From: International Encyclopedia of Linguistics (1992) Oxford University Press.
William Bright (ed)

NLP: **Grammar Formalisms.** Grammar formalisms are artificial languages whose purpose is to characterize precisely other artificial or natural languages. They are used descriptively in linguistics for characterizing—and thereby, under certain assumptions, explaining the properties of—fragments of particular natural languages, and normatively in computer science for characterizing—and thereby providing a reference standard for—computer languages, especially computer programming languages. Their application to natural-language processing (NLP) combines aspects of both of these uses, grammars for this latter purpose requiring descriptive accuracy yet serving as a precise norm for computers to adhere to.

Formalisms for NLP can be evaluated according to the following criteria:

- **Linguistic felicity:** The degree to which descriptions can be directly (or indirectly) stated as linguists tend to state them.
- **Expressiveness:** Which class of analyses can be stated at all in the formalism.
- **Computational effectiveness:** Whether there exist computational devices for interpreting the grammars expressed in the formalism, and, if such devices do exist, what computational limitations inhere in them. (See entries for **Parsing** and **Computational complexity**.)

The tradeoffs among these criteria typically preclude them from coexisting optimally within any single formalism. For instance, as the power of the formalism grows, sufficiently efficient algorithms for parsing may no longer exist. Alternatively, as a formalism becomes oriented toward the style of analysis of one particular linguistic theory, the class of expressible analyses may diminish. Furthermore, the use to which the formalism is to be put may argue for quite different weightings of these criteria. For instance, linguistic theories typically minimize expressiveness as a means of explaining learnability; NLP formalisms, on the other hand, may maximize expressiveness of formalisms as a means of gaining flexibility.

Grammar formalisms can characterize languages **procedurally**, by providing a method or algorithm for generating all of the strings of the language, or **declaratively**, by providing a direct description of the language elements. Further, a procedural description can be **synthetic** in that the algorithm specifies how the strings of the language can be constructed, or **analytic** if the algorithm specifies how the strings can be recognized.

The Rewriting Hierarchy

Early research on formal language theory, motivated by problems in linguistics, led to the codification of a series of four language classes, each a strict subset of the previous. Each member of the so-called **Chomsky hierarchy** of four language classes corresponds to a different grammatical formalism, in particular a rewriting system. A **rewriting system** is a grammar formalism that defines languages procedurally (in fact, synthetically) by giving rules for substituting one substring for another in a given string. (See entry for **Rewriting hierarchy**.) Of particular interest is the formalism corresponding to Type 2 rewrite rules, the so-called context-free grammars (CFG), as the CFG formalism was developed as a codification of the type of immediate-constituent analysis found in structural linguistics, and as CF grammars have served as components of later formalisms in linguistics (e.g., the base component of a transformational grammar) and NLP (as we shall see).

The languages defined by these grammars are specified procedurally as the output of a string-generating algorithm. However, the more

restrictive (non-type-0) classes have an alternative declarative interpretation under which the rules are viewed as admissibility conditions for sets of nodes in parse trees. This distinction in the interpretation method for grammars can have mathematical ramifications—Peters and Ritchie demonstrated that Type 1 grammars under the admissibility interpretation can express only Type 2 languages (see entry on **Computational complexity**)—and therefore highlights the importance of distinguishing procedural and declarative formalisms.

Greibach (1981) provides a detailed history of the origins of formal language theory, including the development of the rewriting hierarchy and categorial grammars and their relationship to mathematical and computational linguistics.

Grammar Formalisms in Linguistic Theory

After the pioneering work on the rewriting hierarchy and its relation to classes of automata, the design of grammatical formalisms diverged in the fields of computer science and linguistics and, within computer science, between those engaged in the study of formal language theory and those interested in natural-language processing. Nonetheless, many NLP researchers have attempted to use formalisms from linguistic theory directly for the purpose of automatic processing of natural language.

Chomsky's immediate use of his results from formal language theory was to motivate the necessity for the more powerful formalism implicit in standard theory **transformational grammar** (q.v.), which augments a context-free or context-sensitive string-rewriting component with tree-rewriting rules of a certain sort, called transformations. Many NLP projects have attempted to make use of transformational grammars (e.g., Zwicky, *et al.*, 1965), in spite of the discovered difficulty of "reversing transformations" (Petrick, 1965). The theories of **Generalized Phrase-Structure Grammar** (GPSG) and **Lexical-Functional Grammar** (LFG) and **Head-Driven Phrase-Structure Grammar** (HPSG) incorporate grammar formalisms without transformations; they have also been used for NLP tasks (see, for instance, the references provided by Gazdar, 1984).

Not all linguistic theories incorporate a grammar formalism. **Relational Grammar** and **Government-Binding Theory** (GB) are examples of theories which state both language-particular and -universal statements in a natural as opposed to artificial metalanguage. However, recent work has attempted to utilize the principles of GB theory, appropriately cast, to aid in NLP (see e.g., Stabler, 1987 and works cited therein).

Grammar Formalisms in NLP

In parallel with the development of grammar formalisms in linguistic theory, several formalisms have been designed specifically for the task of natural-language processing. As early as 1958, Yngve developed the COMIT programming language (Yngve, 1958) as a powerful procedural analytic system for linguists to use in describing natural-language structure to a computer. The language was the precursor of string-processing computer languages such as SNOBOL, whose use went well beyond NLP tasks.

The **Augmented Transition Network** formalism (see **Parsing** entry for an example) is a procedural analytic formalism based on the augmentation of pushdown automata (the automaton equivalent of context-free grammars) with a procedural language for assigning values to registers.

This theme of augmenting context-free grammars

with registers or features that can take on values permeates the computer-science work on grammar formalisms both for NLP and for computer-language specification, as well as the linguistic formalisms such as GPSG, LFG, and HPSG. Other procedural NLP formalisms participating in this theme include the Dialogic formalism (Robinson, 1982) and used to build the large Diagram grammar of English, the Linguistic String Project system (Sager, 1981), Wilensky's phrasal grammar (Wilensky and Arens, 1980) and the Lingol formalism (Pratt, 1973) (although in this last case the augmentations are restricted to being used for preferences among rules and for syntax-directed translation into a semantic representation). For example, the Dialogic formalism allows context-free rules to be augmented with *constructors* that encode in an extension of the LISP programming language constraints on features associated with the constituents. The sentence formation rule found in the **Parsing** entry might be roughly recast in Dialogic as:

```
(S1   S = NP VP;
      CONSTRUCTOR
         (PROGN
            (OR (AGREE NBR NP VP)
               (F.REJECT 'F.NUMAGR))
            (OR (AGREE PER NP VP)
               (F.REJECT 'F.PERAGR))
            (OR (@ VP TENSED)
               (F.REJECT 'F.UNTENSEDS))
            (@SET TYPE 'DECLARATIVE)))
```

Another important strain of NLP research concerns the development of highly procedurally-oriented formalisms that are intended specifically to allow the stating of performance, rather than competence, models of a language. Marcus's PIDGIN formalism (Marcus, 1980) is primary among these efforts. Grammars in the formalism are comprised of instructions as to what actions (such as node creation or attachment) to perform on particular parsing data structures in particular situations. Marcus used the formalism to define a grammar modeling, among other phenomena, the psycholinguistic phenomenon of garden-path sentences.

The move towards context-free-based formalisms augmented with nonterminals structured as sets of features and values led in the mid 1980s to the development of a set of declarative (as opposed to the previously discussed procedural) formalisms, the so-called complex-feature-based or unification-based formalisms, which include the linguistic formalisms used in GPSG, LFG, and HPSG, as well as the NLP formalisms Functional Unification Grammar and PATR-II. These formalisms rely on structured information associated with phrases (as in categorial grammars) as opposed to unstructured symbols (as in context-free grammars). In particular, this set of structures can be thought of as the union of a primitive set of atomic structures and the set of finite functions whose domain is a set of features and whose range is the set of structures itself. These structures—called variously f-structures (in LFG), feature bundles, feature matrices, or categories (in GPSG), attribute-value matrices (in HPSG), or feature structures—can be modeled mathematically in various ways, as finite functions, graph structures of a certain sort, or finite automata. Typically, formalisms use these structures by allowing grammars to specify constraints on the structures, for instance, as equations. The PATR-II formalism, perhaps the simplest member of the class, allows rules to be augmented with equational constraints. For instance, a rule for sentence formation with an added constraint of agreement of subject and predicate could be stated as:

```
S → NP VP
  <NP agreement> = <VP agreement>
  <VP tensed> = yes
  <S type> = declarative
```

Closely related to the unification-based grammar formalisms are the logic grammar formalisms such as Definite-Clause Grammars (DCG) (Pereira and

Warren, 1980), Extraposition Grammars, Slot Grammars, Gapping Grammars, and others. In these, the structured nonterminals are typically first-order terms, and the effect of equational constraints is achieved by the sharing of variables. For example, a DCG rule for sentence formation might be

```
s(Agr, decl) -> np(Agr), vp(Agr, yes).
```

where the sentence term arguments correspond to agreement and type, respectively, the NP argument to agreement alone, and the VP arguments to agreement and tensedness. Shieber (1986a) gives references for unification-based and logic-grammar formalisms.

Comparing Formalisms

In addition to true expressive differences among grammatical formalisms, notational differences are important, as they determine the ease of use of the formalism and may push one towards certain styles of analysis. However, the import of such notational distinctions should not overwhelm that of the analyses. One way to understand the relationships among notationally distinct formalisms is to map from one notation to another. A mapping is revealing, however, only if it preserves some important aspect of the grammars, ideally their interpretation. To rigorously demonstrate such semantic invariance, the input and output formalisms must have explicit semantics defining the interpretations of grammars, an area of research that is only beginning to receive attention (Pereira and Shieber, 1984). Nonetheless, at least informal work on notational reductions has begun—for instance, an attempt to reduce the GPSG formalism to PATR-II (Shieber, 1986b). As a greater understanding of the relationships among grammar formalisms is achieved, it will be possible to place them in their rightful role as ancillary devices, secondary to the linguistic analyses that they are used to encode.

References

Gazdar, Gerald. 1984. Recent computer implementations of phrase-structure grammar. *Computational Linguistics 10*:3–4, 212–214 .

Greibach, Sheila A. 1981. Formal languages: origins and directions. *Annals of the History of Computing* 3:1, 14-41.

Marcus, Mitchell P., 1980. *A Theory of Syntactic Recognition for Natural Language.* Cambridge, Massachusetts: MIT Press.

Pereira, Fernando C. N. and Stuart M. Shieber. 1984. The semantics of grammar formalisms seen as computer languages. In *Proceedings of the 10th International Conference on Computational Linguistics*,Stanford University, Stanford, California, 123–129.

Pereira, Fernando C. N. and David H. D. Warren. 1980. Definite-clause grammars for natural language analysis—a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence 13*:3, 231–278.

Petrick, Stanley R. 1965. *A Recognition Procedure for Transformational Grammars.* Ph.D. thesis, Massachusetts Institute of Technology.

Pratt, Vaughan R., 1973. A linguistics oriented programming language. *Proc of the Third Int'l. Joint Conf. on Artificial Intelligence.* Stanford University, Stanford, California, 372-381.

Robinson, Jane J. 1982. DIAGRAM: A grammar for dialogues. *Communications of the ACM* 25:1, 27–47.

Sager, Naomi. 1981. *Natural Language Information Processing: A Computer Grammar of English and its Applications.* Reading, Massachusetts: Addison-Wesley.

Shieber, S. M., 1986. *An Introduction to Unification-Based Approaches to Grammar.* CSLI Lecture Note Series Number 4. Stanford, California: Center for the Study of Language and Information.

Shieber, Stuart M. 1986. A simple reconstruction of GPSG. In *Proceedings of the 11th International Conference on Computational Linguistics*, University of Bonn, Bonn, West Germany, 211–216.

Stabler, Edward P., Jr., 1987. Restricting logic grammars with government-binding theory. *Computational Linguistics* 13:1–2, 1–10.

Wilensky, Robert, and Yigal Arens. 1980. PHRAN: A knowledge-based approach to natural language analysis. Memorandum No. UCB/ERL M80/34, Electronics Research Laboratory, University of California, Berkeley, California.

Yngve, V.H., 1958. A programming language for mechanical translation. *Mechanical Translation* 5:1, 25-41.

Zwicky, Arnold M., Joyce Friedman, B. C. Hall, and Donald E. Walker. 1965. The Mitre syntactic analysis procedure for transformational grammars. In *Proceedings of the AFIPS 1965 Fall Joint Computer Conference*, volume 27, part 1, 317–326. New York: Spartan Books.