CSCI 340: Computational Models

Turing Machines

Chapter 19

Department of Computer Science

The Turing Machine

- **Regular Expressions**

Acceptor:	FA, TG
Nondeterminism equal?	Yes
Closed Under:	$L_1 + L_2$ $L_1 L_2$ L^* L' $L_1 \cap L_2$
Decidability:	Equivalence, emptiness, finiteness, membership
Examples:	Text editors, Seq. Circuits

- **Context-Free Grammars**

Acceptor:	PDA
Nondeterminism equal?	No
Closed Under:	$L_1 + L_2$ $L_1 L_2$ L^*
Decidability:	Emptiness, finiteness, membership
Examples:	Programming Language Statements, Compilers

The Turing Machine

- **Regular Expressions**

Acceptor:	FA, TG
Nondeterminism equal?	Yes
Closed Under:	$L_1 + L_2 \quad L_1 L_2 \quad L^* \quad L' \quad L_1 \cap L_2$
Decidability:	Equivalence, emptiness, finiteness, membership
Examples:	Text editors, Seq. Circuits

- **Context-Free Grammars**

Acceptor:	PDA
Nondeterminism equal?	No
Closed Under:	$L_1 + L_2 \quad L_1 L_2 \quad L^*$
Decidability:	Emptiness, finiteness, membership
Examples:	Programming Language Statements, Compilers

- **Type 0 Grammars**

Acceptor:	Turing machine, Post machine, 2PDA, *n*PDA
Nondeterminism equal?	Yes
Closed Under:	$L_1 + L_2 \quad L_1 L_2 \quad L^* \quad L_1 \cap L_2$
Decidability:	Not a whole lot
Examples:	Computers

Turing Machines

- We can finally represent and model a computer!
- But when were all of these invented?

1950s: Regular Languages, FAs by Kleene, Mealy, Moore, Rabin, Scott
1960s: CFGs and PDAs by Chomsky, Oettinger, Schützenberger, Evey
1930s: Turing machines and Theory by Turing and Post

Turing Machines

Definition

A **Turing Machine**, denoted TM, is a collection of six things:

1. An alphabet Σ of input letters which does not contain the blank symbol Δ
2. A **TAPE** divided into numbers cells, each containing a character or a blank
3. A **TAPE-HEAD** that can in one step *READ* the contents of a cell, *WRITE* a different character to a cell, and/or *MOVE* left/right one cell. *It cannot move "left" of the beginning of the tape.*
4. An alphabet Γ of characters that can be written to the **TAPE** by the **TAPE-HEAD**. Γ can include Σ. The **TAPE-HEAD** can also print Δ but this is called *erasing*

Turing Machines

Definition

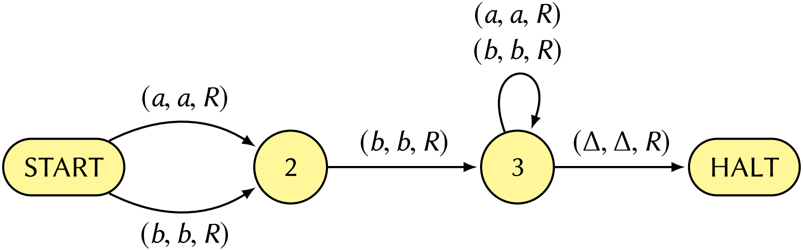
5. A finite set of states including exactly one START state and (maybe) some HALT states that cause execution to terminate.
6. A **program** which is a set of rules to tell us that tell how the state should change
    - Based on the state we are in and the letter the **TAPE-HEAD** has just read, we may change states, print to the **TAPE**, and move the **TAPE-HEAD**.
    - The program is collection of directed edges connecting states together.
    - Each edge is labeled with *(letter, letter, direction)*

Our First Turing Machine

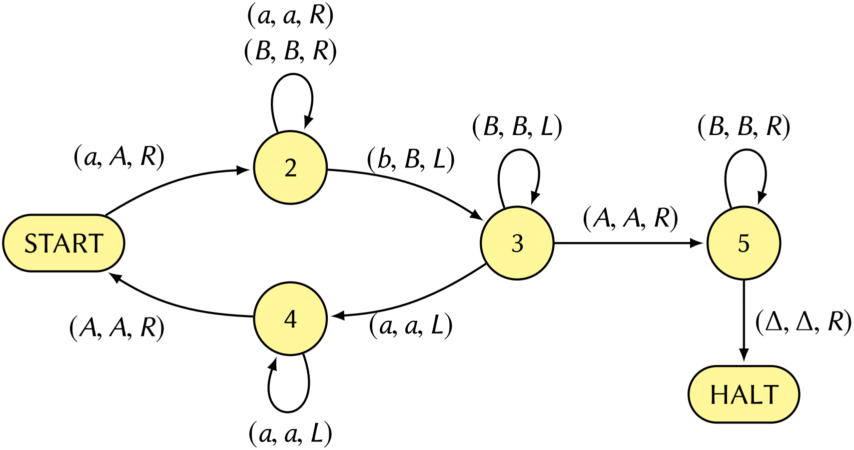
Tape:

a	b	a	Δ	Δ	Δ	

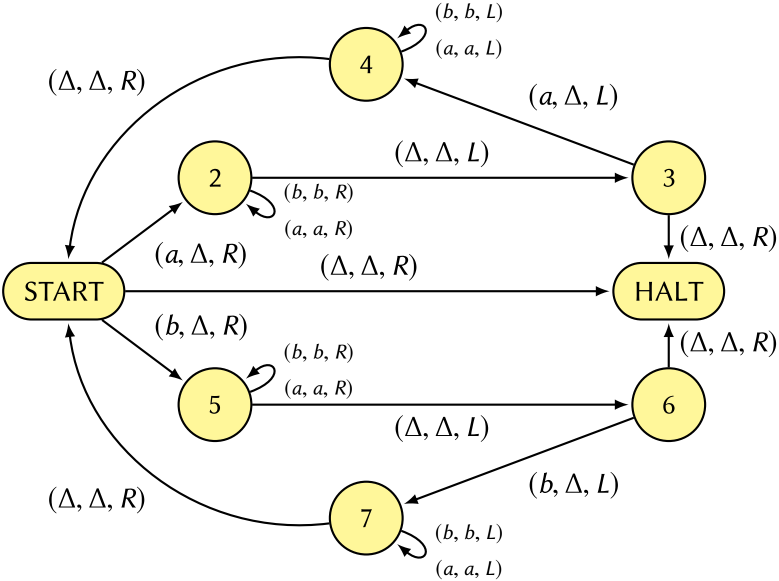
Program:



Another Example — *aaabbb*



Another Another Example — *abaaba*



Regular Languages and Turing Machines

Theorem

Every regular language has a TM that accepts exactly it.

Proof.

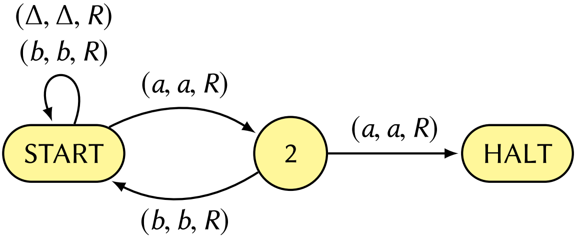
- change all edge labels a and b to (a, a, R) and (b, b, R) respectively
- change the initial state to START
- create a new HALT state
- "toggle" the accepting states and add (Δ, Δ, R) transitions to HALT

□

Example

EVEN-EVEN

Regular Language Example



Consider the following cases:

1. Strings with a double *a*
2. Strings without *aa* that end in *a*
3. Strings without *aa* that end in *b*

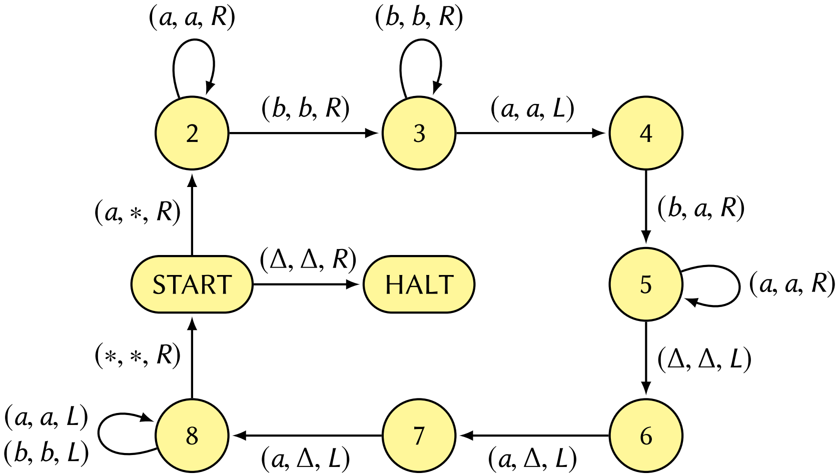
Classes of “Acceptance”

Definition

Every Turing Machine T over the alphabet Σ divides the set of input strings into three distinct classes:

1. **ACCEPT**(T) is the set of all strings leading to a HALT state. This is also called the *language accepted* by T
2. **REJECT**(T) is the set of all strings that crash during execution by either moving left from our first “cell” or by being in a state that has no exit edge by reading the character **TAPE-HEAD** is reading
3. **LOOP**(T) is the set of all other strings, that is, strings that loop forever while running on T

A Turing Machine accepting $L = \{a^n b^n a^n\}$



The INSERT Subprogram

- We would like to be able to *insert* a character into the string on the TAPE where the TAPE-HEAD is currently pointing.
- This action should not otherwise impact the tape in any way — it is *independent*
- We wish to introduce a new “command” or state for our Turing Machine called INSERT.

$\boxed{\text{INSERT } a}$

INSERT

The DELETE Subprogram

- We would also like to be able to *delete* a character from the string on the TAPE where the TAPE-HEAD is currently pointing.
- This action should not otherwise impact the tape in any way — it is *independent*
- We wish to introduce a new "command" or state for our Turing Machine called DELETE.

DELETE

- For example, if the string on our tape is *F***R***IEND* and *R* is where the tape head is pointing, after calling DELETE, *FIEND* is the string on the tape.

DELETE

Homework 10b

3. (5pt) Build a TM that accepts the language of all words that do not contain the substring *bbb*
4. (5pt) Build a TM that accepts $\{ a^n b^{2n} \}$
5. (5pt) Trace *aabbaa* on the Turing Machine on Slide 11
6. (5pt) Trace *aabbaa* on the Turing Machine on Slide 7