CSCI 340: Computational Models

Turing Machine Languages

Chapter 23

Department of Computer Science

Languages: their Definers and Acceptors

- **Regular** languages
  Defined via *regular expressions*
  Accepted by Finite Automata
- **Context-Free** languages
  Defined via *context free grammars*
  Accepted by Push Down Automata
- **???** languages
  Defined via *???*
  Accepted by Turing Machines

Well, what goes above?

Languages Accepted by Turing Machines

Definition

A language L defined over the alphabet Σ is called **recursively enumerable** if there is a Turing Machine T that accepts every word in L and either rejects (crashes) or loops forever for every word in the language L' (the complement of L). *Often abbreviated r.e.*

$$\begin{aligned} \text{accept}(T) &= L \\ \text{reject}(T) + \text{loop}(T) &= L' \end{aligned}$$

Remember the turing machine that "looped" forever on a regular language defined by the regular expression $(\mathbf{a} + \mathbf{b})^*\mathbf{aa}(\mathbf{a} + \mathbf{b})^*$?

accept(T) = all words with *aa*

reject(T) = strings without *aa* ending in *a*

loop(T) = string all without *aa* ending in *b*, or λ

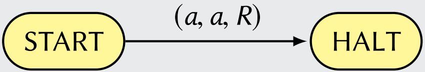
Languages Accepted by Turing Machines

Definition

A language L defined over the alphabet Σ is called **recursive** if there is a Turing Machine T that accepts every word in L and rejects (crashes) for every word in the language L' (the complement of L).

$$\text{accept}(T) = L$$
$$\text{reject}(T) = L'$$
$$\text{loop}(T) = \emptyset$$

Example



Operations on Recursive Languages

Theorem

If the language L is recursive, then its complement (L') is also recursive. In other words, the recursive languages are closed under complementation.

Proof.

- No word will loop when "run" on the machine
- Convert the Turing Machine to a Post Machine
- Introduce a REJECT state and have all unaccounted deterministic paths lead to this new REJECT state
- Relabel all REJECT states as ACCEPT and all ACCEPT states as REJECT
- This new machine ACCEPTs everything the original machine REJECTed and REJECTs everything the original machine ACCEPTed

$\square$

Recursively Enumerable Languages

Theorem

If L is recursively enumerable (r.e.) and L' is also recursively enumerable, then L is recursive

A part of the Proof...

Assuming there is a TM T_1 that accepts L and a TM T_2 that accepts L'. We then construct $T_2{}'$ such that:

$$L' = \text{accept}(T_2) = \text{reject}(T_2{}')$$
$$\text{loop}(T_2) \subset \text{loop}(T_2{}')$$
$$\text{reject}(T_2) \subset \text{loop}(T_2{}')$$

We then construct $T_1{}'$ such that:

$$\text{accept}(T_1{}') = L = \text{loop}(T_2{}')$$
$$\text{loop}(T_1{}') = L' = \text{reject}(T_2{}')$$

Union

Theorem

If T_1 and T_2 are TMs, there exists a TM, T_3 such that

$$\text{accept}(T_3) = \text{accept}(T_1) + \text{accept}(T_2)$$

Proof.

- Make both TMs loop instead of crash
- Nothing stops the two machines from running in alternation given the construction algorithm fully outlined in the prior proof

□

Intersection

Theorem

The intersection of two recursively enumerable languages is also recursively enumerable

Proof.

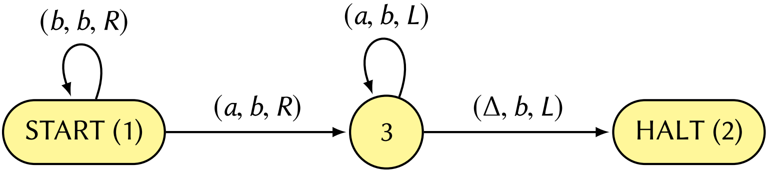
Assume TM_1 is the first TM and TM_2 is the second TM

1. Build a TM preprocessor that takes a two-track TAPE and copies from track 1 to track 2. Always start on TM_1
2. Convert TM_1 to a 2-track TAPE doing all of its processing but referring only to the first track. Change HALT of TM_1 to a state that rewinds the TAPE HEAD to the first cell and branch to the START of TM_2
3. Convert TM_2 into 2-track TAPE doing all of its processing but referring only to the second track.

□

The Encoding of Turing Machines

We can represent Turing Machines as tables rather than as a picture



From	**To**	**Read**	**Write**	**Move**
1	1	b	b	R
1	3	a	b	R
3	3	a	b	L
3	2	Δ	b	L

Coding a Turing Machine

Consider the general row:

From	To	Read	Write	Move
X_1	X_2	X_3	X_4	X_5

- X_1, X_2 are numbers
- $X_3, X_4 \in \{a \;\; b \;\; \#\}$
- $X_5 \in \{L \;\; R\}$

Encoding the row:

- X_1 and X_2 get encoded as: $a^{X_1} b a^{X_2} b$
- X_3 and X_4 get mapped to one of the four strings:
  $a \to aa \qquad b \to ab \qquad \Delta \to ba \qquad \# \to bb$
- X_5 is encoded as a if it's L and b if it's R
- Finally, concatenate all parts together

Coding Example

From	To	Read	Write	Move
6	2	b	a	L

- $X_1, X_2 = a^6ba^2b = aaaaaa\ \ b\ \ aa\ \ b$
- $X_3 =$ "b" $= ab$
- $X_4 =$ "a" $= aa$
- $X_5 =$ "L" $= a$
- Encoding: **aaaaaa b aa b ab aa a**

Every row is a string of a's and b's that is defined by the RE:

$$\mathbf{a}^+\mathbf{b}\mathbf{a}^+\mathbf{b}(\mathbf{a}+\mathbf{b})^5$$

(at least one a) b (at least one a) b (five letters)

Code Word Language

From	**To**	**Read**	**Write**	**Move**	**Code for Each Row**
1	1	b	b	R	*ababababb*
1	3	a	b	R	*abaaabaaabb*
3	3	a	b	L	*aaabaaabaaaba*
3	2	Δ	b	L	*aaabaabbaaba*

One code word for the entire machine is:

ababababbabaaabaaabbaaabaaabaaabaaaabaabbaaba

But this isn't the only code for the machine because the "order" of the rows in the table isn't rigid.

$$\text{CWL} = \text{the language defined by } (\mathbf{a}^{+}\mathbf{b}\mathbf{a}^{+}\mathbf{b}(\mathbf{a}+\mathbf{b})^{5})^{*}$$

A Non-Recursively Enumerable Language

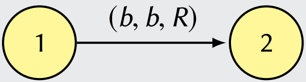
- The code word for a TM contains all the information of the TM
- Since it's just composed of *a*'s and *b*'s it could be used as **input**
- What if we run the TM with the code word as input?

Definition

The Language ALAN is defined as the following:

ALAN = { all the words in CWL that are **not** accepted by the TMs they represent or that do not represent any TM }

Example



Code word: $abaabababb \in ALAN$

More on ALAN

- If a TM accepts everything, then its code word is not in ALAN
- If a TM rejects everything, then its code word is in ALAN
- If a code word is malformed then its in ALAN
- The code word for a TM accepting PALINDROME is not a PALINDROME; therefore, this code word is in ALAN

Approach: We will show that ALAN cannot be r.e. by contradiction

Claim: ALAN **is** recursively enumerable... so there is a TM, T, that accepts it

Question: Is code(T) a word in the language ALAN or not?

ALAN

CASE 1: code(T) is in ALAN

CLAIM	REASON
1. T accepts ALAN.	1. Definition of T.
2. ALAN contains no code word that is accepted by the machine it represents.	2. Definition of ALAN.
3. code(T) is in ALAN.	3. Hypothesis.
4. T accepts the word code(T).	4. From 1 and 3.
5. code(T) is not in ALAN.	5. From 2 and 4.
6. Contradiction.	6. From 3 and 5.
7. code(T) is not in ALAN.	7. The hypothesis (3) must be wrong because it led to a contradiction.

ALAN

CASE 2: code(T) is not in ALAN

CLAIM	REASON
1. T accepts ALAN.	1. Definition of T.
2. If a word is not accepted by the machine it represents, it is in ALAN.	2. Definition of ALAN.
3. code(T) is not in ALAN.	3. Hypothesis.
4. code(T) is not accepted by T.	4. From 1 and 3.
5. code(T) is in ALAN.	5. From 2 and 4.
6. Contradiction.	6. From 3 and 5.
7. code(T) is in ALAN.	7. The hypothesis (3) must be wrong because it led to a contradiction.

ALAN is not R.E. — and UTMs

Theorem

Not all languages are recursively enumerable

See: liar's paradox

The Universal Turing Machine

A **universal TM**, a **UTM**, is a TM that can be fed as input a string composed of two parts:

1. an encoded program of any TM T followed by a marker
2. data

The operation of the UTM is that, no matter what machine T and no matter what the data string is, the UTM will operate *exactly* on the data as if it were T. The TAPE-HEAD would also point to exactly what T would have.

Theorem

Universal Turing Machines exist

UTMs Exist

- We have already defined a UTM.
- We have already defined ALAN as all CWL words that are not accepted by the TMs they might represent.
- Now consider...

Definition

Let MATHISON be the language of all CWL words that *do* represent TMs and *are* accepted by the very machines they do represent

Theorem

MATHISON is recursive enumerable

Proof.

The TM that accepts MATHISON is like a UTM. When we start with an input string, S, we convert the tape to the following:

#	S	\$	S	$\Delta\ldots$

And run the machine □

Recursively Enumerable and Recursive

Theorem

The complement of a recursively enumerable language might not be recursively enumerable

Proof.

Because CWL is regular, CWL′ is also regular. Because CWL′ is regular, it's also recursively enumerable. L = CWL′+ MATHISON is recursively enumerable, but it's complement (L' =ALAN) is not □

Theorem

There are recursively enumerable languages that are not recursive

Proof.

The language L defined is not recursive because that means ALAN would be r.e. (but it is not) □

Decidability

Definition

Suppose we are given an input string w and a TM T. Can we tell whether or not T halts on w? This is called the **halting problem**.

Theorem

There is no TM that can accept any string, w, and any coded TM, T, and always decide correctly whether T halts on w. In other words, the halting problem ***cannot*** *be decided by a TM.*

Proof.

- Assume a TM answers the halting problem – call it *HP*
- Modify *HP* (creating HP_2) by making it loop forever if it was about to print “yes” and halt. If it was to print “no” make no change

Continued...

The Halting Problem

Proof.

- Add a subprogram (preprocessor) to the front of HP_2
- Take the left-of-# part and decide whether it is a word in CWL.
- If the input is, then the preprocessor deletes the w part of the input and puts two copies of the same string onto the TAPE and reruns HP_2
- This means HP_2 will analyze whether the code word passed accepts its own code word as input. If the answer is "yes" then the modified machine loops forever
- If the answer is "no" then it prints "no" and halts.
- HP_2 accepts exactly the language ALAN. But ALAN is not recursively enumerable

□

Other Theorems of Decidability

Theorem

There is no TM that can decide for every TM — fed into it in encoded form — whether or not it accepts the word λ

Theorem

There is no TM that — when fed the code word for an arbitrary TM — can always decide whether the encoded TM accepts ***any*** *words. In other words, the emptiness question for r.e. languages cannot be decided by TM.*

Theorem

There does not exist a TM that can decide — for any encoded TM fed into it — whether or not the language of T is finite or infinite

Homework 11b

3. [4pts each] Show that each of the following languages is recursive by finding a TM that accepts them and crashes on strings in their complement
   - EVEN-EVEN
   - EQUAL
4. [4pts each] Decode the following words from CWL into their corresponding TMs and determine which are in ALAN and which are in MATHISON
   - *abaabbbbab*
   - *abaaabaaabbaaabaababbbb*
   - *abaaabaaabaaaabaababbab*