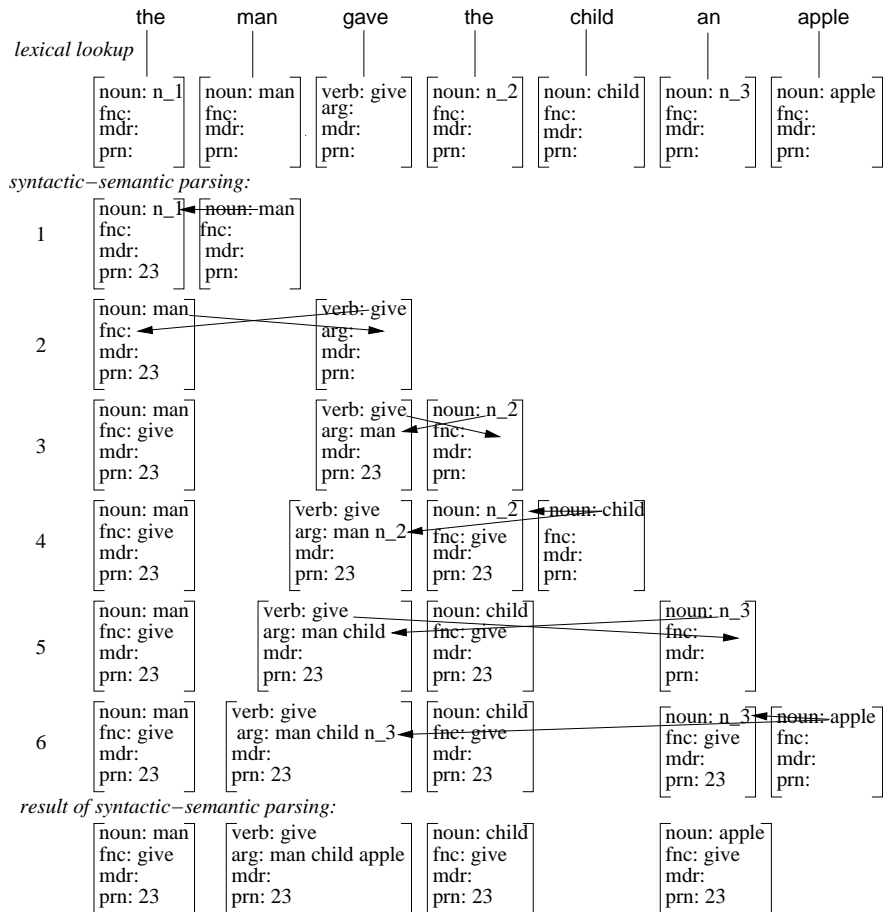


6.2 Determiners

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 *fn*c slot of the noun. In line 3, the value *give* is copied into the *fn*c 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

<i>activated sequence</i>	<i>realization</i>
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) cele-

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).

brated analysis 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., {a₀, a₁, a₂, a₃}, 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 {a₁, a₂} and F(sleep) as {a₀, a₂}. This means that a₁ and a₂ in the model are girls, while a₀ and a₂ 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} {a₀, a₁, a₂, a₃} 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 a₀, then the value a₁, etc. Given the definition of the model @ =_{def} (A,F), we can now check for each such assignment whether or not it makes the formula 6.2.5 true.

For example, the first assignment g'(x) = a₀ makes the formula true: a₀ is not in the set denoted by F(girl) in @; therefore, based on the truth table of p → q in Propositional Calculus, the formula in 6.2.5 is true for this assignment. The second assignment g'(x) = a₁, in contrast, makes the formula in 6.2.5 false: a₁ 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: $\forall x [\text{man}(x) \rightarrow \exists y [\text{woman}(y) \ \& \ \text{love}(x,y)]]$

Reading 2: $\exists y [\text{woman}(y) \ \& \ \forall x [\text{man}(x) \rightarrow \text{love}(x,y)]]$

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 $\text{man}(x)$, $\text{woman}(y)$, and $\text{love}(x,y)$; coordination is used in $[\text{man}(x) \rightarrow P]$ and $[\text{woman}(y) \ \& \ Q]$; and coreference is expressed by the quantifiers and the horizontally bound variables in $\forall x [\text{man}(x) \dots \text{love}(x,y)]$ and $\exists y [\text{woman}(y) \dots \text{love}(x,y)]$.

In DBS, in contrast, the meanings of the determiners *every* and *a* are expressed by atomic values *pl exh* and *indef sg*, respectively, of the noun proplets’ *sem* attribute:

6.2.7 RESULT OF PARSING *Every man loves a woman* IN DBS

sur: noun: <i>man</i> cat: snp sem: pl exh mdr: fnc: love idy: 1 prn: 1	sur: verb: <i>love</i> cat: v sem: pres mdr: arg: man woman prn: 1	sur: noun: <i>woman</i> cat: snp sem: indef sg mdr: fnc: love idy: 2 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 intrapositional 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).

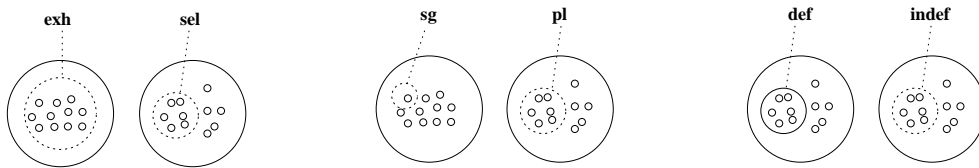
The atomic values *exh* (exhaustive), *sel* (selective), *sg* (singular), *pl* (plural), *def* (definite), and *indef* (indefinite) are used in different combinations to characterize the following kinds of noun phrases in English:

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**.

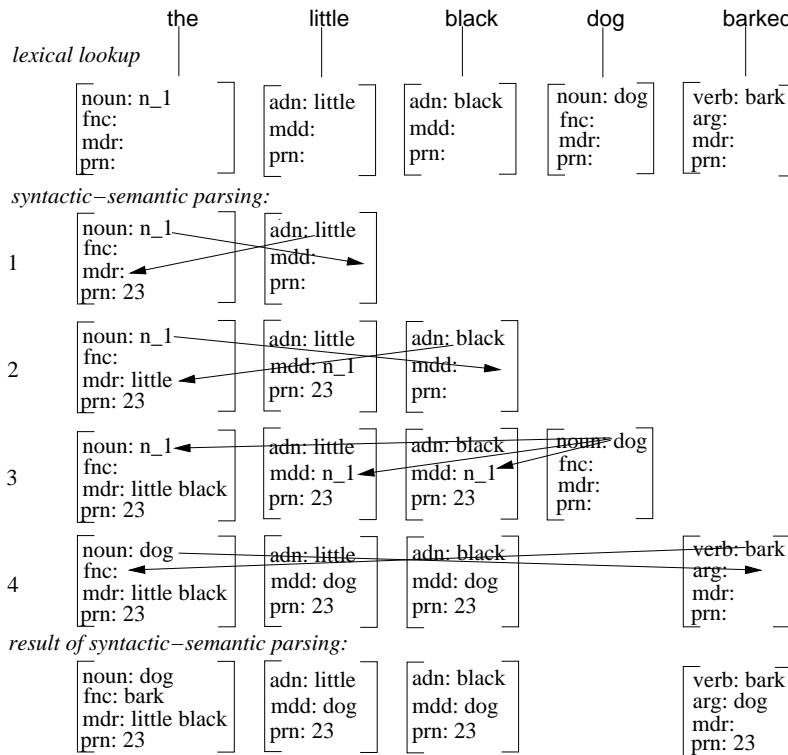
A related difference is that the semantics of Predicate Calculus is based on truth-conditions, while that of Database Semantics is not. Instead, Database Semantics handles truth as procedural assertions. For example, if a robot observes correctly that every girl is sleeping and communicates this fact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