wedge \boxed{1}[\mathsf{mary}_{e}] \\ \wedge \ \mathtt{shifting}(\boxed{1}, \boxed{2}) \end{array}$$

f.
$$\begin{bmatrix} \textit{word} \\ \text{PHON} \ \langle \textit{every} \rangle \\ \text{SYNS LOC CAT} \begin{bmatrix} \text{HEAD} & \textit{det} \\ \text{SUBCAT} & \langle \rangle \\ \text{MARKING} & \textit{unmarked} \end{bmatrix} \\ \text{LF RED AUX} \ \boxed{2} \end{bmatrix} \begin{array}{l} \wedge \boxed{1}\left[\lambda P_{s((se)t)} \lambda Q_{s((se)t)} \forall x_{se} \left[P_{@}(x) \rightarrow Q_{@}(x)\right]\right] \\ \wedge \ \mathtt{shifting}(\boxed{1}, \boxed{2}) \end{array}$$

g. $$\begin{bmatrix} \textit{word} \\ \text{PHON } \langle \textit{every} \rangle \\ \text{SYNS LOC CAT } \begin{bmatrix} \text{HEAD} & \textit{det} \\ \text{SUBCAT} & \langle\rangle \\ \text{MARKING} & \textit{unmarked} \end{bmatrix} \\ \text{LF RED AUX } \boxed{2} \end{bmatrix} \wedge \boxed{1}\left[\lambda P_{s((se)t)}\lambda Q_{s((se)t)}\exists x_{se}\left[P_{@}(x) \wedge Q_{@}(x)\right]\right] \wedge \texttt{shifting}(\boxed{1},\boxed{2})$$

Supposing that the theories of type shifting and lambda conversion are correct, the lexical component of LF-Ty2 is completed with the specification of the lexical entries. The semantic composition at phrases is subject to the SEMANTICS PRINCIPLE, (48a). It relies on the INTENSIONAL FUNCTIONAL APPLICATION PRINCIPLE, (48b), which determines the meaning of the crucial ternary relation `intensional-functional-application`.[10]

(48) a. SEMANTICS PRINCIPLE

$$\textit{phrase} \rightarrow \left(\begin{bmatrix} \text{LF} & \boxed{1} \\ \text{H-DTR LF} & \boxed{2} \\ \text{NH-DTR LF} & \boxed{3} \end{bmatrix} \wedge \texttt{intensional-functional-application}(\boxed{1},\boxed{2},\boxed{3}) \right)$$

b. INTENSIONAL FUNCTIONAL APPLICATION PRINCIPLE

$$\forall\boxed{1}\,\forall\boxed{2}\,\forall\boxed{3}\left(\begin{array}{l} \texttt{intensional-functional-application}(\boxed{1},\boxed{2},\boxed{3}) \leftrightarrow \\ \exists\boxed{4}\,\exists\boxed{5} \\ \left(\begin{array}{l} \left(\boxed{1}\begin{bmatrix} \text{RED AUX} & \begin{bmatrix} \textit{application} \\ \text{FUNCTOR } \boxed{4} \\ \text{ARG} & \begin{bmatrix} \textit{abstraction} \\ \text{VARIABLE } @ \\ \text{ARG } \boxed{5} \end{bmatrix} \end{bmatrix} \end{bmatrix} \wedge \boxed{2}\left[\text{EXPR } \boxed{4}\right] \wedge \boxed{3}\left[\text{EXPR } \boxed{5}\right] \right) \vee \\ \left(\boxed{1}\begin{bmatrix} \text{RED AUX} & \begin{bmatrix} \textit{application} \\ \text{FUNCTOR } \boxed{5} \\ \text{ARG} & \begin{bmatrix} \textit{abstraction} \\ \text{VARIABLE } @ \\ \text{ARG } \boxed{4} \end{bmatrix} \end{bmatrix} \end{bmatrix} \wedge \boxed{2}\left[\text{EXPR } \boxed{4}\right] \wedge \boxed{3}\left[\text{EXPR } \boxed{5}\right] \right) \end{array} \right) \end{array} \right)$$

According to the SEMANTICS PRINCIPLE, the LF value of each phrase stands in the `intensional-functional-application` relation to the LF values of its two daughters. According to the INTENSIONAL FUNCTIONAL APPLICATION PRINCIPLE, this means one of two things:

[10] Note the difference in notation for the description of Ty2 expressions in the INTENSIONAL FUNCTIONAL APPLICATION PRINCIPLE, (48b), and in the exemplary lexical entries. According to the results of Chapter 4, we may simply write expressions of Ty2 instead of their descriptions. But this does not exclude the use of AVM formulae if this is more transparent, such as in the INTENSIONAL FUNCTIONAL APPLICATION PRINCIPLE.

(1) the LF EXPR value of the head daughter is the functor of an application whose argument is the LF EXPR value of the non-head daughter, with lambda abstraction over the designated world variable added; and this application is the LF RED AUX value of the phrase. According to the theories of *reduction* entities and of *complex-lf* entities to be discussed below, this means that the LF EXPR value of the phrase is the expression which we obtain by fully beta reducing the application expression in LF RED AUX.

(2) the LF EXPR value of the non-head daughter is the functor of an application whose argument is the LF EXPR value of the head daughter, with lambda abstraction over the designated world variable @ added; and this application is the LF RED AUX value of the phrase. Again, just as in the first case, the theories of *reduction* entities and of *complex-lf* entities guarantee that the LF EXPR value of the phrase is the expression which we obtain by fully beta reducing the application expression in LF RED AUX.

In short, the LF EXPR value of each phrase is the result of intensional functional application of the LF EXPR value of one of its daughters to the LF EXPR value of the other daughter.

With this result, all that is left to do is to make sure that the type shifting relation works according to the definitions of argument raising and value raising cited above (DEFINITION 33 and DEFINITION 34), and that lambda conversion and alpha conversion behave according to their standard definitions. This is the case if the theory of the configurations under *reduction* entities is a correct theory of lambda and alpha conversion. Moreover, there must be a principle which requires that LF EXPR values be fully beta reduced, as I have presupposed throughout. I will not present all of these principles in detail. Instead, I rely on Sailer's results about them. In the remainder of this section I give a quick overview of all the parts of these theories. The overview will suffice to give the reader an impression of the complexity of the specification of LF-Ty2. First we will take a look at the theory of the shifting relations, followed by the principles governing lambda and alpha conversions. We finish with principles about the admissible form of Ty2 expressions which may occur as LF EXPR values.

The relation `shifting` is needed to write lexical entries which specify the basic translation and the possible type shiftings of the basic translations simultaneously. Suppose that the relation principles for `argument-raising` and `value-raising` are correct. Then the SHIFTING PRINCIPLE, (49), specifies the reflexive transitive closure over the possible shifting operations on a Ty2 expression $\boxed{1}$.

(49) The SHIFTING PRINCIPLE

$$\forall\boxed{1}\,\forall\boxed{2}$$
$$\left(\texttt{shifting}(\boxed{1},\boxed{2}) \leftrightarrow \left(\begin{array}{l} \boxed{1} = \boxed{2} \vee \\ \exists\boxed{3}\ (\texttt{argument-raising}(\boxed{1},\boxed{3}) \wedge \texttt{shifting}(\boxed{3},\boxed{2})) \vee \\ \exists\boxed{3}\ (\texttt{value-raising}(\boxed{1},\boxed{3}) \wedge \texttt{shifting}(\boxed{3},\boxed{2})) \end{array}\right)\right)$$

The VALUE RAISING PRINCIPLE, (50), states that the relation `value-raising` holds between two Ty2 expressions, $\boxed{1}$ and $\boxed{2}$, in a configuration if and only if $\boxed{1}$ stands in the value

raising relation according to DEFINITION 34 with [2], i.e., [2] results from [1] by iteratively adding arguments according to DEFINITION 34.

(50) The VALUE RAISING PRINCIPLE

$$\forall[1]\ \forall[2]$$
$$\left(\texttt{value-raising}([1],[2]) \leftrightarrow \left(\begin{array}{l} \exists[3]\ \left([2]\left[\begin{array}{ll} \textit{abstraction} & \\ \text{VARIABLE} & [3] \\ \text{ARG} & \left[\begin{array}{ll} \textit{application} & \\ \text{FUNCTOR} & \left[\begin{array}{ll} \textit{application} & \\ \text{FUNCTOR} & [3] \\ \text{ARG} & @ \end{array}\right] \\ \text{ARG} & \left[\begin{array}{ll} \textit{abstraction} & \\ \text{VARIABLE} & @ \\ \text{ARG} & [1] \end{array}\right] \end{array}\right] \end{array}\right]\right) \vee \\ \exists[3]\ \exists[4]\ \exists[5] \\ \left([3]\left[\begin{array}{ll} \textit{application} & \\ \text{FUNCTOR} & [1] \\ \text{ARG} & [4] \end{array}\right] \wedge [2]\left[\begin{array}{ll} \textit{abstraction} & \\ \text{VARIABLE} & [4] \\ \text{ARG} & [5] \end{array}\right] \right. \\ \left. \wedge\ \texttt{value-raising}([3],[5])\right) \end{array}\right)\right)$$

The first disjunct to the right of the bi-implication in the VALUE RAISING PRINCIPLE licenses the simple cases of value raising. DEFINITION 34 indicates the type of the input expression as $a_1(\ldots(a_n b)\ldots)$. In the first disjunct, n is zero. It can easily be seen that this case puts uther_e and $\lambda P.P_@(\lambda@.\mathsf{uther}_e)$ in the `value-raising` relation in each configuration (P is the variable referred to by tag [3]). The reader might want to check that the second disjunct licenses more complex cases such as the value raising relationship between $\lambda x_{se}.\mathsf{walk}'_@(x_@)$ and $\lambda y_{se}\lambda u_{s((st)t)}.u_@(\lambda@[(\lambda x_{se}.\mathsf{walk}'_@(x_@))(y)])$.

I omit the ARGUMENT RAISING PRINCIPLE and the auxiliary principle for the relation `ar-aux`. They can be found in [Sailer, 2003, p. 148]. According to Sailer's principles, `argument-raising` holds between two Ty2 expressions in a configuration if and only if they stand in the argument raising relation according to DEFINITION 33.

With the LF-Ty2 theory of type shifting completed, I turn to the theory of lambda conversion and alpha conversion. It is expressed in conditions on the admissible configurations under the three kinds of *reduction* entities which occur as LF RED values of signs.

The theory of *no-reduction* entities is the non-recursive base case of the conversions. The NO REDUCTION PRINCIPLE [Sailer, 2003, p. 174] is illustrated in (51):

(51) The NO REDUCTION PRINCIPLE

$$\textit{no-reduction} \rightarrow \left[\begin{array}{ll} \text{TERM} & [1] \\ \text{AUX} & [1] \end{array}\right]$$

In this simple case, the AUX value and the TERM value are identical. The NO REDUCTION PRINCIPLE is immediately responsible for the fact that the lexical entries in (47)—as disjuncts in the consequent of a WORD PRINCIPLE—license word configurations with basic

translations as LF EXPR value. This is the case because, due to the reflexivity of the `shifting` relation, the lexical entries allow the LF RED AUX value of words to be the basic translation. The NO REDUCTION PRINCIPLE permits the basic translation to appear as the LF RED TERM value. Furthermore, a principle to be introduced below, the COMPLEX LF PRINCIPLE, which governs the configurations under *complex-lf* entities, guarantees that the LF RED TERM and LF EXPR values are always the same Ty2 expression.

The theories of the other two kinds of *reduction* entities, *change-bound-variable* and *β-contraction*, are parallel to the NO REDUCTION PRINCIPLE, except that their conditions are much more complex, because they involve relations and *change-bound-variable* and *β-contraction* configurations are recursively structured due to their REC attribute. Each REC attribute of *reduction* entities of these two species has another *reduction* entity as its value.

The CHANGE BOUND VARIABLE PRINCIPLE [Sailer, 2003, p. 174] needs the relations `subterm`, `free-variable`, `replace` and `replace1`. I ignore the two relations `replace` and `replace1`. They encode relationships between four expressions which help to state that every free occurrence of a variable in a term is replaced by another variable and thus yields a fourth expression. The relation `free-variable` holds between two Ty2 expressions in a configuration if and only if the first is a *variable* entity which occurs free in the second. The `subterm` relation between two Ty2 expressions will become important below. Therefore, I illustrate it here:

(52) The SUBTERM PRINCIPLE

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

The SUBTERM PRINCIPLE uses the relation `ty2-component`, whose meaning in models is determined by the TY2-COMPONENT PRINCIPLE in (34) on page 174. For simplicity, I will later use an infix notation with the symbol '$\triangleleft$' for the `subterm` relation. I will write '$\boxed{1} \triangleleft \boxed{2}$' for '`subterm`$(\boxed{1}, \boxed{2})$'. Two expressions $\boxed{1}$ and $\boxed{2}$ in a configuration are in the `subterm` relation if the first is a subterm of the second, and in no other case.

Based on the relations `subterm`, `free-variable`, `replace` and `replace1` and on the four corresponding relation principles,[11] the CHANGE BOUND VARIABLE PRINCIPLE states about *change-bound-variable* entities in models of LF-Ty2 grammars that (a) the configurations under them do not contain a *β-contraction* entity (ignoring configurations reached by following an EMBEDDED attribute) and that (b) the AUX value and the TERM value stand in the relationship of alpha conversion.

The BETA CONTRACTION PRINCIPLE [Sailer, 2003, p. 176] uses the relations `subterm`, `replace` and `replace1`. It specifies the relationship between the AUX value and the TERM value of *β-contraction* entities as a relationship of a (nonempty) sequence of beta reductions.

With the NO REDUCTION PRINCIPLE, the CHANGE BOUND VARIABLE PRINCIPLE and the BETA CONTRACTION PRINCIPLE, the following picture about configurations under *reduction* entities in models of LF-Ty2 grammars emerges: the relationship between

[11] See [Sailer, 2003, pp. 171–176] for these relation principles.

their AUX expression and their TERM expression is a relationship of finitely many beta contractions and alpha conversions from AUX value to TERM value.

The COMPLEX LF PRINCIPLE presupposes this result and imposes the crucial condition on the properties of the LF EXPR value of all signs. The EXPR value and the RED TERM value of all *complex-lf* entities are identical, and there is no unreduced β redex left in this expression. In effect, this means that the LF EXPR value of each sign equals its LF RED AUX value, which is specified either lexically or by the SEMANTICS PRINCIPLE, except that the expression in LF EXPR is redex free. All possible lambda conversions are performed.

(53) COMPLEX LF PRINCIPLE

$$\textit{complex-lf} \rightarrow \left(\begin{array}{l} \begin{bmatrix} \text{EXPR} & \boxed{1} \\ \text{RED TERM} & \boxed{1} \end{bmatrix} \\ \wedge\, \neg\exists \boxed{2}\ \texttt{subterm}\left(\boxed{2} \begin{bmatrix} \textit{application} \\ \text{FUNCTOR} \quad \textit{abstraction} \end{bmatrix}, \boxed{1} \right) \end{array} \right)$$

Finally, I impose a type restriction on the expressions which may occur as the readings of an unembedded sign configuration. The present version of LF-Ty2 is a theory of sentence semantics. I assume that the readings of sentences are always of type t. The U-SIGN TYPE RESTRICTION PRINCIPLE excludes readings of a higher type. Although they are undesired, they could in principle be derived by the unrestricted application of type shifting to the basic translation of words.

(54) U-SIGN TYPE RESTRICTION PRINCIPLE

$$\textit{u-sign} \rightarrow \left[\text{LF EXPR TYPE} \quad \textit{truth}\right]$$

Summary With the U-SIGN TYPE RESTRICTION PRINCIPLE I have completed the overview of the framework of LF-Ty2, and I can now characterize the general form of LF-Ty2 grammars. An LF-Ty2 grammar $\Gamma_{LF-Ty2} = \langle \Sigma_{LF-Ty2}, \theta_{LF-Ty2} \rangle$ is a grammar which obeys the following conditions:

Σ_{LF-Ty2} is a normal form signature with Ty2 which also includes the specifications in (46). θ_{LF-Ty2} is the theory of a normal form grammar with Ty2. In addition it contains the nine relation principles for the new relations enumerated in (46); the lexical entries in its WORD PRINCIPLE specify the LF RED AUX values of words according to the examples given in (47); and it contains the SEMANTICS PRINCIPLE (48a), the COMPLEX LF PRINCIPLE (53), the U-SIGN RESTRICTION PRINCIPLE (54), the NO REDUCTION PRINCIPLE (51) and the CHANGE BOUND VARIABLE PRINCIPLE and BETA CONTRACTION PRINCIPLE as specified in [Sailer, 2003, p. 174 and 176]. The U-SIGN RESTRICTION PRINCIPLE may be refined for grammars with semantic theories which include more than a sentential semantics and thus might allow utterances whose readings are not specified by an expression of type t. The SEMANTICS PRINCIPLE may be modified with respect to the particular attribute symbols which a grammar uses for the tectogrammatical structure. It can also be generalized to non-binary branching tectogrammatical structures.

As mentioned earlier already, LF-Ty2 has already been employed in empirical studies. It was first presented in a [Richter and Sailer, 1999a], a study of sentential negation and negative concord in French. The study [Richter and Sailer, 1999b] analyzes similar data in Polish and reveals typological differences in the negation system of French and Polish on the basis of the common semantic system of the two studies. [Sailer, 2003] gives the most comprehensive account of LF-Ty2 and investigates various possibilities to modify the system. In the second part of [Sailer, 2003], LF-Ty2 provides the theoretical foundation for an analyses of idiomatic expressions. [Trawiński et al., 2004] is the latest study to employ LF-Ty2. The purpose of LF-Ty2 in this study is to demonstrate how Trawiński's HPSG analysis of collocational prepositional phrases can be augmented by a compositional semantics.

5.3 Discussion

The advantages of LF-Ty2 compared to the uninterpreted semantic representations of [Pollard and Sag, 1994] and other HPSG theories which employ some form of uninterpreted semantic representations are clear. They range from practical aspects such as direct access to the results of theories formulated outside of the HPSG framework to abstract considerations such as the compatibility with theories of compositional semantics. Extending a list compiled in [Sailer, 2003, p. 383–384] and adapting it to the more comprehensive grammar theory of normal form HPSG grammars, the following points deserve to be noted:

The semantic representations of LF-Ty2 belong to the standard repertoire of formal semantics. Using them builds a bridge across the boundaries of linguistic frameworks, no matter whether the boundary is due to different underlying syntactic frameworks or to the distinction between a model-theoretic and a generative-enumerative framework. On the basis of Ty2 any semantic theory about a certain empirical domain in LF-Ty2 can be compared to alternative theories and their predictions. This should be in the best interest of all parties concerned.

Ty2 is, of course, a well defined representation language. This imposes clear restrictions on what kinds of extensions of the semantic theory are possible or make sense in the given framework. As long as one works with representations which are only given by example, both aspects remain unclear. To be more specific, the weak intensionality of Ty2 is accompanied by well known problems in the analysis of the semantics of necessary truths under opaque predicates. Anybody interested in this problem and solutions to it knows about these limitations of Ty2 in advance and may want to choose a different logical language from the start. The example set by the integration of Ty2 with normal form grammars indicates for a large class of logical languages how their integration with normal form HPSG grammars can proceed. It is simple to follow this example if one would like to work with a different logical language for semantic representations in HPSG.

LF-Ty2 establishes a direct link between the logical form of an unembedded sign and the model-theoretic meaning of the sign. Among other things, this allows us to express restrictions on meanings by formulating restrictions on admissible logical forms. In LF-Ty2

this is particularly simple since syntactic structures and semantic structures are uniformly specified in the language of the same description logic. The formulation of constraints such as scope island constraints at the syntax-semantics interface can thus be explored quite easily. The empirical value of the theories can be tested by considering the model-theoretic meaning of the unembedded signs in minimal exhaustive models of the grammar.

LF-Ty2 accounts for scope ambiguities on the basis of the sparse syntactic structures which are typical for the HPSG framework. The theory of scope ambiguities does not need tree-geometric operations such as quantifier raising. It can even avoid a Cooper store.

The fact that no Cooper store is necessary to analyze scope ambiguities without postulating a (tectogrammatical) syntactic ambiguity leads to a framework-internal improvement in the architecture of semantic representations. In Pollard and Sag's architecture of signs, semantic representations are distributed over several attribute values. In LF-Ty2 everything is contained under the LF attribute. The configurations under the LF attribute may even be reduced further to just one Ty2 expression if one adopts Sailer's chain encoding of beta conversion and alpha conversion, which eliminates the *reduction* configurations of the version of LF-Ty2 presented in Section 5.2.

LF-Ty2 also gives a clear answer to questions about the ontological status of the LF value. All LF EXPR values are logical forms. Since they are logical forms and they consist of configurations of abstract entities, it is conceivable that principles of grammar exist which restrict logical forms in certain syntactic configurations, or *vice versa*.

As a system with logical forms and the potential for interaction between syntax and logical form, LF-Ty2 differs from Montague Grammar. In Montague Grammar syntax is the basis for interpretation, and a level of semantic representations is dispensable. As long as there are no grammatical principles which interfere with semantic composition along tectogrammatical trees, LF-Ty2 is still a strictly compositional semantics according to the results of [Hendriks, 1993, Chapter 2] about FMG. However, semantic composition in LF-Ty2 is embedded in an environment of unembedded sign configurations. In unembedded sign configurations there is no ontological distinction between syntactic entities and the entities in logical forms. Since all components in these configurations are specified as models of a theory in a description logic, it is not even clear what we gain by keeping to a relationship between syntactic and semantic configurations that preserves the homomorphism condition of compositionality. It seems as if the principle of compositionality, valuable as a strategy and guiding idea in other syntactic frameworks, looses at least some of its value in a model-theoretic syntactic framework.

Traditionally the lambda calculus ensures that semantic composition goes hand in hand with the syntactic combinatorics to derive the readings for utterances. This is true in Montague Grammar, in Hendriks's Flexible Montague Grammar, in von Stechow's theory of Transparent Logical Form, and in systems which combine a Transformational Grammar with a Montague Semantics such as [Cooper, 1975]. But in a model-theoretic syntax the syntactic combinatorics do not have the same status as in these frameworks. What appears as the syntactic combinatorics in *u-sign* configurations is ultimately a reflection of how linguists interpret these configurations. Linguists recognize tectogrammatical patterns which are reminiscent of the phrase structure rule systems or of the categorial syntax in the other

frameworks. But these patterns emerge from the model-theoretic interpretation of a set of grammar principles. They are not a direct reflection of a generative mechanism. As a result, there is no obvious need to specify the Ty2 expressions in unembedded sign configurations by an encoding of the lambda calculus which implements the derivation steps for readings at each phrase in the configuration. In particular, compositionality as a major motivation for using the lambda calculus in the aforementioned systems does not have the same importance in normal form grammars with Ty2.

Although LF-Ty2 restricts the different ways in which the combinatorial system of FMG can derive the same readings of a given sentence, it does not eliminate the possibility of infinitely many derivations. The additional restriction in LF-Ty2 comes from the lexicalization of type shifting. The remaining issue is the iterated application of the type shifting rules to functors and their arguments which cancels out the higher types of functors and arguments. The example of the two derivations for the reading of the sentence *Uther walks* in Figure 5.1 and Figure 5.4 is only a very simple example of this situation. While this is not a problem from the perspective of FMG, it is a problem in the model-theoretic syntactic framework. Our meta-theory of normal form grammars states that two distinct unembedded sign configurations in a minimal exhaustive grammar model should constitute distinct predictions of the grammar. Since the derivational history of the LF EXPR value of each unembedded sign is an intrinsic part of the structure of the sign, different derivations lead to distinct unembedded sign configurations. However, the structural differences do not constitute an empirical difference.

Viewed from the level of unembedded sign configurations in a minimal exhaustive grammar model, we obtain the following picture. There are infinitely many *u-sign* configurations with isomorphic syntax and differently configured LF values. Although the LF values of the unembedded signs with isomorphic syntax are differently configured, the LF EXPR values of infinitely many of them are pairwise isomorphic, indicating that these different *u-sign* configurations have the same linguistic meaning. In other words, the infinitely many *u-sign* configurations with isomorphic syntax fall into classes with infinitely many members. Each class of configurations has members with isomorphic LF EXPR values and different LF configurations in at least some signs. For example, we obtain two such infinite classes of configurations for the utterance *Every student reads some book*, one class for each reading of the sentence. This is not what we intuitively expect in our linguistic framework. The situation is methodologically problematic, because there does not seem to be any way to distinguish empirically between these different *u-sign* configurations with isomorphic LF EXPR values and non-isomorphic LF configurations in at least some of their signs. What our linguistic framework suggests is that, as far as syntactic and semantic structure are concerned, there should be exactly one unembedded sign configuration in a minimal exhaustive model for each expression with a certain syntactic configuration and a certain LF EXPR value.

There might be ways to fix this problem. For example, we could use the insights from computational work on FMG in [Bouma, 1994] and build restrictions on the number of possible derivations into LF-Ty2. Or we could characterize equivalence classes of derivations and treat classes of unembedded sign configurations which fall into the same derivation

class as empirically equivalent. However, the fact remains that something fundamental in the architecture of LF-Ty2 is at odds with the model-theoretic perspective on the description of language. This suspicion can be confirmed upon closer inspection. There are differences in the approach to grammar specifications between LF-Ty2 and 'normal' HPSG specifications.

First of all, the specification of Ty2 expressions in the lexical entries for words is systematically specific up to isomorphism classes of expressions. From a formal point of view, the Ty2 terms are specified by large, unambiguous descriptions. As we know, these descriptions can even be determined by a function from Ty2 models to descriptions. This form of description rarely occurs in HPSG principles. HPSG principles typically use very general descriptions of large classes of configurations in order to state generalizations over large classes of data. Maximal specificity of descriptions is not suitable for doing this.

There is, of course, a reason for the specificity of the Ty2 descriptions in LF-Ty2. The basic translations feed the type shifting rules, and the type shifting rules act on the form of expressions. Since the exact form of the expressions determines their type and their interpretation, it is essential. It is ultimately a reflection of what Partee calls one of the great achievements of working with the lambda calculus in semantics: "lambdas provide a particularly perspicuous tool for representing and working with function-argument structures explicitly and compositionally" [Partee, 1996, p. 24]. An integration of the lambda calculus in the grammar models as configurations of entities lifts the importance of form to the level of the description language. Form is typically not important at the description level of HPSG grammars.

The type shifting mechanism exhibits another property which is alien to traditional HPSG grammars. It encodes a derivational history in the linguistic configurations of entities. The standard perspective of HPSG emphasizes a direct description of the intended structures, and this emphasis includes the view that all specified structures have empirical significance. This is not the case for the type shifting mechanism and the lambda conversion mechanism of LF-Ty2.

It would probably be misguided to overestimate the HPSG strategy of only specifying structures with empirical significance. After all, classical HPSG grammars use relations such as `append` and `member` whose immediate empirical significance is questionable at best. However, these relations are employed to specify structures of immediate significance such as the order of phonological entities on PHON lists, and they do not introduce structural ambiguities in the configurations. They are auxiliary constructions used for making precise predictions. As far as their basic function is concerned, the same is true for the type shifting and the lambda conversion mechanism. They are supposed to specify the right readings of phrases as a consequence of the readings of their parts. However, in contrast to the `append` relation, the type shifting mechanism has structural consequences beyond its immediate empirical purpose of specifying each possible reading in a particular way. From the model-theoretic perspective, it is not a good technique for the purpose of uniquely specifying a particular intended unembedded sign configuration. It fails to fulfill the requirement that it make only empirically relevant structural distinctions.

Type shifting, lambda conversion and alpha conversion have another property which

usually does not enter into HPSG grammar specifications. They can be viewed as a computational processes. The configurations which express them in LF-Ty2 models are in a sense structural counterparts to computations. The declarative perspective of constraint-based grammar, on the other hand, is typically concerned with seeking principles which do not have any computational implications. They are thought of as declarative characterizations of empirical structures. Computing with these structures (or with their descriptions) should be a completely independent issue. Computing with type shifting and lambda conversion, however, is clearly not independent of the configurations specified by LF-Ty2.

None of these arguments against the architecture of LF-Ty2 is completely devastating or poses an insurmountable obstacle for a pragmatic perspective on LF-Ty2 or an elaborate abstract construction which can save the underlying meta-theoretic conception. Despite all differences from common practice in HPSG, LF-Ty2 is a viable framework for working with model-theoretic semantics in HPSG. However, our observations beg the question of whether there is an alternative way to model-theoretic semantics with a type theory in model-theoretic syntax, a way whose architecture and strategies stay closer to the standards of grammar specification in this framework.

This question might find an answer if we consider the potential which the techniques of model-theoretic syntax have for developing new approaches to old problems in a system of semantic composition based on the lambda calculus. An alternative system of semantic composition could take advantage of the techniques developed for the combinatorics of the uninterpreted semantic representations in [Kasper, 1996] or in MRS [Copestake et al., 2003] and of the mechanism of structural identities which is a central feature of many syntactic HPSG principles. It could use methods of underspecification which are natural in the HPSG framework and have been successfully applied in theoretical and computational semantic theories. It could approach difficult empirical problems in the syntax and semantics of negative concord phenomena from a new angle by taking a resource sensitive position and envisaging structural identities in semantic representations as an additional analytical possibility not formerly available. And, finally, such an alternative system could take seriously the meta-theory of the constraint-based framework and target unembedded sign configurations comprising syntactic and semantic subconfigurations which stand in one-to-one correspondence to the empirical predictions of the grammar. Instead of a combinatorial system which produces the readings, it could be a combinatorial system which emerges from an accumulation of restrictions which are tied to certain syntactic and semantic environments. With these ideas in mind, we will now turn to LRS.

Bibliography

[Asudeh and Crouch, 2002] Asudeh, Ash and Crouch, Richard 2002. Glue semantics for HPSG. In Dorothee Beermann Frank van Eynde, Lars Hellan (ed), *Proceedings of the 8th International Conference on Head-Driven Phrase Structure Grammar*, 1–19. CSLI Publications.

[Barwise and Perry, 1983] Barwise, Jon and Perry, John 1983. *Situations and Attitudes*. Cambridge, Mass.: The MIT Press.

[Bouma, 1994] Bouma, Gosse 1994. Calculated flexibility. In Harry Bunt, Reinhart Muskens, and Gerrit Rentier (eds), *Proceedings of the International Workshop on Computational Semantics*, 32–40. Katholieke Universiteit Brabant.

[Bouma, 2003] Bouma, Gosse 2003. Verb clusters and the scope of adjuncts in Dutch. In Pieter A. M. Seuren and Gerard Kempen (eds), *Verb Constructions in German and Dutch*, 5–42. Amsterdam and Philadelphia: Benjamins.

[Carnap, 1956] Carnap, Rudolf 1956. *Meaning and Necessity*. The University of Chicago Press, Chicago.

[Carpenter, 1992] Carpenter, Bob 1992. *The Logic of Typed Feature Structures*. Cambridge University Press. Cambridge, Massachusetts, USA.

[Chomsky, 1965] Chomsky, Noam 1965. *Aspects of the Theory of Syntax*. The M.I.T. Press.

[Cooper, 1983] Cooper, Robin 1983. *Quantification and Syntactic Theory*. D. Reidel Publishing Company, Dordrecht.

[Cooper, 1975] Cooper, Robin H. 1975. *Montague's Semantic Theory and Transformational Syntax*. PhD thesis, University of Massachusetts at Amherst.

[Copestake, 2002] Copestake, Ann 2002. *Implementing Typed Feature Structure Grammars*, (= *CSLI Lecture Notes*, 110). Stanford: CSLI Publications.

[Copestake et al., 2003] Copestake, Ann, Flickinger, Dan, Pollard, Carl, and Sag, Ivan A. 2003. Minimal Recursion Semantics: An introduction. *Note:* Journal submission, November 2003.

[Corblin and Tovena, 2001] 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*, 87–115. John Benjamins, Amsterdam.

[Curry, 1961] Curry, Haskell B. 1961. Some logical aspects of grammatical structure. In Jacobson (ed), *Structure of Language and its Mathematical Aspects: Proceedings of the Twelfth Symposium in Applied Mathematics*, 56–68. American Mathematical Society.

[Dalrymple, 1999] Dalrymple, Mary (ed) 1999. *Semantics and syntax in lexical functional grammar: the resource logic approach*. MIT Press.

[Daniels and Meurers, 2002] Daniels, Mike and Meurers, W. Detmar 2002. Improving the efficiency of parsing with discontinuous constituents. In Shuly Wintner (ed), *Proceedings of NLULP'02: The 7th International Workshop on Natural Language Understanding and Logic Programming*, (= *Datalogiske Skrifter*, 92), 49–68, Copenhagen: Roskilde Universitetscenter.

[Davidson, 1980] Davidson, Donald 1980. *Essays on Actions and Events*. Clarendon Press, Oxford.

[Dowty et al., 1981] Dowty, David R., Wall, Robert E., and Peters, Stanley 1981. *Introduction to Montague Semantics*. D. Reidel Publishing Company.

[Ebert, 2003] Ebert, Christian 2003. On the expressive completeness of underspecified representations. In *Proceedings of the 14th Amsterdam Colloquium*, Amsterdam.

[Egg, 1998] Egg, Markus 1998. *Wh*-questions in Underspecified Minimal Recursion Semantics. *Journal of Semantics*, 15:37–82.

[Egg et al., 2001] Egg, Markus, Koller, Alexander, and Niehren, Joachim 2001. The Constraint Language for Lambda Structures. *Journal of Logic, Language and Information*, 10.4:457–485.

[Frank and Reyle, 1995] Frank, Anette and Reyle, Uwe 1995. Principle based semantics for HPSG. In *Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics*, 9–16. Association for Computational Linguistics.

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

[Gazdar et al., 1985] Gazdar, Gerald, Klein, Ewan, Pullum, Geoffrey K., and Sag, Ivan A. 1985. *Generalized Phrase Structure Grammar*. Harvard University Press. Cambridge Massachusetts.

[Ginzburg and Sag, 2000] Ginzburg, Jonathan and Sag, Ivan A. 2000. *Interrogative Investigations. The Form, Meaning, and Use of English Interrogatives*. CSLI Publications.

[Groenendijk and Stokhof, 1982] Groenendijk, Jeroen and Stokhof, Martin 1982. Semantic analysis of *wh*-complements. *Linguistics and Philosophy*, 5:175–233.

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

[Halvorsen and Ladusaw, 1979] Halvorsen, Per-Kristian and Ladusaw, William A. 1979. Montague's 'Universal Grammar': An introduction for the linguist. *Linguistics and Philosophy*, 3:185–223.

[Hamm, 1999] Hamm, Fritz 1999. Modelltheoretische Untersuchungen zur Semantik von Nominalisierungen. SfS–Report–01–00, Seminar für Sprachwissenschaft der Universität Tübingen. *Note:* Habilitationsschrift.

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

[Hindley and Seldin, 1986] Hindley, J. Roger and Seldin, Jonathan P. 1986. *Introduction to Combinators and the lambda-Calculus*. Cambridge University Press.

[Höhle, 1999] Höhle, Tilman N. 1999. An architecture for phonology. In Robert D. Borsley and Adam Przepiórkowski (eds), *Slavic in Head-Driven Phrase Structure Grammar*, 61–90. Stanford: CSLI Publications.

[Janssen, 1983] Janssen, Theodoor Maria Victor 1983. *Foundations and Applications of Montague Grammar*. PhD thesis, University of Amsterdam.

[Janssen, 1997] Janssen, Theo M. V. 1997. Compositionality. In Johan van Benthem and Alice ter Meulen (eds), *Handbook of Logic and Language*, 417–473. Elsevier Science B.V.

[Johnson, 1988] Johnson, Mark 1988. *Attribute-Value Logic and the Theory of Grammar*. CSLI Publications.

[Kasper, 1996] Kasper, Robert T. 1996. Semantics of recursive modification. *Note:* Unpublished Manuscript, September 11th, 1996. The Ohio State University.

[Kathol, 1995] Kathol, Andreas 1995. *Linearization-Based German Syntax*. PhD thesis, Ohio State University.

[Kathol, 2000] Kathol, Andreas 2000. *Linear Syntax*. Oxford University Press.

[Kathol and Pollard, 1995] Kathol, Andreas and Pollard, Carl 1995. Extraposition via complex domain formation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 174–180.

[Kepser, 2004] Kepser, Stephan 2004. On the complexity of RSRL. In Lawrence S. Moss and Richard T. Oehrle (eds), *Electronic Notes in Theoretical Computer Science*. Elsevier.

[Kepser and Mönnich, 2003] Kepser, Stephan and Mönnich, Uwe 2003. Graph properties of HPSG feature structures. In Gerhard Jäger, Paola Monachesi, Gerald Penn, and Shuly Wintner (eds), *Proceedings of Formal Grammar 2003*, 115–124.

[King, 1989] King, Paul J. 1989. *A logical Formalism for Head-Driven Phrase Structure Grammar*. PhD thesis, University of Manchester.

[King, 1999] King, Paul J. 1999. Towards Truth in Head-driven Phrase Structure Grammar. In Valia Kordoni (ed), *Tübingen Studies in Head-Driven Phrase Structure Grammar*, (= *Arbeitspapiere des SFB 340, Nr. 132, Volume 2*), 301–352. Eberhard-Karls Universität Tübingen.

[Klein and Sag, 1985] Klein, Ewan and Sag, Ivan A. 1985. Type-driven translation. *Linguistics and Philosophy*, 8:163–201.

[Meurers, 2000] Meurers, Walt Detmar 2000. *Lexical Generalizations in the Syntax of German Non-Finite Constructions*. Phil. dissertation, Eberhard-Karls Universität Tübingen. *Note:* Published as: Arbeitspapiere des SFB 340, Nr. 145.

[Meurers et al., 2002] Meurers, W. Detmar, Penn, Gerald, and Richter, Frank 2002. A web-based instructional platform for constraint-based grammar formalisms and parsing. In Dragomir Radev and Chris Brew (eds), *Effective Tools and Methodologies for Teaching NLP and CL*, 18–25, New Brunswick, NJ: The Association for Computational Linguistics. *Note:* Proceedings of the Workshop held at the 40th Annual Meeting of the Association for Computational Linguistics. 7.–12. July 2002. Philadelphia, PA.

[Montague, 1974a] Montague, Richard 1974a. English as a formal language. In Richmond H. Thomason (ed), *Formal Philosophy. Selected Papers of Richard Montague*, 188–221. Yale University Press.

[Montague, 1974b] Montague, Richard 1974b. The proper treatment of quantification in ordinary English. In Richmond H. Thomason (ed), *Formal Philosophy. Selected Papers of Richard Montague*, 247–270. Yale University Press.

[Montague, 1974c] Montague, Richard 1974c. Universal grammar. In Richmond H. Thomason (ed), *Formal Philosophy. Selected Papers of Richard Montague*, 222–246. Yale University Press.

[Moshier, 1988] Moshier, Michael Andrew 1988. *Extensions to Unification Grammar for the Description of Programming Languages*. PhD thesis, University of Michigan.

[Müller, 1999] Müller, Stefan 1999. *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*, (= *Linguistische Arbeiten*, 394). Max Niemeyer Verlag, Tübingen.

[Müller, 2004] Müller, Stefan 2004. Continuous or discontinuous constituents? A comparison between syntactic analyses for constituent order and their processing systems. *Research on Language and Computation*, 2:209–257.

[Muskens, 2001] Muskens, Reinhard 2001. Talking about trees and truth-conditions. *Journal of Logic, Language and Information*, 10.4:417–455.

[Nerbonne, 1992] Nerbonne, John 1992. Constraint-based semantics. In Paul Dekker and Martin Stokhof (eds), *Proceedings of the Eighth Amsterdam Colloquium*, 425–444. Institute for Logic, Language and Information.

[Partee, 1975] Partee, Barbara 1975. Montague grammar and transformational grammar. *Linguistic Inquiry*, 6.2:203–300.

[Partee, 1976] Partee, Barbara H. 1976. *Montague Grammar*. Academic Press, New York.

[Partee, 1996] Partee, Barbara H. 1996. The development of formal semantics in linguistic theory. In Shalom Lappin (ed), *The Handbook of Contemporary Semantic Theory*, 11–38. Blackwell Publishers.

[Partee, 2004a] Partee, Barbara H. 2004a. Compositionality. In *Compositionality in Formal Semantics*, 153–181. Blackwell Publishing Ltd.

[Partee, 2004b] Partee, Barbara H. 2004b. *Compositionality in Formal Semantics*. Blackwell Publishing Ltd.

[Partee and Hendriks, 1997] Partee, Barbara H. and Hendriks, Herman L. W. 1997. Montague grammar. In Johan van Benthem and Alice ter Meulen (eds), *Handbook of Logic and Language*, 5–91. Elsevier Science B.V.

[Penka and von Stechow, 2001] Penka, Doris and von Stechow, Arnim 2001. Negative Indefinita unter Modalverben. In Reimar Müller and Marga Reis (eds), *Modalität und Modalverben im Deutschen*, (= *Linguistische Berichte. Sonderheft 9*), 263–286. Helmut Buske Verlag, Hamburg.

[Penn, 1999a] Penn, Gerald 1999a. A Generalized-Domain-Based Approach to Serbo-Croation Second Position Clitic Placement. In Gosse Bouma, Erhard Hinrichs, Geert-Jan M. Kruijff, and Richard T. Oehrle (eds), *Constraints and Resources in Natural Language Syntax and Semantics*, 119–136. CSLI Publications.

[Penn, 1999b] Penn, Gerald 1999b. Linearization and *WH*-Extraction in HPSG: Evidence from Serbo-Croatian. In Robert D. Borsley and Adam Przepiórkowski (eds), *Slavic in Head-Driven Phrase Structure Grammar*, 149–182. CSLI Publications.

[Penn, 2004] Penn, Gerald 2004. Balancing clarity and efficiency in typed feature logic through delaying. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 240–247.

[Penn and Haji-Abdolhosseini, 2003a] Penn, Gerald and Haji-Abdolhosseini, Mohammad 2003a. *ALE. The Attribute Logic Engine User's Guide with TRALE extensions*. *Note:* Version 4.0 Beta. With contributions by other authors.

[Penn and Haji-Abdolhosseini, 2003b] Penn, Gerald and Haji-Abdolhosseini, Mohammad 2003b. Topological parsing. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics, April 12–17, 2003, Budapest, Hungary*, 283–290.

[Penn and Richter, 2004] Penn, Gerald and Richter, Frank 2004. Lexical Resource Semantics: From theory to implementation. In *The 11th International Conference on Head-driven Phrase Structure Grammar. Abstracts*, 10–14. University of Leuven. Centre for Computational Linguistics.

[Pollard and Sag, 1987] Pollard, Carl and Sag, Ivan A. 1987. *Information-Based Syntax and Semantics. Vol.1: Fundamentals*. CSLI Lecture Notes 13.

[Pollard and Sag, 1994] Pollard, Carl and Sag, Ivan A. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

[Pollard, 1999] Pollard, Carl J. 1999. Strong generative capacity in HPSG. In Gert Webelhuth, Jean-Pierre Koenig, and Andreas Kathol (eds), *Lexical and Constructional Aspects of Linguistic Explanation*, 281–297. CSLI Publications.

[Pollard and Yoo, 1998] Pollard, Carl J. and Yoo, Eun Jung 1998. A unified theory of scope for quantifiers and *wh*-phrases. *Journal of Linguistics*, 34:415–445.

[Przepiórkowski, 1998] Przepiórkowski, Adam 1998. 'A unified theory of scope' revisited. Quantifier retrieval without spurious ambiguities. In Gosse Bouma, Geert-Jan M. Kruijff, and Richard T. Oehrle (eds), *Proceedings of the FHCG-98, 14–16 August 1998, Saarbrücken*, 185–195.

[Pullum and Scholz, 2001] Pullum, Geoffrey K. and Scholz, Barbara C. 2001. On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In Philippe de Groote, Glyn Morrill, and Christian Retoré (eds), *Logical Aspects of Computational Linguistics: 4th International Conference*, 17–43. Berlin: Springer-Verlag.

[Reape, 1989] Reape, Mike 1989. A logical treatment of semi-free word order and bounded discontinuous constituency. In *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics*, 103–110.

[Reape, 1990] Reape, Mike 1990. Getting things in order. In *Proceedings of the Symposium on Discontinuous Constituency*. Institute for Language Technology and Artificial Intelligence.

[Reape, 1992] Reape, Mike 1992. *A Formal Theory of Word Order: A Case Study of West Germanic*. PhD thesis, University of Edinburgh.

[Reape, 1994] Reape, Mike 1994. Domain Union and Word Order Variation in German. In John Nerbonne, Klaus Netter, and Carl Pollard (eds), *German in Head-Driven Phrase Structure Grammar*, 151–197. CSLI Publications.

[Reinhard, 2001] Reinhard, Sabine 2001. *Deverbale Komposita an der Morphologie-Syntax-Semantik-Schnittstelle: ein HPSG-Ansatz*. Phil. dissertation, Eberhard-Karls Universität Tübingen.

[Reyle, 1993] Reyle, Uwe 1993. Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics*, 10.2:123–179.

[Richter, 1997] Richter, Frank 1997. Die Satzstruktur des Deutschen und die Behandlung langer Abhängigkeiten in einer Linearisierungsgrammatik. Formale Grundlagen und Implementierung in einem HPSG-Fragment. In Erhard Hinrichs, Detmar Meurers, Frank Richter, Manfred Sailer, and Heike Winhart (eds), *Ein HPSG-Fragment des Deutschen, Teil 1: Theorie*, 13–187. Eberhard-Karls Universität Tübingen.

[Richter, 2004a] Richter, Frank 2004a. *A Mathematical Formalism for Linguistic Theories with an Application in Head-Driven Phrase Structure Grammar*. Phil. dissertation (2000), Eberhard-Karls Universität Tübingen.

[Richter, 2004b] Richter, Frank 2004b. A web-based course in grammar formalisms and parsing. *Note:* Electronic textbook distributed over the e-learning platform ILIAS at the Seminar für Sprachwissenschaft of the Eberhard-Karls Universität Tübingen.

[Richter et al., 2002] Richter, Frank, Ovchinnikova, Ekaterina, Trawiński, Beata, and Meurers, W. Detmar 2002. Interactive graphical software for teaching the formal foundations of Head-Driven Phrase Structure Grammar. In Gerhard Jäger, Paola Monachesi, Gerald Penn, and Shuly Wintner (eds), *Proceedings of Formal Grammar 2002*, 137–148.

[Richter and Sailer, 1999a] Richter, Frank and Sailer, Manfred 1999a. A lexicalist collocation analysis of sentential negation and negative concord in French. In Valia Kordoni (ed), *Tübingen Studies in Head-driven Phrase Structure Grammar*, 231–300. Eberhard-Karls Universität Tübingen.

[Richter and Sailer, 1999b] Richter, Frank and Sailer, Manfred 1999b. LF conditions on expressions of Ty2: An HPSG analysis of negative concord in Polish. In Robert D. Borsley and Adam Przepiórkowski (eds), *Slavic in Head-Driven Phrase Structure Grammar*, 247–282. Stanford: CSLI Publications.

[Richter and Sailer, 1999c] Richter, Frank and Sailer, Manfred 1999c. Underspecified semantics in HPSG. In Harry C. Blunt and Reinhard Muskens (eds), *Computing Meaning*, 95–112. Dordrecht: Kluwer Academic Publishers.

[Richter and Sailer, 2001] 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*, 257–300. Stanford: CSLI Publications.

[Richter and Sailer, 2003] Richter, Frank and Sailer, Manfred 2003. Cranberry words in formal grammar. In Claire Beyssade, Olivier Bonami, Patricia Cabredo Hofherr, and Francis Corblin (eds), *Empirical Issues in Formal Syntax and Semantics 4*, 155–171. Presses de l'Université de Paris-Sorbonne.

[Richter and Sailer, 2004a] Richter, Frank and Sailer, Manfred 2004a. Basic concepts of Lexical Resource Semantics. In Arnold Beckmann and Norbert Preining (eds), *ESSLLI 2003 – Course Material I*, (= *Collegium Logicum*, 5), 87–143. Kurt Gödel Society Wien.

[Richter and Sailer, 2004b] Richter, Frank and Sailer, Manfred 2004b. Polish negation and lexical resource semantics. In Lawrence S. Moss and Richard T. Oehrle (eds), *Electronic Notes in Theoretical Computer Science*. Elsevier.

[Richter et al., 1999] Richter, Frank, Sailer, Manfred, and Penn, Gerald 1999. A formal interpretation of relations and quantification in HPSG. In Gosse Bouma, Erhard Hinrichs, Geert-Jan M. Kruijff, and Richard Oehrle (eds), *Constraints and Resources in Natural Language Syntax and Semantics*, 281–298. Stanford: CSLI Publications.

[Sailer, 2003] 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, 2004] Sailer, Manfred 2004. Propositional relative clauses in German. In *The 11th International Conference on Head-driven Phrase Structure Grammar. Abstracts*, 155–159. University of Leuven. Centre for Computational Linguistics.

[Sailer and Richter, 2002a] Sailer, Manfred and Richter, Frank 2002a. Collocations and the representation of polarity. In Gábor Alberti, Kata Balogh, and Paul Dekker (eds), *Proceedings of the Seventh Symposium on Logic and Language*, 129–138, Pécs.

[Sailer and Richter, 2002b] Sailer, Manfred and Richter, Frank 2002b. Not for love or money: Collocations! In Gerhard Jäger, Paola Monachesi, Gerald Penn, and Shuly Wintner (eds), *Proceedings of Formal Grammar 2002*, 149–160.

[Smolka, 1992] Smolka, Gert 1992. Feature-constraint logics for unification grammars. *Journal of Logic Programming*, 12:51–87.

[Soehn, 2004] Soehn, Jan-Philipp 2004. *Über Bärendienste und erstaunte Bauklötze. Idiome ohne freie Lesart in der HPSG*. PhD thesis, Friedrich-Schiller-Universität Jena. *Note:* In preparation. Version of July 27th, 2004.

[Soehn and Sailer, 2003] Soehn, Jan-Philipp and Sailer, Manfred 2003. At first blush on tenterhooks. About selectional restrictions imposed by nonheads. In Gerhard Jäger, Paola Monachesi, Gerald Penn, and Shuly Wintner (eds), *Proceedings of Formal Grammar 2003*, 149–161.

[Suhre, 2000] Suhre, Oliver 2000. Computational aspects of a grammar formalism for languages with freer word order. Diplomarbeit, Eberhard-Karls Universität Tübingen. *Note:* Published in Arbeitspapiere des SFB 340, Nr. 154.

[Thomason, 1974] Thomason, Richmond H. (ed) 1974. *Formal Philosophy. Selected Papers of Richard Montague*. Yale University Press.

[Trawiński et al., 2004] Trawiński, Beata, Sailer, Manfred, and Soehn, Jan-Philipp 2004. Combinatorial aspects of collocational prepositional phrases. In Patrick Sain-Dizier (ed), *Computational Linguistics Dimensions of Syntax and Semantics of Prepositions*. Kluwer Academic Press. *Note:* To appear.

[van Deemter and Peters, 1996] van Deemter, Kees and Peters, Stanley (eds) 1996. *Semantic Ambiguity and Underspecification*. CSLI Publications.

[von Stechow, 1991] Stechow, Arnim von 1991. Theorie der Satzsemantik. In Arnin von Stechow and Dieter Wunderlich (eds), *Semantik: Ein internationales Handbuch der zeitgenössischen Forschung*, 90–148. Walter de Gruyter.

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

[Zimmermann, 1989] Zimmermann, Thomas Ede 1989. Intensional logic and two-sorted type theory. *The Journal of Symbolic Logic*, 54:65–77.