2nd Lt David Crow

ENG/20M

CSCE 532 Homework 4

Problem 2.15: Give a counterexample to show that the following construction fails to prove that the class of context-free languages is closed under star. Let be a CFL that is generated by the CFG . Add the new rule and call the resulting grammar . This grammar is supposed to generate .

Let be the following CFG over the alphabet :

This grammar generates the language consisting of the single string “a.” When we add the new rule , we obtain the following grammar :

This grammar generates the language consisting of strings of one or more *a*s*.* In other words, cannot generate the empty string, which is necessarily in the language . We’ve now shown that adding the rule to a grammar that generates a language does not necessarily give a grammar that generates the language . Thus, the given construction is not sufficient to prove that the class of CFLs is closed under star.

Problem 3.8c: Give an implementation-level description of a Turing machine that decides the language over the alphabet .

“On input string :

1. Sweep left to right across the tape. Mark the first unmarked . Return the head to the left end of the tape. If there are no unmarked s, go to stage 4.
2. Sweep left to right across the tape. Mark the first unmarked . If there are no unmarked s, accept.
3. Return the head to the left end of the tape and repeat stage 2. Return the head to the left end of the tape and go to stage 1.
4. Sweep left to right across the tape. If there are no unmarked s, reject. Otherwise, accept.”

In each iteration, this Turing Machine crosses off one and two s. If there are no remaining s after it’s crossed off all s in , then has exactly twice as many s as s, and so the machine rejects. In all other cases, the machine accepts.

Problem 3.9: Let a -PDA be a pushdown automaton that has stacks. Thus a -PDA is an NFA and a -PDA is a conventional PDA. You already know that -PDAs are more powerful (recognize a larger class of languages) than -PDAs.

1. Show that -PDAs are more powerful than -PDAs.

A 2-PDA can simulate a 1-PDA simply by utilizing just one of its two stacks. In other words, 2-PDAs are at least as powerful as 1-PDAs.

To show that 2-PDAs are strictly more powerful than 1-PDAs, we must show a language that a 2-PDA recognizes but that a 1-PDA does not recognize. Let . We know (from previous examples) that is not context-free, so we know that is not recognized by any 1-PDA. However, we can create a 2-PDA that recognizes .

For some input , push onto both stacks. Begin reading s in . For each read, push an onto both stacks. If we read anything other than an or a , reject. When all s have been read, begin reading s.

For each read, pop an off stack 1. If we still have s to read and is on the top of stack 1, reject. If we read anything other than a or a , reject. When all s have been read, begin reading s.

For each read, pop an off stack 2. If we still have s to read and is on the top of stack 2, reject. If we read anything other than a , reject. If, after all s have been read, is on the top of both stacks, accept.

Because a 2-PDA can recognize but no 1-PDA can, we know that 2-PDAs are more powerful than 1-PDAs.

1. Show that -PDAs are not more powerful than -PDAs. (Hint: Simulate a Turing machine tape with two stacks.)

We know that a Turing machine can simulate a -PDA if is finite. In other words, 3-PDAs are no more powerful than Turing machines.

We now show that a 2-PDA can simulate a Turing machine. In doing so, we show that 2-PDAs are just as powerful as 3-PDAs.

Let stack 1 contain all characters left of the tape head. Characters closer to the tape head are higher on the stack (that is, the character directly to the left of the tape head is on the top of stack 1). Let stack 2 contain all characters right of the tape head and the character under the tape head. Like before, characters closer to the tape head are higher on the stack. To simulate the tape head’s left movement, we pop a character off stack 1 and push it onto stack 2. To simulate the tape head’s right movement, we pop a character off stack 2 and push it onto stack 1. To simulate the act of writing to the tape, we pop off stack 2 and push onto stack 2.

In doing so, we can simulate a Turing machine with a 2-PDA. Thus, 2-PDAs are just as powerful as 3-PDAs.

Problem 3.15b: Show that the collection of decidable languages is closed under the operation of concatenation.

Let and be decidable languages. This means there must exist 1-tape Turing machines and such that decides and decides . If the collection of decidable languages is closed under concatenation, then there must exist a Turing machine such that decides . In describing , we show that the collection of decidable languages is closed under concatenation.

Let be a 2-tape Turing machine. For some input , non-deterministically read the first characters into tape 1. Read the remaining characters into tape 2. Run on tape 1. If it accepts, continue; otherwise, reject. Run on tape 2. If it accepts, accept; otherwise, reject.

We’ve now described a Turing machine such that . Thus, the collection of decidable languages is closed under concatenation.

Problem 3.16b: Show that the collection of Turing-recognizable languages is closed under the operation of concatenation.

This problem is almost exactly the same as the one above. Again, we have recognizable languages and , and we know that there exist Turing machines and such that and . Construct like before. For some input , non-deterministically read the first characters onto ’s tape 1. Read the remaining characters onto ’s tape 2. Then, as before, run on ’s tape 1. If it accepts, run on ’s tape 2. If it accepts, accept. If either or reject or run forever on their portion of , will reject or run forever on . We now say that recognizes . Thus, the collection of Turing-recognizable languages is closed under concatenation.