TREE-STRUCTURED CHART PARSING

Paul W. Placeway
Language Technologies Institute, School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213 USA
pwp@cs.cmu.edu

Abstract

We investigate a method of improving the memory efficiency of a chart parser. Specifically, we propose a technique to reduce the number of active arcs created in the process of parsing. We sketch the differences in the chart algorithm, and provide empirical results that demonstrate the effectiveness of this technique.

One basic shortcoming of a classic chart parser [6, 1, 10] is that it does not make efficient use of its grammar. In grammars used to parse natural languages, there is quite often a substantial amount of redundancy in the prefixes of the rule right-hand-sides. A naïve implementation of a chart parser will not take advantage of this redundancy. In contrast, a shift-reduce parser [4, 9, 2, 10] will often use a grammar that has been optimized to eliminate this redundancy [4]. Since chart parsing and shift-reduce parsing are substantially similar [10], many techniques used in shift-reduce parsing can be applied to a chart parser, including this particular optimization.

Tree-Structured Grammar

Consider a context-free grammar represented as follows: we will refer to a sequence of children (the "right-hand-side" of a rule) as a sequence of *shifts*, and the parent (or "left-hand-side") as the *reduce* operation. We write the rules with the children on the left leading to the parent reduction on the right. Finally, a child symbol can have multiple shifts *and* multiple reductions to its right.

standard representation				*tree representation*			
S	$\Leftarrow$	NP	VP	NP	VP	$\Rightarrow$	S
NP	$\Leftarrow$	NP	PP		$\searrow PP$	$\Rightarrow$	NP

The tree grammar is then constructed in the straight-forward way, compressing the left prefixes of the right-hand-sides as much as possible.

Using the Tree-Structured Grammar

Parsing with the tree grammar is quite straightforward. The principle difference between this algorithm and the classic chart algorithm [1] is that in the classic implementation, extending an active arc results in *one* new arc, whereas when using the tree-grammar, extending an arc may result in *several* new arcs. Finally, since one active arc could spawn multiple arcs, if we must keep track of of children used to create an arc (e.g. to resolve unifications), we must do so using an up-tree [3]. The resulting inner loop remains quite straight-forward:

```
while the agenda is not empty, do:
  let e = next entry from agenda
  add e to chart.
  foreach arc in continued by e, do:
    foreach tnode in arc.tnode.shiftlist, do:
      new-arc = make-arc (e.end, tnode, traceback = (cons(e, arc.traceback))
      arc-add (new-arc)
    foreach rule in arc.tnode.reducelist
      let new-children = reverse (cons(e, arc.traceback))
      new = make-entry (first(new-children).start,
                        e.end, rule.LHS, children = list (new-children))
      agenda-add (new)
```

This technique was evaluated in a chart parser with unification, left-corner and look-ahead constraints, among other features. We used a large-scale English grammar for machine-translation of heavy equipment manuals [7, 5], and a test-set of 2524 sentences (22,558 words). Without any special restrictions, when compared to the naïve implementation, the tree-structured grammar reduced the number of active arcs created by 23%, and when employing full left-corner and look-ahead constraints [11, 8, 4] on the parser, the tree-grammar gave a 40% reduction.

Acknowledgements

This work was supported by the CMU Language Technologies Institute. The author wishes to thank Dr. Eric Nyberg, Dr. Alon Lavie, Dr. Carolyn Rosé, and the anonymous reviewers for comments that improved this paper. A full version of this paper is available as technical report from:
`http://www.cs.cmu.edu/~pwp/papers/tree_parse_tr.ps`

References

[1] James Allen. *Natural Language Understanding.* Benjamin/Cummings, Redwood City, CA, second edition, 1995.

[2] John Andrew Carroll. *Practical Unification-based Parsing of Natural Language.* PhD thesis, University of Cambridge, Computer Laboratory, September 1993.

[3] Thomas H. Cormen, Charles E. Leiserson, and Ronald L Rivest. *Introduction to Algorithms.* McGraw-Hill and MIT Press, Cambridge, MA, 1990.

[4] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Series in Computer Science. Addison-Wesley, Reading, MA, 1979.

[5] Kamprath, Adolphson, Mitamura, and Nyberg. Controlled Language for Multilingual Document Production: Experience with Caterpillar Technical English. In *Proc. Second Int. Workshop on Controlled Language Applications (CLAW '98)*, 1998.

[6] Martin Kay. Algorithm schemata and data structures in syntactic processing. In *Readings in Natural Language Processing.* Morgan Kaufmann, San Mateo, CA, 1986 (1980).

[7] T. Mitamura, E. Nyberg, and J. Carbonell. An efficient interlingua translation system for multi-lingual document production. In *Proc. 3rd Machine Translation Summit*, 1991.

[8] Carolyn P. Rosé and Alon Lavie. LCFLEX: An efficient robust left-corner parser, 1999.

[9] Masaru Tomita. *Efficient Parsing for Natural Language.* Kluwer, Boston, 1986.

[10] G. van Noord, M-J. Nederhof, R. Koeling, and G. Bouma. Conventional Natural Language Processing in the NWO Priority Programme on Language and Speech Technology. Technical report, Rijksuniversiteit Groningen, Vakgroep Alfa-informatica & BCN, March 1996.

[11] Gertjan van Noord. An efficient implementation of the head-corner parser. *Computational Linguistics*, 23(3):425–456, March 1997.