Yet More PDAs

CS154

Chris Pollett

Apr 2, 2007.

Outline

- Different proof that for recognized by PDA =>CFL.
- Grammars for DCFLs
- Pumping Lemma for CFLs

- My laptop died so there will be no JFLAP demo today.

Different proof that recognized by PDA => CFL

- Last day, we gave a proof that being recognized by a PDA => CFL.
- Today, we will look at a different proof.
- In this case the resulting grammar will be Greibach Normal Form.
- So since we have already shown CFL => recognized by PDA, this will show every CFL is equivalent to one in Greibach Normal Form.

Simplifying Assumptions

- Let M be a PDA
- As in our first proof, we will make some simplifying assumptions:
  - It was a single final accept state q_f which is entered iff the stack is empty.
  - For $a \in \Sigma \cup \{\lambda\}$, all transitions must have the form $\partial(q_i, a, A) = \{c_1, .., c_m\}$, where $c_i = (q_j, \lambda)$ or $c_i = (q_j, BC)$, so a move at most increases or decreases the stack size by 1.
- It turns out any language recognized by PDA can be recognized by a PDA with these restrictions: We can take a PDA and modify it with two new states empty string transitions from the old final states to the first new state, transitions that empty the stack in this state, then a transition from this state to the second new state which is a final state. Similarly, we can split up a more complicated stack move into a sequence of single character stack moves to handle the second case.

Theorem

If L=L(M) for some PDA M, then L is CFL.

Proof. Assume M satisfies the simplifying assumptions. Let q_0 be the initial state. The variables of our grammar will be of the form (q_iAq_j) where q_i and q_j are states of M and where A is a stack symbol. This is supposed to represent one can transition from q_i q_j by a sequence of moves, removing A from the top of the stack. Our start variable will be $(q_0\ \$, q_f)$ where \$ is the start of stack symbol. We will then have the productions: $(q_iAq_j) \to a$ if (q_j, λ) is in $\partial(q_i, a, A)$. This handles the first type of rule. And we will have productions $(q_iAq_j) \dashrightarrow a(q_jBq_l)\ (q_l\,Cq_k)$ to handle transitions where (q_j, BC) is in $\partial(q_i, a, A)$. We have this for each state k. Notice this grammar is in Greibach Normal form since each production begins with a terminal which is followed by a string of variables.

Grammars for DCFLs

- Last day we examined deterministic PDAs whose languages are called the deterministic CFLs.
- We said these languages are important for compilers because one could hope to get parsers for them which run in closer to linear time then to cubic time.
- Although DPDAs don’t have nondeterministic moves they do have empty state transitions.
- It is interesting to ask what grammars correspond to DPDAs?

Grammars for DCFLs cont'd

- Recall s-grammars have rules of the form A-->ax where x is a string of variables and where each pair (A, a) occurs in at most one rule.
- If we apply our construction CFL=> PDA to such a grammar we will get a DPDA.
- Essentially, while scanning a string w left to right we only need to look at the next character to determine which rule to use.
- However, there are more grammars for which this is true, in particular those where we can look ahead k symbols and fix the rule.
- For instance, say a grammar is an LL(k) grammar if whenever we have two leftmost derivations of a string S=>wAx =>wyx=> ..=> ws and S=>wAv =>wzv=> ..=> wt, the equality of the k leftmost symbols of s and t implies y and z are equal.
- LL(k) grammars and LR grammars are what are actually used in compilers.

Languages that are not Context Free

- We can prove languages are not context free by using the Pumping Lemma for context-free languages:

Pumping Lemma for Context Free Languages: If A is a context free language, then there is a number p (the pumping length) where, if s is any string A of length at least p, then s maybe divided into five pieces s= uvxyz satisfying the conditions:

1. for each $i>=0$, uv^ixy^iz is in A.
2. $|vy| > 0$, and
3. $|vxy| <= p$.

Example use of the CFL Pumping Lemma

- Let $C = \{a^i b^j c^k \mid 0 <= i <= j <= k\}$
- Argue by contradiction. Let p be the pumping length of C and consider the string $s=a^p b^p c^p$.
- Then s can be written as uvxyz. There are two cases:

1. Both v and y contain only one type of alphabet symbol. So one of a, b, or c does not appear in v or y. So there are three subcases
   a) The a's do not appear. By the pumping lemma, $uv^0xy^0z= uxz$ must be in the language. This string has the same number of a's but fewer b's or c's so cannot be in C giving a contradiction.
   b) The b's do not appear. Then either a's or c's must appear in v and y. If a's appear, then uv^2xy^2z will have more a's then b's giving a contradiction. If c's appear, then uv^0xy^0z will have more b's then c's giving a contradiction.
   c) The c's do not appear. Then uv^2xy^2z will have more a's or b's then c's giving a contradiction.
2. When either v or y contain more than one symbol uv^2xy^2z will not contain the symbols in the right order giving a contradiction.