CPSC 421/501 Note Sheet for Final Exam, Fall 2014

A set, S, is *countable* if we can write

$$S = \{s_1, s_2, \ldots\},$$

and otherwise *uncountable*, i.e., meaning that any sequence of its elements does not contain all of the set.

The power set of a set, S, denoted $\mathcal{P}ower(S)$ or 2^S is the set of all subsets of S. We know that there is no function $f\colon S \to \mathcal{P}ower(S)$ whose image is all of $\mathcal{P}ower(S)$.

The set of all strings over a countable set is countable. The set of subsets of a countably infinite set is uncountable. In many contexts, the set of "programs" or "algorithms" is countable, while the set of languages is uncountable; in this case, there are many languages which cannot be "recognized" or "solved" by a program or algorithm.

Axiom 1: There exists a Result function, from $\mathcal{P} \times \mathcal{I}$ to $\{\texttt{yes}, \texttt{no}, \texttt{NoHalt}\}$. Axiom 2: There exists a universal program. Axiom 3: One can modify the `yes` and `no` results of a program. Axiom 4: One can modify a program so that inputs of the form $\langle p \rangle$ on the modified program run the original program on the input $\langle p, \langle p \rangle \rangle$. Axiom 5: One can combine two programs and wait for one of them to say `yes`.

A program, $p \in \mathcal{P}$ *recognizes* the language

$$L = L_p = \{i \in \mathcal{I} \mid P[i] = \texttt{yes}\}.$$

A program is a *decider* if on any input its result is either `yes` or `no`. A langauge is *recognizable* if it is recognized by some program, and *decidable* if it is recognized by a some program that is a decider.

If a language is recognizable but not decidable, then its complement is unrecognizable (i.e., not recognized by any element of $\mathcal{P}$).

"421Simple" is an example of a simple programming language that produces algorithms in a similar way to Turing machines. It has the keywords:

`INPUT`, `WORKTAPE`, `OUTPUT`, `RESET`, `AUG`, `DEC`, `LET`, `EOF`, $=$, `IF`, `THEN`, `GOTO`, `END`, `COMMENT`.

A Turing machine is a tuple:

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

and an understood "blank" symbol that is in Γ but not in Σ; Q is the set of states, Σ is the input alphabet, Γ is the worktape alphabet,

$$\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{\texttt{L}, \texttt{R}\}.$$

Savitch's Theorem: $\mathrm{NSPACE}(f(n)) \subset \mathrm{SPACE}((f(n))^2)$ provided that $f(n)$ is computable in $\mathrm{SPACE}((f(n))^2)$. Hence NPSPACE equals PSPACE.

Our half of the Baker-Gill-Soloway Theorem: $\mathrm{P}^A = \mathrm{NP}^A$ where A is any language complete for PSPACE.

There are “easy” examples of NP-complete and PSPACE-complete languages:

$$L_{\text{NP easy}} = \{\langle M, i, 1^t\rangle \mid M \text{ is a non-det TM that accepts } i \text{ in time } t\ \}$$

(the term 1^t is the string of 1’s of length t, i.e., t written out in unary); and

$$L_{\text{PSPACE easy}} = \{\langle M, i, 1^s\rangle \mid M \text{ is a TM that accepts } i \text{ in space } s\ \}.$$

UTM’s (universal Turing machines) are used to prove the Time Hierarchy Theorem. If you simulate s steps of a Turing machine by a UTM in time $O(s^2)$, then you can conclude that $\mathrm{TIME}(n^a)$ is a proper subset of $\mathrm{TIME}(n^b)$ for $b > 2a$. If you simulate s steps of a Turing machine by a UTM in time $O(s \log s)$, then you can conclude that $\mathrm{TIME}(n^a)$ is a proper subset of $\mathrm{TIME}(n^b)$ for $b > a$.

A DFA is a tuple $(Q, \Sigma, \delta, q_0, F)$ where Q and Σ are finite sets, $\delta\colon Q \times \Sigma \to Q$, $q_0 \in Q$, and $F \subset Q$ (F is the set of final or accepting states).

Myhill-Nerode Theorem: For a langauge $L \subset \Sigma^*$ and $w \in \Sigma^*$, we define

$$\mathrm{Future}(L, w) = \{u \in \Sigma^* \mid wu \in L\}.$$

Then the number of different futures of a language is the minimum number of states in a DFA recognizing L.