University of Nevada, Las Vegas Computer Science 456/656 Spring 2024

Answers to Assignment 7: Due Saturday April 27, 2024

1. Prove that every decidable language is enumerated in canonical order by some machine.

   Let L be a decidable langague over an alphabet Σ. Let $w_1, w_2, \ldots$ be the enumeration of Σ^* in canonical order. The following program enumerates L in canonical order.

   ```
   For all $i$ from 1 to $\infty$
       If $w_i \in L$ write $w_i$
   ```

2. Prove that every language that is enumerated in canonical order by some machine is decided by some other machine.

   Suppose some machine writes Let $w_1, w_2, \ldots,$ the canonical order of L. If L is finite, let w_n be the last item in the enumeration, otherwise, the enumeration is infinite. The following program decides whether a given $w \in \Sigma^*$ is a member of L.

   ```
   Read $w$
   For $i$ from 1 to $n$ if $L$ is finite, otherwise from 1 to $\infty$.
       If $w = w_i$
           Halt and accept $w$
       Else if $w \leq w_i$ (canonical order)
           Halt and reject $w$
   ```

   Prove that eny language accepted by any machine can be enumerated by some other machine.

   Let L be a language and M a machine that accepts L. If $w \in L$, then M will accept w in finitely many steps.

   Let $w_1, w_2, \ldots$ be the enumeration of Σ^* in canonical order. The following program enumerates L.

   ```
   For $t$ from 1 to $\infty$
       For $i$ from 1 to $t$
           If $M$ accepts $w_i$ within $t$ steps
               write $w_i$
   ```

   Each member of L will be written infinitely many times, but that is allowed for an enumeration.

3. Prove that any language which is enumerated by some machine is accepted by some other machine.

   Let M be a machine which writes an enumeration of L, $w_1, w_2, \ldots$ The following program will accept any string $w \in L$, and will not accept any string not in L.

   ```
   Read $w$
   For $i = 1$ to $\infty$
   ```

```
If $w = w_i$
    Halt and accept $w$
```

4. I have repeatedly stated in class that no language that has parentheses can be regular. For that to be true, there must be parenthetical strings of arbitrary nesting depth. (If you don't know what nesting depth is, look it up.)

   Some programming languages have limitations on nesting depth. For example, I have read that ABAP has maximum nesting depth of 256. (Who would ever want to go that far!)

   The Dyck language is generated by the following context-free grammar. (As usual, to make grading easier, I use a and b for left and right parentheses.)

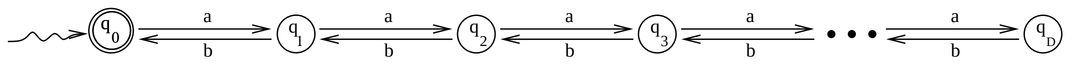
   1. $S \to aSbS$
   2. $S \to \lambda$

   (a) Use the pumping lemma to prove that the Dyck language is not regular.

   Let L be the Dyck language. Suppose that L is regular. Let p be a positive integer which is a pumping length of L. Let $w = a^p b^p$ which is member of L. Then there exist strings x, y, z such that
   1. $w = xyz$
   2. $|xy| \leq p$
   3. $|y| \geq 1$
   4. For any integer $i \geq 0$, $xy^i z \in L$.
   By 2., xy is a prefix of $xyz = w = a^p b^p$ of length at most p, hence consists entirely of a's. Thus $y = a^k$, and $k > 0$ by 3. By 4. $xyyz = a^{p+k} b^p \in L$, contradiction.

   (b) Let D be any finite integer. Let L be the language consisting of all members of the Dyck language whose nesting depth does not exceed D. Prove that L is regular.

   The following DFA accepts L. It has one live state for each possible nesting depth of the current input.



5. We know that context-free languages are exactly those which are accepted by push-down automata. We now define a new class of machines, which we call "limited push-down automata." An LPDA is exactly the same as a PDA, but with the restriction that the stack is never allowed to be larger than some given constant. What is the class of languages accepted by limited push-down automata? Prove your answer.

   The class of regular languages.

   Proof: If L is regular, L is accepted by some NFA M_1 Let M_2 be the PDA whose states are the states of M_1, and where, at each step, the bottom-of-stack symbol z is popped off the stack and then pushed back on, and nothing else is done with the stack. Then M_2 is an LPDA and is basically the same as M_1, hence accepts L.

   Conversely, suppose M_1 is an LPDA where the stack size is limited to D. Suppose M_1 has n states, and the stack alphabet of M_1 has m symbols. Recall that an **i.d.** of a PDA is a pair consisting of the current state and the stack contents. Thus the number of instantaneous desciptions of M_1 is not greater than nm^D, which is finite. Let M_2 be the DFA which one state for each **i.d.** of M_1, where a transition given an input a corresponds to a move by M_1 given input a. Then L is accepted by an NFA, hence regular. ▌

6. Prove that every context-sensitive language is decidable. The way to do this is to start with an arbitrary context-sensitive grammar, using the definition I gave in class (that's not the only definition) namely that the right side of any production must be at least as long as the left side, and then design a program which decides whether any given string is generated by that grammar.

   Let G be a context-sensitive grammar, and $L = L(G)$. Let Σ be the terminal alphabet of G, that is, $L \subseteq \Sigma^*$. Let Γ be the variable alphabet of G, and $S \in \Gamma$ the start symbol of G. By definition, a *sentential form* of G is any string which is derived from S in any number of steps, using productions of G. Let $\mathfrak{S} \subseteq (\Sigma + \Gamma)^*$ be set of all sentential forms of G. Note that $L = \Sigma^* \cap \mathfrak{S}$. We let $\mathfrak{S}_m$ be the set of all sentential forms of G of length m.

   Let $w \in \Sigma^*$, and let $n = |w|$. The following algorithm decides whether $w \in L$.

   Let $\mathfrak{S}_1 = \{S\}$. For each m from 2 to n, compute $\mathfrak{S}_m$ from $\mathfrak{S}_i$ for all $i < m$, by repeatedly using the following construction. If G has the production $\alpha \to \beta$, and if $\phi\,\alpha\,\psi \in \mathfrak{S}$ for some $\phi, \psi \in (\Sigma + \Gamma)^*$, then $\phi\beta\psi \in \mathfrak{S}$. By the non-decreasing rule of CS productions, $|\phi\,\alpha\,\psi| \leq |\phi\,\beta\,\psi|$. Finally, $w \in L$ if and only if $w \in \mathfrak{S}$.

   The above algorithm is clearly EXP–SPACE, since G has, in general, exponentially many sentential forms of length n.

   Can we do better? Is there a $\mathcal{P}$–SPACE EXP–TIME algorithm for the problem? I'm not sure. What do you think?