An Undecidable Language

A Turing-unrecognizable Language

An Undecidable Language

In this lecture we present an undecidable language.

The language that we prove to be undecidable is a very natural language namely the language consisting of pairs of the form $\langle M, w \rangle$ where M is a TM accepting string w:

$$A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM accepting } w\}$$

The Halting Problem

Since this language requires to decide whether the computation of TM M halts on input w, it is often called ***The Halting Problem***.

Theorem

A_{TM} is Turing-Recognizable.

Proof

Consider a TM U that gets a pair $\langle M, w\rangle$ as input and ***simulates*** the run of M on input w. If M accepts or rejects so does U. Otherwise, U loops. U is an example of **universal Turing machine**.

Note: U recognizes A_{TM} ,since it accepts any pair $\langle M,w\rangle \in L$, that is: any pair in which M accepts input w.

Simulating an Input TM

Earlier we detailed the simulation of a DFA by a TM.

Simulating one TM by another, using the encoding of the first TM is a very similar process. In the next slide we review the main characteristics of TM N simulating TM M, using M's encoding $<M>$.

Simulating an Input TM

TM N works as follows:

1. Mark M's initial state and w's initial symbol as the "current state" and "current head location".
2. Look for M's next transition on the description of its transition function.
3. Execute M's transition.

Simulating an Input TM

4. Move M's "current state" and "current head location" to their new places.
5. If M's new state is a deciding state $decide$ according to the state, otherwise – repeat stages 2-5.

The Language A_{TM} Is Undecidable

So far we proved the existence of a language which is not Turing recognizable. Now we continue our quest to prove:

Theorem

The language

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM accepting } w\}$$

*is **undecidable***.

Proof

Assume, by way of contradiction, that A_{TM} is decidable and let H be a TM deciding A_{TM}.

That is $$H(\langle M,w\rangle)=\begin{cases} accept & \text{if } M \text{ accepts } w \\ reject & \text{if } M \text{ does not accept } w \end{cases}$$

Define now another TM, D, that uses H as a subroutine as follows:

Proof

Define now another TM new D that uses H as a subroutine as follows:

D="On input $\langle M \rangle$ where M is a TM:

1. Run H on input $\langle M, \langle M \rangle \rangle$.

2. Output the opposite of H's output namely: If H accepts *reject*, otherwise *accept*."

Proof

Note: What we do here is taking advantage of the two facts:

Fact1: TM M should be able to compute with any string as input.

Fact2: The encoding of M, $\langle M \rangle$, is a string.

Proof

Running a machine on its encoding is analogous to using a compiler for the computer language Python, to compile itself (the compiler may be written in Python).

Compilers and editors both take programs as inputs.

Proof

What we got now is:

$$D(\langle M\rangle)=\begin{cases} \textit{accept} \text{ if } M \text{ rejects} \langle \mathrm{M}\rangle \\ \textit{reject} \text{ if } M \text{ accepts} \langle \mathrm{M}\rangle \end{cases}$$

Consider now the result of running D with input $\langle D\rangle$. What we get is:

$$D(\langle D\rangle)=\begin{cases} \textit{accept} \text{ if } D \text{ rejects} \langle \mathrm{D}\rangle \\ \textit{reject} \text{ if } D \text{ accepts} \langle \mathrm{D}\rangle \end{cases}$$

Proof

$$D(\langle D \rangle) = \begin{cases} \textit{accept} \text{ if } D \text{ rejects} \langle \mathrm{D} \rangle \\ \textit{reject} \text{ if } D \text{ accepts} \langle \mathrm{D} \rangle \end{cases}$$

So if D *accepts*, it ***rejects*** and if it *rejects* it ***accepts***. ***Contradiction***.

And it all caused by our assumption that TM H exists!!!

Proof Review

1. Define $A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM accepting } w\}$.
2. Assume that A_{TM} is decidable and let H be a TM deciding it.
3. Use H to build TM D that gets a string and behaves exactly opposite to H's behavior, namely:

$$D(\langle M\rangle) = \begin{cases} accept \text{ if } M \textbf{ r}\text{ejects} \langle \mathrm{M}\rangle \\ reject \text{ if } M \text{ accepts} \langle \mathrm{M}\rangle \end{cases}$$

Proof Review

4. Run TM D on its encoding $\langle D \rangle$ and conclude:

$$D(\langle D \rangle) = \begin{cases} \mathit{accept} \text{ if } D \text{ rejects} \langle \mathrm{D} \rangle \\ \mathit{reject} \text{ if } D \text{ accepts} \langle \mathrm{D} \rangle \end{cases}$$

Contradiction.

So Where is the Diagonalization?

The following table describes the behavior of each machine on some machine encodings:

$$
\begin{array}{cccccc}
 & \langle M_1 \rangle & \langle M_2 \rangle & \langle M_3 \rangle & \langle M_4 \rangle & \cdots \\
M_1 & accept & & accept & & \cdots \\
M_2 & accept & accept & accept & accept & \cdots \\
M_3 & & & & & \cdots \\
M_4 & accept & accept & & & \cdots \\
\vdots & \vdots & \vdots & \vdots & &
\end{array}
$$

So Where is the Diagonalization?

This table describes the behavior of TM H.

Note: TM H **rejects** where M_i **does not accept**.

$$
\begin{array}{cccccc}
 & \langle M_1 \rangle & \langle M_2 \rangle & \langle M_3 \rangle & \langle M_4 \rangle & \cdots \\
M_1 & accept & reject & accept & reject & \cdots \\
M_2 & accept & accept & accept & accept & \cdots \\
M_3 & reject & reject & reject & reject & \cdots \\
M_4 & accept & accept & reject & reject & \cdots
\end{array}
$$

Proof Review

Now TM D is added to the table…

$$
\begin{array}{ccccccc}
 & \langle M_1 \rangle & \langle M_2 \rangle & \langle M_3 \rangle & \langle M_4 \rangle & \cdots & \langle D \rangle \\
M_1 & \underline{accept} & reject & accept & reject & \cdots & accept \\
M_2 & accept & \underline{accept} & accept & accept & \cdots & accept \\
M_3 & reject & reject & \underline{reject} & reject & \cdots & reject \\
M_4 & accept & accept & reject & \underline{reject} & \cdots & accept \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \\
D & reject & reject & accept & accept & \cdots & \underline{???} \\
\vdots & \vdots & \vdots & \vdots & \vdots & & & \ddots
\end{array}
$$

$\overline{A_{TM}}$ Is Not Turing-Recognizable

So far we proved the existence of a language which is not Turing recognizable. Now we will prove a theorem that will enable us to identify a specific language, namely $\overline{A_{TM}}$, as a language that is not Turing recognizable.

$\overline{A_{TM}}$ Is Not Turing-Recognizable

(Reminder) A Language L is ***Turing-recognizable*** if there exists a TM recognizing L. That is, a TM that accepts any input $w \in L$.

A Language L is ***co-Turing-recognizable*** if the complement of L, $\overline{L}$, is ***Turing-recognizable***.

$\overline{A_{TM}}$ Is Not Turing-Recognizable

Theorem

A language L is decidable, if and only if it is Turing recognizable and co-Turing recognizable.

Proof ->

If L is decidable then it is Turing-recognizable. Any decidable language is Turing-recognizable.

$\overline{A_{TM}}$ Is Not Turing-Recognizable

Proof ->

If L is decidable so is its complement, $\overline{L}$. A TM to decide $\overline{L}$, is obtained by running a TM deciding L and deciding the opposite. Any decidable language is Turing recognizable, hence $\overline{L}$ is Turing recognizable and L is co-Turing recognizable.

$\overline{A_{TM}}$ Is Not Turing-Recognizable

Proof <-

If L and $\overline{L}$ are both Turing recognizable, there exist two TM-s, M_1 and M_2, that recognize L and $\overline{L}$, respectively. A TM M to decide L can be obtained by running both M_1 and M_2 in parallel and halt when one of them accepts. If M_1 accepts, accept; if M_2 accepts, reject.

$\overline{A_{TM}}$ Is Not Turing-Recognizable

Corollary

The language $\overline{A_{TM}}$ is not Turing recognizable.

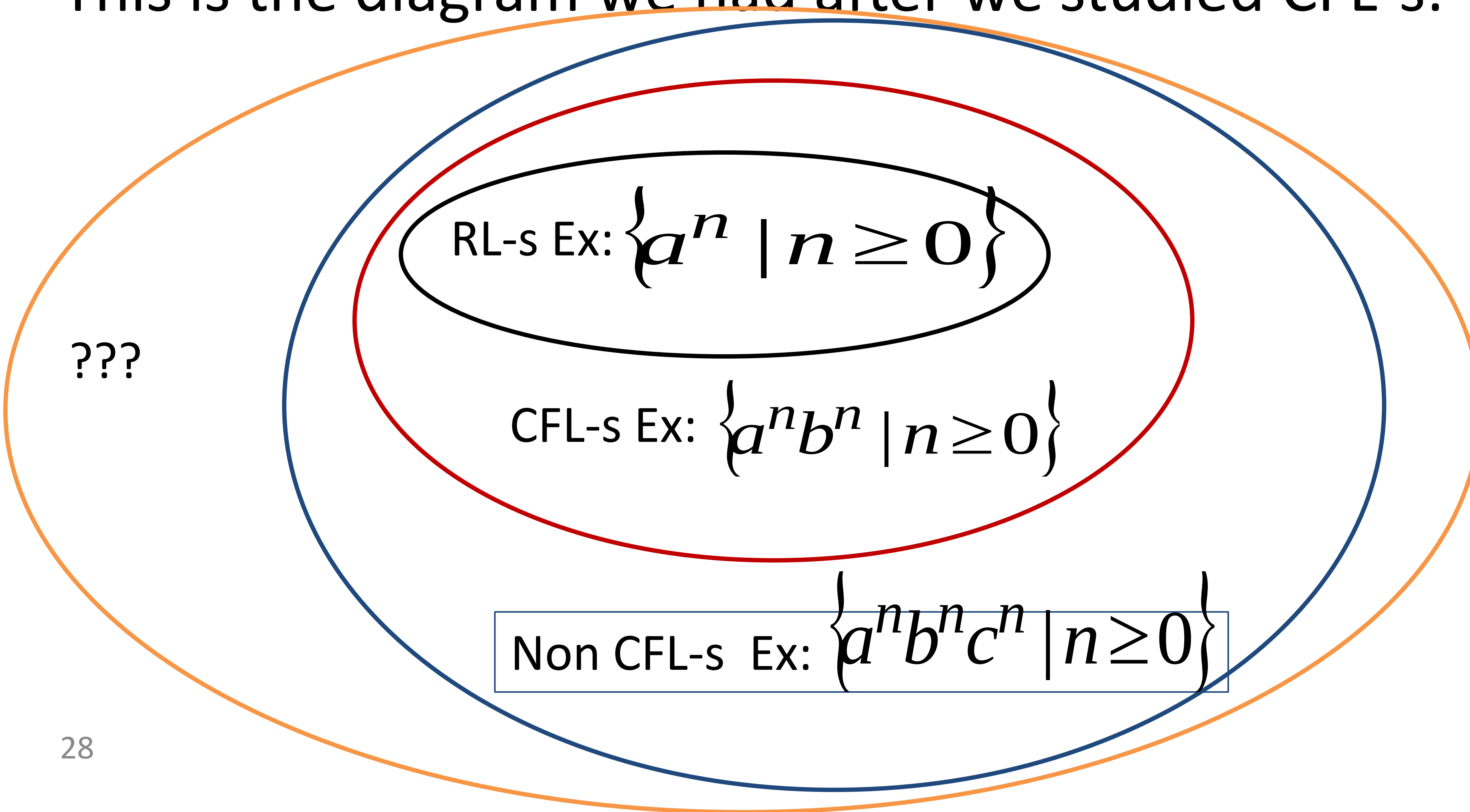
Proof

We already know that A_{TM} is Turing recognizable.

Assume $\overline{A_{TM}}$ is Turing recognizable. Then, A_{TM} is co-Turing recognizable. By the previous theorem A_{TM} is decidable, a contradiction.

Discussion

This is the diagram we had after we studied CFL-s:



Discussion

Now, we can add some more details: