The systematic construction of Earley Parsers: Application to the production of an $O(n^6)$ Earley Parser for Tree Adjoining Grammars

Bernard Lang
INRIA
B.P. 105
78153 Le Chesnay, CEDEX, France
lang@margaux.inria.fr

Logic Programming languages (Prolog) were originally introduced as an extension of context-free (CF) languages. Conversely CF grammars may be seen as logic programs (or Horn clauses) where the predicates are the CF grammar symbols, and where these predicates have arguments corresponding to the boundaries of the input string fragments they derive into.

Earley's parsing algorithm can be generalized to Horn Clauses as a dynamic programming evaluation technique. Keeping in mind the relation between Horn clauses and CF grammars, we suggest encoding similarly in Horn Clauses other syntactic formalisms so as to take advantage of this generalisation of Earley's algorithm to obtain for free efficient parsers for these encoded formalisms.

As an example, we consider the problem of TAG parsing. We show that any TAG can be encoded into a logic program for which there is an evaluation in time $O(n^6)$. We show on this example how the general dynamic programming procedure can be adapted to conform the constraint that sentences be parsed from left to right, even in the presence of interleaved constituents as is the case for TAGs.

The Valid Prefix Property and Parsing Tree Adjoining Grammars

Yves Schabes
Department of Computer and Information Science
R-555 Moore School
University of Philadelphia
220 South Street 33rd Street
Philadelphia, PA 19104-6389, USA
schabes@linc.cis.upenn.edu

The valid prefix property (VPP), capability of a left to right parser to detect errors as soon as possible, is often unobserved in parsing CFGs. Earley's parser for CFG maintains the VPP and obtains a worst case complexity ($O(n^3)$) as good as parsers that do not maintain VPP (as the CKY parser). Contrary of CFGs, maintaining the valid prefix property for TAGs seems costly.

The aim of talk was to informaly explain why the VPP for TAGs seems expensive to maintain and also to introduce a new Earley-style parser for TAGs which has $O(n^6)$ worst case time complexity. The new parser does not maintain VPP but it can behave in linear time on some grammars, in $O(G^2n^4)$ worst time for unambiguous TAGs and in general in $O(G^2n^6)$-time in the worst case. An earlier Earley-type parser that we proposed in 1988 maintains the VPP but at its cost of its worst case complexity ($O(G^2n^9)$-time). To our knowledge, it is the only known polynomial-time general TAG parser that maintains the VPP. Both Earley-style parsers for TAGs use top-down filtering and therefore their behaviors are in practice superior to pure bottom-up parsers (as Joshi's and Vijay-Shanker's adaptation of CKY algorithm to TAG).

In practice, the importance of the VPP varies from grammars and is currently being evaluated on natural language TAG grammars for English and French.

Parallel TAG Parsing on the Connection Machine

Michael Palis, David Wei
Department of Computer and Information Science
School of Engineering and Applied Sciences
R-555 Moore School
University of Philadelphia
220 South Street 33rd Street
Philadelphia, PA 19104-6389, USA
palis@linc.cis.upenn.edu

We present a parallel parsing algorithm for Tree Adjoining Grammars (TAGs) and its implementation on the Connection Machine (CM). The CM TAG parser is designed to handle TAGs of arbitrary size without significant decrease in performance. Specifically, the expected run-time of the parallel algorithm is logarithmic in the grammar size (as opposed to quadratic in a serial implementation).

The CM TAG parser is an emulation of the CRCW PRAM algorithm. The PRAM algorithm is characterized by frequent communication between processors via the shared memory. Moreover, the pattern of inter-processor communication does not have the regular structure often found in many parallel numerical algorithms. Because the CM has a distributed memory, the emulation of the PRAM algorithm can only be realized by explicit message-passing, albeit between non-adjacent processors. Unfortunately, routing messages between non-adjacent processors is time-consuming on the CM. The CM uses a deterministic oblivious routing strategy, which, in the worst-case, can introduce $\sqrt{p}$ delay per emulated step, where p is the number of processors used.

To obtain a more efficient emulation, we employ randomization: i.e., grammar nodes of the TAG are mapped randomly to corresponding CM processors. In theory, this reduces the delay per emulated step to $O(log(p))$ with high probability. In practice, we use randomization as part of a pre-processing step: given a fixed TAG, we generate several random mappings of the TAG to the CM, then choose the most efficient mapping. The most efficient mapping is obtained experimentally by running