0. The NPC Poll

CS 3100, Fall 2021, Finals Prep, Dec 8, 2021

1. Show that $Y = (\lambda f.\ (lambdax.\ f(x\ x))\ (lambdax.\ f(x\ x)))$ is a fixed-point combinator.

2. Show that Ye is a fixed-point combinator for eager languages.

   $Y_eG = (\lambda f.(\lambda x.(xx)[\lambda y.f(\lambda v.((yy)v))])G.$

3. Write a recursive function that sums from N down to 1 and express its using the Ye combinator. Demonstrate, by running, the series summation from a few numbers (e.g., 10, 50, 100) down to 1.

4. Is $L_{emptycfg} = \{\langle G\rangle\ :\ L(G) = \emptyset\}$ recursive? Here, G is a CFG.

   Answer: Yes. Keep sweeping from the start symbol of G to see if G gets caught up in infinite recursion. Actually sweep from the terminals back to the S (start) symbol of G. Somehow if G manages to produce any string s at all, then $L(G) \neq \emptyset$. Else it is equal to $\emptyset$.

5. Is $L_{G1eqG2} = \{\langle G_1, G_2\rangle\ :\ L(G_1) = L(G_2)\}$ RE, where G_1 and G_2 are CFGs?

   Answer: No. If you find a string w accepted by one grammar but not the other, then the grammars are not language-equivalent. But if you keep finding all w within the language of both grammars, you can't stop. Intuitively, grammar equality is not established with one string accepted by both. One has to "exhaust" all strings. The formal proof was discussed as a canvas response but not needed.

6. Prove that there exist non-RE languages. Argue how each RE language can be represented by a bit-vector, calling the bits

```
L0  b00 b01 b01 ...
L1  b10 b11 b12 ...
L2  b21 b22 ...
L3 ...
```

   Now consider the complemented diagonal. Write it out. Argue how it compares with each of the languages L0, L1, ...

   Detailed Answer: Attempt to show that all TMs can be "numbered" (listed out). Thus, all the RE languages can be numbered. That is what I mean by L0, L1, L2, ... above. But now, one can find ONE language not in ANY of these languages. This is discussed in Appendix-C of the book (last "chapter").

7. Prove that the set of Turing machines T that halt when started with input string w, and that do not write outside of positions p_1 and p_2 on the tape ($p_1 \leq p_2$) is recursive.

   Answer:

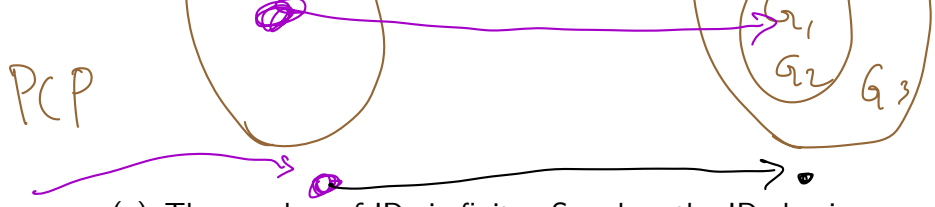
   (a) Notice that the TM is not allowed to write more than a finite number of cells.
   (b) Keep simulating this TM, observing the IDs attained by the TM.

$$\frac{\left(\text{ID}\ \dfrac{q\ a\ b\ c\ d}{a\ q\ c\ d}\right)}{(\lambda x.\, f(xx))\ G}$$

$$= \underline{(\lambda x.\ G(xx))(\lambda x.\ G(xx))}$$

$$= G((\lambda x.\ G(xx))(\lambda x.\ G(xx)))$$

$$= G(YG)$$



(c) The number of IDs is finite. So when the IDs begin repeating and the TM has not yet halted, we can conclude that the TM will never halt.

(d) Following the aforesaid algorithm, we can conclude that this set of $\langle T, w, p_1, p_2 \rangle$ is recursive.

8. Argue that RE sets are closed under union, concatenation, intersection, reversal. Here is a taste for one proof (closure under intersection); write the others:

   Answer:

   - Define the following TM that takes M_1 and M_2:
     - intersectM1M2(M_1,M_2):
     - run M_1 and M_2 in dovetail order on inputs, also generated by the dovetail enumeration procedure
     - When this simulation notices M_1 and M_2 accepting some string, it emits that string
     - This procedure now lists all strings in M_1 and M_2.

9. Present and prove a mapping reduction from A_{TM} to CFL_{TM}.

10. Do Problem 5, Page 232, Exercise 15.2.3 of our book. This asks you to perform a mapping reduction from the PCP to the CFG Ambiguity problem, and thus argue that if there is a decider for the CFG Ambiguity problem, then there will be a decider for PCP also. The languages involved in this exercise are now formally defined: PCP is defined on Page 229, Section 15.2, and reproduced below:
    Given any alphabet Σ such that $|\Sigma| > 1$, consider the *tile alphabet* $\mathcal{T} \subseteq \Sigma^+ \times \Sigma^+$. Define

    $$PCP = \{S \ : \ S \text{ is a finite sequence of elements over } \mathcal{T} \text{ that has a solution}\}.$$

    Also, define

    $$Amb = \{\langle G \rangle \ : \ G \text{ is a } CFG \text{ that has } > 1 \text{ parse for some } w \in \Sigma^*\}$$

    Here is how we can build a mapping reduction from PCP to Amb; please fill in missing steps (if any) and argue that the mapping reduction actually works (achieves its purpose).

    Let

    $$A = w_1, w_2, \ldots, w_n$$

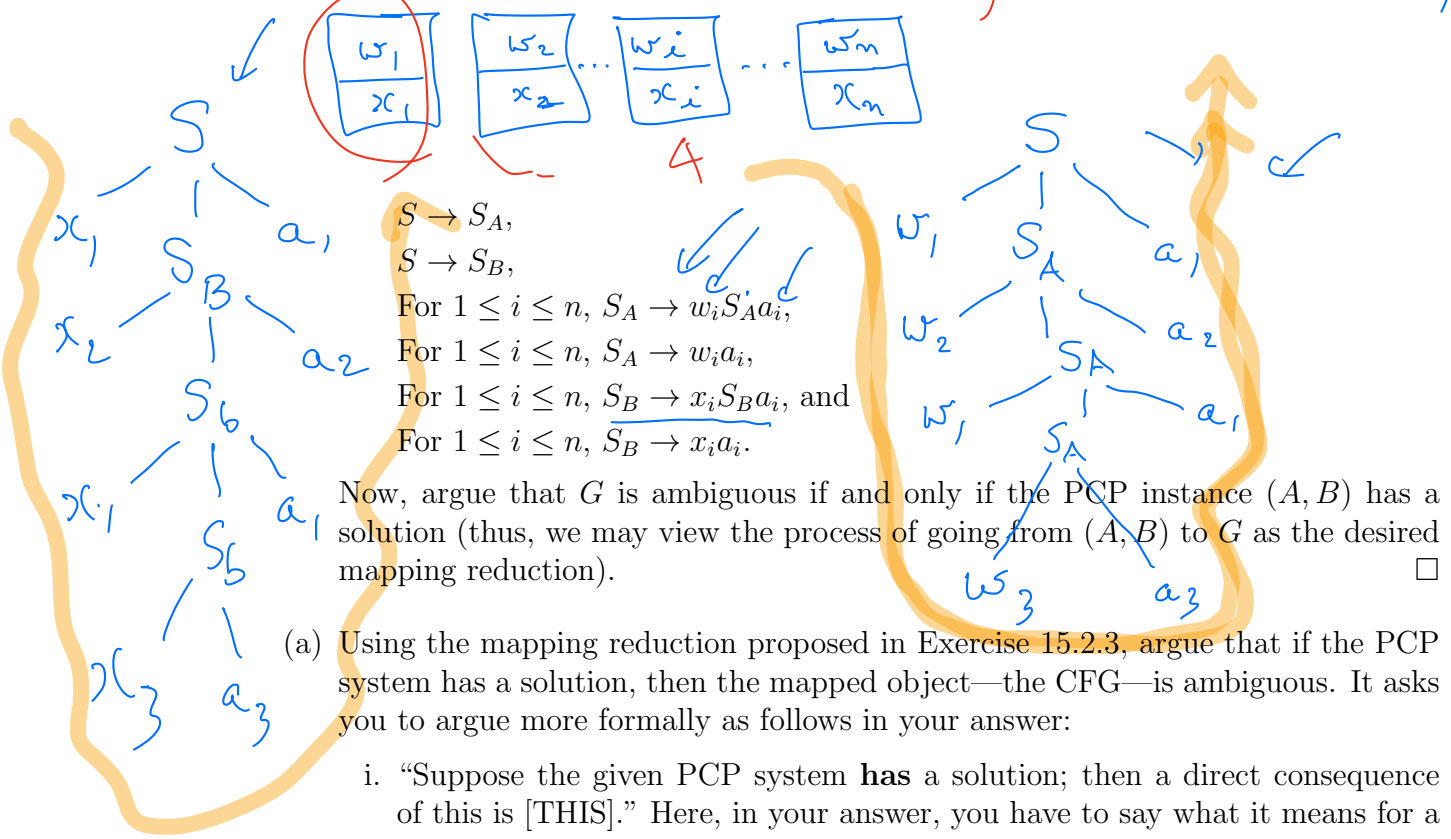
    and

    $$B = x_1, x_2, \ldots, x_n$$

    be two lists of words over a finite alphabet Σ. Let $a_1, a_2, \ldots, a_n$ be symbols that do not appear in any of the w_i or x_i. Let G be a CFG

    $$(\{S, S_A, S_B\}, \ \Sigma \cup \{a_1, \ldots, a_n\}, \ P, \ S),$$

    where P contains the productions



$S \to S_A$,
$S \to S_B$,
For $1 \le i \le n$, $S_A \to w_i S_A a_i$,
For $1 \le i \le n$, $S_A \to w_i a_i$,
For $1 \le i \le n$, $S_B \to x_i S_B a_i$, and
For $1 \le i \le n$, $S_B \to x_i a_i$.

Now, argue that G is ambiguous if and only if the PCP instance (A, B) has a solution (thus, we may view the process of going from (A, B) to G as the desired mapping reduction). □

(a) Using the mapping reduction proposed in Exercise 15.2.3, argue that if the PCP system has a solution, then the mapped object—the CFG—is ambiguous. It asks you to argue more formally as follows in your answer:

i. "Suppose the given PCP system **has** a solution; then a direct consequence of this is [THIS]." Here, in your answer, you have to say what it means for a *common* string being parsed using the grammar resulting from this mapping reduction. Elaborate on "[THIS]" in 3-4 tight sentences.

ii. The translated grammar must then have [THIS] property with respect to ambiguity. Write that down formally.

(b) Show that if the PCP system has no solution, then the mapped object (CFG) is not ambiguous. Argue the following carefully: suppose the PCP instance has no solution; could the generated CFG still admit some common string that produces two parse trees? What rules that out?

Solution

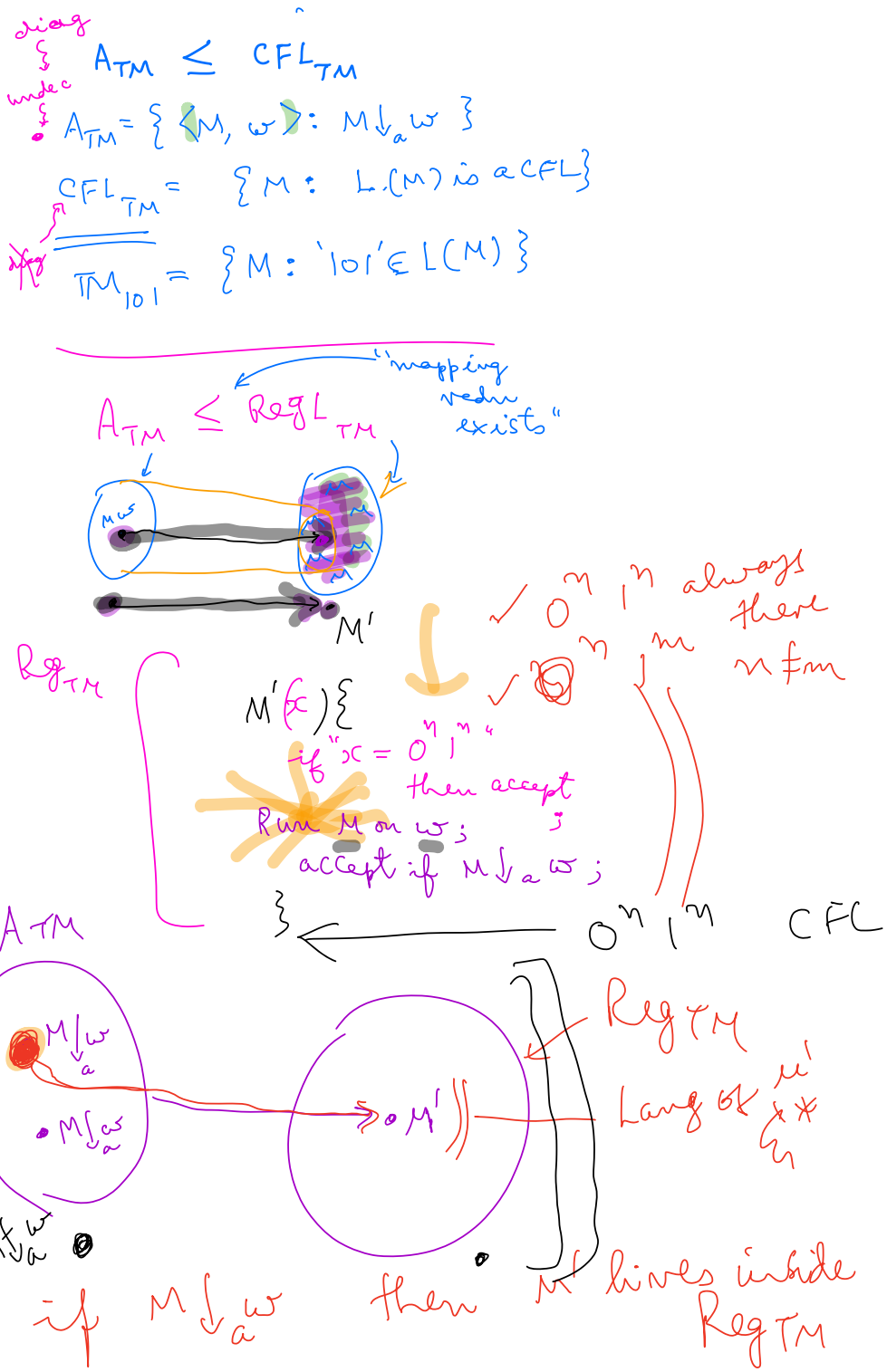
- We design a mapping reduction that takes a PCP instance and emits the desired CFG.
- By our design, we can see that we have woven the PCP instance tile upper and lower halves into two alternate production schemes that split-off at S_A and S_B.
- If the PCP instance has a solution, we can see a way to force ambiguity by forcing S_A and S_B to produce the same output string, namely a sequence of w_i followed by a_i or a sequence of x_i followed by a_i.
- But since the w_i and x_i sequences are the same (PCP instance has a solution), we will obtain it only by matching the a_i sequences; this shows that a parallel production scheme is to be taken via S_A and S_B.

  In more detail,

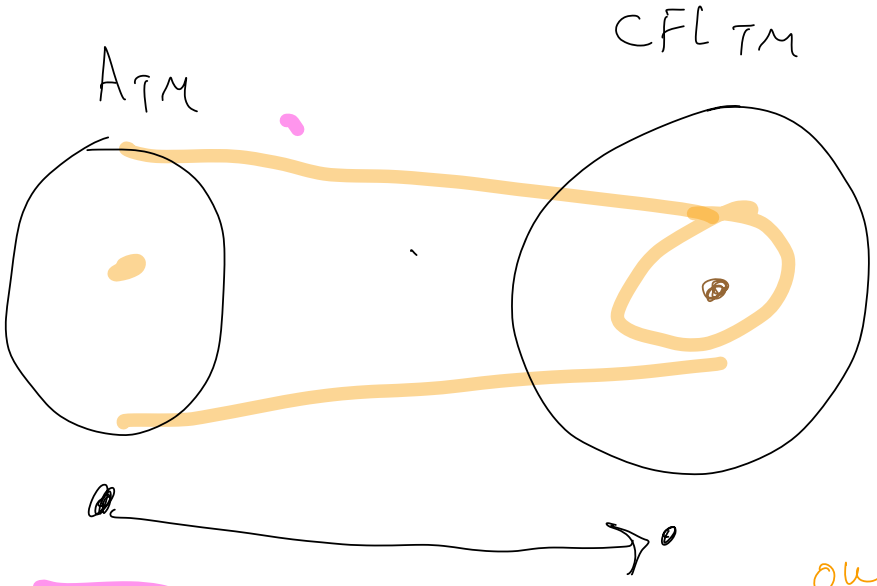
- We can see that if the PCP instance has no solution, there is no alternate production sequence that will match up the derived strings (the a_i tails must still match).
- Thus, there will be no string that has two parses.

- The a_i symbols are not present in Σ.
- Thus, any time a string has two parses, the "tail" a_i sequences must agree.
- Further, the "head" sequences in w_i and x_i must agree, which corresponds to the PCP instance having a solution.
- If the PCP instance has a solution, the mapped CFG will admit a common w_i and x_i sequence. The tail a_i sequence can be matched as well.
- This results in an ambiguous parse for some string that is derived by concatenating the PCP instance solution with a corresponding a_i tail sequence.
- Suppose the PCP instance has no solution.
- Thus, there is no arrangement of the individual w_is and matching x_is such that the w_i sequence agrees with the x_i sequence.
- Is it still possible that there is a common string generated by the CFG? This is impossible as the w_i and x_i do not have common symbols over the a_i.
- Thus the only way common strings can be formed is ruled out by the w_i and x_i sequence matches being ruled out – or that the CFG never generates ambiguous parses.

11. Review the P-time mapping reduction from 3SAT to Clique.
12. Show that if a 3CNF formula is a contradiction, then for any variable assignment, there is one clause that attains the value assignment "TTT" for its literals, and another clause that attains the value assignment "FFF"
13. Review the P-time mapping reduction from 3SAT to Three-color.


diag
undec
$A_{TM} \leq CFL_{TM}$
$A_{TM} = \{ \langle M, w \rangle : M \downarrow_a w \}$
$CFL_{TM} = \{ M : L(M) \text{ is a CFL} \}$
$TM_{101} = \{ M : '101' \in L(M) \}$
"mapping reduc exists"
$A_{TM} \leq RegL_{TM}$
M'
Reg_{TM}
M'(x){
if "x = $0^n 1^n$"
then accept
Run M on w;
accept if $M \downarrow_a w$;
}
$0^n 1^n$ always there
$n \neq m$
$0^n 1^n$ CFL
A_{TM}
$M \downarrow_a w$
$M \not\downarrow_a w$
M'
Reg_{TM}
Lang of M'
if $M \downarrow_a w$ then M' lives inside Reg_{TM}

But if $M \not\downarrow_a w$ then M' outside Reg_{TM}



outside CFL

$M'(x)$ {

if "$x = 0^n 1^n 2^n$" accept

Run M on w

accept if $M \downarrow_a w$ }

Σ^*