

Harvard University
Computer Science 121

Section 6 Notes
Week of 11.2.09

1 Outline

- Deciding/Recognizing languages
- Homomorphisms
- General grammars

2 Deciders and Recognizers

A **Turing-decidable** language is:

A **Turing-recognizable** language is:

- What does it mean for a language to be decidable? How do we show a language is decidable? What are some nice consequences of having a decidable language?
- What is enough to show that a language is recognizable?

2.1 TM Encoding and TMs as Hardware/Software

As discussed in lecture, there is a general way for specifying Turing machines so that their descriptions can be used as inputs to other Turing machines. Let M be a Turing Machine. Then we use $\langle M \rangle$ to mean the string encoding of M . The details of the TM encoding itself are not important. All that is important is that such an encoding exists so that we can feed Turing Machine encodings to other machines.

Note, in particular, that this lets us be much more flexible in designing TMs that use other TMs. Back when we were pasting together NFAs, we had to treat the NFAs we were pasting together as pure hardware: we could reach into them and attach wires here and there, but we couldn't do much besides treat them as black boxes. But because Turing machines are capable of interpreting the encodings of other Turing machines, we can use TMs as *software*, too.

On the one hand, we don't need to know which TM we're simulating until run-time. This is what the universal TM U does: it takes an encoding and simulates it. Note that the M it simulates is *not* built into the states of U . In general, a TM simulating another can do anything a good debugger could do: free run, run until some specified breakpoint, run for some number of steps, run from some particular starting configuration, modify the value of registers, perform logging, or inspect the output and branch based upon it.

2.2 Showing that a language is decidable

To prove that a language is decidable, we must describe a Turing Machine that decides it. This doesn't necessarily mean getting into the nuts and bolts of the Turing Machine, but it simply means describing a general algorithm for finding a yes or no answer to valid input strings. A good example is Theorem 4.5 on page 155 of Sipser.

$EQ_{DFA} = \{\langle A, B \rangle : A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$ is decidable. To prove this we construct a new DFA C from A and B that accepts only those strings that are accepted by either A or B but not both. Thus, if A and B recognize the same language, C will accept nothing. The language of C is $L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$. We can construct C from A and B using constructions for proving regular languages closed under complementation, union, and intersection. These constructions can be carried out by Turing Machines. Then we test to see if $L(C)$ is empty. If so, then $L(A)$ and $L(B)$ must be equal. On input $\langle A, B \rangle$, where A and B are DFAs:

1. Construct DFA C as described.
2. Mark the start state of C .
3. Until no new states get marked, mark any state that has a transition coming into it from any state that is already marked.
4. If no accept state is marked, *accept*; otherwise *reject*.

Exercise 2.1 Show that the language

$$L = \{\langle M, w \rangle \mid M \text{ never moves its head left when running on } w\}$$

is decidable.

2.3 Dovetailing

Dovetailing is an important general technique for simulating an infinite number of computations “in parallel”. It’s explained in the the 10/27 lecture notes, slide 15.

3 Homomorphisms

A *homomorphism* is defined as follows:

Let Σ and Δ be alphabets. Consider a function h from Σ to Δ^* . Extend h to a function from Σ^* to Δ^* by applying it to each symbol in its input. More formally:

$$\begin{aligned} h(\varepsilon) &= \varepsilon \\ h(w\sigma) &= h(w)h(\sigma), \text{ for any } w \in \Sigma^*, \sigma \in \Sigma \end{aligned}$$

Any function $h : \Sigma^* \rightarrow \Delta^*$ defined in this way from a function $h : \Sigma \rightarrow \Delta^*$ is called a **homomorphism**.

Note that homomorphisms can “erase”: $h(w)$ may be ε , even though w is not.

Exercise 3.1 *Show that the regular languages are closed under homomorphisms.*

4 General Grammars

One of the models equivalent to Turing machines that we talked about were the general grammars. Recall that general grammars are a more powerful version of context-free grammars. In any rule $u \rightarrow v$ of a general grammar, unlike context-free grammars, u may be any string of terminals and nonterminals, so long as it contains at least one nonterminal.

Exercise 4.1 *Write a general grammar that generates the language $\{a^{n^2} : n \geq 0\}$.*

Exercise 4.2 *Write a general grammar that generates the language:*

$$\{w\#1^n : n \geq 1 \text{ and } w \text{ is the binary representation of } n\}$$