Book Reviews

Towards a Theory of Cognition and Computing

J. Gerard Wolff
(University of Wales, Bangor)

Chichester, U.K.: Ellis Horwood
Limited (Prentice-Hall), 1991, 194 pp.
Hardbound, ISBN 0-13-925025-5, $52.00

Reviewed by
Philip Swann
University of Geneva

This expensive volume comprises a selection of the author's published papers (Wolff 1982, 1988, 1989, 1990; Wolff and Chipperfield 1990). There is some new material, in the form of a general introduction, a paragraph or so of preface to each paper, and a brief (11-page) additional chapter. None of this supplementary material adds anything of significance to the original papers. Four earlier papers (Wolff 1975, 1976, 1977, 1980) are not included, although the work reported in them is repeatedly referred to. The addition of one or more of these earlier papers would have helped the reader considerably. The papers in the book have been reset, but otherwise reprinted verbatim. As the reader also has to struggle with the author's frequent changes of direction—and with his casual, at times slapdash, expository style—the resulting collage does not make for easy or enjoyable reading. Among computational linguists, probably only those with a special interest in grammar induction will want to know about Wolff's work, and they can limit their attention to Wolff 1982 and 1988.

Wolff started his career as a psychologist with an interest in the computational modeling of first-language acquisition. He then spent four years with a commercial software house before becoming a lecturer in computer systems engineering at the University of Wales. As the title of this book suggests, his goal is to integrate this varied work experience into a general theory that could be considered a contribution to cognitive science. The actual content of the book, however, does not live up to the ambitious title. The first half (Wolff 1982, 1988) reports on the development of a grammar inference program (SNPR) based on statistics, which went through numerous versions over a period of some ten years. The program is informally analyzed in terms of data compression and offered, speculatively, as a model of first-language acquisition and concept formation. In the second half of the book, Wolff generalizes these ideas and methods in the form of a computational notation (SP) analogous to a simplified Prolog. It is this notation that is claimed to be a new language for a unified "theory of cognition and computing" (as well as a potential solution to all the problems of software engineering): but neither real evidence nor a convincing argument is offered in support. The rest of this review will therefore focus on the first half of the book, which contains all that is likely to be of interest to computational linguists.

The SNPR program was built on top of an earlier algorithm (Wolff 1975). Taking an input text without spacing or punctuation, that program (MK10) builds up a grammar as a list of hierarchically bracketed sub-strings. It uses the following algorithm:

- Initialize the grammar to the alphabet used by the text.
- Repeat:

  1. Parse the whole text from left to right, at each point matching against the largest possible grammar element;
  2. calculate the frequency of co-occurrence for all pairs of elements in the parsed text;
  3. add the most frequent pair to the grammar as a new element (choose at random between ties).

For example, starting with the text *abcabc* and the grammar initialized to $\{a, b, c\}$, the first iteration would parse the text as *abcabc* and then add the new element *(ab)* to the grammar. The second iteration would parse the text as *(ab)c(ab)c* and add the element *((ab)(c))* to the grammar. The third iteration would create a full parse and add it to the grammar as *((ab)(c))((ab)(c))*. MK10 was remarkably successful at its assigned task of segmenting text into words, both on artificial texts and natural language (Wolff 1977). If the program is allowed to continue iterating over the segmented text, it will build higher-level constituent structures; but, as might be expected, these do not correspond to a realistic grammatical parse. Interestingly, the program did work quite well on a text transcribed as a sequence of word classes.

The SNPR program (Wolff 1982) adds three new operations to MK10 to be performed on each iteration:

1. "Folding" builds disjunctive rules to capture alternation of elements in a fixed context. For example, if the grammar contains *apb*, *aqb*, and *arb*, then a new element is built, $(X \rightarrow p \mid q \mid r)$, and the three original elements are merged to aXb.

2. "Generalization" finds other occurrences of the elements of the newly created disjunction and replaces them with the nonterminal symbol. So, if xyp is in the grammar, it will be replaced by xyX.

3. "Rebuilding" is used to remove unwanted generalizations created in the previous step. To do this, all instances of a new disjunction in each of its contexts are checked. If the disjunction fails to use all of the constituents in a context, then it is rebuilt for that context without the unused constituent(s). Continuing the example: If cdX always rewrote only as cdp or cdq, then X would be rebuilt as $(Y \rightarrow p \mid q)$, and all occurrences in the grammar of cdX would be replaced by cdY.

Wolff does not give a formal statement or analysis of the full SNPR algorithm, and his description of its implementation is impressionistic and very hard to follow. This is probably not a serious omission, since any implementation of such a complex set of operations is going to be very costly indeed. The program was tested on several miniature finite state grammars. It was reasonably successful with only this one:

$$\begin{aligned}
S &\rightarrow (1)(2)(3) \mid (4)(5)(6)\\
1 &\rightarrow \textit{david} \mid \textit{john}\\
2 &\rightarrow \textit{loves} \mid \textit{hated}\\
3 &\rightarrow \textit{mary} \mid \textit{susan}\\
4 &\rightarrow \textit{we} \mid \textit{you}\\
5 &\rightarrow \textit{walk} \mid \textit{run}\\
6 &\rightarrow \textit{fast} \mid \textit{slowly}
\end{aligned}$$

The sample text for the test was a 2,500-character string, built presumably (no details are given) by stratified random selection of the rules. As might be expected, SNPR does not work on natural language texts.

Since the point of constructing SNPR was to model child language acquisition, Wolff devotes a great deal of space to claiming that this goal was at least partially achieved. To this end, he develops an account of grammar induction as data compression. The "efficiency" of a grammar is defined as its compression capacity divided by its size, where "compression capacity" is the ratio of encoded text to its raw form. He can then claim that:

- efficient data compression is a natural property of biological systems, including children learning their mother tongue;
- children should at all times try to maximize the efficiency of the grammar they are developing;
- SNPR does just that and, therefore, is a valid model of first-language acquisition.

While the last claim would have been easy to test empirically, this was never actually done. It would have been more difficult, but presumably possible, to prove it formally, but this was not done either. As a result, the 35 (very redundant) pages devoted to the topic are largely a waste of the reader's time.

Wolff 1988 also makes a valiant but largely unconvincing attempt to relate SNPR to the literature on child language, with the aim of justifying his belief that the algorithm captures some of the essentials of real language learning. As he notes, his approach is rooted in the pre-Chomsky era of distributional analysis and associationist psychology. The learner (SNPR) is a formal device abstracting patterns according to the frequency distribution of symbols, without the support of semantics, negative evidence, or correction. In the case of word segmentation, the approach works because of the well-known statistical properties of text: words get repeated far more often than do strings of words. The induction of syntactic rules is a much tougher problem, and it is not at all clear that SNPR actually tackles it, since the target text and grammar seem to be closely matched to the learning mechanism.

In conclusion, then, Wolff's early technical work represented a useful contribution to research on grammar induction. It is a pity that he was unable to provide a proper formulation and analysis of SNPR; a good example of what can be achieved in this area is offered by the work of Angluin (1982) and Berwick and Pilato (1987) on the induction of automata. A similar lack of rigor occurs in Wolff's more general theorizing. He brings together interesting ideas from psychology, statistics, and computing in a process of synthesis by loose analogy. In the later part of the book the result is almost comic: "In SP, the boundary between 'knowledge engineering' and other kinds of information engineering breaks down. In SP there is the potential for full integration of artificial intelligence, software engineering, and other aspects of computing—with Shannon's information theory as the unifying framework. SP also offers a bridge between 'connectionist' and 'symbolic' views of computing" (page 101). One is left with the feeling that the drift of the book is away from, rather than toward, a general theory of cognition.

References

Angluin, Dana (1982). "Inference of reversible languages." *Journal of the Association for Computing Machinery*, **29**(3), 741–765.

Berwick, Robert C., and Pilato, Sam (1987). "Learning syntax by automata induction." *Machine Learning*, **2**, 9–38.

Wolff, J. Gerard (1975). "An algorithm for the segmentation of an artificial language analogue." *British Journal of Psychology*, **66**, 79–90.

Wolff, J. Gerard (1976). "Frequency, conceptual structure and pattern recognition." *British Journal of Psychology*, **67**, 377–390.

Wolff, J. Gerard (1977). "The discovery of segments in natural language." *British Journal of Psychology*, **68**, 97–106.

Wolff, J. Gerard (1980). "Language acquisition and the discovery of phrase structure." *Language and Speech*, **23**, 255–269.

Wolff, J. Gerard (1982). "Language acquisition, data compression and generalization." *Language and Communication*, **2**(1), 57–89.

Wolff, J. Gerard (1988). "Learning syntax and meanings through optimization and distributional analysis." In *Categories and processes in language acquisition*, edited by Y. Levy, I. M. Schlesinger, and M. D. S. Braine, 179–215. Lawrence Erlbaum.

Wolff, J. Gerard (1989). "The management of risk in system development: 'Project SP' and the 'new spiral model.' " *Software Engineering Journal*, **4**(3), 134–142.

Wolff, J. Gerard (1990). "Simplicity and power: Some unifying ideas in computing." *Computer Journal*, **33**(6), 518–534.

Wolff, J. Gerard, and Chipperfield, Andrew J. (1990). "Unifying computing: Inductive learning and logic." In *Research and development in expert systems VII*, edited by T. R. Addis and R. M. Muir, 263–276. Cambridge University Press.

Philip Swann is a Research Fellow in the Department of Psychology and Education at the University of Geneva. He holds a Ph.D. in Computer-Assisted Learning from the Open University. His main research interest is in the psycholinguistics of language learning. Swann's address is: FPSE, University of Geneva, 1211 Geneva 4, Switzerland. e-mail: swann@divsun.unige.ch