6. Intrapropositional Functor–Argument Structure

The major constructions of natural language consist of the primary relations of (i) functor–argument structure and (ii) coordination, and the secondary relation of (iii) coreference, occurring (a) intra- as well as (b) extrapropositionally. Analyzing these six basic constellations requires the use of examples. Moreover, in order to be concrete, these examples must be from at least one real natural language. Thus, the task at hand has a strongly sign-oriented aspect.

Nevertheless, our approach is also agent-oriented insofar as the examples are analyzed in the speaker mode and the hearer mode. In the hearer mode, an LA-hear grammar maps the language-dependent surface into a set of proplets suitable to be stored in a Word Bank. In the speaker mode, an LA-think grammar navigates along the grammatical relations established between the language-independent proplets in the Word Bank while an LA-speak grammar realizes the traversed proplets in a natural language of choice, here English.

In this chapter, we begin the systematic investigation of the major constructions with the analysis of intrapropositional functor–argument structure. The grammatical analyses are presented at an intermediate level of abstraction, using the formats illustrated in 3.4.2 and 3.5.3 for the hearer mode and the speaker mode, respectively.

6.1 Overview

In Database Semantics, the analysis of a grammatical construction comprises three basic tasks: (i) Designing the semantic representation of the relevant natural language example as a set of proplets, (ii) automatically deriving the semantic representation from the example surface (hearer mode) and (iii) automatically deriving the surface from the semantic representation (speaker mode).

The semantic representation of an intrapropositional functor–argument structure is the set of proplets of an elementary proposition (cf. FoCL'99, p. 62). It consists of one verb (relation, functor) with one, two, or three nouns (objects, arguments), the number of which is determined by the verb. There may be adjectives (properties, modifiers) modifying either the verb (adverbial use) or a noun (adnominal use).

To emphasize that the set of proplets serving as a semantic representation is unordered, the following examples present the proplets in the alphabetical order of their core values (rather than the order suggested by the natural surface).

6.1.1 Examples of intrapropositional functor–argument structure

1. Representing The man gave the child an apple (three-place verb)

noun: apple	noun: child	verb: give	noun: man
fnc: give	fnc: give	arg: man child apple	fnc: give
mdr:	mdr:	mdr:	mdr:
prn: 1	prn: 1	prn: 1	prn: 1

2. Representing The little black dog barked (adnominal adjectives)

verb: bark	adj: black	adj: little	noun: dog
arg: dog	mdd: dog	mdd: dog	fnc: bark
mdr:	mdr:	mdr:	mdr: little black
prn: 2	prn: 2	prn: 2	prn: 2

3. Representing Julia has been sleeping deeply (adverbial adjective)

adj: deep	noun: Julia	verb: sleep
mdd: sleep	fnc: sleep	arg: Julia
mdr:	mdr:	mdr: deep
prn: 3	prn: 3	prn: 3

These semantic representations show the coding of grammatical relations. The function words have been absorbed by fusing them with the associated content words. The contributions of the function words as well as other morpho-syntactic details are specified in the attributes cat (for category) and sem (for semantics), which are omitted here for simplicity.[1] For complete semantic representations see Part III.

In Example 1, the verb *give* takes the noun arguments *man*, *child*, and *apple*. These grammatical relations are coded by the value give in the fnc slot of the noun proplets *man, child*, and *apple*, and by the values man, child, and apple in the arg slot of the verb proplet *give*. The different case roles of the arguments are specified in terms of their order in the arg slot of the verb proplet. The lexical proplets of the determiners have been fused with their noun proplets, whereby their contributions are represented as values of the cat and sem attributes (not shown, cf. Sect. 13.4).

In Example 2, the noun *dog* is modified by the adnominal adjectives *little* and *black*. These grammatical relations are coded by the value dog in the mdd slot of the adjective proplets *little* and *black*, and by the values little and black in the mdr slot of the noun proplet *dog*. The adnominal use of the adjectives is specified as the value adn of the cat attribute (not shown, cf. Sect. 13.3) of the adjective proplets.

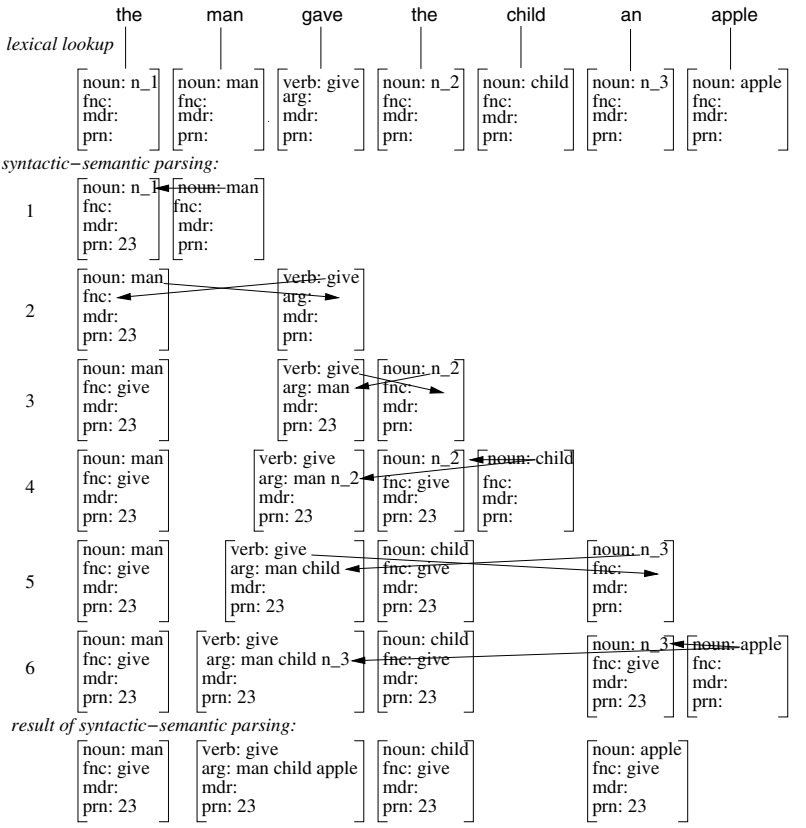
In Example 3, the verb *sleep* is modified by the adverbial adjective *deep*. This relation is coded by the value sleep in the mdd slot of the proplet *deep*, and by the value deep in the mdr slot of the verb proplet *sleep*. The adverbial use of the adjective is specified as the value adv of the cat attribute (not shown) of the adjective proplet.

[1] For example, in the derivation of Example 3 from the surface Julia has been sleeping deeply, the contributions of the auxiliaries has been and the progressive form are specified in the verb proplet by the feature [sem: hv_pres perf prog] to indicate the tense of the input sentence (cf. 13.4.7).

6.2 Determiners

We turn now to deriving the semantic representation 6.1.1 (1) from a suitable English surface (hearer mode). The result is an intrapropositional functor–argument structure consisting of a three-place verb and its arguments. The derivation illustrates how determiners and their nouns are fused by absorbing the noun into the determiner.

6.2.1 THREE-PLACE PROPOSITION: The man gave the child an apple



That the derivation 6.2.1 is time-linear (cf. 1.6.5) is apparent from the stair-like structure resulting from adding exactly one new "next word" in each new proplet line. The derivation is also surface compositional (cf. 1.6.1) because each word form in the surface has a lexical analysis and there are no "zero elements" postulated in the input.

In line 1, the substitution value n_1 of the determiner proplet is replaced by the value man of the noun proplet, which is then discarded (function word absorption). The result is shown in the first proplet of line 2. In the combination of the proplets *man* and *give*, the core value of *man* is copied into the arg slot of the verb and the core

value of *give* into to fnc slot of the noun. In line 3, the value give is copied into the fnc attribute of the next word proplet *the*, and the substitution value n_2 of *the* is copied into the arg slot of the verb.[2] In line 4, both instances of the substitution value n_2 are replaced by the value child of the next word. Integration of the noun proplet *apple* in lines 5 and 6 is analogous to that of *child* in lines 3 and 4. The result of the derivation is shown in the bottom line, using the natural surface order.

Once these proplets have been stored in the Word Bank, they support various LA-think traversals which use the relations between proplets for retrieval. One of these traversals is the standard VNNN navigation (cf. Appendix A), where the first N is the subject, the second N the indirect object, and the third N the direct object. Consider the production of the English input sentence from such a VNNN navigation:

6.2.2 Schematic production from a three-place proposition

activated sequence					*realization*
i					
...	V				
i.1		d			d
	V	N			
i.2		d nn			d nn
	V	N			
i.3	fv	d nn			d nn fv
	V	N			
i.4	fv	d nn	d		d nn fv d
	V	N	N		
i.5	fv	d nn	d nn		d nn fv d nn
	V	N	N		
i.6	v	d nn	d nn	d	d nn v d nn d
	V	N	N	N	
i.7	fv	d nn	d nn	d nn	d nn fv d nn d nn
	V	N	N	N	
i.8	fv p	d nn	d nn	d nn	d nn fv d nn d nn p
	V	N	N	N	

Like 3.5.3, this derivation shows the handling of word order, function word precipitation, and the switching between LA-think and LA-speak. The abstract surfaces d, nn, fv, and p stand for determiner, noun, finite verb, and punctuation, respectively.

In Database Semantics, the meaning of the determiners (here the) is handled in terms of the atomic values exh, sel, sg, pl, def, and indef (cf. 6.2.9) in the sem attribute (cf. 6.2.7) of the noun proplets. This is different from the treatment of determiners as quantifiers in Predicate Calculus, beginning with Russell's (1905) celebrated analysis

[2] To distinguish different determiners, the substitution values n_1, n_2, n_3, etc., are automatically incremented during lexical lookup (cf. 13.3.5, 13.5.4, and 13.5.6).

of "definite descriptions" and still being expanded within the framework associated with Montague (1974), e.g., Barwise and Perry (1983); Kamp and Reyle (1993); and others. Consider the following example:

6.2.3 Predicate calculus analysis of All girls sleep

$$\forall x\ [\text{girl}(x) \rightarrow \text{sleep}(x)]$$

The interpretation of such a formula is defined with respect to a model and a variable assignment. Following Montague (1974), the model @ is defined as a tuple (A,F), where A is a set of individuals, e.g., $\{a_0, a_1, a_2, a_3\}$, and F is an assignment function which assigns to every one-place predicate in the formal language an element of 2^A (i.e., the power set of A) as an interpretation (and accordingly for two-place predicates, etc.). For example, F(girl) might be defined in @ as $\{a_1, a_2\}$ and F(sleep) as $\{a_0, a_2\}$. This means that a_1 and a_2 in the model are girls, while a_0 and a_2 are sleeping.

The dependence of the truth-value of a formula on the actual definition of the model and a variable assignment is represented by Montague by adding @ and g as superscripts to the end of the formula:

6.2.4 Interpretation relative to a model

$$\forall x\ [\text{girl}(x) \rightarrow \text{sleep}(x)]^{@,g}$$

The interpretation of the quantifier $\forall$ is based on the variable assignment g as follows: The whole formula is true relative to the model @ if it holds for *all* possible variable assignments g' that the formula without the outermost quantifier is true:

6.2.5 Elimination of the outermost quantifier

$$[\text{girl}(x) \rightarrow \text{sleep}(x)]^{@,g'}$$

The purpose of eliminating the quantifier is to reduce the Predicate Calculus formula to Propositional Calculus and its truth tables (cf. Bochenski 1961). This is achieved by systematically assigning all possible values in the set of individuals $A =_{def} \{a_0, a_1, a_2, a_3\}$ to the variable x and determining the truth-value of the subformulas girl(x) and sleep(x) for each assignment. Thus, g' first assigns to the variable x the value a_0, then the value a_1, etc. Given the definition of the model $@ =_{def}$ (A,F), we can now check for each such assigment whether or not it makes the formula 6.2.5 true.

For example, the first assignment $g'(x) = a_0$ makes the formula true: a_0 is not in the set denoted by F(girl) in @; therefore, based on the truth table of $p \rightarrow q$ in Propositional Calculus, the formula in 6.2.5 is true for this assignment. The second assignment $g'(x) = a_1$, in contrast, makes the formula in 6.2.5 false: a_1 is in the set denoted by F(girl), but not in the set denoted by F(sleep) in @. Having shown that *not all* variable assignments g' make the formula in 6.2.5 true, the interpretation of the formula in 6.2.3 is determined to be false relative to @. Given how the model $@ =_{def}$ (A,F) was defined, this is in accordance with intuition.

This method of treating determiners at the highest level of the logical syntax leads to ambiguities because the quantifiers may have different orders.[3] For example, the Predicate Calculus analysis of Every man loves a woman has the following readings:

6.2.6 ANALYZING Every man loves a woman IN PREDICATE CALCULUS

Reading 1: ∀x [man(x) → ∃y [woman(y) & love(x,y)]]
Reading 2: ∃y [woman(y) & ∀x [man(x) → love(x,y)]]

On reading 1, it holds for every man that there is some woman whom he loves. On reading 2, there is a certain woman, e.g., Marilyn Monroe, who is loved by every man.

The two formulas of Predicate Calculus are based on the notions of functor–argument structure, coordination, and coreference, though in a manner different from their use in Database Semantics. Functor–argument structure is used in man(x), woman(y), and love(x,y); coordination is used in [man(x) → P] and [woman(y) & Q]; and coreference is expressed by the quantifiers and the horizontally bound variables in ∀x [man(x) ... love(x,y)] and ∃y [woman(y)... love(x,y)].

In Database Semantics, in contrast, the meanings of the determiners every and a are expressed by atomic values pl exh of the sem attribute of the noun proplets:

6.2.7 RESULT OF PARSING Every man loves a woman IN DBS

sur:	sur:	sur:
noun: *man*	verb: *love*	noun: *woman*
cat: snp	cat: v	cat: snp
sem: pl exh	sem: pres	sem: indef sg
mdr:	mdr:	mdr:
fnc: love	arg: man woman	fnc: love
idy: 1	prn: 1	idy: 2
prn: 1		prn: 1

In this analysis, the sentence is not ambiguous: It has only reading 1 of 6.2.6 – which is entailed by reading 2 (i.e., reading 1 is true whenever reading 2 is true, but not vice versa). In other words, whether or not some (or even all) of the men happen to love the same woman is treated as a private matter in Database Semantics.

Furthermore, the Database Semantics analysis uses only intrapropositional functor–argument structure: As in the natural surface, there is neither coordination nor coreference. Treated as determiners, the "quantifiers" every and a are each fused with their noun into a single proplet (similar to 6.2.1, 6.3.1, 6.5.1, 6.6.2, 8.2.1, and 8.3.2).

The atomic values exh (exhaustive), sel (selective), sg (singular), pl (plural), def (definite), and indef (indefinite) are used in different combinations to characterize the following kinds of noun phrases in English:

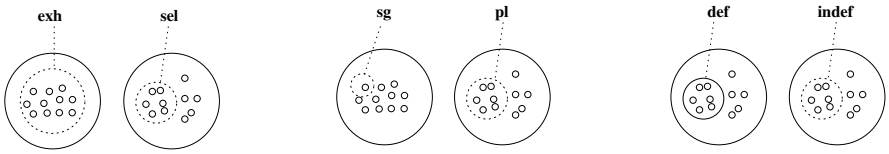
[3] Cf. Kempson and Cormack (1981). In recent years, Minimal Recursion Semantics (MRS, Copestake, Flickinger et al. 2006) has devoted much work to avoid the unnatural proliferation of readings caused by different quantifier scopes, using "semantic under-specification." In MRS, under-specification is limited to quantifier scope (see also Steedman 2005). In Database Semantics, which has neither quantifiers nor quantifier scope, semantic under-specification applies to *all* content coded at the language level and is being used for the matching with a delimited context of use (cf. FoCL'99, Sect. 5.2).

6.2.8 The sem values of different determiner–noun combinations

a girl	[sem: indef sg]
some girls	[sem: indef pl sel]
all girls	[sem: exh pl]
the girl	[sem: def sg]
the girls	[sem: def pl]

The atomic values have the following set-theoretic interpretations:

6.2.9 Set-theoretic interpretation of exh, sel, sg, pl, def, indef



The value exh refers to all members of a set, called the domain, while sel refers only to some. The value sg refers to a single member of the domain, while pl refers to more than one. The value def refers to a prespecified subset of the domain, while no such subset is presumed by indef.

Each value can only be combined with a value from the other pairs. Thus exh cannot combine with sel, sg cannot combine with pl, and def cannot combine with indef. However, the combinations exh pl, sel sg, sel pl, def sg, def pl, indef sg, indef pl, etc., are legitimate and have different meanings. The combination exh sg is theoretically possible, but makes little sense (pace Russell 1905) because the domain would have to be a unit set.

Regarding the interpretation of determiners in Database Semantics during communication, consider a robot in the speaker mode. If it perceives the set-theoretic situation corresponding to exh and pl as shown in 6.2.9, it will use the determiner all, and similarly with the other values. Correspondingly, if a robot in the hearer mode hears the noun phrase all girls, for example, it will be able to draw the corresponding set-theoretic situation or to choose the right schema from several alternatives.

The Database Semantics approach differs from Predicate Calculus in that Predicate Calculus uses the words *some* and *all* in the metalanguage to define the words some and all in the object-language (as shown by the use of the variable assignment function g' described above), while Database Semantics is based on a procedural interpretation. This difference is based on profoundly different ontological assumptions of the two approaches, illustrated in 2.3.1 with the most simple sentence Julia sleeps.

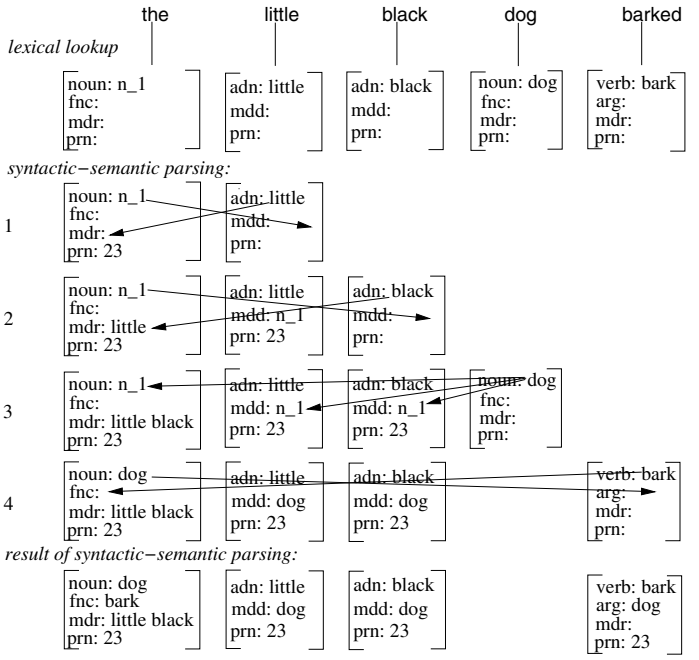
A related difference is that the semantics of Predicate Calculus is based on truth-conditions, while that of Database Semantics is not. Instead, Database Semantics handles truth as procedural assertions. For example, if a robot observes correctly that every girl is sleeping and communicates this fact by saying every girl is sleeping, it is

speaking truly. Semantically, every girl is sleeping asserts that there is a set of more than one girl and all elements of the set participate in whatever is asserted by the verb.

6.3 Adjectives

A form of noun phrase more complex than the determiner–noun combination shown above includes one or more adnominal adjectives, as illustrated by the following example of a (short) modifier recursion:

6.3.1 PARSING The little black dog barked IN THE HEARER MODE



In line 1, the core value of the adnominal adjective *little* is copied into the mdr slot of the determiner, and the substitution value n_1 is copied into the mdd slot of the proplet *little*. In line 2, the core value of the adnominal adjective *black* is copied and added to the mdr slot of the determiner, and the substitution value n_1 is copied into the mdd slot of the proplet *black*. In line 3, all three instances of the substitution value n_1 are simultaneously replaced by the core value of the lexical proplet *dog*, which is then discarded. In line 4, the core value of the former determiner proplet is copied into the arg slot of the verb proplet *bark*, and the core value of the verb is copied into the fnc slot of the former determiner proplet. As a result any adnominal, e.g., *black*,

allows the retrieval of the associated noun, here *dog*, and any noun allows the retrieval of the associated adnominal(s), here *little* and *black*.

The (re)production of the input sentence from a standard VNAA navigation in the speaker mode is characterized below.

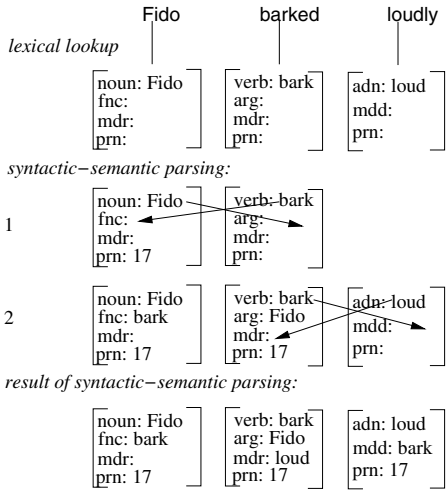
6.3.2 SCHEMATIC PRODUCTION OF THE The little black dog barked

	activated sequence	*realization*
i	V	
i.1	d V N	d
i.2	d an V N A	d an
i.3	d an an V N A A	d an an
i.4	d nn an an V N A A	d an an nn
i.5	fv d nn an an V N A A	d an an nn fv
i.6	fv p d nn an an V N A A	d an an nn fv p

The abstract surfaces d, an, nn, fv, and p stand for determiner, adnominal, noun, finite verb, and punctuation, respectively.

Next consider the time-linear derivation of a sentence with an adverbial adjective:

6.3.3 PARSING Fido barked loudly IN THE HEARER MODE



In adverbial use, the adjective proplet contains a copy of the core value of the modified *verb* in its mdd slot – in contrast to adnominal use, in which the adjective proplet contains a copy of the core value of the modified *noun*. Furthermore, in adverbial use it is the verb which contains a copy of the adjective's core value in its mdr slot, while in adnominal use it is the noun.

The (re)production of the input sentence from a standard VNA navigation in the speaker mode is characterized below:

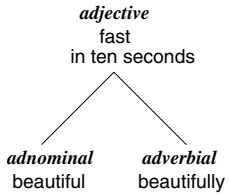
6.3.4 SCHEMATIC PRODUCTION OF THE Fido barked loudly

	activated sequence	*realization*
i	V	
i.1	n V N	n
i.2	fv n V N	n fv
i.3	fv n av V N A	n fv av
i.4	fv p n av V N A	n fv av p

The abstract surfaces n and av stand for name and adverbial, respectively.

In Database Semantics, adnominal and adverbial modifiers are treated as variants of the same part of speech: adjective. The Latin root of this term means "what is thrown in," which aptly characterizes the optional quality of modifiers in general.

6.3.5 RELATION OF THE TERMS Adjective, Adnominal, and Adverbial IN DBS



The morphological motivation for treating adnominals and adverbials as instances of the same part of speech is their similarity. Consider, for example, the adnominal adjective beautiful and the adverbial adjective beautifully in English, or schöne, schöner, schönes, etc., (adnominal adjectives) and schön (adverbial adjective) in German. The two uses resemble each other also in their analytic degrees, as in more beautiful (adnominal) and more beautifully (adverbial). In synthetic degrees, as in faster, the adnominal and the adverbial form are not even distinguished in English.

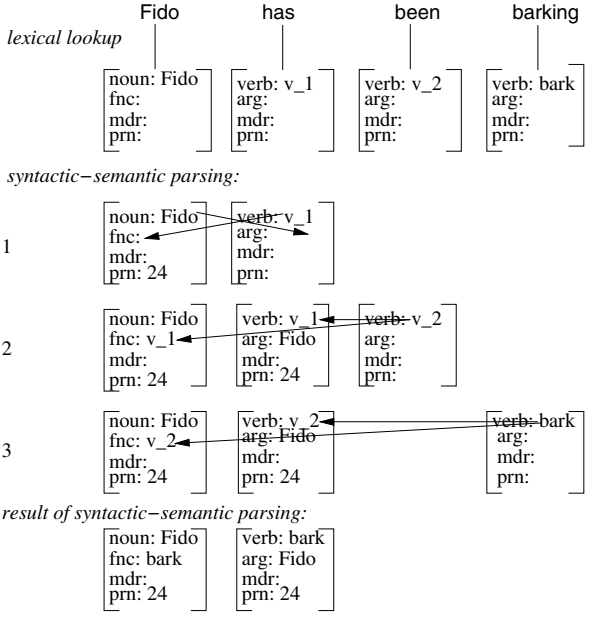
Modifier proplets have the core attribute adj, which represents the part of speech. If there is no morphological distinction between the adnominal and adverbial use of a

modifier and its grammatical role is still undefined, it is called an adjective. Modifiers which are restricted morphologically or by their syntactic environment to adnominal use are called adnominals, and accordingly for adverbials. Just as there are elementary nouns like Fido and complex nouns like the little black dog, there are elementary adjectives like fast and complex adjectives like in ten seconds.

6.4 Auxiliaries

Another kind of function word besides determiners are auxiliaries. They combine with nonfinite forms of the main verb to form complex constructions:

6.4.1 COMPLEX VERB PHRASE: Fido has been barking



Just as the determiners have the substitution values n_1, n_2, etc., (cf. 6.2.1), the auxiliaries have the substitution values v_1, v_2, etc.,, as their core values.

In line 1, the value v_1 is copied into the fnc slot of *Fido*, and the core value of *Fido* is copied into the arg slot of the first auxiliary. In line 2, the two instances of v_1 are replaced by the value v_2 serving as the core value of the second auxiliary. In line 3, the two instances of v_2 are replaced by the core value of the proplet *bark*, which is then discarded (function word absorption). The contribution of the two auxiliaries and the progressive form of the nonfinite main verb is represented by the values of the cat and sem attributes (not shown, cf. Sect. 13.4) of the resulting verb proplet.

The production of auxiliaries (function word precipitation) is derived as follows:

6.4.2 SCHEMATIC PRODUCTION OF Fido has been barking

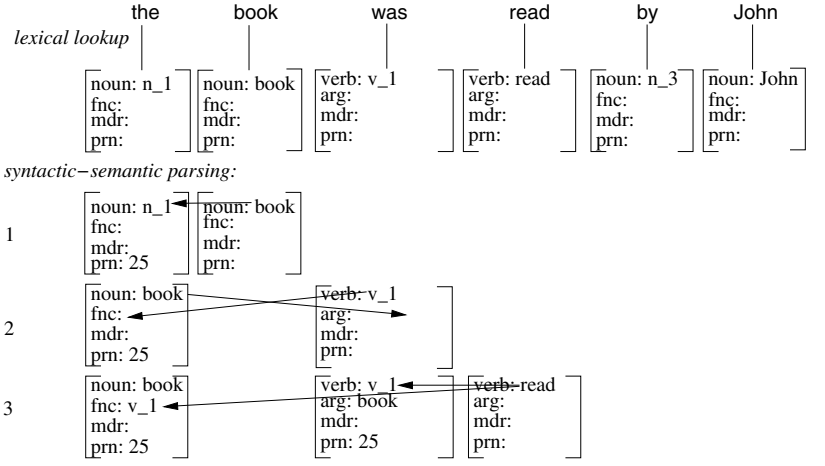
activated sequence			*realization*
i.			
	V		
i.1		n	n
	V	N	
i.2	ax	n	n ax
	V	N	
i.3	ax ax	n	n ax ax
	V	N	
i.4	ax ax nv	n	n ax ax nv
	V	N	
i.6	ax ax nv p	n	n ax ax nv p
	V	N	

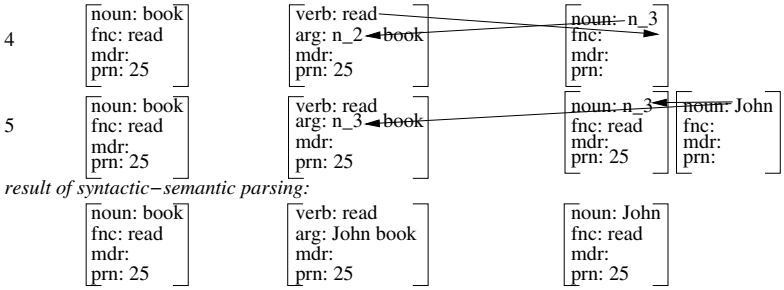
Here, the abstract surfaces n, ax, nv, and p stand for name, auxiliary, nonfinite verb, and punctuation, respectively.

6.5 Passive

Another kind of complex verb construction is passive. In grammar, active and passive are called the *genus verbi* or *voice of the verb*. In Database Semantics, passive is viewed as a certain perspective (cf. Sect. 5.5) on a propositional content, resulting from backward navigation (see also Sect. 9.6).

6.5.1 INTERPRETATION OF PASSIVE: The book was read by John





Up to line 2, the voice of the verb is undecided: The sentence start the book was could be continued as an active, as in the book was lying on the table, or as the passive in question. However, as soon as the past participle of *read* is added in line 3, the order of the values in the arg attribute of the verb proplet is adjusted by inserting the nominal substitution value n_2, fixing the switch to passive. The result is shown by the verbal proplet in line 4.

At this point, the sentence could be completed as The book was read. Such a suppression of the agent is an option typical of passive. In our derivation, it is expressed by the feature [arg: n_2 book] of the verb proplet. It would be up to inferencing to determine the agent by finding a suitable value for n_2. However, as the sentence continues with the *by*-phrase, the value n_2 is replaced by the value n_3. The nominal proplet *John* is added and absorbed into the *by*-proplet, replacing all occurrences of n_3 with the value John. The result is the same set of proplets as would be derived from the corresponding active.

Once the proplets have been stored in the Word Bank, they can be traversed forward or backward. A simple way to characterize a standard forward navigation is by showing the steps as V . VN . VNN, while the alternative backward navigation through the same set of proplets is shown as the steps V . V_N . VNN. This method presupposes that the role of the verbal arguments is expressed by their order: The first N is the agent or (deep) subject and the second N the patient or object; if there is a third N, the second N is the indirect object and the third N is the direct object. In other words, the same order is used in the arg attribute of verbal proplets, regardless of verbal voice.[4]

Based on a V . V_N . VNN navigation, the input sentence is (re)produced as follows:

6.5.2 SCHEMATIC PRODUCTION OF The book was read by John

	activated sequence		*realization*
i			
	V		
i.1		d	d
	V	N	

[4] Expressing the grammatical role in terms of order is a terminological choice. The same content could be expressed by introducing additional attributes like agent. We do not take this option in order to

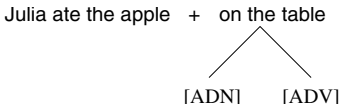
i.2			d nn	d nn
	V		N	
i.3	ax		d nn	d nn ax
	V		N	
i.4	ax nv		d nn	d nn ax nv
	V		N	
i.5	ax nv	by	d nn	d nn ax nv by
	V	N	N	
i.6	ax nv	by n	d nn	d nn ax nv by n
	V	N	N	
i.7	ax nv p	by n	d nn	d nn ax nv by n p
	V	N	N	

The corresponding active is produced from the same set of proplets, but is activated in the standard V . VN . VNN navigation order. A production of passive in the interrogative mood is shown in A.5.2. For a more detailed analysis of passive in Database Semantics, see Twiggs (2005).

6.6 Prepositions

We conclude our survey of intrapropositional functor–argument structures with prepositional phrases in adnominal and adverbial use ("PP attachment"). In certain positions, both interpretations are possible. For example, in Julia ate the apple on the table, the prepositional phrase on the table can modify *eat* (in the sense of Julia sitting on the table while eating) or *apple* (in the sense of eating the apple which was on the table). In DBS, this ambiguity is handled in terms of semantic doubling:[5]

6.6.1 Semantic doubling of a prepositional phrase



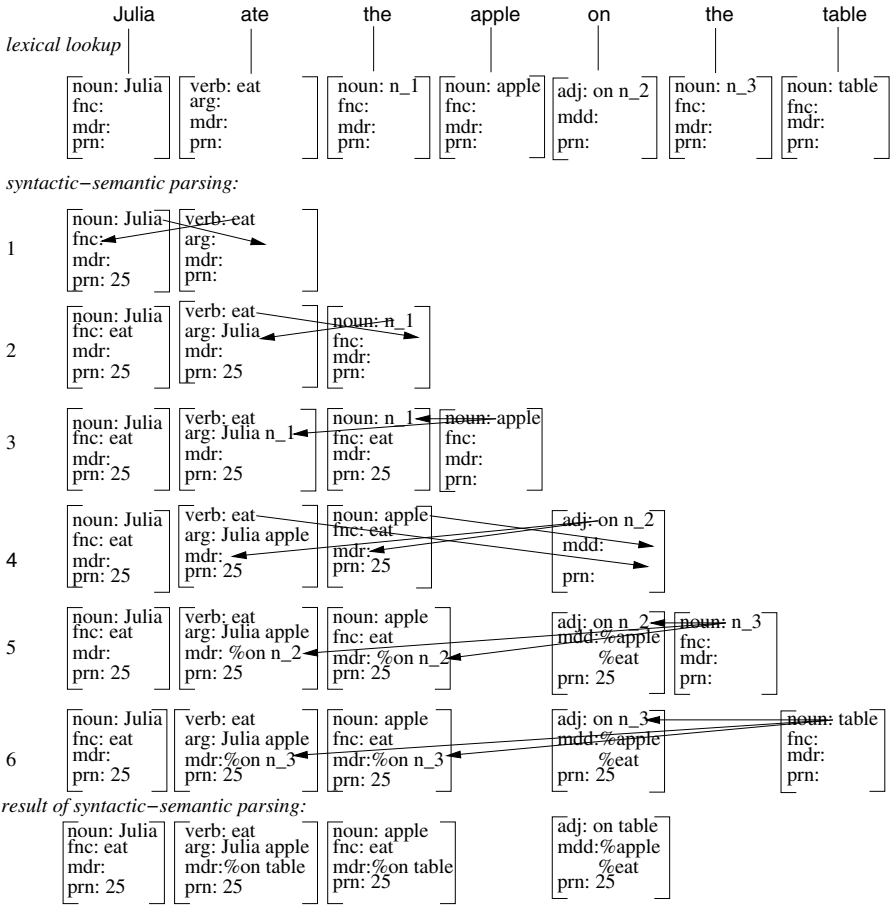
According to this analysis, the example, is unambiguous syntactically because only one representation is derived. It is ambiguous semantically, however, in that the prepositional phrase has an ADN (adnominal) and an ADV (adverbial) meaning attached to it. It is assumed that the alternative ADN and ADV meanings are positioned relative to the context of use, whereby the choice between the alternatives is based on the principle of best match.

The formal realization of semantic doubling is shown by the following derivation:

maintain the flat (nonrecursive) feature structure of proplets.

[5] The method of semantic doubling was first proposed in CoL'89, pp. 219–232 and pp. 239–247. See also FoCL'99, pp. 234 ff.

6.6.2 ADJECTIVAL PHRASE: Julia ate the apple on the table

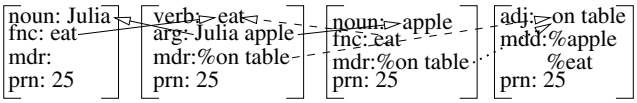


In line 4, the adj value of the preposition is copied into the mdr slot of both the verb proplet *eat* and the noun proplet *apple*. Furthermore, the core values of the verb and the noun are both copied into the mdd slot of the preposition *on*. To indicate that the copied values are only one of several possible interpretations (here two), they are preceded by % (cf. line 5). The fusion of the proplets *on*, *the*, and *table* into one is another instance of function word absorption.

When the resulting proplets are stored, the ambiguity may be resolved by an inference which removes a pair of %-values, for example, %on table in the mdr-slot of the *eat* proplet and %eat in the mdd-slot of the *on table* proplet. However, the proplets may also remain as they are, leaving the ambiguity in place.

Turning to the (re)production of the input sentence in its two interpretations, consider the retrieval connections of the output of 6.6.2:

6.6.3 RETRIEVAL DURING NAVIGATION



Starting with the V proplet *eat*, the navigation retrieves the N proplet *Julia*, returns to the V, and retrieves the N proplet *apple*. At this point it may either return to the V to continue from there to the adverbial interpretation of *on table* (dashed arrows), or it may continue directly to the adnominal interpretation (dotted arrow). Either case produces a standard VNNA navigation, serving as the basis of the following schematic production:

6.6.4 PRODUCTION OF Julia ate the apple on the table

	activated sequence				*realization*
i					
	V				
i.1		n			n
	V	N			
i.2	fv	n			n fv
	V	N			
i.3	fv	n	d		n fv d
	V	N	N		
i.4	fv	n	d n		n fv d n
	V	N	N		
i.5	fv	n	d n	pp	n fv d n pp
	V	N	N	A	
i.6	fv	n	d n	pp d	n fv d n pp d
	V	N	N	A	
i.7	fv	n	d n	pp d nn	n fv d n pp d nn
	V	N	N	A	
i.8	fv p	n	d n	pp d nn	n fv d n pp d nn p
	V	N	N	A	

The abstract surface pp stands for a preposition.

Treating "PP-attachment" by means of semantic doubling is especially beneficial in more complex examples containing potentially unlimited sequences of prepositional phrases, as in *Julia ate the apple on the table in the garden behind the tree* .... There, semantic doubling reduces the mathematical complexity from the exponential complexity of the Nativist approach to linear complexity.[6] For a detailed analysis of "prepositional phrase attachment" in the hearer mode see Chapt. 15.

[6] Cf. FoCL'99, Sect. 12.5.