Context Sensitive Grammar and Linear Bounded Automata

Anup Kumar Sah
Hemanth Kumar A N

Department of Computer Science & Automation
Indian Institute of Science, Bangalore

December 4, 2013

Overview

Introduction

- Hierarchy of grammar and languages.
  - Type-0 $\rightarrow$ Recursively enumerable language
    $(N \cup \Sigma)^+ \rightarrow (N \cup \Sigma)^*$
  - Type-1 $\rightarrow$ Context Sensitive language
  - Type-2 $\rightarrow$ Context Free language
  - Type-3 $\rightarrow$ Regular language
- As we move up in hierarchy restrictions on form of the production increases and power of grammar to represent languages decreases.
- We discuss Context Sensitive Language and corresponding state machine, (Linear Bounded Automaton(LBA)) and properties of Context Sensitive Languages.

Formal Definition

- Context Sensitive Grammar(CSG) is a quadruple G=(N,Σ,P,S), where
  - N is set of non-terminal symbols
  - $\sum$ is set of terminal symbols
  - S is set of start symbol
  - P's are of the form $\alpha A\beta \rightarrow \alpha\gamma\beta$ where $\gamma \neq \epsilon$ where $(\alpha, \beta, \gamma) \in (N \cup \Sigma)^*$ and $(A \in N)$
- Why Context Sensitive??
  - Given a production : $\alpha A\beta \rightarrow \alpha\gamma\beta$ where $\gamma \neq \epsilon$. During derivation non-terminal A will be changed to γ only when it is present in context of α and β
- As a consequence of $\gamma \neq \epsilon$ we have $\alpha \rightarrow \beta \Rightarrow |\alpha| \leq |\beta|$ (Noncontracting grammar)

Context Sensitive Languages

- The language generated by the Context Sensitive Grammar is called context sensitive language.
- If G is a Context Sensitive Grammar then
  - $L(G) = \{w | \quad w \in \sum^* \quad and \quad S \Rightarrow_G^+ w\}$
- CSG for L = $\{ a^n b^n c^n | \text{ n} \geq 1 \}$
  - $N : \{S, B\} \quad and \quad \sum = \{a, b, c\}$
  - P : S $\rightarrow$ aSBc | abc cB $\rightarrow$ Bc bB $\rightarrow$ bb
- Derivation of *aabbcc* :
  - $S \Rightarrow aSBc \Rightarrow aabcBc \Rightarrow aabBcc \Rightarrow aabbcc$

Closure Properties

Context Sensitive Languages are closed under

- Union
- Concatenation
- Reversal
- Kleene Star
- Intesection

All of the above except Intersection can be proved by modifying the grammar.
Proof of Intersection needs a machine model for CSG.

Recursive v/s Context Sensitive

Theorem

Every CSL is recursive

- Construct a 3-tape nondeterministic TM M to simulate the derivations of G.
  - First tape holds the input string
  - Second tape holds the sentential form generated by the simulated derivation
  - Third tape is for the derivation.
- On any input string w, a computation of the nondeterministic TM M consists of the following sequence of steps.

Recursive v/s Context Sensitive

- Tape-3 initially contains S#.
- A rule $\alpha \rightarrow \beta$ is nondeterministically chosen from tape 2.
- Let $\gamma\#$ be the most recent string written on tape 3. An instance of the string α in γ is chosen, if exists (i.e. $\gamma = \delta\alpha\sigma$ for some δ and σ). Otherwise, go to rejecting state.
- $\delta\alpha\sigma\#$ is written on tape 3 immediately after $\gamma\#$ (indicating the application of the rule to to produce the next sentential form $\delta\beta\sigma$)
- Accept/Reject
  - If $\delta\beta\sigma =$ w, the computation of M halts in an accepting state
  - If $\delta\beta\sigma$ occurs at other position on tape 3, the computation halts in a rejecting state.
  - If $|\delta\beta\sigma| > |w|$, then the computation of M halts in a rejecting state.
- Repeat 2 through 6.

Recursive v/s Context Sensitive

Theorem

There is a recursive language that is not context-sensitive.

- Enumerate all the halting TMs for the CSLs over the alphabet $=\{a, b\}$
- Every CSG G can be described using its production $\alpha_i \rightarrow \beta_i$, for i=1,..., m . So all the productions of the grammar can be represented as a string $\alpha_1 \rightarrow \beta_1 \# \alpha_2 \rightarrow \beta_2 \# ... \# \alpha_m \rightarrow \beta_m$
- The above string can be encoded as a binary string which uniquely represents G
  - h(a) = 010
  - h(b) = 0110
  - h($\rightarrow$) = 01110
  - h(#) = 011110
  - h($A_i = 01^{i+5}0, \forall A_i \in N \quad where \quad N = A_0, A_1, .., A_N$)

Recursive v/s Context Sensitive

- Consider a proper ordering on $\{0,1\}^+$. So we can write $w_1, w_2, \ldots$ in an order.
- If a binary string w_j represents a CSG, lets call it Grammar G_j
- Define a new language $L = \{w_i : w_i$ defines a CSG G_i and $w_i \notin L(G_i)\}$

Claim

L is recursive

- Construct a Membership algorithm
  - Given w_i we can verify whether it defines a CSG G_i.
  - If it does the we can use previous membership algorithm to check if $w_i \in G_i$
  - If $w_i \in G_i$ then $w_i \in L$ else $w_i \notin G_i$.

Recursive v/s Context Sensitive

Claim

L is not context-sensitive

- Assume, for contradiction, that L is a CSL.Then there exists some w_j such that $L = L(G_j)$.
- Now consider : Does $w_j \in L(G_j)$?
- If answer is true then by definition of L we have $w_j \notin L$, but $L = L(G_j)$, so we have a contradiction.
- If answer is false, by definition of L, $w_j \in L$. Since $L = L(G_j)$, we again have a contradiction.
- So L is not CSL.

Chomsky Hierarchy

- CSL is clearly powerful than CFL.
- From previous theorem : CSL has less expressive power than Recursive languages.

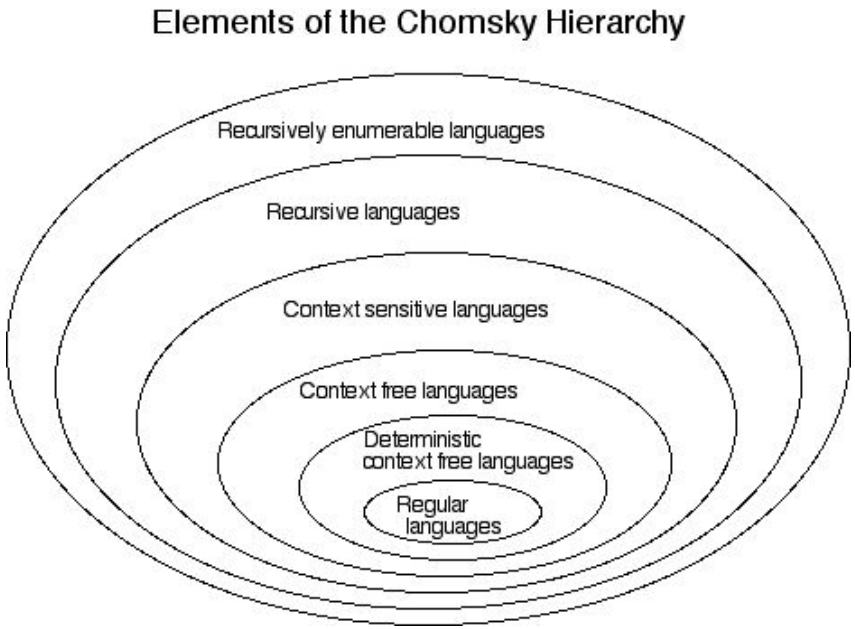

Figure: Chomsky Hierarchy

Linear Bounded Automata - Definition

Linear Bounded Automata is a single tape Turing Machine with two special tape symbols call them left marker $<$ and right marker $>$.

The transitions should satisfy these conditions:

- It should not replace the marker symbols by any other symbol.
- It should not write on cells beyond the marker symbols.

Thus the initial configuration will be:

$<$ q0a1a2a3a4a5.....an $>$

Formal Definition

Formally Linear Bounded Automata is a non-deterministic Turing Machine, M=(Q, $\sum$, Γ, δ,$\bigsqcup$,q0,$<$, $>$, t,r)

- Q is set of all states
- $\sum$ is set of all terminals
- Γ is set of all tape alphabets $\sum \subset \Gamma$
- δ is set of transitions
- $\bigsqcup$ is blank symbol
- q0 is the initial state
- $<$ is left marker and $>$ is right marker
- t is accept state
- r is reject state

Myhill-Landweber-Kuroda Theorem

- A language is accepted by LBA iff it is context sensitive language.

Proof:

- If L is a CSL it is accepted by a LBA.

We can construct a Turing Machine for L which will take the string as input that randomly tries to apply productions backwards to get finally S.
Now the productions are of the form $\alpha \Rightarrow \beta$ where $|\alpha| \leq |\beta|$ so at every stage the length of string in tape will be non increasing so none of the cells beyond the initial input string cells will be required. This is the required condition for LBA, so we can have a LBA for context sensitive language.

Myhill-Landweber-Kuroda Theorem

Proof:

- If L is accepted by a LBA, M, then L-$\{\epsilon\}$ is CSL.

In LBA the initial configuration will be <q0w >
w will be accepted if <q0w $> \vdash *$ <xqfy >, where q0 is initial and qf is final state, w is the input string.
We need to encode the transitions using context sensitive grammar.

Myhill-Landweber-Kuroda Theorem

The grammar will start with S and generate w if w is accepted by LBA.

$S \Rightarrow *q0w \Rightarrow *xqfy \Rightarrow *w$

Encoding

How the grammar will remember the input string which is modified while mimicking the transitions of LBA?? We introduce two variables V_{aib} and V_{ab} where $a \in \sum \cup \{\sqcup\}$, $b \in \Gamma$ and all i such that $qi \in Q$.

First index of V is used to store the initial input and rest indices encode the configuration of LBA at an instant.

Myhill-Landweber-Kuroda Theorem

Procedure to get CSG from LBA:

1. $S \rightarrow V_{\sqcup\sqcup}S \mid SV_{\sqcup\sqcup} \mid T$
   $T \rightarrow TV_{aa} \mid V_{a0a}$ for all a $\in \sum$
2. for each δ(qi,c) = (qj,d,R) introduce
   $V_{aic}V_{pq} \rightarrow V_{ad}V_{pjq}$
3. for each δ(qi,c) = (qj,d,L) introduce
   $V_{pq}V_{aic} \rightarrow V_{pjq}V_{ad}$ for all a, p $\in \sum \cup \{\sqcup\}$, q $\in \Gamma$
4. for every qj $\in$ F introduce
   $V_{ajb} \rightarrow$ a
5. also introduce $cV_{ab} \rightarrow$ca, $V_{ab}c \rightarrow$ac
   for all a, c $\in \sum \cup \{\sqcup\}$, b $\in \Gamma$

Myhill-Landweber-Kuroda Theorem

- In all the productions introduced the length of left side is less than or equal to right side. So the grammar generated in CSG.
- It copies the transitions of LBA and when LBA reaches final state we are generating the string back. So grammar will generate strings accepted by LBA.

Intersection closure of CSL

Given CSL L1 and CSL L2 we can have a LBA(M1) and LBA(M2) for it. We can construct a new LBA for L1 $\bigcap$ L2 by using a 2-track tape. One track will simualate M1 and other will simulate M2. If both of them accepts then string is accepted by intersection.

Thank You

References

- An Introduction to Formal Languages and Automata - Peter Linz
- *nptel.iitm.ac.in/courses/Webcourse − contents/IIT − 20Guwahati/afl/index.htm*
- *www.cs.cmu.edu/ ./FLAC*
- *www.princeton.edu/achaney/tmve/wiki100k/docs/Context−sensitive − grammar.html*