7. Generative grammar

7.1 Language as a subset of the free monoid

7.1.1 Definition of language

A language is a set of word sequences.

7.1.2 Illustration of the free monoids over LX = {a,b}

ε
a, b
aa, ab, ba, bb
aaa, aab, aba, abb, baa, bab, bba, bbb
aaaa, aaab, aaba, aabb, abaa, abab, abba, abbb, . . .
. . .

7.1.3 Informal description of the artificial language $a^k b^k$ (with $k \geq 1$)

Its wellformed expressions consist of an arbitrary number of the word a followed by an equal number of the word b.

7.1.4 Wellformed expressions of $a^k b^k$

a b, a a b b, a a a b b b, a a a a b b b b, etc.,

7.1.5 Illformed expressions of $a^k b^k$

a, b, b a, b b a a, a b a b, etc.,

7.1.6 PS-grammar for $a^k b^k$

$$S \rightarrow a\ S\ b$$
$$S \rightarrow a\ b$$

A formal grammar may be viewed as a filter which selects the wellformed expressions of its language from the free monoid over the language's lexicon.

7.1.7 Elementary formalisms of generative grammar

1. Categorial or C-grammar
2. Phrase-structure or PS-grammar
3. Left-associative or LA-grammar

7.1.8 Algebraic definition

The algebraic definition of a generative grammar explicitly enumerates the basic components of the system, defining them and the structural relations between them using only notions of set theory.

7.1.9 Derived formalisms of PS-grammar

Syntactic Structures, Generative Semantics, Standard Theory (ST), Extended Standard Theory (EST), Revised Extended Standard Theory (REST), Government and Binding (GB), Barriers, Generalized Phrase Structure Grammar (GPSG), Lexical Functional Grammar (LFG), Head-driven Phrase Structure Grammar (HPSG)

7.1.10 Derived formalisms of C-grammar

Montague grammar (MG), Functional Unification Grammar (FUG), Categorial Unification Grammar (CUG), Combinatory Categorial Grammar (CCG), Unification-based Categorial Grammar (UCG)

7.1.11 Examples of semi-formal grammars

Dependency grammar (Tesnière 1959), systemic grammar (Halliday 1985), stratification grammar (Lamb ??)

7.2 Methodological reasons for generative grammar

7.2.1 Grammatically well-formed expression

the little dogs have slept earlier

7.2.2 Grammatically ill-formed expression

* earlier slept have dogs little the

7.2.3 Methodological consequences of generative grammar

- *Empirical*: formation of explicit hypotheses

  A formal rule system constitutes an explicit hypothesis about which input expressions are well-formed and which are not. This is an essential precondition for incremental improvements of the empirical description.

- *Mathematical*: determining formal properties

  A formal rule system is required for determining mathematical properties such as decidability, complexity, and generative capacity. These in turn determine whether the formalism is suitable for empirical description and computational realization.

- *Computational*: declarative specification for parsers

  A formal rule system may be used as a declarative specification of the parser, characterizing its necessary properties in contrast to accidental properties stemming from the choice of the programming environment, etc. A parser in turn provides the automatic language analysis needed for the verification of the individual grammars.

7.3 Adequacy of generative grammars

7.3.1 Desiderata of generative grammar for natural language

The generative analysis of natural language should be simultaneously

- defined *mathematically* as a formal theory of low complexity,
- designed *functionally* as a component of natural communication, and
- realized *methodologically* as an efficiently implemented computer program in which the properties of formal language theory and of natural language analysis are represented in a modular and transparent manner.

7.4 Formalism of C-grammar

7.4.1 The historically first generative grammar

Categorial grammar or C-grammar was invented by the Polish logicians LEŚNIEWSKI 1929 and AJDUKIEWICZ 1935 in order to avoid the Russell paradox in formal language analysis. C-grammar was first applied to natural language by BAR-HILLEL 1953.

7.4.2 Structure of a logical function

$$\begin{tikzcd}
\text{1. function name:} & \text{2. domain} & \Longrightarrow & \text{3. range} \\
 & & \text{4. assignment} \arrow[u] &
\end{tikzcd}$$

7.4.3 Algebraic definition of C-grammar

A C-grammar is a quintuple $<$ W, C, LX, R, CE $>$.

1. W is a finite set of word form surfaces.
2. C is a set of categories such that
   (a) *basis*
   u and v ϵ C,
   (b) *induction*
   if X and Y ϵ C, then also (X$/$Y) and (X$\backslash$Y) ϵ C,
   (c) *closure*
   Nothing is in C except as specified in (a) and (b).
3. LX is a finite set such that LX $\subset$ (W $\times$ C).
4. R is a set comprising the following two rule schemata:
   $\alpha_{(Y/X)} \circ \beta_{(Y)} \Rightarrow \alpha\beta_{(X)}$
   $\beta_{(Y)} \circ \alpha_{(Y\backslash X)} \Rightarrow \beta\alpha_{(X)}$
5. CE is a set comprising the categories of *complete expressions*, with CE $\subseteq$ C.

7.4.4 Recursive definition of the infinite set C

Because the start elements u and v are in C so are (u/v), (v/u), (u\v), and (v\u) according to the induction clause. This means in turn that also ((u/v)/v), ((u/v)\u), (u/(u/v)), (v/(u/v)), etc., belong to C.

7.4.5 Definition of LX as finite set of ordered pairs

Each ordered pair is built from (i) an element of W and (ii) an element of C. Which surfaces (i.e. elements of W) take which elements of C as their categories is specified in LX by explicitly listing the ordered pairs.

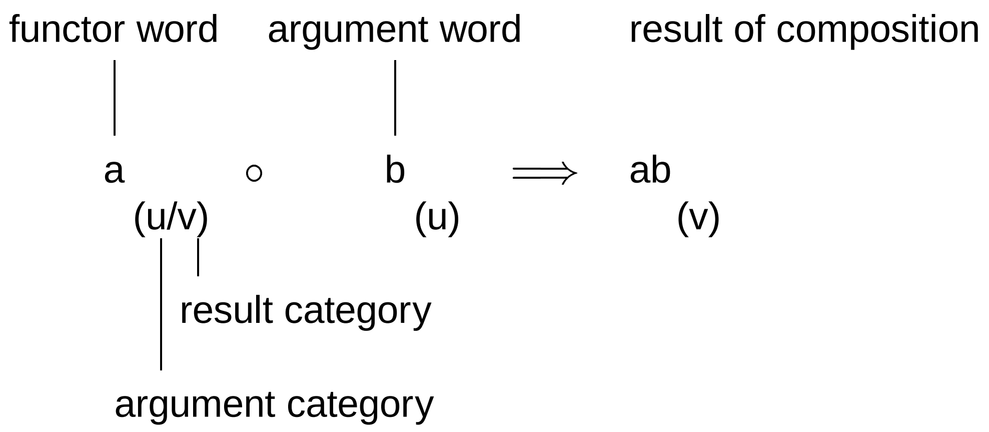
7.4.6 Definition of the set of rule schemata R

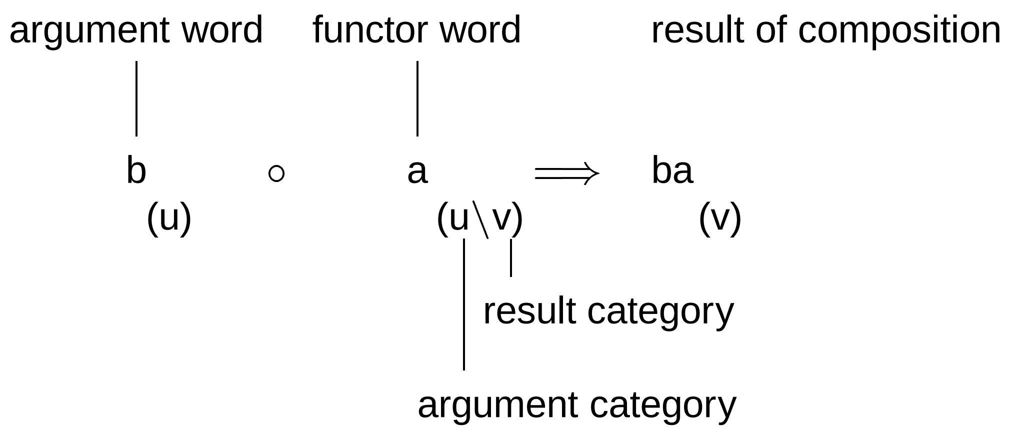
The rule schemata use the variables α and β to represent the surfaces of the functor and the argument, respectively, and the variables X and Y to represent their category patterns.

7.4.7 Definition of the set of complete expressions CE

Depending on the specific C-grammar and the specific language, this set may be finite and specified in terms of an explicit listing, or it may be infinite and characterized by patterns containing variables.

7.4.8 Implicit pattern matching in combinations of bidirectional C-grammar



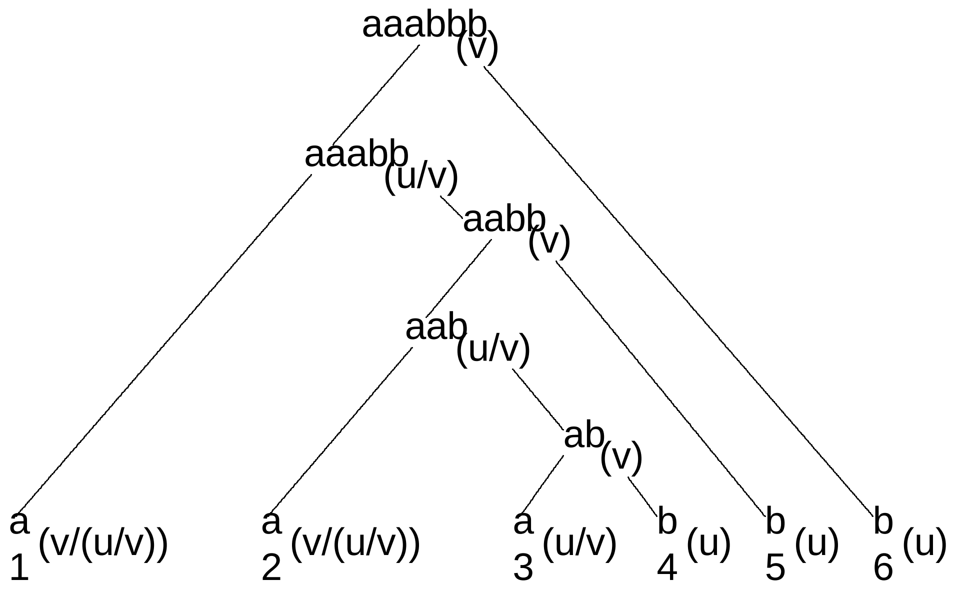


7.4.9 C-grammar for $a^k b^k$

$$LX =_{def} \{a_{(u/v)}, b_{(u)}, a_{(v/(u/v))}\}$$
$$CE =_{def} \{(v)\}$$

The word a has two lexical definitions with the categories (u/v) and (v/(u/v)), respectively, for reasons apparent in the following derivation tree.

7.4.10 Example of $a^k b^k$ derivation, for $k = 3$

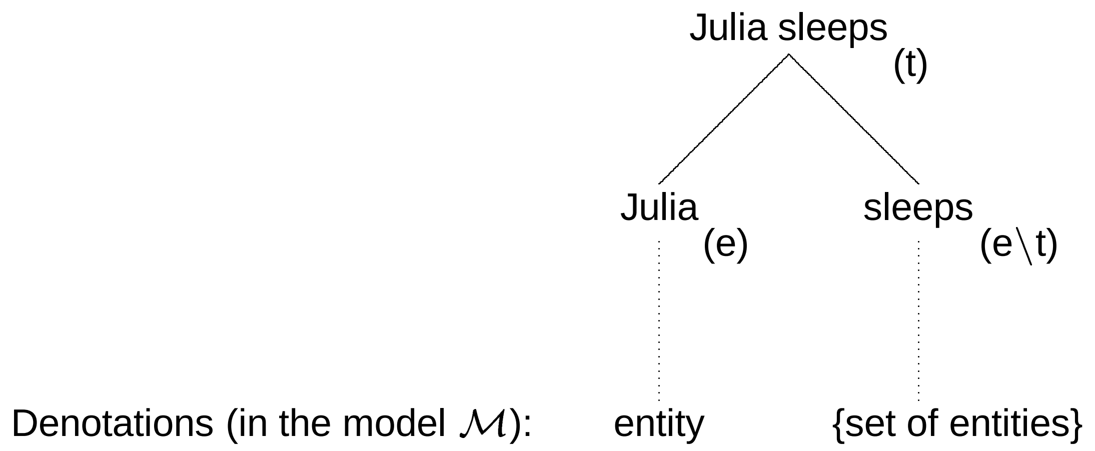


7.5 C-grammar for natural language

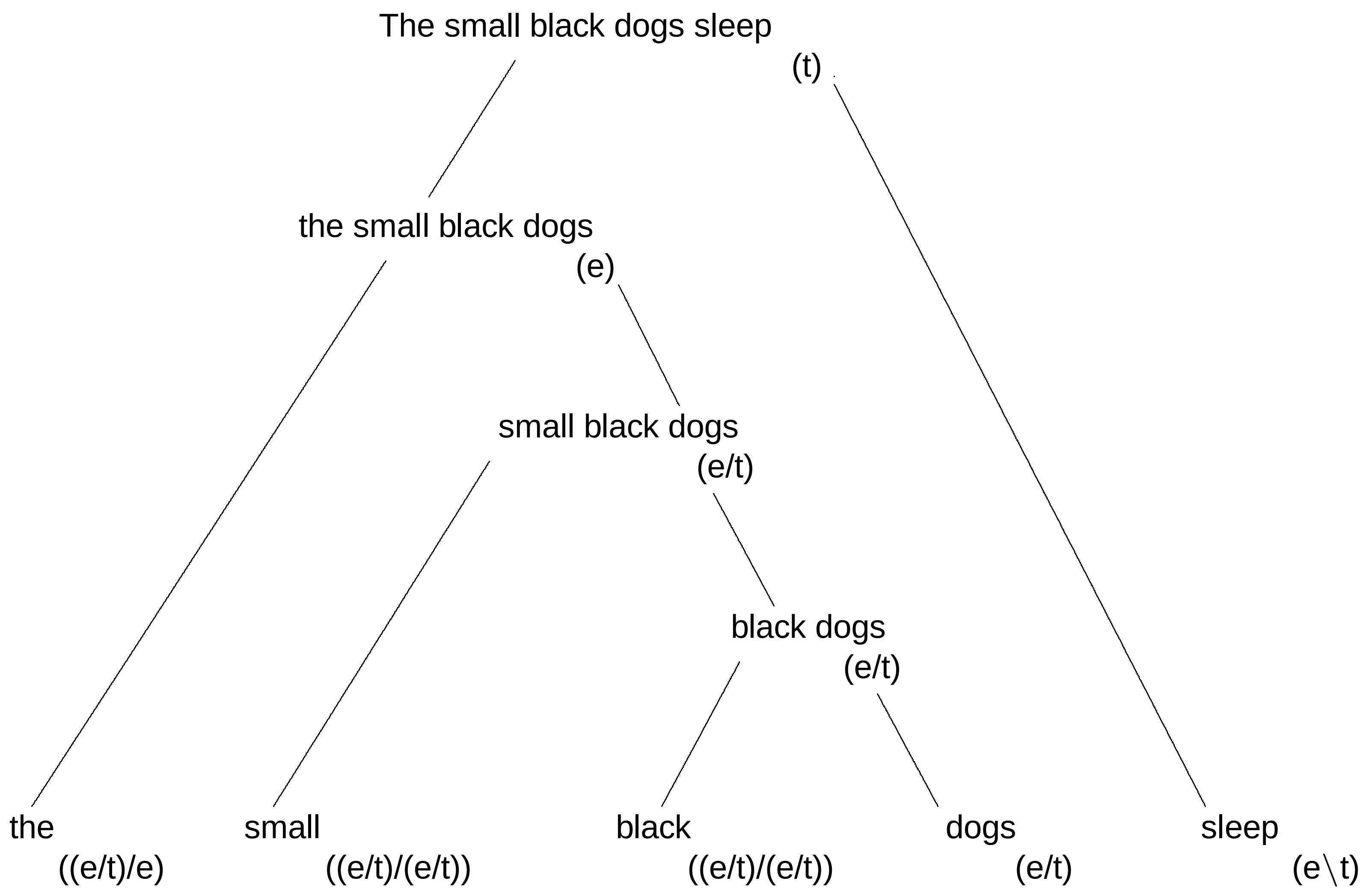
7.5.1 C-grammar for a tiny fragment of English

LX $=_{def}$ $\{W_{(e)} \cup W_{(e\backslash t)}\}$, where
$W_{(e)}$ = {Julia, Peter, Mary, Fritz, Suzy ...}
$W_{(e\backslash t)}$ = {sleeps, laughs, sings ...}
CE $=_{def}$ {(t)}

7.5.2 Simultaneous syntactic and semantic analysis



7.5.3 C-analysis of a natural language sentence



7.5.4 C-grammar for example 7.5.3

LX $=_{def}$ {$W_{(e)} \cup W_{(e\backslash t)} \cup W_{(e/t)} \cup W_{((e/t)/(e/t))} \cup W_{((e/t)/t)}$ }, where

$W_{(e)}$ = {Julia, Peter, Mary, Fritz, Suzy . . . }

$W_{(e\backslash t)}$ = {sleeps, laughs, sings . . . }

$W_{(e/t)}$ = {dog, dogs, cat, cats, table, tables . . . }

$W_{((e/t)/(e/t))}$ = {small, black . . . }

$W_{((e/t)/t)}$ = {a, the, every . . . }

CE $=_{def}$ {(t)}

7.5.5 Empirical disadvantages of C-grammar for natural language

- Deriving expressions relative to a C-grammar has the character of problem solving.
- The handling of alternative word orders and agreement phenomena requires an extremely high degree of lexical ambiguities.