Elizabeth Scott Explained

Parsing from Earley Recognisers

Zoe Wheeler
University of Texas at Austin
zoe.donnellon.wheeler@gmail.com

Walter Xia
University of Texas at Austin
swilery@utexas.edu

Abstract

Earley's Algorithm is able to recognize general context-free grammars in $O(n^3)$, where n is the size of the string to be recognized. However, there are times in which we want more than just a yes or no answer. There are times in which we want an actual parse tree, and for ambiguous grammars, there are times in which we want all possible parse trees. Fortunately, there is a paper by Dr. Elizabeth Scott, [2], that presents a technique to produce a data structure known as a Shared Packed Parse Forest (SPPF), able to represent even an infinite number of parse trees. Unfortunately this paper is poorly written, making it very difficult to understand. Our paper is a re-explanation of Scott's techniques. It is agreed by many that Earley's Algorithm is also difficult to understand. Fortunately, there exists a data structure due to Dr. Gianfranco Bilardi and Dr. Keshav Pingali, [1], known as Grammar Flow Graphs (GFGs) that significantly ease the understanding of the algorithm by reformulating parsing problems as path problems in a graph. Our technique will use GFGs.

Categories and Subject Descriptors F.7.2 [*Semantics and Reasoning*]: Program Reasoning–Parsing

General Terms Context-Free Languages, Cubic Generalized Parsing, Earley Parsing

Keywords Earley Sets, Grammar Flow Graphs, Non-Deterministic Finite Automaton, Shared Packed Parse Forest

1. Introduction

It is important here for us to distinguish between recognisers and parsers for a grammar. Recognizers determine whether or not a string is part of a language defined by a grammar whereas parsers construct parse trees that reveal *how* a string satisfies the syntax dictated by a grammar. For about the past five decades, there already exist general recognizers like Cocke-Younger-Kasami (CYK) and Earley's Algorithms that run cubic relative to the size of the string to be recognized. Alternatively, Generalized LR (GLR) is an algorithm that produces parsers but has the very undesirable property that it is unbounded. Dr. Elizabeth Scott extended the Earley Recogniser into a parser that is able run in cubic space and time, [2]. The challenge was to successfully apply the parser to ambiguous grammars that produces multiple, perhaps infinite, parse trees for a string in the grammar. Note that simply disallowing ambiguous grammars is not a solution since there exists grammars that are intrinsically ambiguous. The solution she used used was a representation known as a Shared Packed Parse Forest (SPPF), which is in essence a Directed Acyclic Graph (DAG).

Earley's Algorithm is a highly complex algorithm. To dramatically simplify its understanding, we view it from the perspective of Grammar Flow Graphs (GFGs) that restructure parsing as finding certain paths within the graph, [1]. For those of you familiar with automata theory, GFGs play the same role for context-free grammars as finite-state automota play for regular grammars. The rest of the paper is organized as follows:

- Section 2 will introduce GFGs
- Section 3 will introduce Earley's Algorithm using GFGs
- Section 4 will introduce SPPFs
- Section 5 will introduce Dr. Scott's Algorithm for producing SPPFs
- Section 6 will discuss our implementation
- Section 7 will discuss our results
- Section 8 will conclude

2. Grammar Flow Graphs

Let us begin with the standard definition of a context-free grammar.

Definition: A *context-free grammar*, CFG, is a tuple (N, T, P, S), where, [1]:

- ▷ N is a finite set of elements called *nonterminals*,
- ▷ T is a finite set of elements called *terminals*,
- ▷ $P \subseteq N \times (N \cup T)^*$ is the set of *productions* that map nonterminals to a sequence of nonterminals or terminals, and
- ▷ $S \in N$ is the unique *start symbol* that appears once on the left-hand side of a single production.

An example of a grammar is the following, where | signifies or:

$$S \longrightarrow N\ t \mid t\ N$$
$$N \longrightarrow t\ t$$

Now we are in a position to introduce the GFG.

Definition: Let $CFG = (N, T, P, S)$ be a context-free grammar and let ϵ denote the empty string. The *grammar flow graph (GFG)* of CFG, $GFG(CFG) = (V(CGF), G(CFG))$, is the smalled directed graph that has the following properties, [1]:

- ▷ For each nonterminal $M \in N$, there exist $\bullet M, M\bullet \in V(CFG)$ called *start nodes* and *end nodes* respectively,

CONF 'yy, Month d–d, 20yy, City, ST, Country.

http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

- ▷ For each production $(M \longrightarrow \epsilon) \in P$, there exists $(M \longrightarrow \bullet) \in V(CFG)$ and $(\bullet M, M \longrightarrow \bullet), (M \longrightarrow \bullet, M\bullet) \in E(CFG)$
- ▷ For each production $(M \longrightarrow q_1 q_2 \dots q_r)$ where $q_i \neq \epsilon$:
  - ◇ $(M \longrightarrow \bullet q_1 q_2 \dots q_r), (M \longrightarrow q_1 \bullet q_2 ... q_r), \dots, (M \longrightarrow q_1 q_2 ... q_r \bullet) \in V(CFG)$, where the first node is called an *entry node* and the last node is called an *exit node*,
  - ◇ $(\bullet M, M \longrightarrow \bullet q_1 q_2 \dots q_r), (M \longrightarrow q_1 q_2 \dots q_r \bullet, M\bullet) \in E(CFG)$ called *entry edges* and *exit edges* respectively,
  - ◇ For each $t \in T$, $(M \longrightarrow \dots \bullet t \dots, M \longrightarrow \dots t \bullet \dots) \in E(CFG)$ called *scan edges* labeled t, where $(M \longrightarrow \dots \bullet t \dots)$ is called a *scan node*, and
  - ◇ For each $K \in N$, $(M \longrightarrow \dots \bullet K \dots, \bullet K)$, $(K\bullet, M \longrightarrow \dots K \bullet \dots,)$ called *call edges* and *return edges* respectively, where $(M \longrightarrow \dots \bullet K \dots)$ is called a *call node* that is matched with the *return node* $(M \longrightarrow \dots K \bullet \dots)$, and
- ▷ Edges not scan edges are labeled ϵ.

A. Appendix Title

This is the text of the appendix, if you need one.

Acknowledgments

Acknowledgments, if needed.

References

[1] Gianfranco Bilardi, and Keshav Pingali. Parsing with Pictures. UTCS Tech Reports, 2012. This is a full TECHREPORT entry.

[2] Elizabeth Scott. SPPF-Style Parsing From Earley Recognisers. *Electronic Notes in Theoretical Computer Science*, 203(53-67), 2008. This is a full ARTICLE entry.