Context Sensitive Grammars and Linear Bounded Automata

Rajesh Verma, Mayank Sati, Himanshu Kumar

Computer Science & Automation
Indian Institute of Science, Bangalore

December 6,2021

Overview

Introduction

Till now:
$L_1 = \{a^n b^n c^n \mid n > 0\}$ is this a CFL?
can be shown not CFL using pumping lemma.
Solution: To deal with problems like this one, we need to strengthen our grammars. The key is to remove the constraint of being "context-free."

Chomsky Hierarchy

Level	Language type	Grammars	Accepting Automaton
3	Regular	$X \to \epsilon, X \to Y,$ $X \to aY$ (regular)	Finite State A
2	Context-free	$X \to \beta$	Pushdown A
1	Context-Sensitive	$\alpha A \beta \to \alpha \gamma \beta$ where $\gamma \neq \epsilon$	Linear Bounded A
0	Recursively enumerable	$\alpha \to \beta$ (unrestricted)	Turing Machine

Formal Definition

- Context Sensitive Grammar (CSG) is a 4-tuple G = (N,Σ,P,S), where
  - N is a non empty set of non-terminal symbols.
  - Σ is a non empty set of terminal symbols.
  - S is the start symbol and S ∈ N.
  - P is the non empty set of productions of the form:

$$\alpha \underline{A} \beta \rightarrow \alpha \underline{\gamma} \beta$$

where $A \in N, \alpha, \beta \in (N \cup \Sigma)^*$ and $\gamma \in (N \cup \Sigma)^+$

Formal Definition

- Identify which of the following are a CSG production:
  1. aAb → aBb
  2. aAb → bBa
  3. aABb → aBBb
  4. Bc → cB
  5. AB → BA (swapping)

Formal Definition

- Identify which of the following are a CSG production:
  1. aAb $\rightarrow$ aBb ✓
  2. aAb $\rightarrow$ bBa ✗
  3. aABb $\rightarrow$ aBBb ✓
  4. Bc $\rightarrow$ cB ✗
  5. AB $\rightarrow$ BA (swapping) ✗

Context Sensitive Language

- A language L is said to be context-sensitive if there exists a context-sensitive grammar G, such that L $= \mathcal{L}(\text{G})$.
- If G is context-sensitive Grammar then,

$$\mathcal{L}(G) = \left\{ w \mid (w \in \Sigma^*) \wedge (S \overset{+}{\underset{G}{\Longrightarrow}} w) \right\}$$

Context Sensitive Language: Example 1

Example

$L_1 = \{a^n b^n c^n \mid n > 0\}$

The set of Production rules of Context Sensitive Grammar G for L_1:

- S $\rightarrow$ aBC
- S $\rightarrow$ aSBC
- C<u>B</u> $\rightarrow$ C<u>Z</u>
- <u>C</u>Z $\rightarrow$ <u>B</u>Z
- B<u>Z</u> $\rightarrow$ B<u>C</u>
- aB $\rightarrow$ ab
- bB $\rightarrow$ bb
- bC $\rightarrow$ bc
- cC $\rightarrow$ cc

Non-Contracting Grammar

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.

An alternative characterization of context-sensitive languages using non-contracting grammars.

Non-contracting grammar

As a consequence of $\gamma \neq \epsilon$. We have
A formal grammar where production rules are of the form

$$\alpha \rightarrow \beta, where \mid \alpha \mid \leq \mid \beta \mid$$

Non-Contracting ≡ Context Sensitive

Theorem

A language is context sensitive if and only if it can be generated by a non-contracting grammar.

CSG to NCG:

That every production of context-sensitive Grammar can be generated by non-contracting grammar is immediate, since context-sensitive grammars are, by defnition, noncontracting.

Non-Contracting ≡ Context Sensitive

NCG to CSG:

steps:

1. for every terminal symbol a ∈ Σ, add new non terminal $[a]$ and add new rule $[a] \to a$.
2. replace every terminal symbol by its non terminal symbol.
3. Replace each rule $X_1 \ldots X_m \to Y_1 \ldots Y_n$ with following

$$X_1 X_2 \ldots X_{m-1} X_m \to Z_1 X_2 \ldots X_{m-1} X_m$$
$$Z_1 X_2 \ldots X_{m-1} X_m \to Z_1 Z_2 \ldots X_{m-1} X_m$$
$$\vdots$$
$$Z_1 Z_2 \ldots X_{m-1} X_m \to Z_1 Z_2 \ldots Z_{m-1} X_m$$
$$Z_1 Z_2 \ldots Z_{m-1} X_m \to Z_1 Z_2 \ldots Z_{m-1} Z_m Y_{m+1} \ldots Y_n$$
$$Z_1 Z_2 \ldots Z_{m-1} Z_m Y_{m+1} \ldots Y_n \to Y_1 Z_2 \ldots Z_{m-1} Z_m Y_{m+1} \ldots Y_n$$
$$Y_1 Z_2 \ldots Z_{m-1} Z_m Y_{m+1} \ldots Y_n \to Y_1 Y_2 \ldots Z_{m-1} Z_m Y_{m+1} \ldots Y_n$$
$$\vdots$$
$$Y_1 Y_2 \ldots Z_{m-1} Z_m Y_{m+1} \ldots Y_n \to Y_1 Y_2 \ldots Y_{m-1} Z_m Y_{m+1} \ldots Y_n$$
$$Y_1 Y_2 \ldots Y_{m-1} Z_m Y_{m+1} \ldots Y_n \to Y_1 Y_2 \ldots Y_{m-1} Y_m Y_{m+1} \ldots Y_n$$

New Definition

Context Sensitive Grammar

A context-sensitive grammar (CSG) is an unrestricted grammar in which every production has the form $\alpha \to \beta$ with $\mid \alpha \mid \leq \mid \beta \mid$ (where α and β are strings of nonterminals and terminals).

Context Sensitive Language: Example 1 (redefined)

Example

$L_1 = \{a^n b^n c^n \mid n > 0 \}$

The set of Production rules of Non-Contracting grammar G for L_1:

- S $\rightarrow$ abc
- S $\rightarrow$ aSBc
- cB $\rightarrow$ Bc
- bB $\rightarrow$ bb

Example

Derive: $a^3 b^3 c^3$

Context Sensitive Language: Example 1

Example

Derive: $a^3 b^3 c^3$

$$
\begin{aligned}
S &\Longrightarrow a\underline{S}Bc \\
&\Longrightarrow aa\underline{S}BcBc \\
&\Longrightarrow aaab\underline{cB}cBc \\
&\Longrightarrow aaa\underline{bB}ccBc \\
&\Longrightarrow aaabbc\underline{cB}c \\
&\Longrightarrow aaabb\underline{cB}ccc \\
&\Longrightarrow aaab\underline{bB}ccc \\
&\Longrightarrow aaabbbccc
\end{aligned}
$$

Context Sensitive Language: Example 2

Example

$L_2 = \{x \in \{a, b, c\}^* \mid \#_a x = \#_b x = \#_c x\ \}$ - $\{\epsilon\}$

The set of Production rules of non-contracting grammar G for L_2:

- S → SABC / ABC
- XY → YX   for all X,Y ∈ {A,B,C}
- A → a
- B → b
- C → c

Note that the blue production is critical here, that is not allowed in context free grammar.

Closure Properties

Context Sensitive Languages are closed under

- Union
- Intersection
- Complement
- Concatenation
- Kleene Closure
- Reversal

Closure Properties

Union

The class of context-sensitive languages is closed with respect to union.

- Let $G_1 = (N_1, \Sigma_1, P_1, S_1)$ and $G_2 = (N_2, \Sigma_2, P_2, S_2)$ be two CSG s.t $L(G_1) = L_1$ and $L(G_2) = L_2$, W.L.O.G assume $N_1 \cap N_2 = \phi$
- Construct G=$(\{S\} \cup N1 \cup N2, \Sigma_1 \cup \Sigma_2, \{S \to S_1, S \to S_2\} \cup P_1 \cup P_2, S)$
- G is a CSG and any derivation in G is of the form: $S \overset{1}{\Rightarrow} S_1 \underset{G_1}{\overset{*}{\Longrightarrow}} w \in L_1$ or $S \overset{1}{\Rightarrow} S_2 \underset{G_2}{\overset{*}{\Longrightarrow}} w \in L_2$
- The strings derived by G is exactly the strings derived by $L_1 \cup L_2$. Thus $L(G) = L_1 \cup L_2$

Closure Properties

Concatenation

The class of context-sensitive languages is closed with respect to concatenation.

- Let $G_1 = (N_1, \Sigma, P_1, S_1)$ and $G_2 = (N_2, \Sigma, P_2, S_2)$ be two CSG s.t $L(G_1) = L_1$ and $L(G_2) = L_2$, W.L.O.G assume $N_1 \cap N_2 = \phi$
- Construct G=$(\{S\} \cup N1 \cup N2, \Sigma, \{S \to S_1S_2\} \cup P_1 \cup P_2, S)$
- G is a CSG and any derivation in G is of the form: $S \overset{1}{\Rightarrow} S_1S_2 \underset{G_1}{\overset{*}{\Longrightarrow}} w_1S_2 \underset{G_2}{\overset{*}{\Longrightarrow}} w_1w_2$
- There are derivations for w_1 and w_2 from S_1 and S_2 respectively. $\Longrightarrow$ Applying these derivations to S_1S_2 we must get w=w_1w_2.
- Conversely, if w$\in$ L(G) then the rule $S \to S_1S_2$ ensures $w = w_1w_2$ where $w_1 \in L_1$ and $w_2 \in L_2$.

Recursive v/s Context Sensitive

Theorem

Every context-sensitive language is recursive.

- Consider CSL L with an associated CSG G, and look at the derivation of w.
  $S \implies x_1 \implies x_2 \implies \dots \implies x_n \implies w$
- Number of steps in any derivation is a bounded function of $|w|$. Since $|x_j| \leq |x_j + 1|$ (non contracting G)
- There exist some index m=f(G,w) such that $|x_j| < |x_j + m|, \forall j$ where m is bounded on $|N \cup \Sigma| and |w|$
- Therefore, the length of a derivation of w$\in L$ is atmost $|w|.m(|w|)$.
- Check all derivations of length up to $|w|.m(|w|)$. If any of them give w, then $w \in L$, otherwise not.

Recursive v/s Context Sensitive

Theorem

There exists a recursive language that is not context sensitive.

- Code each CSG G on input alphabet {a,b} using its production $\alpha_i \rightarrow \beta_i$, i $\in$ 1,2,..m using # as a separator as a binary string.
- String that code a CSG can be placed in an order. If a binary string w_i represents a CSG, call it G_i.
- Define a new Language L $= \{w_i : w_i$ defines G_i and $w_i \notin L(G_i)\}$
- Claim: L is recursive and not Context Sensitive

Recursive v/s Context Sensitive

Claim

L is recursive and not Context Sensitive

L $= \{w_i : w_i$ defines G_i and $w_i \notin L(G_i)\}$

L is Recursive:

- Given w_i, verify whether it defines a CSG G_i.
- If w_i does not define a CSG, then $w_i \notin L$. If w_i defines a CSG, use Membership Algorithm defined to find out if $w_i \in L(G_i)$. If $w_i \notin L(G_i)$, then $w_i \in L$.
- L is well defined and is recursive

Recursive v/s Context Sensitive

Theorem

There exists a recursive language that is not context sensitive.

L $= \{w_i : w_i$ defines G_i and $w_i \notin L(G_i)\}$

L is not Context Senstive

- Proof by contradiction, assume that L is a CSL.Then there exists some CSG w_k such that L=L(G_k) for some k.
- If $w_k \in L(G_k)$, then $w_k \notin$L (by def. of L). But L=L(G_k). $\implies$ Contradiction.
- If $w_k \notin L(G_k) \implies w_k \in L$. But L=L($G_k$). $\implies$ Contradiction
- So L is not context sensitive.

Expressive Power of CSL

- Every Context sensitive language is recursive and there exists a recursive language that is not context sensitive. Thus, CSL has less expressive power than Recursive languages.

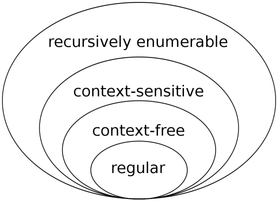

Figure: Chomsky Hierarchy

Linear Bounded Automata

A non-deterministic single tape Turing machine that uses only the tape space occupied by the input is called a linear-bounded automaton (LBA).

$\vdash$	a_1	a_2	.	...	.	a_n	$\dashv$

where, $a_1, a_2, ..., a_n \in A$

Equivalent Definition

- An equivalent definition of an LBA is that it uses only constant (c) times the amount of space occupied by the input string, where c is a constant fixed for the particular machine.
- The machine therefore has just linear amount of memory, bounded by the length of the input string. We call this a linear bounded automaton.

Formal definition

A Linear Bounded Automaton is a non-deterministic Turing Machine,

$$M = (Q, A, \Gamma, s, \delta, \vdash, \dashv, B, t, r)$$

1. Q is a finite non empty set of states.
2. A is the finite non empty set of input alphabet.
3. Γ is the finite tape alphabet which contains A.
4. $s \in Q$ is the start state.
5. δ is the set of transitions.
6. $\vdash \in \Gamma$ is the left-end marker
7. $\dashv \in \Gamma$ is the right-end marker.
8. $B \in \Gamma$ is the blank tape symbol.
9. $t \in Q$ is the accept state.
10. $r \in Q$ is the reject state and r $\neq$ t.

Formal Definition

- The Transition should satisfy the following conditions:
  - It should not replace the marker symbols by any other symbol.
  - The tape head should not move left of $\vdash$ and right of $\dashv$.
- Thus, the initial configuration on input x will be:

$$(s, \vdash x \dashv, 0)$$

- The linear-bounded M accept w. if,

$$\left\{ (s, \vdash w \dashv, 0) \xrightarrow[M]{*} (t, \vdash \alpha \dashv, i), \text{for some } \alpha \text{ and i} \right\}$$

Number of Configurations

- Let a given LBA M has
  - q states.
  - m characters in the tape alphabet.
  - and the input length is n.
- Then M can be in at most

  $$\alpha(n) = m^n \times q \times n$$

  configurations.

Results about LBA

Theorem

On an input of length n, if the LBA M does not halt after $m^n nq$ steps, then M cannot accept the input.

Proof.

The computation of M begins with the start configuration. When M performs a step, it goes from one configuration to another. If M does not halt after $m^n nq$ steps, some configuration has repeated. Then M will repeat this configuration over and over. $\implies$ Loop. $\square$

Results about LBA

Halting Problem

$\text{HALT}_{LBA} = \{< M, w >|$ M is a LBA and M halts on input w $\in A^*\}$

Results about LBA

Halting Problem

The halting problem is solvable for linear bounded automata.

- HALT$_{LBA}$ $= \{< M, w >|$ M is a LBA and M halts on input w $\in A^*\}$ is decidable.
- An LBA that stops on input w must stop in at most $\alpha(| w |)$ steps.

Results about LBA

Membership Problem

$A_{LBA} = \{< M, w >|$ M is an LBA and M accepts w $\in A^*$ $\}$

Results about LBA

Membership Problem

The membership problem is decidable for linear bounded automata.

- $A_{LBA} = \{< M, w >|$ M is an LBA and M accepts w $\in A^*$ $\}$ is decidable

Proof.

Simulate M on w for $m^n nq$ steps (n $= \mid w \mid$) or until it halts.
If M halts and accepts w, Accepted!
else, Rejected! □

Language accepted by LBA

- The language accepted by LBA M is denoted L(M) and is the set of strings accepted by M.
- A language L $\subseteq A^*$ is called Context Sensitive Language (CSL) if it is accepted by some Linear Bounded Automaton M.

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 $\cap$ 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.

Curiosity

- At the bottom level of the Chomsky hierarchy, it makes no difference: every NFA can be simulated by a DFA.
- At the top level, the same happens. Any nondeterministic Turing machine can be simulated by a deterministic one.
- At the context-free level, there is a difference: we need NPDAs to account for all context-free languages.
- What about the context-sensitive level? Are NLBAs strictly more powerful than DLBAs? Asked in 1964, and still open!!

References

- John Myhill (June 1960). Linear Bounded Automata
- Linear Bounded Automata by Forbes D. Lewis
- https://en.wikipedia.org/wiki/Context-sensitive_language
- https://www.cs.cmu.edu/ sutner/CDM/notes/70-cont-sens.pdf
- John E. Hopcroft; Jeffrey D. Ullman (1979). Introduction to Automata Theory, Languages, and Computation
- An Introduction to Formal Languages and Automata by Peter Linz
- and old seminars.

Thank You!