Reading:

The rest of the reading is: Chapter 5, sections 5.1 and 5.3, and Chapter 7, but taking the proof of the NP-completeness of SAT to be the “alternative proof” in Chapter 9. Chapter 6 is skipped, likewise sections 5.2 (and the “computation histories” part of 5.1 may be treated as a skim). One thing to be aware of is that the diagonal language $D$ is implicit in the text’s treatment of the undecidability of the halting problem (in whose proof it is counterfactually referred to as a “machine”); next Tuesday’s lecture will first re-state the fact that $D$ is not even Turing-acceptable and derive the undecidability of the halting problem as a consequence. The second thing is that the key definition of chapter 5 comes in section 5.3 not 5.1, namely mapping reductions. I prefer talking about them up-front.

Also review these notes in connection with the Thu. 4/20 lecture notes and problem (2): Although the intersection of two DCFls need not even be a CFL, the DCFLs are closed under any Boolean operation with a regular set. For instance, let $M_1$ be a DPDA and let $M_2$ be a DFA. Using the Cartesian product idea, we can combine the code of $M_2$ into that of $M_1$ so that whenever the DPDA $M_1$ makes an $R$-move upon reading a character $c$ on the input tape, the DFA $M_2$ also gets stepped forward on $c$. If $M_1$ stays stationary, $M_2$ does not get stepped but is held paused. Provided $M_1$ obeys “good housekeeping” so that it always reads all of its input $x$ (without being stuck in a loop, that is), $M_2$ is able to read all of $x$ as well. Then the combined machine $M_3$ gives an answer according to the Boolean combination of final states $q_1, q_2$ reached by $M_1$ and $M_2$ in the same way as when we had two DFAs, e.g. $(q_1 \in F_1 \land q_2 \in F_2)$ for intersection, $(q_1 \in F_1 \lor q_2 \in F_2)$ for union, and $(q_1 \in F_1 \text{ XOR } q_2 \in F_2)$ for symmetric difference. (The main reason why this doesn’t work for two DPDAs is that they might “clash” both for control of the stack, one wanting to push where the other wants to pop. Also note that if $M_1$ is an NPDA then the idea still works for $\cap$ and $\cup$ but not for symmetric difference—indeed, it can’t work for symmetric difference because $L \triangle \Sigma^* = \tilde{L}$ but the CFLs are not closed under complements.)

Assignment 9, due in hardcopy and in class Thu. 5/4*

(••except R4 may submit Fri. 5/5 before 2pm to my Davis 326 office)

Please write your name, Student ID#, and recitation attended atop your HW.

(1) Consider the following decision problem: Given a CFG $G = (V, \Sigma, R, S)$ with $\Sigma = \{a, b\}$, is $L(G) \cap a^+ \neq \emptyset$? That is, does $S$ generate one or more strings that consist only of one or more $a$’s? Sketch a decision procedure in prose similar to what the text does in section 4.1 with $E_{\text{CFG}}$ and what my lectures did when comparing the latter to the algorithm for whether $\epsilon \in L(G)$.

(You may if you wish use the conversion from $G$ to a $G'$ without $\epsilon$-rules as the first step of your procedure with no other comment needed on how to do it, but further steps should show all details including sketching any while-loops that may be needed. 24 pts.)

(2) Consider the following decision problem:

INSTANCE: A DFA $M = (Q, \Sigma, \delta, s, F)$ and two strings $u, v \in \Sigma^*$.

QUESTION: Does $L(M)$ contain all strings that can be formed by concatenating $u$ and $v$ as often as desired in any order?

(a) Give in pseudocode a decision procedure for this problem. (18 pts.)

(b) With reference to the above notes, tweak your procedure so that it solves the somewhat more general problem where $M$ can be given as a DPDA not just a DFA. (6 pts.)

(c) With reference to the above notes and facts in the lecture notes it references, say why the problem becomes undecidable when $M$ is allowed to be an NPDA not just a DPDA. (6 pts., for 30 total and 54 total on the set)