18.404/6.840 Lecture 10

Last time:

- The Reducibility Method for proving undecidability and T-unrecognizability
- General reducibility
- Mapping reducibility

Today:

- The Computation History Method for proving undecidability
- The Post Correspondence Problem is undecidable
- Linearly bounded automata
- Undecidable problems about LBAs and CFGs

Remember

To prove some language is undecidable, show that (or any known undecidable language) is reducible to .

Revisit Hilbert's 10th Problem

Recall polynomial has integer solution)

Hilbert's 10th problem (1900): Is decidable?

Theorem (1971): No

Proof: Show is reducible to . [would take entire semester]

Do toy problem instead which has a similar proof method.

Toy problem: The Post Correspondence Problem.

Method: The Computation History Method.

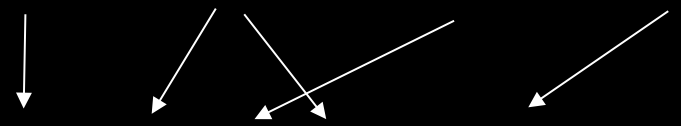
Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

a match is a finite sequence of dominos in (repeats allowed)
where the concatenation of the 's = the concatenation of the 's.

Match = where

Example:



Match:

Check-in 10.1

Let

Does have a match?

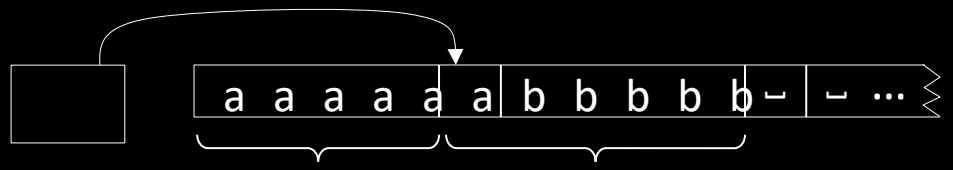
(a) Yes.

(b) No.

Check-in 10.1

TM Configurations

Defn: A configuration of a TM is a triple where
= the state,
= the head position,
= tape contents
representing a snapshot of the TM at a point in time.



Configuration:

Encoding as a string:

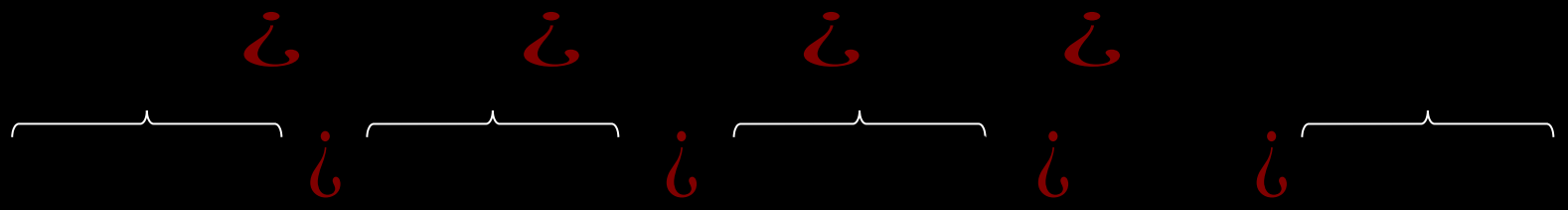
Encode configuration as the string where
and the head position is on the first symbol of .

TM Computation Histories

Defn: An (accepting) computation history for TM on input is a sequence of configurations that enters until it accepts.

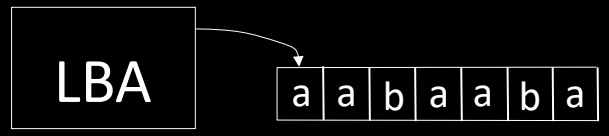
Encode a computation history as the string where each configuration is encoded as a string.

A computation history for
on .
Here say
and .



Linearly Bounded Automata

Defn: A linearly bounded automaton (LBA) is a 1-tape TM that cannot move its head off the input portion of the tape.



Tape size adjusts to length of input.

Let LBA accepts }

Theorem: is decidable

Proof: (idea) If on runs for long, it must be cycling.

Claim: For inputs of length , an LBA can have only different configurations.

Therefore, if an LBA runs for longer, it must repeat some configuration and thus will never halt.

Decider for :

"On input

1. Let .
2. Run on for steps.
3. If has accepted, *accept*.
4. If it has rejected or is still running, *reject*."

must be looping

is undecidable

Let is an LBA and }

Theorem: is undecidable

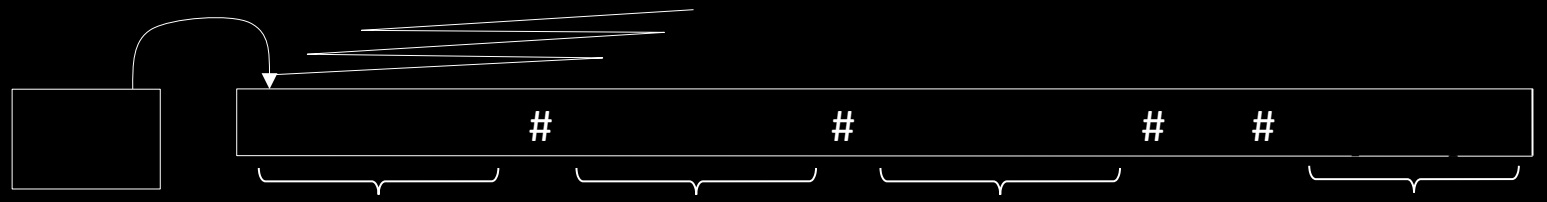
Proof: Show is reducible to . Uses the computation history method.

Assume that TM decides

Construct TM deciding

"on input

1. Construct LBA which tests whether its input is an accepting computation history for on , and only accepts if it is.
2. Use to determine whether .
3. *Accept* if no. *Reject* if yes."



Check-in 10.2

What do you think of the Computation History Method? Check all that apply.

(a) Cool !

(b) Just another theorem.

(c) I'm baffled.

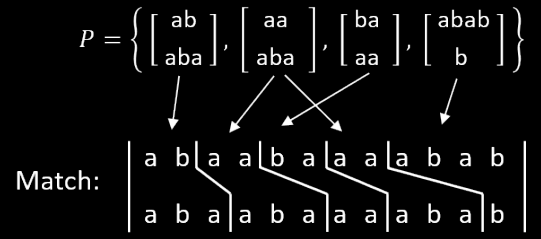
(d) I wish I was in 6.046.

N[illegible]


Coffee
Break

is undecidable

Recall has a match



Theorem: is undecidable

Proof: Show is reducible to . Uses the computation history method.

Technical assumption: Match must start with . Can fix this assumption.

Assume that TM decides

Construct TM deciding

"on input

1. Construct PCP instance where a match corresponds to a computation history for on .
2. Use to determine whether has a match.
3. *Accept* if yes. *Reject* if no."

Constructing

Make where a match is a computation history for on .

(starting domino)

For each and where

put in and put ir

(Handles right moves

Ending dominos to allow a match if accepts:

Check-in 10.3

What else can we now conclude?

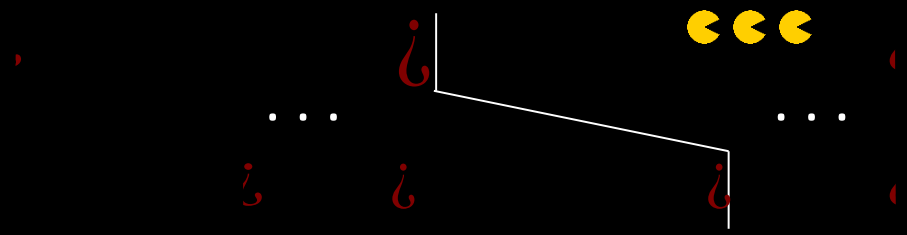
Choose all that apply.

(a) is T-unrecognizable.

(b) is T-unrecognizable.

(c) Neither of the above.

Illustration:



Match completed!

… one detail needed.

is undecidable

Let is a CFG and

Theorem: is undecidable

Proof: Show is reducible to via the computation history method.

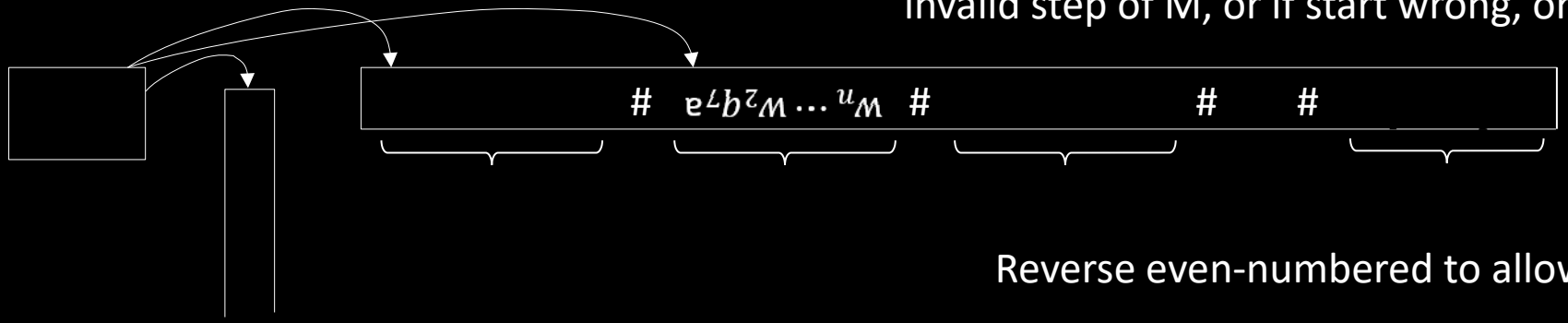
Assume TM R decides and construct TM deciding .

"On input

1. Construct PDA which tests whether its input is an accepting computation history for M on w, and only accepts if it is NOT.
2. Use to determine whether .
3. *Accept* if no. *Reject* if yes."

operation:

Nondeterministically push some and pop to compare with . *Accept* if invalid step of M, or if start wrong, or if end isn't accepting.



Reverse even-numbered to allow comparing with via stack.

Computation History Method - recap

Computation History Method is useful for showing the undecidability of problems involving testing for the existence of some object.

- Is there an integral solution (to the polynomial equation)?
- Is there some accepted string (for the LBA)?
- Is there a match (for the given dominos)?
- Is there some rejected string (for the CFG)?

In each case, the object is the computation history in some form.

Quick review of today

1. Defined configurations and computation histories.
2. Gave The Computation History Method to prove undecidability.
3. is decidable.
4. is undecidable.
5. is undecidable.
6. is undecidable.

Eliminating the technical assumption

Technical assumption: Match must start with .
Fix this assumption as follows.

Let where we require match to start with .

Create new

For any string , let

Then let