Harvard CS 121 and CSCI E-121

Lecture 12: Turing Machines

Harry Lewis

October 15, 2013

- **Reading:** Sipser, §3.1.

CF Recognition in Practice

In compilers for programming languages, parsing is done via algorithms that correspond to *Deterministic* PDAs (DPDAs).

- What is the advantage over CNF algorithm?

Our CFG$\mapsto$PDA construction is highly nondeterministic.

- Constructs parse tree "top-down" from start variable; input might not be used until the very end.

A dual, "bottom-up" approach sometimes yields a DPDA.

- Construct parse tree starting from input string.
- Yields a DPDA if G is a "DCFG".
- We can design programming languages to ensure the DCFG property. (DCFLs are a strict subset of CFLs.)

The Top-Down CFG $\rightarrow$ PDA construction, revisited

- $\delta(q_{\text{start}}, \varepsilon, \varepsilon) = \{(q_{\text{loop}}, S\$)\}$

  "Start by putting $S\$$ on the stack, & go to q_{loop}"

- for each $A \in V$,
  $\delta(q_{\text{loop}}, \varepsilon, A) = \{(q_{\text{loop}}, w) : A \rightarrow w \text{ is a rule of } R\}$

  "Remove a variable from the top of the stack and replace it with a corresponding righthand side"

- for each $\sigma \in \Sigma$, $\delta(q_{\text{loop}}, \sigma, \sigma) = \{(q_{\text{loop}}, \varepsilon)\}$

  "Pop a terminal symbol from the stack if it matches the next input symbol"

- $\delta(q_{\text{loop}}, \varepsilon, \$) = \{(q_{\text{accept}}, \varepsilon)\}$.

  "Go to accept state if stack contains only $\$$."

The Dual Bottom-Up CFG → PDA Construction

- $\delta(q_{\text{start}}, \varepsilon, \varepsilon) = \{(q_{\text{loop}}, \$)\}$

  "Start by putting $\$$ on the stack, & go to q_{loop}"

- for each $\sigma \in \Sigma$, $\delta(q_{\text{loop}}, \sigma, \varepsilon) = \{(q_{\text{loop}}, \sigma)\}$

  "Shift input symbols onto the stack"

- for each $A \in V$, $\delta(q_{\text{loop}}, \varepsilon, w^R) = \{(q_{\text{loop}}, A) : A \to w \in R\}$

  "Reduce right-hand sides on the stack to corresponding left-hand sides"

- $\delta(q_{\text{loop}}, \varepsilon, S\$) = \{(q_{\text{accept}}, \varepsilon)\}$

  "Accept if the stack consists just of S above the bottom-marker"

Summary of Context-Free Recognition

- CFL to PDA reduction yields nondeterministic automaton
- By use of Chomsky Normal Form and dynamic programming, there is a general $O(n^3)$ non-stack-based algorithm
- The deterministic CFLs are the languages recognizable by deterministic PDAs
- E.g. $\{wcw^R : w \in \{a, b\}^*\}$ is a deterministic CFL but $\{ww^R : w \in \{a, b\}^*\}$ (even palindromes) is not
- Methods used in compliers are deterministic stack-based algorithms, requiring that the source language be deterministic CF or a special type of deterministic CF (LR(k), etc.).

Beyond Context-Free Languages

- A **Context-Sensitive Grammar** allows rules of the form $\alpha \rightarrow \beta$, where α and β are strings and $|\alpha| \leq |\beta|$, so long as α contains at least one nonterminal.

- The possibility of using rules such as $aB \rightarrow aDE$ makes the grammar "sensitive to context"

- Is there an algorithm for determining whether $w \in L(G)$ where G is a CSG?

- But the field moved, and now we also move, from syntactic structures to computational difficulty.

Turing Machines

Objective: Define a computational model that is

- General-purpose:

  (as powerful as programming languages)

- Formally Simple:

  (we can prove what cannot be computed)

The Origin of Computer Science

Alan Mathison Turing

"On Computable Numbers, with an Application to the Entscheidungsproblem" 1936

What Problem Was Turing Trying to Solve?

- David Hilbert

  “Mathematical Problems” 1900

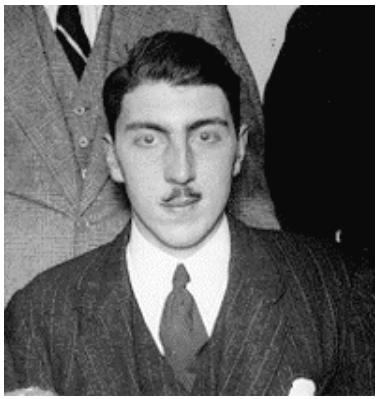
The Logicians

- Kurt Gödel

Library of Congress

"On Formally Undecidable Propositions . . ." 1931

- Alonzo Church



"An Unsolvable Problem of Elementary Number Theory" 1936

The Cliff's Notes Version of History