

Reconciling Expressivity and Tractability: Finite State Methods in Natural Language Analysis

AAAS Symposium: Monday, February 18, 2002, 9:30-12:30

This symposium is about the consequences for natural language structure and natural language processing of an intrinsic tension in logical and computational systems—the trade-off between expressive power and computational complexity. We attempt to describe here in a qualitative, non-technical way how this trade-off arises and its bearing on the scientific study of natural language and the computational modeling of natural language processing and natural language databases. We then list the talks to be presented in the symposium, with links to on-line resources which offer more detailed information about the speakers, to abstracts, and to a list of references.

The issues we are concerned with arose in the mathematical and computation study of formal languages (which are 'formal' in two senses: they are rigorously defined and they are uninterpreted). A formal language depends on a finite vocabulary Σ —a set of atomic symbols. For example, we can take Σ to be the set $\{a, b\}$ containing just the two distinct elements a and b or we can take Σ to be the set of words of some natural languages (assuming that this can be fixed in some determinate way and ignoring any possible internal word structure). We assume in addition that we may make complex symbols from the atomic elements of Σ by *concatenation* (an associative binary operation). Taking Σ to be $\{a, b\}$, this yields (among many others) $a, b, aa, ab, ba, bb, aaa$ (not in any particular order). The set of *all* such sequences of elements of Σ is called Σ^* . A *formal language over Σ* is a subset of Σ^* , that is, a collection of sequences of elements of Σ .

Two obvious extremes are the empty set \emptyset , which contains no elements of Σ^* (not even the empty sequence ϵ) and Σ^* itself. In between are all the finite subsets of Σ^* and the (countably) infinite subsets of Σ^* . A finite subset of Σ^* can in principle be specified by a list. To describe some of the infinite subsets, we use a convenient exponential notation: $a^1 = a$ and $a^{k+1} = a^k \cdot a$ (for k a positive integer). Thus, $a^2 = aa$, $a^3 = aaa$, and $\{a^n \mid n \in N^+\}$ is the set of all finite sequences of the symbol a . We conventionally denote this set simply by a^n (taking for granted that the variable n ranges over the positive natural numbers N). Using this notation, we have such infinite languages as a^n (all finite sequences of a), $(ab)^n$ (all finite sequences of ab), $a^n b^m$ (all sequences consisting of some a 's followed by some b 's), $a^n b^n$ (all sequences consisting of some a 's followed by exactly the same number of b 's).

The descriptive advantages of the set-theoretical notation obscure basic computational questions: what resources do procedures that characterize these formal languages require? are these procedures all of the same kind? Two basic paradigms for the general investigation of these questions arose quite early in computer science: grammatical rewriting systems and automata. Both give rise to classes of languages which can be strictly ranked in terms of expressive power and which share significant common structure. We sketch this ordering—the *Chomsky hierarchy*—qualitatively in the table below.

The center column of the table specifies a characteristic language or class of languages. The left column describes the hierarchy from the point of characteristic restrictions on production rules in grammatical rewriting systems and the third column does the same for automata. We include information about rewriting systems because the ranking of production rule types shows an intuitive increase in complexity as one moves from the bottom of the table to the top. Rewriting systems contain *production rules* involving both the finite vocabulary of terminal elements Σ and a finite set \mathcal{C} of *non-terminal elements* or categories (disjoint from Σ). Using lower-case letters for elements of Σ , upper-case letters for categories, and lower-case Greek letters for sequences of elements drawn from Σ and \mathcal{C} . The general form of a production is $\alpha A \beta \rightarrow \gamma$, but each of the special cases illustrated in the table below imposes restrictions on the formal languages that can be characterized by productions of this form. In the hierarchy of automata, the main role is played by restrictions on memory and ‘scratch space’. (Further details can be found in the references at the URL listed below.)

production rule type	formal language exemplar(s)	corresponding automata
$\alpha A \beta \rightarrow \gamma$ (unrestricted)	recursively-enumerable languages	Turing machines
$\alpha A \beta \rightarrow \alpha \gamma \beta$ (context-sensitive)	$a^n b^n c^n$	linear bounded automata
$A \rightarrow \gamma$ (context-free)	$a^n b^n$	push-down automata
$A \rightarrow Ba$ or $A \rightarrow a$ (left-linear)	$(ab)^n$	finite automata (regular languages)
$A \rightarrow aB$ or $A \rightarrow a$ (right-linear)	$(ab)^n$	finite automata (regular languages)

Note: the two lines before the final line—representing left-linear and right-linear grammars—are equivalent. Otherwise, as one ascends the steps of the hierarchy, the computational properties of the production rules and the automata increase in expressive power, meaning vthat it is possible to make finer discriminations. For example, the languages $a^n b^m$ and $a^n b^n$ both involve sequences containing only a and b , with all the a preceding all the b ’s, but $a^n b^n$ imposes an additional restriction which is beyond the discriminating power of left-linear grammars, right-linear grammars, or finite-state automata.

Increasing expressive power comes at a significant cost. At the bottom of the hierarchy, determining whether or not a candidate expression e of length k is a member of a regular language can be done in time that does not significantly differ from k . The best algorithms for parsing context-free languages require in the worst case a time proportional to the *cube* of the length of the input (though for many special kinds of context-free languages, performance can be significantly improved). Even for some of the tamer sub-classes of context-sensitive languages (the *mildly context-sensitive languages*), parsing algorithms can require time proportional to n^6 for input of length n . And as we ascend higher in the hierarchy, things do not improve.

When we consider natural languages against this perspective of mathematical and computational results, we are faced with a puzzle: natural languages are reasonably expressive (including constructions which cannot be characterized beneath the mildly context sensitive level of the Chomsky hierarchy), yet they are parsed and processed in very close to real time. This combination of properties raises interesting scientific and engineering challenges. The work reported in this symposium addresses these challenges in a variety of ways. One way is to show that phenomena thought to be characterizable at one level in the Chomsky hierarchy can actually be adequately characterized at a lower level of expressive power. A second way is to extend the computational techniques of a given level to an interesting and significant sub-class of a higher level. A third way is to approximate the properties of a higher class while keeping strictly within a lower class. The presentations in this symposium will exemplify these methods across a spectrum of linguistic phenomena. The individual talks and speakers are listed below. Abstracts of the talks and references may be found at: <http://www.cs.indiana.edu/fgmol/>

- *Finite-State Methods and Their Role in Computational Linguistics*
Aravind Joshi, University of Pennsylvania
<http://www.cis.upenn.edu/~joshi/home.html>
- *Regular Relations and Morphological Analysis*
Ron Kaplan, Xerox PARC
<http://www.parc.xerox.com/istl/members/kaplan/>
- *Linguistically-Motivated Extensions of Finite-State Methods*
Lauri Karttunen, Xerox PARC
<http://www.parc.xerox.com/istl/members/karttune/>
- *Finite-State Aspects of Annotation Graphs*
Steven Bird, University of Pennsylvania
<http://www ldc.upenn.edu/sb/>
- *The Linguistic Significance of Finite-State Techniques*
Richard Sproat, AT&T Labs
<http://www.research.att.com/~rws>