Pāṇini’s *Aṣṭādhyāyī* and Linguistic Theory

Brendan S. Gillon

© Springer Science+Business Media B.V. 2007

**Introduction**

The *Aṣṭādhyāyī*, the world’s earliest, extant grammar, is a grammar of the Sanskrit language as spoken by the descendants of the Indo-Aryan tribe living in the northwest of the Indian subcontinent around the 5th century BCE. Of its author, Pāṇini, a speaker of that language, we know almost nothing. The grammar is neither a list of observations about the language, nor is it a descriptive grammar of the kind compiled by modern field linguists; rather, it comprises a set of rules by means of which all and only well-formed Sanskrit sentences are constructed and paired with the situations they express. It is, in short, a generative grammar.

As remarked by Leonard Bloomfield almost a century ago, this work is ‘one of the greatest monuments of human intelligence’ (1933 p.11). It ranks in its sophistication with that of Euclid’s *Elements*. It would be as remise of anyone interested in the philosophy of language or the history of linguistics to know nothing of the *Aṣṭādhyāyī* as it would be of anyone with a serious interest in the history or philosophy of mathematics to know nothing of the *Elements*.

In what follows, I show that *Aṣṭādhyāyī* is indeed a generative grammar. I shall also show that two problems it addressed over 2,000 years ago are problems for which no adequate solution exists even today. The paper comprises three parts: the first provides a brief presentation of what a generative grammar is; the second gives a general idea of the grammar, showing that it is
indeed a generative one; the last presents two problems: the problem of the semantics of nominal compounds and that of implicit arguments.

**Generative Grammars**

The notion of ‘generative’, as used in the expression ‘generative grammar’, is an algebraic one. One set of elements generates another just in case the closure of the former under some specified set of operations yields the latter. For example, the set of positive integers is generated from the set of prime numbers together with the number one under the operation of multiplication. The reason for this is simple: every positive integer greater than one is either itself a prime number or the product of prime numbers. Thus, for example, seventeen (17) is a prime number and 2310 is the product of two times three five seven times eleven (2310 = 2 × 3 × 5 × 7 × 11). The same set can be generated from the set containing just the number 1 under the operation of immediate succession. Seventeen, for example, is got from one after 16 applications of the operation of immediate succession; and 2310 is got from one after 2309 applications of the operation of immediate succession.

The syntax of a generative grammar of a language comprises a set of basic elements, the morphemes of a language, and a set of operations, the grammatical rules, which, when applied recursively to the set of basic elements, yields all and only the language’s well-formed expressions.

To see the basic idea of a generative syntax, consider the set of expressions $SL$. It comprises strings of one or more tokens of the letters in $\{A, B, C, D\}$, which we shall call $L$. Members of $SL$ include $A, B, C, D, DA, ABC, BCAAD$, etc. It includes neither $BE$ nor $BCCFAA$. We can give a generative specification of $SL$ by the following recursive definition:

1. If $x \in L$, then $x \in SL$;
2. If $y \in L$ and $z \in SL$, then $yz \in SL$;
3. Nothing else is a member of $SL$.

Thus, the set $SL$ is said to be generated from the set $L$ by the rules in (1).

Below a tree is used to show how the expression $BACD$ is obtained from the clauses in (1). The leaves of the tree are labelled with various elements of $L$. This labelling corresponds to the application of clause (1.1). Each convergence of two branches corresponds to an application of clause (1.2). The fact that the bottom of the tree is labelled with the expression $BACD$ shows that it is an expression of $SL$, as defined in (1).