CMPE 350 - Spring 2018

PS 10 - 26.04.18

2.25 For any language A, let SUFFIX$(A) = \{v | uv \in A$ for some string $u\}$. Show that the class of context-free languages is closed under the SUFFIX operation

Let A be a context-free language recognized by the PDA M_1. We are going to construct a PDA M recognizing the language SUFFIX(A) to show that the language SUFFIX(A) is also context-free.

Let M_2 be identical to M_1. We update the transitions of M_2 so that the transitions are performed without consuming any input symbol. That is, any transition of the form $a, b \to c$ is replaced with $\varepsilon, b \to c$. We add the transition $\varepsilon, \varepsilon \to \varepsilon$ from each state in M_2 to the corresponding state in M_1. M_1 and M_2 together form the PDA M. Start state of M is the start state of M_2.

When M starts reading a string v, it will construct the stack as it was reading u without reading u and without consuming any input symbol. This is possible since we updated all transitions in M_2 accordingly. Then at some point, by following the transition $\varepsilon, \varepsilon \to \varepsilon$, the computation continues in M_1 by reading the string v. So, the only accepted strings are those which are the suffixes of the strings in A.

We conclude that the set of context-free languages is closed under the SUFFIX operation.

• Prove or disprove: "The class of non-context-free languages is closed under complementation."

We are going to prove that the class of non-context-free languages is closed under complementation. First let us prove that the set of context-free languages is not closed under complement.

We will first prove that the set of context-free languages is not closed under intersection. Let $L_1 = \{a^i b^i c^j | i, j \geq 0\}$ and $L_2 = \{a^i b^j c^j | i, j \geq 0\}$. L_1 and L_2 are context-free, which can be easily shown by constructing CFG's or PDA's. $L_1 \cap L_2 = \{a^n b^n c^n, n \geq 0\}$ which is known to be non-context-free. Hence we conlcude that the set of context-free languages is not closed under intersection.

Now suppose for a contradiction that the set of context-free languages is closed under complement. Let A_1 and A_2 be two context-free languages. Then A_1^c and A_2^c are also context-free. $A_1^c \cup A_2^c$ is also context-free since the set of context-free languages is closed under union. Taking one more complement we obtain $(A_1^c \cup A_2^c)^c$ which is also context free and equal to $A_1 \cap A_2$ using De Morgan's law. But since we have proven that the set of context-free languages is not closed under intersection, we obtain a contradiction.

We have proven that the set of context-free languages is not closed under complement. (Otherwise, it would be closed under intersection as well) It means that there exists some context-free language L whose complement L^c is non-context-free. If the set of non-context-free languages were closed under complement, then $L^{cc} = L$ would also be non-context-free but this is not the case. Hence, we can conclude that the set of non-context-free languages is not closed under complement.

Alternatively, you can pick $L = \{a^n b^n c^n\}$ or $L = \{ww | w \in \{a, b\}^*\}$ which are non-context-free and prove that their complements are context-free.

• Let $A = \{1^p | p$ is a prime number greater than $2^{100}\}$ and $B = \{1^k | 0 \leq k < 2^{100}\}$. Is the language $A \cup B$ context-free? Prove your answer.

Suppose for a contradiction that $A \cup B$ is context-free and the Pumping Lemma holds. Let p be the pumping length. Let $s = a^n$ where $n \geq p$ and n is a prime number greater than 2^{100}. Since there are infinitely many prime numbers this is possible. According to Pumping Lemma, $s = a^n$ can be written as $s = uvxyz$ where $|vy| > 0$ and $|vxy| \leq p$. We can write $v = a^j$ and $y = a^k$ for some $j, k \geq 0$ (both not equal to 0 at the same time).

We usually pick $i = 2$ but this time $uv^2xy^2z = a^{n+j+k}$ and we can not guarantee that it does not belong to $A \cup B$. Let us look at uv^ixy^iz in general.

$uv^ixy^iz = a^{n+j(i-1)+k(i-1)} = a^{n+(i-1)(j+k)}$. Let $i = n+1$. Then $uv^ixy^iz = a^{n+n(j+k)} = a^{(n+1)(j+k)} \notin A \cup B$. $a^{(n+1)(j+k)} \notin A$ since $(n+1)(j+k)$ is not a prime number as it is the product of two integers and $\notin B$ since $(n+1)(j+k)$ is greater than 2^{100}. We conclude that $A \cup B$ is not context-free.

Bu soruda tek başına A'yı alıp context-free olmadığını göstermek yeterli değil. Bu yapılacaksa, non-context-free ve finite iki dilin birleşiminin her zaman non-context-free olduğunu ıspatlamak gerekir. String seçerken uzunluğunu 2^{100}'den büyük seçtik ki pump ettiğimizde uzunluğunun 2^{100}'den büyük olmasını garanti edelim ve B'nin içinde kalmasın. Pumping lemma'yı A yerine $A \cup B$ üzerinde kullanmanın farkı bu. Burada i'nin seçimi önemli. i'yi $n-1$ seçtik ki a'nın üssü iki sayının çarpımı olarak yazılabilsin.

• **Consider the language $R = \{w | w$ is a regular expression on the alphabet $\{0,1\}\}$. Is R context-free?**

We can construct a CFG generating R. Here is the grammar:

$S \to (S \cup S)|(S^*)|(SS)|\emptyset|\varepsilon|0|1$

Bu soruda S variable, $\cup, *, \emptyset, `\varepsilon' \ldots$ terminal. Dilin elemanı olan stringler valid birer regular expression. Örneğin $((0 \cup 1)0), (1^*)1$ dilin elemanı iken $)1\cup, *1$ dilin elemanı değil. Amacımız da dilin elemanı olan stringleri üreteren bir gramer yazmak. Buradaki ε gramer kurallarındaki empty string değil. Yani $\cup S$ gibi bir şey üretilemiyor. ε regular expression kurallarındaki empty string. Regular expression = regular language o da zaten context-free gibi bir yaklaşım yanlış. Çünkü burada verilen dil regular değil, verilen dil regular expression dili.

• **Prove that there exists a Turing machine M whose language L is decidable, but M is not a decider.**

Let M be a Turing machine which loops on every string. (For example, stay at the start start state which is not an accept state, move right at each step and don't change the tape) Then the language recognized by M is the empty set and M is not a decider since it loops. On the other hand, empty set is decidable (even regular).

Yani decidable (hatta regular) bir dil için de loopa giren bir makine yapılabilir. Bu o dilin decidable olmadığını göstermez. Makinenin 'kötü' yapıldığını gösterir.

3.18 Show that a language is decidable iff some enumerator enumerates the language in the standard string order

We need to show equivalence between a Turing machine that decides a language and an enumerator that enumerates it. Thus we need show the proof in both directions.

Let L be a decidable language and let M be its decider. We can use M to construct an enumerator E as follows. We generate strings in lexicographic order, and input each string into M. If M accepts, E prints the string. Thus, E prints all strings of L in lexicographic order.

Now we need to show the other direction. i.e., if we have an enumerator E for a language L,

then we can use E to construct a Turing machine that decides L. If L is a finite language, it is decidable because all finite languages are decidable. If L is infinite, a decider M for L operates as follows. On receiving input w, M runs E until a string greater than w in the lexicographic order is printed by E. This must eventually occur since L is infinite. If w has appeared in the enumeration already, then M accepts w. If w has not appeared yet, then it will never appear, and hence M rejects.