

**Harvard CS 121 and CSCI E-207**  
**Lecture 14:**  
**Decidability, Recognizability,  
and a Universal Turing Machine (II)**

Harry Lewis

October 29, 2009

- Reading: Sipser §4.1.

## Decidability

- Recall that a language  $L \subseteq \Sigma^*$  is decidable if there is a TM that always halts when started on an input in  $\Sigma^*$ , in either  $q_{\text{accept}}$  if  $w \in L$  or  $q_{\text{reject}}$  if  $w \notin L$ .
- **Proposition:** Every regular language is decidable.

**Proof:** (By “coding” a DFA as a TM.)

## Asking questions about arbitrary finite automata

- **Q:** What if the DFA  $D$  is part of the input? That is can we design a single TM that, given two inputs,  $D$  and  $w$ , decides whether  $D$  accepts  $w$ ?
  - The TM needs to use a fixed alphabet & state set for all inputs  $D, w$ .
- **Q:** How to represent  $D = (Q, \Sigma_D, \delta, q_0, F)$  and  $w$ ? List each component of the 5-tuple, separated by |'s.
  - Represent elements of  $Q$  as binary strings over  $\{0, 1\}$ , separated by , 's.
  - Represent elements of  $\Sigma_D$  as binary strings over  $\{0, 1\}$ , separated by , 's.
  - Represent  $\delta : Q \times \Sigma_D \rightarrow Q$  as a sequence of triples  $(q, \sigma, q')$ , separated by , 's, etc.

We denote the encoding of  $D$  and  $w$  as  $\langle D, w \rangle$ .

## A “Universal” algorithm for deciding regular languages

- **Proposition:**  $A_{\text{DFA}} = \{\langle D, w \rangle : D \text{ a DFA that accepts } w\}$  is decidable.

### Proof sketch:

- First check that input is of proper form.
- Then simulate  $D$  on  $w$ . Implementation on a multitape TM:
  - Tape 2: String  $w$  with head at current position (or to be precise, its representation).
  - Tape 3: Current state  $q$  of  $D$  (i.e., its representation).
- Could work with other encodings, e.g. transition function as a matrix rather than list of triples.

## Representation independence

- **General point:** Notions of computability (e.g. decidability and recognizability) are independent of data representation.
- A TM can convert any reasonable encoding to any other reasonable encoding.
- We will use  $\langle \cdot \rangle$  to mean “any reasonable encoding”.
- We’ll need to revisit representation issues again when we discuss computational *speed*.
- For the moment when we are interested only in whether problems are decidable, undecidable, recognizable, etc., so we can be content knowing that there is *some* representation on which an algorithm could work.

# Describing Turing Machines

## Formal Description

- 7-tuple or state diagram
- Most of the course so far

## Implementation Description

- Prose description of tape contents, head movements
- Previous lecture and today's lecture so far

## High-Level Description

- Does not refer to specific computational model, data representation
- From now on!

## More Decidable Problems

- $\{\langle R, w \rangle : R \text{ is a regular expression that generates } w\}$ .
- $\{\langle X \rangle : X \text{ is an DFA/NFA/RE such that } L(X) = \emptyset\}$ .
- $\{\langle X \rangle : X \text{ is a DFA/NFA/RE such that } |L(X)| = \infty\}$ .
- $\{\langle M, w \rangle : M \text{ is a PDA that accepts } w\}$ .
- Every context-free language.

## A Universal Turing machine

**Theorem:** There is a Turing machine  $U$ , such that when  $U$  is given  $\langle M, w \rangle$  for any TM  $M$  and  $w$ ,  $U$  produces the same result (accept/reject/loop) as running  $M$  on  $w$ .

**Proof:** Initially,

- First tape contains  $\langle M \rangle$ , including in particular its transition function  $\delta_M$ .
- Second tape contains  $\langle w \rangle$ .
- Third tape contains  $\langle q_{\text{start}} \rangle$ .
- Simulate steps of  $M$  by multiple steps of  $U$ .

(Brief return to implementation description.)

$\Rightarrow$  Turing machines can be “programmed”.

# Consequences of the Existence of Universal Turing Machines

- **Corollary:**  $A_{\text{TM}} = \{\langle M, w \rangle : M \text{ accepts } w\}$  is Turing-recognizable (r.e.).
- **Corollary:**  $HALT_{\text{TM}} = \{\langle M, w \rangle : M \text{ eventually halts on } w\}$  (“The Halting Problem”) is r.e.
- **Q:** What about  $\{\langle M, w, n \rangle : M \text{ halts on } w \text{ in at most } n \text{ steps}\}$ ?
- **Q:** Are  $A_{\text{TM}}$  and  $HALT_{\text{TM}}$  decidable?
- **Q:** Are there undecidable languages?

## Three basic facts on the recursive vs. r.e. languages

1. If  $L$  is recursive, then  $L$  is r.e.

Proof:

## Three basic facts on the recursive vs. r.e. languages

1. If  $L$  is recursive, then  $L$  is r.e.

Proof:

If  $M$  decides  $L$ , then a machine can recognize  $L$  by running  $M$ , and then going into an infinite loop if  $M$  would have halted in the  $q_{\text{accept}}$  state.

2. If  $L$  is recursive then so is  $\bar{L}$ .

Proof:

## Three basic facts on the recursive vs. r.e. languages

1. If  $L$  is recursive, then  $L$  is r.e.

Proof:

If  $M$  decides  $L$ , then a machine can recognize  $L$  by running  $M$ , and then going into an infinite loop if  $M$  would have halted in the  $q_{\text{accept}}$  state.

2. If  $L$  is recursive then so is  $\bar{L}$ .

Proof:

A machine can decide  $\bar{L}$  by running  $M$  and then giving a “no” answer when  $M$  would give “yes” and vice versa.

3.  $L$  is recursive if and only if both  $L$  and  $\bar{L}$  are r.e.

Proof: . . .