Noncomputability lecture summary

M. George

October 20, 2014

1 Why are automata useful?

- They are a nice abstraction to have in mind when building certain kinds of programs
  - Network protocols (alphabet is send and receive messages)
  - Games (alphabet is player actions, state is character)
  - Thread state in an operating system (started, running, waiting for input ...)
  - For one way to use these ideas in your programs, see the "State design pattern".
- They are closely related to regular expressions, which are a useful tool for text processing (including compilers).
- Having a formal model of computation allows us to prove things about *all possible programs*. For example, we can prove that there are languages that cannot be recognized by any program.

2 Pumping lemma

We expanded on Sid's proof that a machine with n states cannot count past n. This is formalized in the *pumping lemma*:

Pumping lemma: Given a machine M having m states, and a string $x \in L(M)$ with $|x| > m$, there exists strings u, v, and w, with $|v| \geq 1$, such that $x = uvw$ (the concatenation of u, v and w), and for all $c \geq 0$, $uv^c w \in L$.

The proof is similar to the proof given by Sid; since $|x| \geq m$, we must repeat some state while processing x. Let v be the substring of x that is processed within the loop, and let u be the string that takes the machine from q_0 to q_i and w the string that takes q_j to a final state $q_f \in F$. The fact that we can "pump" v corresponds to the fact that we can go around the loop any number of times.

Note: once we introduce a bit more formal notation, it would be a good exercise to write out this proof more formally.

3 Using the pumping lemma

We proved that the language $L = \{0^n 1^n \mid n \in \mathbb{N}\}$ is not recognizable.

Proof: Suppose (for contradiction) that there is a machine M with $L(M) = L$. Suppose M has m states. We know $0^m 1^m \in L$. But $|0^m 1^m| > m$ (which is the number of states of M), so we can apply the pumping lemma to split x into uvw. v can occur in one of three places in the string x; it can contain only zeros, only ones, or some number of zeros followed by some number of ones. The pumping lemma tells us that $uvvw \in L(M)$, but this gives a contradiction in each of the three cases. In the first case, $uvvw$ has more zeros than ones, so it is not in L. In the second case, $uvvw$ has more ones than zeros, and is thus not in L. In the third case, $uvvw$ contains some ones followed by some zeros, and is thus not in L. In all three possible cases we have shown that $uvvw \notin L$ which contradicts the fact that $uvvw \in L(M)$. Thus there can be no machine M.

4 Non-computability

The above shows that not all languages are DFA-recognizable. This raises the question: are there more powerful models that can recognize any language?

Choose your favorite language (say Java). We have a function L from Java programs to sets of strings: on a given program P,

$$\begin{aligned} L : &\quad \text{The set of all programs} \to 2^{\Sigma^*} \\ L : &\quad P \mapsto \{x \in \Sigma^* \mid P(x) \text{ returns true}\} \end{aligned}$$

Asking whether there are any uncomputable sets is the same as asking if this function is surjective. Answer: it cannot be, because the set of all programs is a subset of the set of all strings, and thus is countable (you proved this on the homework), but the set of all languages is uncouontable (you also proved this on the homework).

Thus there are sets that Java programs cannot recognize.