

Center Fragments for Upscaling and Verification in Database Semantics

Roland Hausser

Universität Erlangen-Nürnberg
Abteilung Computerlinguistik (CLUE)
rrh@linguistik.uni-erlangen.de

Abstract

The notion of a fragment was coined by Montague 1974 to illustrate the formal handling of certain puzzles, such as *de dicto/de re*, in a truth-conditional semantics for natural language. The idea of a fragment is methodological: given the vastness of a natural language, one begins with a system which is of limited data coverage, but formally explicit and functionally complete relative to a certain goal or standard.

Once a small fragment has been defined, there arises the task of *upscaling*, such as adding the treatment of another puzzle. Unfortunately, however, upscaling may turn out to be difficult or even impossible, depending on the assumptions of the initial fragment and the standard of functional completeness. For example, despite half a century of intensive research there is still no coherent formal account of a single natural language, verified computationally as nearly complete.

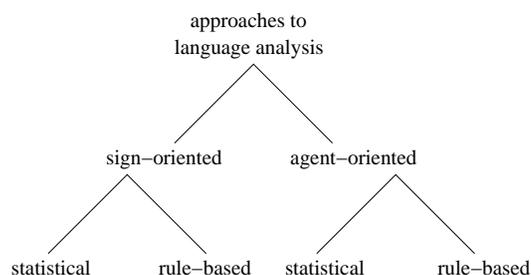
This paper proposes a new kind of fragment, called *center fragment*, designed to ensure the longterm success of upscaling. Its functional standard is the model of natural language communication of agent-oriented Database Semantics (DBS), based on the algorithm of time-linear LA-grammar and the data structure of proplets. Its language data are selected to represent the primary semantic relations of natural language, namely (a) functor-argument structure and (b) coordination, in their most basic form. The approach is illustrated with applications to four languages with distinctively different word orders, namely English, German, Korean, and Russian.

1 Sign-Oriented vs. Agent-Oriented Grammar

Of the many different ways to analyze natural language, the most basic distinction is between sign-oriented and agent-oriented approaches.¹ These may in turn each be divided into statistical and rule-based methods. For example, a sign-oriented approach based on statistics is corpus linguistics (e.g., Kučera and Francis 1967), a rule-based sign-oriented approach is the analysis of sentences within Nativist generative grammar (e.g., Chomsky 1965), an agent-oriented approach based on statistics is current work in robotics (e.g., Roy 2003), and a rule-based agent-oriented model is Database Semantics (e.g., NLC'06):

¹A related distinction is that between *language as product* and *language as action* of Clark 1996. However, while Clark counts Speech Act Theory (Austin 1962, Grice 1965, Searle 1969) among the language as action theories, it is nevertheless sign-oriented insofar as Speech Act Theory is based on enriching the analyzed language sign with performative clauses, such as *I declare*, *I promise*, etc. For a more detailed review of Ordinary Language Philosophy see FoCL'99, p. 84 f.

1.1 BASIC APPROACHES TO THE ANALYSIS OF LANGUAGE



While the statistical method is based on frequency, the rule-based method models functions or structures the importance of which may be in inverse proportion to their frequency.

The two rule-based approaches indicated in 1.1 are best distinguished in terms of their components. In a sign-oriented approach, the components are as follows:

1.2 COMPONENTS OF GRAMMAR IN A SIGN-ORIENTED APPROACH

lexicon: list of analyzed words

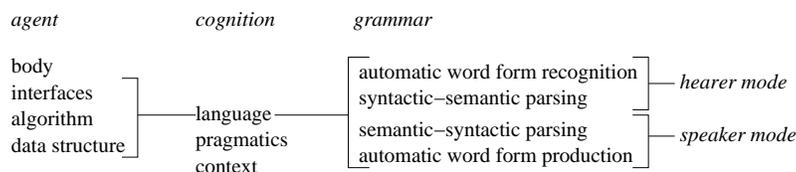
morphology: rule-based analysis of word forms

syntax: rule-based composition of word forms into sentences

semantics: rule-based interpretation of sentences

In an agent-oriented approach, the components of grammar are reorganized into the *speaker mode* and the *hearer mode*, and are embedded into a cognitive agent with language:

1.3 COMPONENTS OF A TALKING AGENT IN DATABASE SEMANTICS



At the highest level of abstraction, the agent consists of a (i) body with (ii) external interfaces for recognition, e.g., vision and hearing, and action, e.g. locomotion and manipulation, an internal database with a suitable (iii) data structure, and an (iv) algorithm for reading content into and out of the database.

The content stored in the database is called the *context* and its processing is called the pragmatics. In agents with language, the context component is complemented by a language component,² designed as a grammar with the purpose of modeling natural language communication as a computer program in a talking robot. This functionality of the agent-oriented approach profoundly changes the ontological, empirical, and methodological foundations of linguistic research as compared to the sign-oriented approaches.

First, there is a procedural notion of basic meaning: instead of the truth conditions of signs, meanings are defined in terms of the concepts used for realizing recognition and action. Second, the components of the grammar must be connected to the external interfaces of the agent, and functionally integrated into the speaker mode (recognition) and the hearer mode (action). Third, there must be a computationally well-defined interaction between the levels of language and context, such that reference is based solely on the procedures of cognition – without postulating any external relation between the sign and its referent(s) in the world.³

²For a detailed description see NLC'06, Chapters 1–3.

³Autonomy from the metalanguage, cf. FoCL'99, p. 64, 382.

2 Coding Contents as Sets of Proplets

For representing content, Database Semantics combines the traditional notion of a proposition with a modern data structure designed for indexing, storage, and retrieval: content is coded as a set of non-recursive feature structures, called *proplets*.⁴ There are three basic kinds of proplets, called nouns, verbs, and adjectives, which are combined into basic propositions. In the following examples, each set of proplets is preceded by a paraphrase which characterizes the content with an equivalent expression of English:

2.1 PROPOSITIONS WITH ONE-, TWO-, AND THREE-PLACE VERBS

1. The girl dreams.

[sur: noun: girl fnc: dream prn: 1]	[sur: verb: dream arg: girl prn: 1]
--	--

2. The man sees the girl.

[sur: noun: man fnc: see prn: 2]	[sur: verb: see arg: man girl prn: 2]	[sur: noun: girl fnc: see prn: 2]
---	--	--

3. The man gives the girl a flower.

[sur: noun: man fnc: give prn: 3]	[sur: verb: give arg: man girl flower prn: 3]	[sur: noun: girl fnc: give prn: 3]	[sur: noun: flower fnc: give prn:3]
--	--	---	--

These examples represent content at the context level because their **sur** (for *surface*) attributes have the value NIL (represented by empty space). If the proplets were to represent content at the language level, the **sur** attributes would have non-NIL values, for example [sur: träumt].

The remaining attributes of the simplified proplets in 2.1 are interpreted as follows: The second attribute, **noun** or **verb**, is called the *core* attribute of a proplet, specifies the part of speech, and takes a basic meaning, e.g., a concept, as its value. The third attribute, **fnc** or **arg**, is called a *continuation* attribute and specifies the proposition's functor-argument structure. The fourth attribute, **prn**, is called a *book-keeping* attribute and takes a proposition number as its value; proplets belonging to the same proposition have the same proposition number.

As sets, proplets are unordered, and may be stored and retrieved according to the needs of one's database. Nevertheless, proplets code the traditional semantic relations of functor-argument structure and coordination, at the context level as well as the language level.

The method of coding semantic relations is called *symbolic bidirectional pointering*. It is symbolic because the relation between, e.g., a functor like **dream** and an argument like **girl** is represented by symbols (in the sense of computer science) serving as values. It is bidirectional because any functor specifies its argument(s) in its **arg** slot and any argument specifies its functor in its **fnc** slot, and similarly for modifiers and modifieds.

This method is suitable also for coordination,⁵ as illustrated by the following example:

⁴The term proplet was introduced in Hausser 1996 and coined in analogy to "droplet." Proplets are so-called because they are the elementary items constituting a proposition.

⁵Just as the basic notion of functor-argument structure has been modeled in Predicate Calculus, the basic notion of coordination as been modeled in Propositional Calculus. However, while Predicate Calculus and

2.2 REPRESENTING Julia sleeps. John dreams. Susanne sings.

$\left[\begin{array}{l} \text{sur:} \\ \text{noun: Julia} \\ \text{fnc: sleep} \\ \text{prn: 4} \end{array} \right]$	$\left[\begin{array}{l} \text{sur:} \\ \text{verb: sleep} \\ \text{arg: Julia} \\ \text{pc:} \\ \text{nc: 5 dream} \\ \text{prn: 4} \end{array} \right]$	$\left[\begin{array}{l} \text{sur:} \\ \text{noun: John} \\ \text{fnc: dream} \\ \text{prn: 5} \end{array} \right]$	$\left[\begin{array}{l} \text{sur:} \\ \text{verb: dream} \\ \text{arg: John} \\ \text{pc: 4 sleep} \\ \text{nc: 6 sing} \\ \text{prn: 5} \end{array} \right]$	$\left[\begin{array}{l} \text{sur:} \\ \text{noun: Susanne} \\ \text{fnc: sing} \\ \text{prn: 6} \end{array} \right]$	$\left[\begin{array}{l} \text{sur:} \\ \text{verb: sing} \\ \text{arg: Susanne} \\ \text{pc: 5 dream} \\ \text{nc:} \\ \text{prn: 6} \end{array} \right]$
---	---	--	---	--	--

The coordination is coded in the *pc* (for previous conjunct) and the *nc* (for next conjunct) slot of the verb of the elementary propositions, using a proposition number, e.g., 5, and a verb, e.g., *dream*, as values.⁶ Note that the initial verb proplet of this extrapositional coordination has an empty *pc* slot, while the last verb proplet has an empty *nc* slot.⁷

3 Center Fragments for Different Word Orders

The semantic analysis of natural language meaning has two basic aspects, (i) lexical semantics and (ii) compositional semantics. In Database Semantics, the aspect of lexical semantics is treated in terms of the core value of the proplets, while the aspect of compositional semantics is treated in terms of the continuation values. Accordingly, the three propositions coordinated in Example 2.2 differ in their lexical semantics, but their respective intrapropositional compositional semantics are equivalent.

A center fragment focuses on the compositional aspects of natural language semantics, and may be characterized as follows:

3.1 THE SEVEN CRITERIA OF A WELL-DEFINED CENTER FRAGMENT

1. Semantic relations

A center fragment handles the primary semantic relations of natural language, namely functor-argument structure and coordination. These are assumed to be universal in the sense that they may be found in all natural languages.

2. Data coverage

A center fragment handles the primary semantic relations in principle, but in their simplest form. Therefore a center fragment is purposely limited to functor-argument structure at the level of *elementary* nouns, verbs, (e.g., Example 2.1) and adjectives, and to coordination at the *clausal* level (e.g., Example 2.2).

3. Functional standard

The functional standard of a center fragment is that of Database Semantics, i.e., modeling the mechanism of natural language communication in the speaker mode (language production) and the hearer mode (language interpretation).

4. Formal standard

A center fragment must be specified as an explicit formal definition suitable as the declarative specification for computational implementations.

Propositional Calculus are sign-oriented, metalanguage-based, and truth-conditional, Database Semantics is agent-oriented, procedural, and based on bidirectional pointering.

⁶For a detailed analysis of intra- and extrapositional coordination in English, including gapping, see NLC'06, Chapters 8 and 9.

⁷This is similar to the linear data structure of a linked list.

5. Equivalence

The center fragments of different natural languages are called equivalent if they handle the same grammatical relations at the level of proplets. Thus, equivalent center fragments map a given set of proplets into equivalent surfaces of different languages (speaker mode) and equivalent surfaces of different languages into the same – or rather equivalent– sets of proplets (hearer mode).

6. Upscaling

There are two systematic ways of upscaling, namely grammar-based and frequency-based. Grammar-based upscaling consists in complementing the elementary nouns, verbs, and adjectives with their phrasal and clausal counterparts, and complementing clausal coordination with the coordination of elementary and phrasal nouns, verbs, and adjectives. Frequency-based upscaling consists in ordering the constructions found in a corpus and integrating those not yet handled, beginning with the most frequent.

7. Verification

A center fragment of a natural language and each subsequent step of upscaling must be verified computationally by implementing the system as a running program and testing it automatically on the data selected. Thereby the functional (cf. 3) and formal (cf. 4) standards of Database Semantics must be maintained in full.

Based on the representation of one-place, two-place, and three-place propositions in 2.1, let us define four equivalent center fragments for languages with different word orders.⁸ The following schema shows the word order variations of English, German, Korean, and Russian:

3.2 SCHEMATIC COMPARISON OF WORD ORDERS

	<i>V-first</i>	<i>V-second</i>	<i>V-third</i>	<i>V-last</i>
<i>one-place</i>	VS	SV	English	(SV)
<i>two-place</i>	VSO	SVO	German	SOV
	VOS	OVS	Korean	OSV
<i>three-place</i>	VSID	SVID	SIVD	SIDV
	VSDI	SVDI	SDVI	SDIV
	VISD	IVSD	ISVD	ISDV
	VIDS	IVDS	IDVS	IDSV
	VDSI	DVSI	DSVI	DSIV
	VDIS	DVIS	DIVS	DISV

S stands for Subject or nominative, V for Verb, O for Object,⁹ D for Direct object or accusative, and I for Indirect object or dative. The permutations are ordered (i) according to the places of the verb (one-place in line 1, two-place in lines 2 and 3, and three-place in lines 4–9), and (ii) according to the position of the verb in the sentence (initial position in column 1, second position in column 2, third position in column 3, and last position in column 4).¹⁰

⁸For reasons of space, elementary adjectives and extrapositional coordination are omitted, and the formal center fragments are specified in the hearer mode only. In NLC'06, Chapters 11–14, complete fragments with LA-hear, LA-think, and LA-speak grammars are explicitly defined as declarative specifications which have been verified by a concomitant implementation in Java (Kycia 2004).

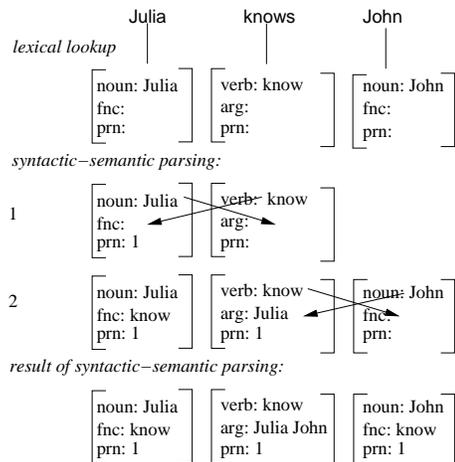
⁹The terminology of S, V, and O follows Greenberg 1963.

¹⁰Note that S V occurs twice in this table, once as verb-second (English and German) and once as verb-last (Korean and Russian).

Due to different word order possibilities, the three propositions of 2.1 have a total of three surfaces¹¹ in English, of nine in German and Korean, and of thirty-two in Russian.

The challenge presented by the four different center fragments is to provide the correct semantic relations for the different word orders, using a strictly time-linear derivation order in the hearer mode. Consider the following example of an SVO derivation (e.g., English):

3.3 TIME-LINEAR HEARER-MODE ANALYSIS OF AN SVO SURFACE

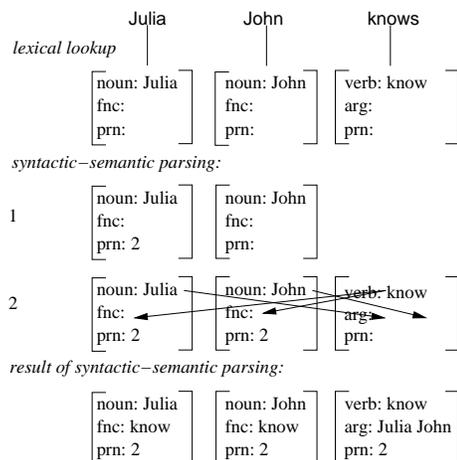


The derivation begins with an incremental lexical lookup, which relates each incoming surface to a corresponding proplet. These lexical proplets are called *isolated* proplets because their continuation attributes have no values yet. The isolated proplets are turned into *connected* proplets by a variant of LA-grammar, called LA-hear.

The task of an LA-hear grammar is to provide the most direct buildup of semantic relations possible in a strictly time-linear derivation order. This is done mainly by copying values between proplets. For example, in line 1 the core value of the *Julia* proplet is copied into the *arg* slot of the *know* proplet, and the core value of the *know* proplet is copied into the *fnc* slot of the *Julia* proplet (symbolic bidirectional pointering), and accordingly in line 2. The result is a content coded as a set of proplets ready to be sorted into the database.

While the SVO order permits immediate cross-copying between the verb and its arguments, this is not possible in the SOV order (e.g., Korean), as shown by the following example:

3.4 TIME-LINEAR HEARER-MODE ANALYSIS OF AN SOV SURFACE



¹¹For better comparison of the four center fragments, additional surfaces such as *The girl was seen by the man* (passive) or *the man gave a flower to the girl* (prepositional dative) are disregarded here.

In line 1, the first and the second noun cannot be related in terms of functor-argument structure (no verb yet). Such adding of a lexical proplet without building a semantic relation to any proplet(s) in the sentence start is called *suspension* (cf. NLC'06, Section 7.6). After a suspension, all the necessary cross-copying takes place as soon as possible and at once, as shown in line 2. The result is a content equivalent to the one derived in Example 3.3.

4 Fixed Noun Order, Verb Second: English

Having specified the language phenomena to be handled by our four center fragments intuitively, we turn now to defining formally explicit LA-hear grammars for them. An LA-hear grammar is a symbolic system which consists of (i) word form recognition, (ii) a variable definition, and (iii) syntactic-semantic rules. It is embedded into a language-independent software machine called the *motor*, currently implemented in Java.

In the hearer mode, the motor takes two kinds of language-dependent input, namely (a) an LA-hear grammar (prior to run time) and (b) unanalyzed surfaces of the language in question (during run time). The output of the motor are proplets which represent the content¹² coded by the input surfaces by means of symbolic bidirectional pointering.

The linguistic analysis of an LA-hear grammar differs from the sign-oriented approaches of Nativism and Modeltheoretic Semantics in three fundamental ways:

4.1 CHARACTERISTIC PROPERTIES OF LA-HEAR GRAMMARS

1. *Rules have an external interface based on pattern matching*

The rules of LA-grammar in general and LA-hear in particular analyze input based on pattern matching, whereas the rules of the sign-oriented approaches are typically based on possible substitutions without any external interface.¹³

2. *No separation of syntactic composition and semantic interpretation*

LA-hear treats the syntactic composition and the semantic interpretation simultaneously, whereas the sign-oriented approaches typically handle the syntactic composition first and then add a separate semantic interpretation.

3. *Strictly time-linear derivation order*

LA-hear integrates the syntactic-semantic composition into a strictly time-linear derivation order which computes possible continuations, whereas the sign-oriented approaches combine the words and phrases according to what belongs together semantically, resulting in irregular hierarchies called constituent structure.

As a reference grammar for comparing different kinds of time-linear syntactic-semantic compositions in the following sections, let us define an LA-hear grammar for the center fragment of English, represented by the sentences analyzed in Example 2.1. To simplify matters, the word form recognition system for these sentences is (i) defined as a full-form lexicon and (ii) treats the noun phrases `the_girl`, `the_man` and `a_flower` as elementary proplets.¹⁴ To illustrate the handling of grammatical agreement, the word form recognition also handles the plural `the_girls` and the unmarked verb form `dream`:

¹²I.e., the compositional semantics (grammatical relations) of the input expressions.

¹³This applies specifically to the rewrite rules of context-free phrase structure grammar. Though Chomsky's transformations (cf. FoCL'99, Example 8.5.3) have an interface based on matching, it is only an *internal* interface (i.e., transformations modify phrase structure trees rather than taking external language input). The rules of Categorical Grammar (cf. FoCL'99, Sections 7.5, 7.5) have external interfaces, but they compute possible substitutions rather than possible continuation and are therefore not time-linear.

¹⁴For the proper incremental treatment see NLC'06, Chapter 13.

4.2 WORD FORM RECOGNITION SYSTEM OF LAH.English.1

sur: the_girl noun: girl cat: snp fnc: prn:	sur: the_girls noun: girl cat: pnp fnc: prn:	sur: the_man noun: man cat: snp fnc: prn:	sur: a_flower noun: flower cat: snp fnc: prn:
sur: dreams verb: dream cat: ns3' v arg: prn:	sur: dream verb: dream cat: n-s3' v arg: prn:	sur: sees verb: see cat: ns3' a' v arg: prn:	sur: gives verb: give cat: ns3' d' a' v arg: prn:

A system of automatic word form recognition like 4.2, based on lexical lookup from a full-form lexicon, is technically very simple: it consists of matching an incoming unanalyzed surface, e.g., *dreams*, with the *sur* value of the corresponding lexical full-form proplet.¹⁵

The second component of an LA-grammar, i.e., the variable definition, is needed for specifying the patterns which provide the LA-grammar rules with an external interface. In order for a rule pattern and a language proplet to successfully match, (i) the attributes of the rule pattern must be a subset of the attributes of the language proplet and (ii) the values of the rule pattern must be compatible with the corresponding values of the language proplet.

The values of a rule pattern consist of variables and constants. Because of the variables, a rule pattern (type) can match any number of language proplets (token) of the same kind. The compatibility of corresponding values at the rule and the language level is defined as follows: any constant at the rule level must be matched by the same constant at the language level, and any variable at the rule level must be matched by a constant at the language level which is in the *restriction set* of the variable.

Variables with general restrictions are written as lower case Greek letters like α , β , γ , which match any string of characters, or as X, Y, Z, which match any sequence of zero, one, two, or three constants. Variables with specific restrictions are defined in the variable definition:

4.3 VARIABLE DEFINITION OF LAH.English.1

1. Variable restrictions:
 NP \in {snp, pnp}
 NP' \in {ns3', n-s3', d', a'}
2. Variable constraints (agreement conditions):
 If NP \in {snp}, then NP' \in {ns3', d', a'}.
 If NP \in {pnp}, then NP' \in {n-s3', d', a'}.

The variable NP (for noun phrase) is restricted to the category segments snp (for singular noun phrase) and pnp (for plural noun phrase), which represent nominal valency *fillers*. The variable NP' is restricted to the category segments ns3' (for nominative singular 3rd person, e.g., *dreams*), n-s3' (for nominative minus singular 3rd person, e.g., *dream*), d' (for dative), and a' (for accusative), which represent nominal valency *positions* in the category of a verb.

According to the constraint, any rule containing the variables NP as well as NP' will only match the input at the language level if the variable NP is bound either (i) to the constant snp and the variable NP' is bound to a constant ns3', d', or a', or (ii) NP is bound to the constant pnp and NP' is bound to a constant n-s3', d', or a'. In this way, agreement violations as in *The girl dream or *The girls dreams will not be accepted. As shown in FoCL'99,

¹⁵For an overview of different methods of automatic word form recognition see FoCL'99, Chapters 13-15.

17.4.1, and NLC'06, 13.2.1, the variable definition provides a simple and effective method for handling grammatical agreement in all its forms, without cluttering up the rule system.

The rule system of an LA-*hear* grammar defines (i) a set of start states ST_S , (ii) a set of rules, and (iii) a set of final states ST_F . Consider the following example:

4.4 RULE SYSTEM OF LAH.English.1

$ST_S =_{def} \{ ([cat: X] \{1 N+V\}) \}$

$N+V \quad \{2 V+N\}$

$\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: NP} \\ \text{fnc:} \end{bmatrix}$	$\begin{bmatrix} \text{verb: } \beta \\ \text{cat: NP}' Y v \\ \text{arg:} \end{bmatrix}$	<p>delete NP' nw.cat acopy α nw.arg ecopy β ss.fnc copy_{ss} copy_{nw}</p>
---	---	--

$V+N \quad \{3 V+N\}$

$\begin{bmatrix} \text{verb: } \alpha \\ \text{cat: NP}' Y v \\ \text{arg:} \end{bmatrix}$	$\begin{bmatrix} \text{noun: } \beta \\ \text{cat: NP} \\ \text{fnc:} \end{bmatrix}$	<p>delete NP' ss.cat acopy β ss.arg ecopy α nw.fnc copy_{ss} copy_{nw}</p>
--	--	--

$ST_F =_{def} \{ ([cat: v] rp_{N+V}), ([cat: v] rp_{V+N}) \}$

A start state specifies (i) a pattern for a possible first word of a sentence or text and (ii) a rule package listing all the rules which may apply to the initial word.¹⁶ A rule consists of (i) a rule name, (ii) a rule package, (iii) a pattern for the sentence start, (iv) a pattern for the next word, and (v) a set of operations.¹⁷ A final state specifies a pattern characterizing a complete sentence and the rule package of a sentence-completing rule.¹⁸

As an illustration of a rule application, consider the following example. It shows the rule $N+V$ defined in 4.4 applying to the proplets *man* and *see* defined in 4.2:

4.5 EXAMPLE OF AN LA-HEAR RULE APPLICATION

	$N+V$	$\{2 V+N\}$	
<i>rule level</i>	$\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: NP} \\ \text{fnc:} \end{bmatrix}$	$\begin{bmatrix} \text{verb: } \beta \\ \text{cat: NP}' Y v \\ \text{arg:} \end{bmatrix}$	<p>delete NP' nw.cat acopy α nw.arg ecopy β ss.fnc copy_{ss} copy_{nw}</p>
<i>language level</i>	$\begin{bmatrix} \text{sur: the_man} \\ \text{noun: man} \\ \text{cat: snp} \\ \text{fnc:} \\ \text{prn: 7} \end{bmatrix}$	$\begin{bmatrix} \text{sur: sees} \\ \text{verb: see} \\ \text{cat: ns3' a' v} \\ \text{arg:} \\ \text{prn:} \end{bmatrix}$	

Matching between the rule level and the language level is successful because the attributes of the rule patterns are subsets of the attributes of the corresponding language proplets. Further-

¹⁶Here, the pattern $[cat: X]$ will accept any first word. The rule package, $\{1 N+V\}$, contains only one rule and ensures that any derivation will begin with an application of the rule $N+V$ (for noun plus verb).

¹⁷For example, the first rule of the above LA-*hear* grammar has (i) the name $N+V$, (ii) the rule package $\{2 V+N\}$, (iii) a sentence start pattern for a noun proplet, (iv) a next word pattern for a verb proplet, and (v) four operations.

¹⁸In the above example, there are two final states, one ending with an application of the rule $N+V$ (in an intransitive sentence), the other ending in an application of the rule $V+N$ (in a transitive sentence). The pattern $[cat: v]$ represents a verb proplet with no uncanceled valency positions.

more, the restrictions of the variables NP, NP', and Y as well as the constraint defined in 4.3 are satisfied.

During matching, the variables α and β are vertically bound¹⁹ to the values *man* and *see*, respectively, and the variables NP, NP' and Y are vertically bound to the values *snp*, *ns3'*, and *a'*, respectively. The vertical binding of rule level variables to language level values is the precondition for executing the operations at the language level. The assignment to a variable (scope) holds within a rule application.

The operation *delete NP' nw.cat* deletes the value assigned to the variable NP', i.e., *sn3'*, in the *cat* attribute of the next word proplet. The operation *acopy α nw.arg* adds the value assigned to the variable α , i.e., *man*, to the *arg* slot of the next word proplet, while the operation *ecopy β ss.fnc* copies the value assigned to the variable β , i.e., *see*, to the empty *fnc* slot of the sentence start. The operations *copy_{ss}* *copy_{nw}* retain the proplets of the sentence start and the next word proplet in the output.

The successful rule application shown in Example 4.5 has the following result:

4.6 RESULT OF THE LA-HEAR RULE APPLICATION

[sur: the_man]	[sur: sees]
noun: man	verb: see
cat: snp	cat: a' v
fnc: see	arg: man
prn: 7	prn: 7

In addition to the execution of the operations, the control structure of the motor provides the *prn* attribute of the next word with a value, here 7. Next, the rule V+N contained in the rule package of N+V is applied to the verb proplet *see* in 4.6 and to the lexical proplet *girl* defined in 4.2. The overall derivation corresponds to 3.3.

5 Free Noun Order, Verb Second: German

Because English is a fixed word order language with the finite verb in post-nominative position in declarative sentences, each kind of functor-argument structure in Example 2.1 has only one corresponding surface. The nouns are morphologically unmarked for case,²⁰ and get their syntactic-semantic case role assigned by the valency position they fill in the verb.

German, in contrast, is a language with a comparatively free word order: the arguments (nouns) may occur in any order, while the finite verb must be in second position in declarative main clauses. This results in three times as many surfaces as in English:

5.1 THE 9 BASIC FUNCTOR-ARGUMENT SURFACES OF GERMAN

- one-place verb: 1 surface

man_{nom} dream N+V

- two-place verb: 2 surfaces

man_{nom} see girl_{acc} N+V, V+N

girl_{acc} see man_{nom}

¹⁹The vertical binding of variables in Database Semantics is in contradistinction to the horizontal binding of variables by quantifiers in Predicate Calculus (cf. NLC'06, Section 5.3).

²⁰With the exception of the personal pronouns, which are morphologically distinguished between nominative and oblique case, e.g., *she* vs. *her*. For a detailed discussion see FoCL'99, Section 17.2.

- three-place verb: 6 surfaces

man_ *nom* give girl_ *dat* flower_ *acc* N+V, V+N, V+N
 man_ *nom* give flower_ *acc* girl_ *dat*
 girl_ *dat* give man_ *nom* flower_ *acc*
 girl_ *dat* give flower_ *acc* man_ *nom*
 flower_ *acc* give girl_ *dat* man_ *nom*
 flower_ *acc* give man_ *nom* girl_ *dat*

The examples are ordered into blocks, each followed by the associated rule sequence.

The free order of nouns in German is supported by morphological case marking. This is why the schematic full-form lexicon of English in Example 4.2 has only one entry for each singular noun, while the nouns in the corresponding full-form lexicon of German each have three entries (cf. 5.2 below), assuming unambiguous morphological case marking.

Besides German, Korean and Russian also have morphologically case-marked nouns, though each by different means: German by means of determiners and inflectional endings (though highly defective), Korean by agglutinative endings (cf. Lee 2004), and Russian by inflectional endings alone. Rather than defining the following word form recognition system only for German, let us define it for case marking languages in general.

Limited to the word forms needed for the respective center fragments of German, Korean, and Russian (cf. 3.2), this generic word form recognition system for case marking languages handles three nouns unmarked for number, each in the nominative, dative, and accusative, and three verbs with valencies for one, two, and three arguments:

5.2 WORD FORM RECOGNITION SYSTEM FOR CASE-MARKED NOUNS

[sur: man_ <i>nom</i> noun: man cat: n fnc: prn	[sur: man_ <i>dat</i> noun: man cat: d fnc: prn	[sur: man_ <i>acc</i> noun: man cat: a fnc: prn
[sur: girl_ <i>nom</i> noun: girl cat: n fnc: prn	[sur: girl_ <i>dat</i> noun: girl cat: d fnc: prn	[sur: girl_ <i>acc</i> noun: girl cat: a fnc: prn
[sur: flower_ <i>nom</i> noun: flower cat: n fnc: prn	[sur: flower_ <i>dat</i> noun: flower cat: d fnc: prn	[sur: flower_ <i>acc</i> noun: flower cat: a fnc: prn
[sur: dream_ <i>n</i> verb: dream cat: n' v arg: prn	[sur: see_ <i>n+a</i> verb: see cat: n' a' v arg: prn	[sur: give_ <i>n+d+a</i> verb: give cat: n' d' a' v arg: prn

The following variable definition is also generic, applying to all three of the case-marking languages in question:

5.3 VARIABLE DEFINITION FOR CASE-MARKING LANGUAGES

1. Variable restriction:

$$\begin{aligned} \text{NP} &\in \{n, d, a\} \\ \text{NP}' &\in \{n', d', a'\} \end{aligned}$$

2. Variable constraint (agreement conditions)

$$\begin{aligned} \text{If } \text{NP} \in \{n\}, &\text{ then } \text{NP}' \in \{n'\}. \\ \text{If } \text{NP} \in \{d\}, &\text{ then } \text{NP}' \in \{d'\}. \\ \text{If } \text{NP} \in \{a\}, &\text{ then } \text{NP}' \in \{a'\}. \end{aligned}$$

According to the variable constraint, an n' valency position in the verb can only be canceled by an n noun, and similarly for the other cases. Thus a rule with the variables NP and NP' will not match any language proplets which would, for example, assign the value d to NP and the value a' to NP' .

Assuming the abstract word form analysis 5.2 and the variable definition 5.3, the rule system of LAH.German.1 is defined as follows:

5.4 RULE SYSTEM OF LAH.German.1

$$\text{ST}_S =_{\text{def}} \{ ([\text{cat}: X] \{1 \text{ N+V}\}) \}$$

$$\text{N+V} \quad \{2 \text{ V+N}\}$$

$$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: NP} \\ \text{fnc:} \end{array} \right] \left[\begin{array}{l} \text{verb: } \beta \\ \text{cat: X NP' Y v} \\ \text{arg:} \end{array} \right] \quad \begin{array}{l} \text{delete } \{\text{NP}'\} \text{ nw.cat} \\ \text{acopy } \alpha \text{ nw.arg} \\ \text{ecopy } \beta \text{ ss.fnc} \\ \text{copy}_{ss} \text{ copy}_{nw} \end{array}$$

$$\text{V+N} \quad \{3 \text{ V+N}\}$$

$$\left[\begin{array}{l} \text{verb: } \alpha \\ \text{cat: X NP' Y v} \\ \text{arg:} \end{array} \right] \left[\begin{array}{l} \text{noun: } \beta \\ \text{cat: NP} \\ \text{fnc:} \end{array} \right] \quad \begin{array}{l} \text{delete } \text{NP}' \text{ ss.cat} \\ \text{acopy } \beta \text{ ss.arg} \\ \text{ecopy } \alpha \text{ nw.fnc} \\ \text{copy}_{ss} \text{ copy}_{nw} \end{array}$$

$$\text{ST}_F =_{\text{def}} \{ ([\text{cat}: v] \text{rp}_{\text{N+V}}), ([\text{cat}: v] \text{rp}_{\text{V+N}}) \}$$

Even though the rule system 4.4 for English handles three surfaces with a fixed noun order and the rule system 5.4 for German handles nine surfaces with a free noun order, the grammars are very similar. In fact, the only difference is in the respective **cat** values of the rules' verb patterns. In English, this pattern is **[cat: NP' Y v]**, whereas in German it is **[cat: X NP' Y v]**.

In other words, in the English pattern, the valency position NP' is not preceded by the variable X , while in German it is. Thus, given the proper word form analyses of nouns and the associated variable definitions, a noun in English always cancels the leftmost valency position in the verb, thereby obtaining its case role. In German, in contrast, any valency position of the verb may be canceled during a time-linear derivation as long as the filler is of a corresponding case.

6 Free Noun Order, Verb Final: Korean

Unlike English, but similar to German, Korean has a free order of nouns. In contrast to German, however, the verb is in final position. Thus, the surfaces of Korean coding the basic functor-argument structures corresponding to Example 5.1 are as follows:

6.1 THE 9 BASIC FUNCTOR-ARGUMENT SURFACES OF KOREAN

- one-place verb: 1 pattern

man_nom dream N+V

- two-place verb: 2 patterns

man_nom girl_acc see N+N, N+V
girl_acc man_nom see

- three-place verb: 6 patterns

man_nom girl_dat flower_acc give N+N, N+N, N+V
man_nom flower_acc girl_dat give
girl_dat man_nom flower_acc give
girl_dat flower_acc man_nom give
flower_acc girl_dat man_nom give
flower_acc man_nom girl_dat give

These surfaces are analyzed by only two rules. However, while these rules are called N+V and V+N in English and German, they are called N+N and N+V in LAH.Korean.1.

Because Korean nouns are case-marked,²¹ the definition of LAH.Korean.1 may reuse the generic word form recognition 5.2 and the generic variable definition 5.3. The main difference between LAH.Korean.1 on the one hand, and LAH.English.1 and LAH.German.1 on the other, is in the rule system:

6.2 RULE SYSTEM OF LAH.Korean.1

$ST_S =_{def} \{ ([cat: X] \{1 N+N, 2 N+V\}) \}$

N+N {3 N+N, 4 N+V}

$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: NP}_1 \end{array} \right] \left[\begin{array}{l} \text{noun: } \beta \\ \text{cat: NP}_2 \end{array} \right] \text{copy}_{ss} \text{copy}_{nw}$

N+V { }

$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: NP}_1 \\ \text{fnc:} \end{array} \right] \left[\begin{array}{l} \text{noun: } \beta \\ \text{cat: NP}_2 \\ \text{fnc:} \end{array} \right] \left[\begin{array}{l} \text{noun: } \gamma \\ \text{cat: NP}_3 \\ \text{fnc:} \end{array} \right] \left[\begin{array}{l} \text{verb: } \delta \\ \text{cat: } \{ \text{NP}'_1 \text{ NP}'_2 \text{ NP}'_3 \} \text{v} \\ \text{arg:} \end{array} \right] \text{cancel } \{ \text{NP}'_1 \text{ NP}'_2 \text{ NP}'_3 \} \text{nw.cat}$
acopy $\alpha \beta \gamma$ nw.arg
ecopy δ ss.fnc
copy_{ss} copy_{nw}

$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: NP}_1 \\ \text{fnc:} \end{array} \right] \left[\begin{array}{l} \text{noun: } \beta \\ \text{cat: NP}_2 \\ \text{fnc:} \end{array} \right] \left[\begin{array}{l} \text{verb: } \gamma \\ \text{cat: } \{ \text{NP}'_1 \text{ NP}'_2 \} \text{v} \\ \text{arg:} \end{array} \right] \text{cancel } \{ \text{NP}'_1 \text{ NP}'_2 \} \text{nw.cat}$
acopy $\alpha \beta$ nw.arg
ecopy γ ss.fnc
copy_{ss} copy_{nw}

$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: NP} \\ \text{fnc:} \end{array} \right] \left[\begin{array}{l} \text{verb: } \beta \\ \text{cat: NP}' \text{v} \\ \text{arg:} \end{array} \right] \text{delete NP}' \text{nw.cat}$
acopy α nw.arg
ecopy β ss.fnc
copy_{ss} copy_{nw}

$ST_F =_{def} \{ ([cat: v] \text{rp}_{N+V}) \}$

One difference is in the finite state transition networks defined by the rules and rule packages of LAH.English.1 and LAH.German.1 (cf. 8.2) compared to that of LAH.Korean.1 (cf. 8.3):

²¹See Choe et al. 2007 for a detailed analysis of Korean noun phrases within Database Semantics.

First, in LAH.Korean.1 there are two start states, one for intransitive sentences like *man sleep*, beginning with N+V, the other for transitive sentences like *man girl see* or *man girl flower give*, beginning with N+N. Second, the rule package of N+N contains two rules, N+N for adding another noun, as in *man girl + flower*, and N+V for adding the final verb, as in *man girl + see*. Third, the rule N+V has an empty rule package, indicating the end of the derivation.²²

The other difference is in the patterns and operations of the rules themselves: The rule N+N simply adds the next word to the sentence start, without any cross-copying (suspension, cf. 3.4). Minimal change is provided by the control structure of the motor, which adds the current proposition number to the *prn* slot of the next word proplet.

The rule N+V has three subclauses, one for three-place verbs, one for two-place verbs, and one for one-place verbs. These clauses are tried on the input in sequence: If the clause for a three-place verb does not match, the one for a two-place verb is tried; if the clause for a two-place clause does not match, the one for a one-place verb is tried. If this one fails as well, the application of the rule N+V fails as whole.

As in German, the noun fillers may cancel valency positions in any order. Unlike German, however, the canceling takes place all at once at the end. For this, the matching between the rule patterns and corresponding language proplets needs to be adapted in N+V: the set parentheses { } in the *cat* pattern of the verb in the first clause, i.e., {NP₁' NP₂' NP₃'} v, indicate that the surface order of the noun fillers NP₁, NP₂, and NP₃ may differ from the order of the corresponding valency positions in the category of the verb, and similarly for the second clause. Consider the following example:

6.3 N+V MATCHING FREE NOUN ORDER AND THREE-PLACE VERB

<i>rule patterns</i>	[noun: α cat: NP ₁ fnc:]	[noun: β cat: NP ₂ fnc:]	[noun: γ cat: NP ₃ fnc:]	[verb: δ cat: {NP ₁ ' NP ₂ ' NP ₃ '} v arg:]
<i>language proplets</i>	[sur: flower_acc noun: flower cat: a fnc: prn]	[sur: man_nom noun: man cat: n fnc: prn]	[sur: girl_dat noun: girl cat: d fnc: prn]	[sur: give_s3+d+a verb: give cat: n' d' a' v arg: prn]

Here, NP₁, NP₂, and NP₃ are bound to the values *a*, *n*, and *d*, respectively, thus restricting NP₁', NP₂', and NP₃' to the values *a'*, *n'*, and *d'*, respectively (variable constraint of 5.3). Nevertheless, the *cat* values of the verb pattern are compatible with the *cat* values *n' d' a' v* of the *give* proplet, due to the set parentheses in the verb pattern. The new operation *cancel* differs from *delete* in that it fills more than one valency position.

7 Completely Free Word Order: Russian

In Russian, not only the noun order is free, but also the position of the verb. Given the three propositions underlying the center fragments of English (3 surfaces), German (9 surfaces), and Korean (9 surfaces), this results in a total of 32 surfaces (cf. 3.2). They are analyzed by three rules, called N+V, N+N, and V+N.

²²This, however, is only temporary and due to the tiny size of the current fragment. For example, as soon as clause-final modifiers, interpunctuation, or extrapositional coordination are added, the rule package of N+V will not be empty, but contain the associated rules.

Because Russian nouns are case-marked, the definition of LAH.Russian.1 may reuse the generic word form recognition system 5.2 and the generic variable definition 5.3 (like LAH.-Korean.1). The LAH.Russian.1 rule system has a start state with a rule package containing N+N for verb third or fourth (last), N+V for verb second, and V+N for verb first:

7.1 RULE SYSTEM OF LAH.Russian.1

$$\mathbf{ST}_S =_{def} \{ ([cat: X] \{1 N+N, 2 N+V, 3 V+N\}) \}$$

$$\mathbf{N+N} \quad \{4 N+N, 5 N+V\}$$

$$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: NP}_1 \end{array} \right] \left[\begin{array}{l} \text{noun: } \beta \\ \text{cat: NP}_2 \end{array} \right] \quad \text{copy}_{ss} \text{ copy}_{nw}$$

$$\mathbf{N+V} \quad \{6 V+N\}$$

$$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: NP}_1 \\ \text{fnc:} \end{array} \right] \left[\begin{array}{l} \text{noun: } \beta \\ \text{cat: NP}_2 \\ \text{fnc:} \end{array} \right] \left[\begin{array}{l} \text{noun: } \gamma \\ \text{cat: NP}_3 \\ \text{fnc:} \end{array} \right] \left[\begin{array}{l} \text{verb: } \delta \\ \text{cat: } \{ \text{NP}'_1 \text{ NP}'_2 \text{ NP}'_3 \} \text{ v} \\ \text{arg:} \end{array} \right] \quad \begin{array}{l} \text{cancel } \{ \text{NP}'_1 \text{ NP}'_2 \text{ NP}'_3 \} \text{ nw.cat} \\ \text{acopy } \alpha \beta \gamma \text{ nw.arg} \\ \text{ecopy } \delta \text{ ss.fnc} \\ \text{copy}_{ss} \text{ copy}_{nw} \end{array}$$

$$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: NP}_1 \\ \text{fnc:} \end{array} \right] \left[\begin{array}{l} \text{noun: } \beta \\ \text{cat: NP}_2 \\ \text{fnc:} \end{array} \right] \quad \left[\begin{array}{l} \text{verb: } \gamma \\ \text{cat: } \{ X \text{ NP}'_1 \text{ NP}'_2 \} \text{ v} \\ \text{arg:} \end{array} \right] \quad \begin{array}{l} \text{cancel } \{ \text{NP}'_1 \text{ NP}'_2 \} \text{ nw.cat} \\ \text{acopy } \alpha \beta \text{ nw.arg} \\ \text{ecopy } \gamma \text{ ss.fnc} \\ \text{copy}_{ss} \text{ copy}_{nw} \end{array}$$

$$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: NP} \\ \text{fnc:} \end{array} \right] \quad \left[\begin{array}{l} \text{verb: } \beta \\ \text{cat: } \{ X \text{ NP}' \} \text{ v} \\ \text{arg:} \end{array} \right] \quad \begin{array}{l} \text{cancel } \{ \text{NP}' \} \text{ nw.cat} \\ \text{acopy } \alpha \text{ nw.arg} \\ \text{ecopy } \beta \text{ ss.fnc} \\ \text{copy}_{ss} \text{ copy}_{nw} \end{array}$$

$$\mathbf{V+N} \quad \{7 V+N\}$$

$$\left[\begin{array}{l} \text{verb: } \alpha \\ \text{cat: } X \text{ NP}' \text{ Y v} \\ \text{arg:} \end{array} \right] \left[\begin{array}{l} \text{noun: } \beta \\ \text{cat: NP} \\ \text{fnc:} \end{array} \right] \quad \begin{array}{l} \text{cancel } \text{NP}' \text{ ss.cat} \\ \text{acopy } \beta \text{ ss.arg} \\ \text{ecopy } \alpha \text{ nw.fnc} \\ \text{copy}_{ss} \text{ copy}_{nw} \end{array}$$

$$\mathbf{ST}_F =_{def} \{ ([cat: v] \text{ rp}_{N+V}), ([cat: v] \text{ rp}_{V+N}) \}$$

The rule packages of LAH.Russian.1 call a total of 7 rules, compared to 4 in LAH.Korean.1 and 3 in LAH.German.1.

The rule N+N is the same as in LAH.Korean.1: it simply collects the nouns, relying on a later application of N+V to provide the bidirectional pointer for coding the required semantic relations. The rule V+N is the same as in LAH.German.1: it utilizes the adjacency of the verb and the noun to establish the relevant aspect of the functor-argument structure between the two. The rule N+V has three clauses as in Korean; also, the set parentheses in the verb pattern indicate that the order of the fillers may differ from the order of the valency positions in the verb. Unlike N+V for Korean, however, N+V for Russian has a non-empty rule package, containing the possible successor rule V+N; also, the second and third clause have the additional variable X in the cat attribute of the verb pattern, formally indicating that in Russian there may be a remainder of uncanceled valency positions.

This variable provides for an important difference between Korean and Russian: in Korean, the sentence-final application of N+V requires that all valencies are being canceled and all functor-argument structures are being completed at once, while in Russian N+V may also

apply prefinally, with the verb in second or third position and one or two subsequent applications of V+N.

8 Comparing the Four LAH Center Fragments

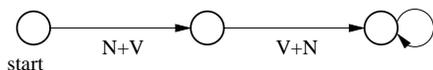
The four center fragments code the same content, but differ in whether nouns are case-marked (German, Korean, Russian) or not (English). They also differ in whether the order of nouns is fixed (English) or not (German, Korean, Russian). Finally, they differ in whether the position of the verb is fixed (English, German, Korean) or not (Russian):

8.1 SCHEMATIC DISTINCTION BETWEEN THE 4 CENTER FRAGMENTS

	case-marked	fixed noun order	fixed verb position
English	no	yes	yes
German	yes	no	yes
Korean	yes	no	yes
Russian	yes	no	no

In an LA-*hear* grammar, the distinction between nouns with and without case marking shows up in the automatic word form recognition and the variable definition. The distinction between fixed and variable noun order in English vs. German is treated minimally in terms of the absence vs. presence of the variable X in the *cat* pattern of the verb. Therefore, LAH.English.1 and LAH.German.1 have the same finite state transition network (FSN):

8.2 FSN UNDERLYING LAH.ENGLISH.1 AND LAH.GERMAN.1

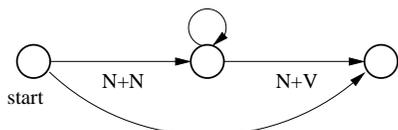


The network consists of states, represented by circles, and of rule applications, represented by arrows. Arrows going into a state correspond to the same rule called from different predecessor states, while arrows going out of a state (cf. 8.3, 8.4 below) correspond to the different rules of that state's rule package. With this in mind, it is sufficient for a complete characterization of the network to explicitly name only one of the arrows going into a state.

The rule N+V is different in English and German in that it assigns a nominative in English, whereas in German the case-marked noun cancels a corresponding valency position in the verb, and accordingly for the rule V+N. In sentences with a three-place verb, V+N applies to its own output, as indicated by the circle arrow. Given that an V+N application requires an uncanceled valency, the number of V+N repetitions is limited to one.

The rule systems of English and German on the one hand, and Korean on the other differ in that the former have two *simple cross-copying* rules (cf. 3.3), N+V and V+N, while Korean has a *suspension* rule N+N and a *multiple cross-copying* rule (cf. 3.4) N+V with three clauses. The finite state transition network underlying LAH.Korean.1 is as follows:

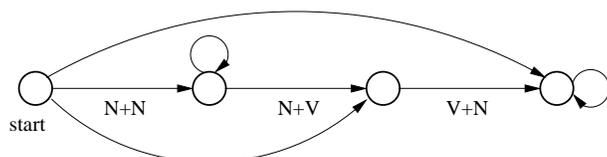
8.3 FSN UNDERLYING LAH.KOREAN.1



As indicated by the circle arrow, N+N may apply to its own output, but only once and only if this is subsequently licensed by a three-place verb. If the verb is one-place, N+V is called by the start state.

The rule system of Russian combines the simple cross-copying rule V+N of English and German with the suspension rule N+N and the multiple cross-copying rule N+V of Korean, with concomitant changes of the rule packages:

8.4 FSN UNDERLYING LAH.RUSSIAN.1



The respective rules N+V of LAH.Korean.1 and LAH.Russian.1 differ minimally in the absence vs. presence of the variable X in the *cat* pattern of the verb (compare 6.2 and 7.1), thus accounting for the fixed verb position in Korean and the free verb position in Russian.

We are now in a good position to establish the mathematical complexity of the four center fragments. This is important for the next steps, namely a systematic upscaling: If the center fragments can be shown to be of a low, i.e., linear, complexity, then each extension of the fragment should be designed in such a way that this degree of complexity is maintained.

The complexity of a grammar formalism – defined as the upper bound on the number of “primitive”²³ computational operations required in the analysis of arbitrary input – depends on the following two parameters:

8.5 PARAMETERS OF COMPLEXITY

- The *amount* of computation per rule application required in the worst case.
- The *number* of rule applications relative to the input length needed in the worst case.

These parameters are independent of each other and apply in principle to any rule-based grammar formalism (cf. FoCL9, Section 11.2).

In LA-grammar, the amount parameter depends on (i) the cost of matching the rule patterns with the input and (ii) the cost of applying the rule’s operations.²⁴ In basic propositions without modifiers, the maximal number of proplets is 4, namely one verb and at most three arguments, thus limiting the cost of matching with a small constant. Furthermore, the number of operations in each rule is finite. Therefore, as long as it holds for each available operation that the cost of its application is finite, the cost of applying the operations of any rule is finite.

Because each rule application in LA-grammar requires the consumption of a next word, the number parameter depends solely on the degree of ambiguity in a derivation. In the derivation of an unambiguous sentence, the maximal number of rule applications (including unsuccessful attempts) is $O \cdot (n-1)$, whereby O is a constant representing the number of rules in the largest rule package of the grammar and n is the length of the sentence. In other words, the number of rule applications in an unambiguous LA-grammar grows linearly with the length of the input. It has been proven that the number of rule application grows only linearly even in ambiguous LA-grammars as long the ambiguities are non-recursive (cf. FoCL’99, 11.3.7, Theorem 3).

²³This terminology follows Earley 1970.

²⁴These conditions differ from those of the earlier stage of LA-grammar, presented in NEWCAT’86 and analyzed in terms of complexity in TCS’92 and FoCL’99, Part II.

It follows that LAH.English.1, LAH.German.1, LAH.Korean.1, and LAH.Russian.1 are all of linear complexity. The cost of applying any of their rules is constant because the maximal number of proplets to be matched by the sentence start pattern is 3, the number of operations in each rule is finite, and the cost of applying them is finite as well (amount parameter). Furthermore, the LA-hear grammars in question are unambiguous because they don't have any rule package containing input-compatible²⁵ rules (number parameter).

9 Upscaling

After completing the center fragments of English, German, Korean, and Russian presented above,²⁶ they can handle a substantial number of grammatical constructions. Furthermore, the number of contents recognized and produced by a center fragment is infinite because there is no grammatical limit on the number of sentences in an extrapositional coordination (text).

Nevertheless, the coverage of a center fragment is small compared to the whole of a natural language. This is in part because a natural language has nouns, verbs, and adjectives at the elementary, phrasal, and clausal level, while a center fragment is deliberately limited to the elementary level. Furthermore, natural language has extra- as well as intrapositional coordination, while a center fragment is deliberately limited to extrapositional coordination.

One direction to increase the coverage of a center fragment is by adding to the automatic word form recognition and production. Given (i) a suitable software, (ii) a traditional dictionary of the natural language existing online in public domain, and (iii) a well-trained computational linguist, a highly detailed word form recognition and production covering more than 90% of the word form types (!) in a sizeable corpus can be done in less than half a person year. The other direction to increase coverage is by extending the LA-hear, LA-think, and LA-speak grammars to additional constructions of the language in question. This may be done in two systematic ways, namely (i) grammar-based and (ii) frequency-based.

A grammar-based extension begins with a center fragment, which means that functor-argument structure is restricted to *elementary* nouns, verbs, and adjectives, while coordination is restricted to the sentence level. The first step of systematic grammar-based upscaling consists in adding *phrasal* nouns, verbs, and adjectives, and their *intrapositional* coordination.²⁷ Then *clausal* nouns and adjectives are added (subordinate clauses):

9.1 EXAMPLES OF ELEMENTARY, PHRASAL, AND CLAUSAL NOUNS

Elementary: Julia saw *Fido*
Phrasal: Julia saw *a grumpy old dog*
Clausal: Julia saw *that Fido barked*

²⁵Cf. FoCL'99, 11.3.2.

²⁶The extrapositional coordination of basic propositions, illustrated in Example 2.2, requires two additional rules, called S+IP (sentence plus interpunctuation) and IP+START (interpunctuation plus sentence start), which are the same for all four fragments, and formally defined in NLC'06, Example 11.4.1.

Elementary modifiers, formally represented by proplets with the core attribute *adj*, for adjective, may be used adnominally or adverbially (cf. NLC'06, Section 6.3). While elementary propositions must have one verb and one, two, or three nouns (depending on the verb), adjectives are optional and may be stacked, with no grammatical limit on their number. Adding elementary adjectives in adnominal use requires two rules, DET+NN (determiner plus noun) and DET+ADN (determiner plus adnominal), which are formally defined in NLC'06, Example 13.2.4. A detailed hearer mode analysis of elementary and phrasal adjectives in adnominal and adverbial use may be found in NLC'06, Chapter 15.

²⁷These include gapping constructions. Even though gapping constructions may be found very rarely in a corpus, their semantic relations are intuitively quite clear to the native speakers and hearers of a wide range of different languages.

Also, the sentential mood of declarative is supplemented with other moods, such as interrogative and imperative, and their interpretation and production in dialogue. Such a grammar-based extension has been presented in NLC'06²⁸ for English.

A frequency-based extension begins with an almost complete grammar-based extension, including a largely complete automatic word form recognition of the language in question. The first step of a frequency-based extension is to apply the automatic word form recognition to a corpus. This allows to establish sequences of word form *categories* (rather than letter sequences representing surfaces), ordered according to frequency. Next, the grammar-based extension is tested automatically on this list of constructions (verification), starting with the most frequent ones. In this way, constructions not yet handled by the current system may be found and added to the existing DBS system.

Grammar-based extensions are important because they are based on semantic relations which are intuitively clear from the outset. Therefore, they may be implemented in the speaker and the hearer mode, and allow straightforward upscaling while maintaining the standard of functional completeness: an infinite set of constructions is composed recursively using only functor-argument structure and coordination, intrapropositionally and extrapropositionally. In addition, there is the secondary semantic relation of inference, which includes the handling of coreference.²⁹

Frequency-based extensions are necessary for a achieving realistic coverage. This requires not only an extension of the grammar system to the new structures found in some corpus, but also a systematic development of the corpus itself. Such a corpus development should be based on a synchronic reference corpus structured into domains with subsequent annual monitor corpora. In this way, the coverage of a rule-based agent-oriented language analysis may be maintained continuously, extending its full functionality into the future. The possibilities of using such a system of Database Semantics for practical applications are enormous, and include all aspects of man-machine communication and natural language processing.

10 Conclusion

The starting point for the definition of equivalent center fragments are the most basic *contents* of compositional semantics. These are traditionally intrapropositional functor-argument structure (i.e., one-place, two-place, and three-place propositions) and extrapropositional coordination (concatenation of propositions as in a text). The scientific challenge is

1. to represent basic contents as a data structure suitable for storage and retrieval,
2. to decode the contents from surfaces with different word orders in the hearer mode,
3. to encode the contents into surfaces with different word orders in the speaker mode,
4. and to do the encoding and decoding in a strictly time-linear derivation order.

In this paper, different word orders are exemplified by the natural languages English, German, Korean, and Russian. For reasons of space, the definition of their equivalent center fragments is limited to the hearer mode and to elementary functor-argument structures without modifiers. The four center fragments are presented as formal definitions of partial DBS-systems, each consisting of an automatic word form recognition, a variable definition, and an LA-hearer grammar. These definitions serve as the *declarative specifications* for corresponding implementations (here in Java).

²⁸NLC'06 contains over 100 detailed examples, analyzed in the hearer mode and the speaker mode.

²⁹Cf. NLC'06, Chapter 10.

The theoretical as well as practical purpose of center fragments for natural languages is to ensure the success of long-term upscaling. Proof of concept is provided by NLC'06, in which a center fragment of English is systematically upscaled by extending functor-argument structure from elementary nouns, verbs, and adjectives to their phrasal and clausal counterparts, and by extending coordination from the clausal level to its phrasal and elementary counterparts, including gapping constructions. Equivalent upscaled fragments have been defined and computationally verified for German (Mehlhoff 2007) and Chinese (Hua Mei 2007).

References

- Austin, J.L. (1962) *How to Do Things with Words*. Oxford, England: Clarendon Press
- Choe, Jae-Woong, and Hausser, R. (2007) "Treating quantifiers in Database Semantics," in M. Duži et al. (eds) *Information Modelling and Knowledge Bases XVIII*, Amsterdam: IOS Press
- Chomsky, N. (1965) *Aspects of the Theory of Syntax*, MIT Press, Cambridge, Mass.
- Clark, H. H. (1996) *Using Language*. Cambridge, UK: Cambridge Univ. Press
- Earley, J. (1970) "An efficient context-free parsing algorithm," *Commun. ACM* 13.2:94–102
- Greenberg, J. (1963) "Some universals of grammar with particular reference to the order of meaningful elements," in J. Greenberg (ed.) *Universals of Language*. Cambridge, MA: MIT Press
- Grice, P. (1965) "Utterer's meaning, sentence meaning, and word meaning," *Foundations of Language* 4:1–18
- Hausser, R. (1992) "Complexity in Left-Associative Grammar," *Theoretical Computer Science*, Vol. 106.2:283-308, Amsterdam: Elsevier (TCS'92)
- Hausser, R. (1996) "A database interpretation of natural language," *Korean Journal of Linguistics* 106(2), 29-55
- Hausser, R. (1999) *Foundations of Computational Linguistics, 2nd ed. 2001*, Berlin Heidelberg New York: Springer (FoCL'99)
- Hausser, R. (2001) "Database Semantics for natural language," *Artificial Intelligence*, Vol. 130.1:27–74, Amsterdam: Elsevier (AIJ'01)
- Hausser, R. (2006) *A Computational Model of Natural Language Communication*, Berlin Heidelberg New York: Springer (NLC'06)
- Kučera, H. and W.N. Francis (1967) *Computational analysis of present-day English*, Brown U. Press, Providence, Rhode Island
- Kycia, A. (2004) *A Java Implementating of the Speaker, Think, and Hearer Modes in Database Semantics*. MA-thesis, CLUE, Universität Erlangen–Nürnberg [in German]
- Mehlhoff, J. (2007) *An Implementation of a Center Fragment of German in Database Semantics*. MA-thesis, CLUE, Universität Erlangen–Nürnberg [in German]
- Mei, Hua (2007) *An Implementation of a Center Fragment of Chinese in Database Semantics*. MA-thesis, CLUE, Universität Erlangen–Nürnberg [in German]
- Montague, R. (1974) *Formal Philosophy*, Yale U. Press, New Haven
- Roy, D. (2003) "Grounded spoken language acquisition: experiments in word learning," *IEEE Transactions on Multimedia* 5.2:197–209
- Searle, J.R. (1969) *Speech Acts*. Cambridge, England: Cambridge Univ. Press