CS 154 – Introduction to Automata and Complexity Theory

Section notes, May 15, 2000

Closure properties This is essentially a clarification of what closure properties are, and what they can be used for. Both versions of the midterm discussed certain closure properties for regular sets. We will see more of closure properties with regard to context-free languages, so I thought it would be worthwhile to clear up any lingering confusion.

Let's take the set union operation, which is closed for both regular sets and context-free languages. The closure property for the union operation for regular sets may be stated as:

$$\forall \text{ regular sets } L_1 \text{ and } L_2\text{: } L_3 = L_1 \cup L_2 \text{ is also regular}$$

However, that closure property has no relevance to any of the following questions, where $L_3 = L_1 \cup L_2$:

- If L_3 is regular, are L_1 and/or L_2 regular?
- If L_1 is nonregular and L_2 is regular, is L_3 regular?
- If L_1 and L_2 are nonregular, is L_3 nonregular?

The answer to all three of these questions is "it depends on what L_1, L_2, and L_3 are." The point here is that it is impossible in these three cases to draw a generalized conclusion such as a closure property. In particular, you may not appeal to the closure property described above, as it simply does not apply to any of the three questions.

Pumping lemma for context free languages The pumping lemma for CFL's is very similar to the pumping lemma for regular sets, and those students who were present at section indicated that they are comfortable with the PL for CFL's on a conceptual level. Consequently, section focused on the practical application of the PL for CFL's.

Generally speaking, proofs appealing to the PL for CFL's are much more involved than proofs based on the PL for regular sets. This difficulty results primarily from the fact that the PL for CFL's is much less restrictive about the substrings we use to pump. More specifically, given our choice of a string z, we must examine all choices of substrings u,v,w,x, and y, where $z = uvwxy$ and the following conditions hold:

$$\begin{aligned} &1)\ |vx| \geq 1 \\ &2)\ |vwx| \leq n \end{aligned}$$

We must then show

$$3)\ uv^iwx^iy \in L, \forall i \geq 0$$

for all substrings which satisfy the above two conditions. Note that there are no particular constraints on u or y, so we have to consider cases such as $u = \epsilon$ and $y = \epsilon$. In other words, although we only really care about the vwx portion of our string z, that section may be anywhere in the string - beginning, middle, or end - and we have to deal with every possibility.

Here is an example problem: prove that the language $\{a^ib^j \mid j = i^2\}$ is not context-free, using the pumping lemma for context-free languages.

We start by selecting the string $z = a^nb^{n^2}$. We now have to consider all possible $uvwxy = z$, subject to the two conditions mentioned above. Here is an enumeration of all the cases we need to consider, where p, q, r are integers greater than 0:

- $v = \epsilon$, $x = a^p$
- $v = \epsilon$, $x = b^p$
- $v = \epsilon$, $x = a^pb^q$
- $v = a^p$, $x = \epsilon$
- $v = b^p$, $x = \epsilon$
- $v = a^pb^q$, $x = \epsilon$
- $v = a^p$, $x = a^q$
- $v = b^p$, $x = b^q$
- $v = a^p$, $x = b^q$
- $v = a^pb^q$, $x = b^r$
- $v = a^p$, $x = a^qb^r$

Note that one (but not both) of v and x may be empty. Otherwise, we have to consider every possible case for v and x given the restraint that the larger string z is of the form $a^nb^{n^2}$. Proving each of these individual cases is easy, as in the following examples:

If one of the strings v or x contains both a's and b's, clearly it cannot be pumped to produce a string in L. (Doing so will introduce a's after b's, which is not allowed in our language.) This covers four of the enumerated cases.

If one of the strings v or x is empty, then clearly you cannot pump the other (nonempty) string to produce a string in L. To see why, take the case where v is empty and x is all a's. Pumping x will introduce some additional number of a's (let's say, c instances of the symbol a, where c<0), and clearly $(n + c)^2 \neq n^2$. The reciprocal argument holds where v is empty and x is all b's, or v is all a's and x is empty, or v is all b's and x is empty. This covers an additional four cases.

We still have three cases left to cover, but I will skip those, as they are similar to the ones above. The point is that the most difficult part is often enumerating all of the cases. My example solution is perhaps more exhaustive than is strictly necessary - several of those cases could be collapsed into each other - but it gives you an idea of how to proceed in a PL for CFL proof.

Turing machine example In the last five minutes of section, I gave a rough run-through of how a Turing machine can be designed to solve a particular problem. Note that the description is a high-level one, and that the details of implementation are omitted.

Design a Turing machine to accept the language L:

$$\{ww^R \mid w \in (0+1)^*\}$$

The general strategy here is to read the leftmost symbol, scan across to the rightmost symbol, and check to make sure that they match. If they don't, we can stop now, as the string does not satisfy the description above. If they do, we then proceed to the second leftmost and second rightmost symbols and compare those. In this fashion, we can process the entire string and determine whether or not it is in L.

Now that we have an idea about our strategy, we can consider some implementation details.

- How can we keep track of which symbols we have already checked?
- How can we scan from the current left-hand symbol to the current right-hand symbol, and vice-versa?
- How can we keep track of the left-hand symbol, so that we can compare it to the right-hand symbol once we get there?
- How do we know when we are done?

Time did not permit much discussion of these individual points, so they are left for the consideration of the student.