11. DBS.1: Hearer Mode

The declarative specification (cf. Sect. 1.2) of a system of natural language communication must be defined at a level of detail sufficient for a straightforward implementation in a programming language of choice. Given the large structural variety of natural language expressions, the size of the lexicon, and the complexity of natural language communication, let us begin with a small "fragment."

By a fragment we mean a system of natural language communication which has limited coverage, but is functionally complete in that it models the hearer mode, the think mode (defined as navigating through the content of the database), and the speaker mode. Given the limitation of having to use standard computers rather than robots, the following fragments do not include contextual recognition and action (i.e., they are capable of mediated reference only, cf. 2.5.1). Therefore, the Word Bank may be simplified to store language proplets without a separate context section (cf. 3.3.1).

The first fragment of English defined in Database Semantics is called DBS.1. For simplicity, the coverage of DBS.1 is restricted to the sentence sequence

Julia sleeps. John sings. Susanne dreams.

The focus of DBS.1 is on concatenating propositions, as in a text (extrapropositional coordination, cf. Sect. 9.2). DBS.1 consists of components for automatic word form recognition and production, and three LA-grammars, called LA-hear.1, LA-think.1, and LA-speak.1. In Chaps. 13 and 14, DBS.1 will be extended into DBS.2 without losing any of the previously achieved functionality or coverage.

11.1 Automatic Word Form Recognition

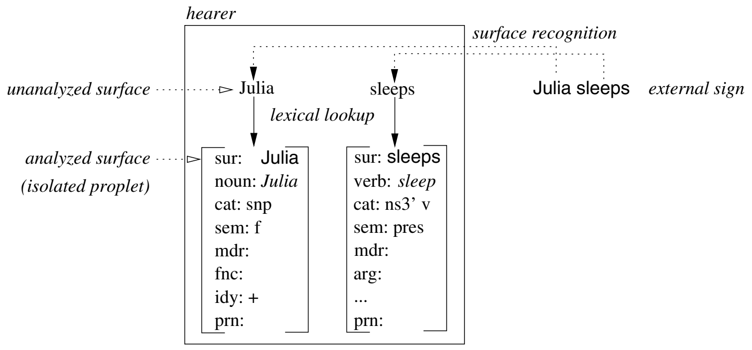
For a cognitive agent with language, the cycle of communication may begin in the speaker mode or in the hearer mode. However, before an agent can produce language, its Word Bank must be filled with content. In order to read content automatically into the system, let us begin the definition of the fragment with the hearer mode.

An LA-hear system consists of two components: (i) a system of word form recognition and (ii) an LA-hear grammar for syntactic–semantic interpretation. Word form recognition takes unanalyzed natural language surfaces as input and produces corresponding lexically analyzed types as output.[1]

[1] For a detailed description of automatic word form recognition see FoCL'99, Chaps. 13–15.

The basic functioning of automatic word form recognition may be characterized schematically as a combination of surface recognition and lexical lookup:

11.1.1 Surface recognition and lexical lookup



In *surface recognition*, the external sign surfaces (token) are matched with corresponding internal sign surfaces (types) provided by the agent's Word Bank (cf. Sects. 4.3–4.5). This process may be achieved by speech recognition, optical character recognition (OCR), or by simply typing the letters of the external surface into the computer. In the latter procedure, the external surface is converted directly into a digitally coded, modality-independent form – as opposed to speech recognition or OCR, which are modality-dependent (cf. Sect. 2.2).

The internal representation of the unanalyzed surface is submitted to *lexical lookup*. Lexical lookup is based on an electronic lexicon defined as a list of lexical entries. Assuming that each entry is defined as an isolated proplet, lexical lookup compares the unanalyzed surface with the value of the sur attribute of each entry. The proplet with a sur value matching the unanalyzed surface is returned as the result.

Depending on whether lexical lookup is based on full word forms or on morphological analysis, there are different kinds of automatic word form recognition (cf. FoCL'99, Sect. 13.5). The full form method is the simplest, but it can only recognize a finite number of items. An infinite number of word forms can be recognized by systems based on morphologically analyzing the surface into smaller parts called allomorphs, e.g., learn, ing, or ed. Such a compositional approach is especially important for languages with a morphology richer than English.

No matter which method of automatic word form recognition is used, it should work for any language. Changing to another language should require no more than providing another language-specific lexicon and, if present, other rules of morphological analysis. Also, it must be possible at any time to replace the current component of automatic word form recognition by a more advanced one (modularity). For this, the new component must merely be input/output equivalent with the old one: It should take unanalyzed surface tokens as input and render lexically analyzed surface types (e.g., isolated proplets) as output, which are passed to the parser.

11.2 Lexicon of LA-hear.1

In order for an LA-hear grammar to parse a sentence or text, it must be able to lexically recognize all the word forms occurring in it. Let us therefore define the word forms needed for the small fragment of DBS.1. For simplicity, we use full-form lookup.

To facilitate storage and retrieval, the entries in a lexicon are usually ordered alphabetically, using the word form surface as the key. However, to show the structure of the different kinds of words, the following entries are ordered according to the part of speech, i.e., first all the nouns, then all the verbs, and finally the punctuation sign.

11.2.1 The lexical entries of **LA-hear.1**

proper names:

sur: Julia	sur: John	sur: Susanne
noun: *Julia*	noun: *John*	noun: *Susanne*
cat: nm	cat: nm	cat: nm
sem: f	sem: m	sem: f
mdr:	mdr:	mdr:
fnc:	fnc:	fnc:
idy:	idy:	idy:
prn:	prn:	prn:

finite one-place verbs, inflected for third-person singular

sur: sleeps	sur: sings	sur: dreams
erb: *sleep*	verb: *sing*	verb: *dream*
cat: ns3′ v	cat: ns3′ v	cat: ns3′ v
sem: pres	sem: pres	sem: pres
mdr:	mdr:	mdr:
arg:	arg:	arg:
nc:	nc:	nc:
pc:	pc:	pc:
prn:	prn:	prn:

punctuation sign

sur: .
cat: v′ decl

The three kinds of lexical entries have in common that their first attribute, called sur, takes the surface of the word form as its value (cf. 4.1.2, 1). It is printed here in sans serif font. In the hearer mode, it serves as the key to be matched with the unanalyzed input surface (cf. 11.1.1).

The nouns and verbs are examples of content words, while the punctuation sign is an example of a function word. In content words, the second attribute is the core attribute (cf. 4.1.2, 2). It specifies the basic part of speech, i.e., noun, verb, or adj, and may take a concept, a name marker, or an index as its value. For simplicity, concepts are represented by a corresponding English word serving as a placeholder (cf. 4.1.4 ff.) and are printed in italics.

The next two attributes, cat and sem, are called grammatical attributes (cf. 4.1.2, 3). cat specifies the syntactic category of the word form. It retains the notation of category segments familiar from the older syntactic systems of LA-grammar (NEWCAT'84 and CoL'89). In the simple LA-hear.1 grammar defined in 11.4.1, the following *category segments* are used:

11.2.2 Values of the cat attribute

decl	= declarative sentence
nm	= proper name
ns3′	= nominative third-person singular valency position
v	= verb, unmarked for sentence mood
v′	= valency position for verb unmarked for sentence mood

Being constants, the category segments are written in lowercase Roman letters.

The sem attribute specifies semantic properties of the word form. In LA-hear.1, these are the gender values in noun proplets and a tense value in verb proplets:

11.2.3 Values of the sem attribute

f	= femininum
m	= masculinum
pres	= present tense

The remaining attributes of the isolated proplets in the lexicon 11.2.1 do not have values yet, but will receive values during syntactic–semantic parsing. The attributes in question are of three kinds. The first specifies *intrapropositional relations* between content words in terms of traditional valency relations (cf. 4.1.2, 4):

11.2.4 Intrapropositional continuation attributes

arg:	specifies the arguments of verbs (list)
fnc:	specifies the functor of nouns (atom)
mdr:	specifies the modifiers of nouns and verbs (list)
mdd:	specifies the modified of adnominals and adverbials (atom)

The value restrictions specified above will have to be revised for handling the constructions presented in Part II. For example, the attribute fnc must be able to take nonatomic values, e.g., [fnc: 28 amuse], as in 7.1.2, 1.

The second specifies *extrapropositional relations* (cf. 4.1.2, 5). These are the identity relation between nouns and the coordination relation between verbs.

11.2.5 Extrapropositional continuation attributes

idy:	identity between nouns. Takes a number as value.
nc:	next conjunct. Takes an optional conjunction, a proposition number, and a verb concept as value.
pc:	previous conjunct. Takes values similar to nc.

The attribute idy is restricted to nouns and prepositional phrases (complex adjectives), while the attributes nc and pc are used by DBS-1 only in verbs.

The third kind of attribute with a nonlexical value is a bookkeeping attribute (cf. 4.1.2, 6):

11.2.6 Bookkeeping attribute of **LA-hear.1**

prn:	proposition number

This attribute is common to all proplets representing content words. Analyzed content words belonging to the same proposition have a common prn value.

Finally, consider the feature structure of the function word in 11.2.1. The punctuation sign has only two attributes, one for the surface and one for the syntactic category. This is sufficient because the punctuation sign is absorbed into the proplet of the verb (e.g., 11.5.2).

11.3 Preamble of LA-hear.1

The rules of LA-hear.1 are based on patterns defined as flat (nonrecursive) feature structures. The patterns of the rule level differ from their counterparts at the language level by using *variables* rather than constants as the values of some of their attributes. These variables are defined in the preamble of an LA-hear grammar, including the restrictions on their domain and their agreement conditions. In this way, different rules may utilize the same variable definitions provided by the preamble.

The rules of LA-hear.1 use the following binding variables:

11.3.1 List of binding variables

SM	= sentence mood
VT	= verb type filler
VT′	= verb type valency position
NP	= noun phrase filler
NP′	= noun phrase valency position
α, β, γ, etc.,	= for individual concepts like *Julia, sleep, young,* etc.,
i, j, k	= for number values of the prn and idy attributes

As explained in Sect. 3.2, the matching of a proplet pattern at the rule level with a proplet at the language level requires (i) that all attributes in the rule pattern have counterparts in the language proplet and (ii) that the attribute values of the language proplet satisfy the restrictions of the corresponding variables of corresponding attributes in the rule patterns. The variables of LA-hear.1 have the following restrictions:

11.3.2 Restriction of binding variables

SM ϵ {decl}
VT ϵ {v}
VT′ ϵ {v′}
NP ϵ {nm}
NP′ ϵ {ns3′}

The set of values for these variables will be extended in later versions of the grammar.

The role of variable restrictions may be shown schematically as follows:

11.3.3 Matching between a variable and a constant

$$\begin{array}{rll} \textit{rule level:} & \begin{bmatrix}\text{cat: NP}\end{bmatrix} & \\ & & \textit{matching?} \\ \textit{language level:} & \begin{bmatrix}\text{cat: nm}\end{bmatrix} & \end{array}$$

The match is successful because the constant value nm of the attribute cat at the language level is defined in 11.3.2 to be in the restriction set of the variable value NP of the corresponding attribute at the rule level.

The definition of restricted variables may be extended to a simple, general treatment of *agreement*.[2] This is done by defining conditionals of the following form:

If variable X has the values a, b, or c, then the variable Y must have the values p or q.

Such a conditional relates two variables X and Y at the rule level by further refining their restrictions.

The agreement conditions of LA-hear.1 are very simple and relate valency fillers to suitable valency positions.

11.3.4 Agreement conditions

if $\text{VT} \in \{\text{v}\}$, then $\text{VT}' \in \{\text{v}'\}$
if $\text{NP} \in \{\text{nm}\}$, then $\text{NP}' \in \{\text{ns3}'\}$

For a more detailed analysis of agreement regarding the restrictions between nominal valency fillers and the verb, between finite auxiliaries and the nonfinite main verb, and between determiners and nouns in English see the preamble of LA-hear.2 defined in Sect. 13.2.

11.4 Definition of LA-hear.1

For parsing simple texts like

Julia sleeps. John dreams. Susanne sings.

LA-hear.1 requires three rules, called NOM+FV (nominative plus finite verb), S+IP (sentence plus interpunctuation), and IP+START (interpunctuation plus start of the new sentence). The states and rules of this input grammar are based on the definition

[2] For a complete treatment of agreement in English see FoCL'99, Sect. 16.3, pp. 307–310, and Chap. 17, pp. 331–332.

of attributes and values in Sect. 11.2 and on the definition of variables and agreement in Sect. 11.3.

NOM+FV adds a finite verb to a sentence start consisting of a noun, S+IP adds a full stop to a sentence start ending in a finite verb, and IP+START adds a noun to a sentence start ending in a full stop. Then the cycle starts over again: Treating the noun as the new sentence start, NOM+FV adds a finite verb, etc.

In addition to these three rules, LA-hear.1 defines a start state ST_S which ensures that a derivation begins with the rule NOM+FV, and a final state ST_F which ensures that a derivation ends with an application of the rule S+IP.

11.4.1 Formal definition of **LA-hear.1**

$$\mathbf{ST}_S =_{def} \{\ (\ [\text{cat: X}]\ \{1\ \text{NOM+FV}\})\ \}$$

NOM+FV {2 S+IP}

$$\begin{bmatrix}\text{noun: }\alpha\\ \text{cat: NP}\\ \text{fnc:}\\ \text{prn: k}\end{bmatrix}\ \begin{bmatrix}\text{verb: }\beta\\ \text{cat: NP}'\ \text{X VT}\\ \text{arg:}\\ \text{nc:}\\ \text{pc:}\end{bmatrix}\qquad \begin{array}{l}\text{delete NP}'\ \text{nw.cat}\\ \text{acopy }\alpha\text{ nw.arg}\\ \text{ecopy }\beta\text{ ss.fnc}\\ \text{ecopy PC nw.pc}\\ \text{acopy PCV nw.pc}\\ \text{set }\beta\text{ NCV}\\ \text{copy}_{ss}\ \text{copy}_{nw}\end{array}$$

S+IP {3 IP+START}

$$\begin{bmatrix}\text{verb: }\alpha\\ \text{cat: VT}\\ \text{prn: k}\end{bmatrix}\ \begin{bmatrix}\text{cat: VT}'\ \text{SM}\end{bmatrix}\qquad \begin{array}{l}\text{replace SM VT}\\ \text{set k PC}\\ \text{set }\alpha\text{ PCV}\\ \text{copy}_{ss}\end{array}$$

IP+START {1 NOM+FV}

$$\begin{bmatrix}\text{verb: }\alpha\\ \text{cat: SM}\\ \text{nc:}\end{bmatrix}\ \begin{bmatrix}\text{noun: }\beta\\ \text{cat: NP}\\ \text{prn: k}\end{bmatrix}\qquad \begin{array}{l}\text{increment nw.prn}\\ \text{ecopy k ss.nc}\\ \text{acopy `NCV' ss.nc}\\ \text{copy}_{ss}\ \text{copy}_{nw}\end{array}$$

$$\mathbf{ST}_F =_{def} \{(\ [\text{cat: decl}]\ \text{rp}_{\text{S+IP}})\}$$

The rules in the rule packages of the grammar are numbered in sequence.[3] These numbers are useful in debugging because a rule may be called from different rule packages.[4]

Each rule consists of a rule name, e.g., NOM+FV, a rule package, e.g., {2 S+IP}, a pattern for the sentence start, a pattern for the next word, and a list of operations, e.g., delete NP' nw.cat.

The operations of the rules have the following interpretation (see also Appendix B.5.2–B.5.5):

[3] During grammar development, the contents of the rule packages may frequently change. Therefore the grammar-specific numbering of the rules should be provided automatically by the system. Otherwise, the grammar writer has to adjust the numbering by hand after each revision.

[4] This is not illustrated by LA-hear.1, due to the fact that each of its rule packages has only one rule and each rule is called from only one rule package. See the definition of LA-hear.2 in 13.2.4 instead.

11.4.2 Definitions of the operations

delete variable proplet-attr. =	delete the value corresponding to the variable in the corresponding attribute of the corresponding proplet at the level of language.
Example:	delete NP$'$ nw.cat (in NOM+FV)
acopy variable proplet-attr. =	additively copy value(s) corresponding to the variable into the specified target slot; default position is at the end of the target slot.
Examples:	acopy α nw.arg (in NOM+FV)
	acopy PCV nw.pc (in NOM+FV)
	acopy 'NCV' ss.nc (in IP+START)
ecopy variable proplet-attr. =	exclusively copy value(s) corresponding to the variable into the specified proplet attribute.
Examples:	ecopy β ss.fnc (in NOM+FV)
	ecopy PC nw.pc (in NOM+FV)
	ecopy k ss.nc (in IP+START)
set value variable =	rule-based (rather than matching-based) binding of a variable to a value.
Examples:	set β NCV (in NOM+FV)
	set k PC (in S+IP)
	set α PCV (in S+IP)
replace variable2 variable1 =	replace value of variable1 with value of variable2.
Examples:	replace SM VT (in S+IP)
increment proplet-attr. =	increment numerical value of corresponding attribute by one.
Example:	increment nw.prn (in IP+START)
copy$_{ss}$ =	include the proplets of the sentence start in the result.
Examples:	NOM+FV, S+IP, IP+START
copy$_{nw}$ =	include the proplet of the next word in the result.
Examples:	NOM+FV, IP+START

To add a value "additively" with acopy means that the target attribute may already have non-NIL values. To add a value "exclusively" with ecopy means that the target attribute may not have any non-NIL values. Instead of referring to a value by means of a variable, e.g., acopy α nw.arg, the operations may also refer to a value by specifying a proplet attribute, e.g., acopy ss.noun nw.arg.

The set operation allows binding a value to a variable by a rule (rather than binding a value to a variable by matching rule patterns with the language level). The variables NCV (next conjunct verb), PC (previous conjunct prn), and PCV (previous conjunct verb) are a special kind of global variable. They are called *loading variables* because they are assigned values at certain stages of the derivation by certain rules, such that the values can be (i) used or (ii) supplied at other stages. Loading variables occur at the level of language and the level of rules, in contrast to the binding variables, which occur only at the level of rules.

The operations apply in the order of their appearance in the rule. Their functioning will be explained in connection with the detailed sample derivation presented in the following section.

11.5 Interpreting a Sequence of Sentences

An LA-hear derivation is activated by the first word form passed to the grammar by automatic word form recognition. This analyzed word form (lexical proplet) is matched with the pattern(s) of the start state ST_S. If the match is successful, the associated rule package of the start state ST_S is activated, the second word is provided by automatic word form recognition, and the rules of the rule package are applied to the first and second proplet serving as *ss* and *nw* (cf. 3.4.3, Appendix B.1).

Given the definition of LA-hear.1 (cf. 11.4.1) and the first word form of our sample text, this operation is successful: The pattern [cat: X] of the start state trivially matches the lexical type of *Julia*, activating lexical lookup of the second word form *sleep* and the rule package {1 NOM+FV}. The single rule in this rule package is applied to the lexical proplets of the first two words:

11.5.1 COMBINING Julia AND sleeps

NOM+FV {2 S+IP}

$$
\begin{bmatrix}\text{noun: } \alpha\\ \text{cat: NP}\\ \text{fnc:}\\ \text{prn: k}\end{bmatrix}
\quad
\begin{bmatrix}\text{verb: } \beta\\ \text{cat: NP}'\text{ X VT}\\ \text{arg:}\\ \text{nc:}\\ \text{pc:}\end{bmatrix}
\qquad
\begin{array}{l}\text{delete NP}'\text{ nw.cat}\\ \text{acopy } \alpha \text{ nw.arg}\\ \text{ecopy } \beta \text{ ss.fnc}\\ \text{ecopy PC nw.pc}\\ \text{acopy PCV nw.pc}\\ \text{set } \beta \text{ NCV}\\ \text{copy}_{ss}\ \text{copy}_{nw}\end{array}
$$

$$
\begin{bmatrix}\text{sur: 1.1 Julia}\\ \text{noun: } \textit{Julia}\\ \text{cat: nm}\\ \text{sem: f}\\ \text{mdr:}\\ \text{fnc:}\\ \text{idy: 1}\\ \text{prn: 1}\end{bmatrix}_1
\begin{bmatrix}\text{sur: 1.2 sleeps}\\ \text{verb: } \textit{sleep}\\ \text{cat: ns3}'\text{ v}\\ \text{sem: pres}\\ \text{mdr:}\\ \text{arg:}\\ \text{nc:}\\ \text{pc:}\\ \text{prn:}\end{bmatrix}_2
\Longrightarrow
\begin{bmatrix}\text{sur:}\\ \text{noun: } \textit{Julia}\\ \text{cat: nm}\\ \text{sem: f}\\ \text{mdr:}\\ \text{fnc: sleep}\\ \text{idy: 1}\\ \text{prn: 1}\end{bmatrix}_1
\begin{bmatrix}\text{sur:}\\ \text{verb: } \textit{sleep}\\ \text{cat: v}\\ \text{sem: pres}\\ \text{mdr:}\\ \text{arg: Julia}\\ \text{nc:}\\ \text{pc:}\\ \text{prn: 1}\end{bmatrix}_2
$$

The proplets at the language level have a *word number* as subscript. It refers to the word which the proplet represents, indicating the word's position in the surface.

The precondition for applying the operations is the successful matching between the rule patterns and the language proplets in accordance with the matching conditions 3.2.3. The attribute condition is fulfilled insofar as the attributes of the patterns are a subset of the attributes of the corresponding language proplets. The value condition includes the requirement that the agreement condition between the values of the variables is fullfilled. Given that cat values satisfy the restiction of the variable NP to nm and and the variable NP' to ns3′ (cf. 11.3.4), the operations may apply.

The effect of the operations is shown in the proplets to the right of the arrow. The first operation, delete NP′ nw.cat, deletes the category segment ns3′ in the cat attribute of the verb. The second operation, acopy α nw.arg, copies the value Julia of the noun attribute into the arg attribute of the proplet *sleep*. The third operation, ecopy β ss.fnc, copies the value sleep of the verb attribute into the fnc attribute of the proplet *Julia*. Linguistically, the first operation models the canceling of the nominative valency position, while the second and third operations realize the functor–argument structure holding between the first and the second proplet.

The following three operations ecopy PC nw.pc, acopy PCV nw.pc, and set β NCV are for establishing the extrapropositional relations between the present and the previous proposition. However, given that the current sentence is initial, the three operations apply vacuously in this particular case.[5] A nonvacuous application of the three operations in question is shown in 11.5.4.

The last two operations copy$_{ss}$ and copy$_{nw}$ specify that both input proplets are in the output of the rule. The proplets in the output differ from the input proplet in that the latter have been subjected to the rule operations.

The application of NOM+FV activates its rule package, which contains only one rule: S+IP. The input to S+IP consists of the output of NOM+FV and the new next word "." (punctuation, full stop). The application of S+IP in the derivation of Julia sleeps + . is illustrated below:

11.5.2 Combining Julia sleeps and .

S+IP {3 IP+START}

$$\begin{bmatrix}\text{verb: }\alpha\\ \text{cat: VT}\\ \text{prn: k}\end{bmatrix} \quad \begin{bmatrix}\text{cat: VT}'\ \text{SM}\end{bmatrix} \qquad \begin{array}{l}\text{replace SM VT}\\ \text{set k PC}\\ \text{set }\alpha\text{ PCV}\\ \text{copy}_{ss}\end{array}$$

$$\begin{bmatrix}\text{sur:}\\ \text{noun: }\textit{Julia}\\ \text{cat: nm}\\ \text{sem: f}\\ \text{mdr:}\\ \text{fnc: sleep}\\ \text{idy: 1}\\ \text{prn: 1}\end{bmatrix}_1 \begin{bmatrix}\text{sur:}\\ \text{verb: }\textit{sleep}\\ \text{cat: v}\\ \text{sem: pres}\\ \text{mdr:}\\ \text{arg: Julia}\\ \text{nc:}\\ \text{pc:}\\ \text{prn: 1}\end{bmatrix}_2 \quad \begin{bmatrix}\text{sur: 1.3 .}\\ \text{cat: v}'\ \text{decl}\end{bmatrix}_3 \Longrightarrow \begin{bmatrix}\text{sur:}\\ \text{noun: }\textit{Julia}\\ \text{cat: nm}\\ \text{sem: f}\\ \text{mdr:}\\ \text{fnc: sleep}\\ \text{idy: 1}\\ \text{prn: 1}\end{bmatrix}_1 \begin{bmatrix}\text{sur:}\\ \text{verb: }\textit{sleep}\\ \text{cat: decl}\\ \text{sem: pres}\\ \text{mdr:}\\ \text{arg: Julia}\\ \text{nc:}\\ \text{pc:}\\ \text{prn: 1}\end{bmatrix}_2$$

The first operation replaces the cat value v of the verb with decl. The second and the third operations introduce the loading variables PC and PCV and assign values to them. These values, namely PC = 1 and PCV = sleep, are for the pc slot of the verb of the next proposition. Because the proplets of the next proposition are not yet available (and might not come at all), the patterns of the current rule S+IP have no place for the variables PC and PCV. This is why PC and PCV cannot be assigned values via pattern matching with the language level, but by means of the rule operations in question.

[5] If there were a previous proposition, the PC and PCV variables would have been bound to values, which would now be copied into the pc slot of the current proposition (forward loading). Also, the NCV variable would have been copied as a name into the nc slot of the previous proposition; it would now be provided with a value (backward loading) .

The last operation copies only the sentence start into the result. This is because punctuation is treated as a function word which is absorbed into the verb.

The application of S+IP activates its rule package, which contains only one rule: IP+START. The input to IP+START consists of the output of S+IP and the new next word, i.e., *John*. The application of IP+START in the derivation of Julia sleeps. + John is illustrated below:

11.5.3 COMBINING Julia sleeps. AND John

3 IP+START {1 NOM+FV }

$$\begin{bmatrix}\text{verb: }\alpha\\ \text{cat: SM}\\ \text{nc: }\end{bmatrix} \quad \begin{bmatrix}\text{noun: }\beta\\ \text{cat: NP}\\ \text{prn: k}\end{bmatrix} \quad \begin{array}{l}\text{increment nw.prn}\\ \text{ecopy k ss.nc}\\ \text{acopy 'NCV' ss.nc}\\ \text{copy}_{ss}\ \text{copy}_{nw}\end{array}$$

$$\begin{bmatrix}\text{sur: }\\ \text{verb: }\textit{sleep}\\ \text{cat: decl}\\ \text{sem: pres}\\ \text{mdr: }\\ \text{arg: Julia}\\ \text{nc: }\\ \text{pc: }\\ \text{prn: 1}\end{bmatrix}_2 \begin{bmatrix}\text{sur: 2.1 John}\\ \text{noun: }\textit{John}\\ \text{cat: nm}\\ \text{sem: m}\\ \text{mdr: }\\ \text{fnc: }\\ \text{idy: }\\ \text{prn: }\end{bmatrix}_4 \Longrightarrow \begin{bmatrix}\text{sur: }\\ \text{verb: }\textit{sleep}\\ \text{cat: decl}\\ \text{sem: pres}\\ \text{mdr: }\\ \text{arg: Julia}\\ \text{nc: 2 NCV}\\ \text{pc: }\\ \text{prn: 1}\end{bmatrix}_2 \begin{bmatrix}\text{sur: }\\ \text{noun: }\textit{John}\\ \text{cat: nm}\\ \text{sem: m}\\ \text{mdr: }\\ \text{fnc: }\\ \text{idy: 2}\\ \text{prn: 2}\end{bmatrix}_4$$

The operation increment nw.prn in IP+START increments the proposition number of the *nw*, which happens to be the first word of the next proposition. The second operation copies the incremented prn value of the *nw* into the nc slot of the *ss*, i.e., the verb proplet *sleep*. The third operation adds the variable name NCV to the nc slot of the *ss*. NCV will be provided with a value in the following proposition (backward loading) by the rule NOM+FV, which applies next.

11.5.4 COMBINING Julia sleeps. John AND sings

1 NOM+FV {2 S+IP}

$$\begin{bmatrix}\text{noun: }\alpha\\ \text{cat: NP}\\ \text{fnc: }\\ \text{prn: k}\end{bmatrix} \quad \begin{bmatrix}\text{verb: }\beta\\ \text{cat: NP}'\text{ X VT}\\ \text{arg: }\\ \text{nc: }\\ \text{pc: }\end{bmatrix} \quad \begin{array}{l}\text{delete NP}'\text{ nw.cat}\\ \text{acopy }\alpha\text{ nw.arg}\\ \text{ecopy }\beta\text{ ss.fnc}\\ \text{ecopy PC nw.pc}\\ \text{acopy PCV nw.pc}\\ \text{set }\beta\text{ NCV}\\ \text{copy}_{ss}\ \text{copy}_{nw}\end{array}$$

$$\begin{bmatrix}\text{sur: }\\ \text{noun: }\textit{Julia}\\ \text{cat: nm}\\ \text{sem: f}\\ \text{mdr: }\\ \text{fnc: sleep}\\ \text{idy: 1}\\ \text{prn: 1}\end{bmatrix}_1 \begin{bmatrix}\text{sur: }\\ \text{verb: }\textit{sleep}\\ \text{cat: decl}\\ \text{sem: pres}\\ \text{mdr: }\\ \text{arg: Julia}\\ \text{nc: 2 NCV}\\ \text{pc: }\\ \text{prn: 1}\end{bmatrix}_2 \begin{bmatrix}\text{sur: }\\ \text{noun: }\textit{John}\\ \text{cat: nm}\\ \text{sem: m}\\ \text{mdr: }\\ \text{fnc: }\\ \text{idy: 2}\\ \text{prn: 2}\end{bmatrix}_4 \begin{bmatrix}\text{sur: 2.2 sings}\\ \text{verb: }\textit{sing}\\ \text{cat: ns3}'\text{ v}\\ \text{sem: pres}\\ \text{mdr: }\\ \text{arg: }\\ \text{nc: }\\ \text{pc: }\\ \text{prn: }\end{bmatrix}_5 \Longrightarrow \begin{bmatrix}\text{sur: }\\ \text{noun: }\textit{Julia}\\ \text{cat: nm}\\ \text{sem: f}\\ \text{mdr: }\\ \text{fnc: sleep}\\ \text{idy: 1}\\ \text{prn: 1}\end{bmatrix}_1 \begin{bmatrix}\text{sur: }\\ \text{verb: }\textit{sleep}\\ \text{cat: decl}\\ \text{sem: pres}\\ \text{mdr: }\\ \text{arg: Julia}\\ \text{nc: 2 sing}\\ \text{pc: }\\ \text{prn: 1}\end{bmatrix}_2$$

$$\begin{bmatrix}\text{sur:}\\ \text{noun: }\textit{John}\\ \text{cat: nm}\\ \text{sem: m}\\ \text{mdr:}\\ \text{fnc: sing}\\ \text{idy: 2}\\ \text{prn: 2}\end{bmatrix}_4 \begin{bmatrix}\text{sur:}\\ \text{verb: }\textit{sing}\\ \text{cat: v}\\ \text{sem: pres}\\ \text{mdr:}\\ \text{arg: John}\\ \text{nc:}\\ \text{pc: 1 sleep}\\ \text{prn: 2}\end{bmatrix}_5$$

The proplets of the previous sentence are shown in addition to the current sentence start in order to illustrate the assignment of the nc (next conjunct) and the pc (previous conjunct) values. Reference to the previous proposition is needed because the coordination of two propositions is coded in their respective verbs, and – due to the word order of English – the verb of the current proposition is only now being provided.

Although the ss-pattern fits two input proplets, *Julia* and *John*, only the proposition number of the *John* proplet is in concord with the counter value of the current sentence start. Furthermore, even though the nw-pattern roughly[6] fits the input proplets *sleep* and *sing*, only *sing* is a possible candidate because the *nw* proplet must be provided by lexical lookup.

As in 11.5.2, the first three operations code the functor–argument relations between the two proplets of proposition 2. The fourth and the fifth operation copy the PC and PCV values set in 11.5.2 by S+IP into the pc slot of the current verb proplet (resulting in pc: 1 sleep in the *sing* proplet). The sixth operation assigns a value to the NCV variable of the previous verb proplet (backward loading, resulting in nc: 2 sing in the *sleep* proplet). Now the first two propositions of the test sample are properly concatenated via the nc and pc values of their verb proplets *sleep* and *sing*.

Given that the test sentences all have the same structure, the remainder of the derivation is analogous to the steps shown above, and is therefore omitted. The sequence of word forms in the test sample is added by the following rule applications:

11.5.5 Sequence of rule applications in deriving the sample of DBS-1

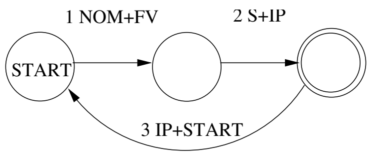
	1.1 Julia
1 NOM+FV	1.2 sleeps
2 S+IP	1.3 .
3 IP+START	2.1 John
1 NOM+FV	2.2 sings
2 S+IP	2.3 .
3 IP+START	3.1 Susanne
1 NOM+FV	3.2 dreams
2 S+IP	3.3 .

The first line contains the first word provided by the start state ST_S. Each subsequent line consists of a rule number, e.g., 1, a rule name, e.g., NOM+FV, and a next word. The latter consists of a proposition.word number, e.g., 1.2, and a surface, e.g., sleeps.

[6] In fact, the nw-pattern would not fit the proplet *sleep* because of conflicting cat values (see variable restrictions 11.3.2).

The format illustrated in 11.5.5 is suited well to show the transitions through the finite-state backbone of an LA-grammar during a derivation. The rules and rule packages of LA-hear.1 form the following finite-state transition network:

11.5.6 FINITE-STATE TRANSITION NETWORK OF **LA-hear.1**



The perplexity of this system is 3 : 3 = 1, that is, only one attempted rule application per composition. The notion of perplexity as applied to an LA-grammar is explained in FoCL'99, p. 339.

With no upper limit on the length, **LA-hear.1** analyzes infinitely many sentence sequences.[7] The simplest way to expand the data coverage of LA-hear.1 is by adding more proper names and more one-place verbs to its lexicon.

11.6 Storing the Output of LA-hear.1 in a Word Bank

The hearer's next step of interpreting a text of natural language consists in sorting the proplets derived into the Word Bank.

11.6.1 SORTING AN **LA-hear** DERIVATION INTO A WORD BANK

owner records (isolated proplets) *member records* (connected proplets)

$$\begin{bmatrix}\text{sur:}\\ \text{verb: }\textit{dream}\\ \text{cat: n-s3' v}\\ \text{sem: pres}\\ \text{mdr:}\\ \text{arg:}\\ \text{nc:}\\ \text{pc:}\\ \text{prn:}\end{bmatrix}\quad\begin{bmatrix}\text{sur:}\\ \text{verb: }\textit{dream}\\ \text{cat: decl}\\ \text{sem: pres}\\ \text{mdr:}\\ \text{arg: Susanne}\\ \text{nc:}\\ \text{pc: 2 sing}\\ \text{prn: 3}\end{bmatrix}$$

$$\begin{bmatrix}\text{sur:}\\ \text{noun: }\textit{John}\\ \text{cat: nm}\\ \text{sem: m}\\ \text{mdr:}\\ \text{fnc:}\\ \text{idy:}\\ \text{prn:}\end{bmatrix}\quad\begin{bmatrix}\text{sur:}\\ \text{noun: }\textit{John}\\ \text{cat: nm}\\ \text{sem: m}\\ \text{mdr:}\\ \text{fnc: sing}\\ \text{idy: 2}\\ \text{prn: 2}\end{bmatrix}$$

[7] This is illustrated by repetitive sequences like Julia sleeps.
Julia sleeps. Julia sleeps.
Julia sleeps. Julia sleeps. Julia sleeps. Etc.

owner records (isolated proplets)	*member records* (connected proplets)
sur:	sur:
noun: *Julia*	noun: *Julia*
cat: nm	cat: nm
sem: f	sem: f
mdr:	mdr:
fnc:	fnc: sleep
idy:	idy: 1
prn:	prn: 1
sur:	sur:
verb: *sing*	verb: *sing*
cat: n-s3’ v	cat: decl
sem: pres	sem: pres
mdr:	mdr:
arg:	arg: John
nc:	nc: 3 dream
pc:	pc: 1 sleep
prn:	prn: 2
sur:	sur:
verb: *sleep*	verb: *sleep*
cat: n-s3’ v	cat: decl
sem: pres	sem: pres
mdr:	mdr:
arg:	arg: Julia
nc:	nc: 2 sing
pc:	pc:
prn:	prn: 1
sur:	sur:
noun: *Susanne*	noun: *Susanne*
cat: nm	cat: nm
sem: f	sem: f
mdr:	mdr:
fnc:	fnc: dream
idy:	idy: 3
prn:	prn: 3

In this Word Bank, complete lexical proplets (rather than just their core values, cf. 3.3.1) are used as the owner records. The owner records of the verbs are represented by the unmarked form of the present tense, e.g., sing, whereby the cat value n-s3’ represents a valency position for “nominative minus singular third-person.”

The LA-hear.1 interpretation shown in 11.5.1–11.5.4 has resulted in a *set* of proplets the elements of which are sorted into the Word Bank 11.6.1 according to their main key, i.e., the value of their core attribute. Sorting proplets into a Word Bank is completely automatic. It constitutes the first step of the hearer’s pragmatic interpretation and corresponds to the down arrow in 2.5.1 (simplified theory of language). Subsequent steps of pragmatic interpretation consist in inferences.