6. Decidable questions

6.1 Membership

Given a context free language L and a string w, there exists a decision procedure that answers the questions, "is w in L?". There are two approaches:

- Find a context-free grammar to generate it
- Find a PDA to accept it

Using a Grammar to Decide

We begin by considering the first alternative. We show a straightforward algorithm for deciding whether a string w is in a language L:

Using facts about every derivation that is produced by a grammar in Chomsky normal form, we can construct an algorithm that explores a finite number of derivation paths and finds one that derives a particular string w iff such a path exists.

decideCFLusingGrammar(*L*: CFL, *w*: string) =

1. If given a PDA, build G so that $L(G) = L(M)$.
2. If $w = \varepsilon$ then if S_G is nullable then accept, else reject.
3. If $w \neq \varepsilon$ then:
   3.1 Construct G' in Chomsky normal form such that $L(G') = L(G) - \{\varepsilon\}$.
   3.2 If G derives w, it does so in $2 \cdot |w| - 1$ steps. Try all derivations in G of $2|w| - 1$ steps. If one of them derives w, accept. Otherwise reject.

Using a PDA to Decide

It is also possible to solve the membership problem using PDAs. We take a two-step approach. We first show that. for every context-free language L it is possible to build a PDA that accepts L - { ε } and that has no ε-transitions. Then we show that every PDA with no ε-transitions is guaranteed to halt.

While not all PDAs halt, it is possible for any context-free language L, to craft a PDA M that is guaranteed to halt on all inputs and that accepts all strings in L and rejects all strings that are not in L.

cfgtoPDAnoeps(G: context-free grammar) =

1. Convert G to Greibach normal form, producing G'.
2. From G' build the PDA M.

Step 2 can be implemented as follows.

$M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$, where Δ contains:

1. The start-up transitions: For each rule $S \rightarrow cs_2 \ldots s_n$, the transition $((p, c, \varepsilon), (q, s_2 \ldots s_n))$.
2. For each rule $X \rightarrow cs_2 \ldots s_n$ (where $c \in \Sigma$ and s_2 through s_n are elements of $V - \Sigma$), the transition $((q, c, X), (q, s_2 \ldots s_n))$.

TechJourney.in

Halting Behavior of PDAs without ε-Transitions

A PDA Without ε-Transitions must halt.

Theorem: Let M be a PDA that contains no transitions of the form $((q_1, \varepsilon, s_1), (q_2, s_2))$, i.e., no ε-transitions. Consider the operation of M on input $w \in \Sigma^*$. M must halt and either accept or reject w. Let $n = |w|$. We make three additional claims:

a. Each individual computation of M must halt within n steps.

b. The total number of computations pursued by M must be less than or equal to b^n, where b is the maximum number of competing transitions from any state in M.

c. The total number of steps that will be executed by all computations of M is bounded by nb^n.

Proof:

a. Since each computation of M must consume one character of w at each step and M will halt when it runs out of input, each computation must halt within n steps.

b. M may split into at most b branches at each step in a computation. The number of steps in a computation is less than or equal to n. So the total number of computations must be less than or equal to b^n.

c. Since the maximum number of computations is b^n and the maximum length of each is n, the maximum number of steps that can be executed before all computations of M halt is nb^n.

The final algorithm can be listed as follows.

decideCFLusingPDA(L: CFL, w: string) =

1. If L is specified as a PDA, use *PDAtoCFG* to construct a grammar G such that $L(G) = L(M)$.
2. If L is specified as a grammar G, simply use G.
3. If $w = \varepsilon$ then if S_G is nullable then accept, otherwise reject.
4. If $w \neq \varepsilon$ then:
   4.1 From G, construct G' such that $L(G') = L(G) - \{\varepsilon\}$ and G' is in Greibach normal form.
   4.2 From G' construct a PDA M such that $L(M) = L(G')$ and M' has no ε-transitions.
   4.3 All paths of M are guaranteed to halt within a finite number of steps. So, run M on w. Accept if it accepts and reject otherwise.

6.2. Decidability of Emptiness and Finiteness

Theorem: Given a context-free language L, there exists a decision procedure that answers each of the following questions:

1. Given a context-free language L, is $L = \emptyset$?
2. Given a context-free language L, is L infinite?

Since we have proven that there exists a grammar that generates L iff there exists a PDA that accepts it, these questions will have the same answers whether we ask them about grammars or about PDAs.

Proof

1. Let $G = (V, \Sigma, R, S)$ be a context-free grammar that generates L. $L(G) = \emptyset$ iff S is unproductive (i.e., not able to generate any terminal strings). The following algorithm exploits the procedure *removeunproductive*, defined in Section 11.4, to remove all unproductive nonterminals from G. It answers the question, "Given a context-free language L, is $L = \emptyset$?"

decideCFLempty(G: context-free grammar) =

1. Let G' = *removeunproductive* (G).
2. If S is not present in G' then return *True* else return *False*.

2. Let $G = (V, \Sigma, R, S)$ be a context-free grammar that generates L. We use an argument similar to the one that we used to prove the context-free Pumping Theorem. Let n be the number of nonterminals in G. Let b be the branching factor of G. The longest string that G can generate without creating a parse tree with repeated nonterminals along some path is of length b^n. If G generates no strings of length greater than b^n, then $L(G)$ is finite. If G generates even one string w of length greater than b^n, then, by the same argument we used to prove the Pumping Theorem, it generates an infinite number of strings since $w = uvxyz$, $|vy| > 0$, and $\forall q \geq 0$ (uv^qxy^qz is in L). So we could try to test to see whether L is infinite by invoking *decideCFL*(L, w) on all strings in Σ^* of length greater than b^n. If it returns *True* for any such string, then L is infinite. If it returns *False* on all such strings, then L is finite.

But, assuming Σ is not empty, there is an infinite number of such strings. Fortunately, it is necessary to try only a finite number of them. Suppose that G generates even one string of length greater than $b^{n+1} + b^n$. Let t be the shortest such string. By the Pumping Theorem, $t = uvxyz$, $|vy| > 0$, and uxz (the result of pumping vy out once) $\in L$. Note that $|uxz| < |t|$ since some nonempty vy was pumped out of t to create it. Since, by assumption, t is the shortest string in L of length greater than $b^{n+1} + b^n$, $|uxz|$ must be less than or equal to $b^{n+1} + b^n$. But the Pumping Theorem also tells us that $|vxy| \leq k$ (i.e., b^{n+1}), so no more than b^{n+1} strings could have been pumped out of t. Thus we have that $b^n < |uxz| \leq b^{n+1} + b^n$. So, if L contains any strings of length greater than b^n, it must contain at least one string of length less than or equal to $b^{n+1} + b^n$. We can now define *decideCFLinfinite* to answer the question, "Given a context-free language L, is L infinite?":

decideCFLinfinite(G: context-free grammar) =

1. Lexicographically enumerate all strings in Σ^* of length greater than b^n and less than or equal to $b^{n+1} + b^n$.
2. If, for any such string w, *decideCFL*(L, w) returns *True* then return *True*. L is infinite.
3. If, for all such strings w, *decideCFL*(L, w) returns *False* then return *False*. L is not infinite.

7. The Undecidable Questions

- Given a context free language L, is $L = \Sigma^*$?
- Given a context free language L, is the complement of L context-free?
- Given a context free language L, is L regular?
- Given a context free language L_1 and L_2,, is $L_1 = L_2$?
- Given a context free language L_1 and L_2, is $L_1 \subseteq L_2$?
- Given a context free language L_1 and L_2, is $L_1 \cap L_2 = \varnothing$?
- Given a context free language L, is L inherently ambiguous?
- Given a context free grammar G, is G ambiguous?

TechJourney.in