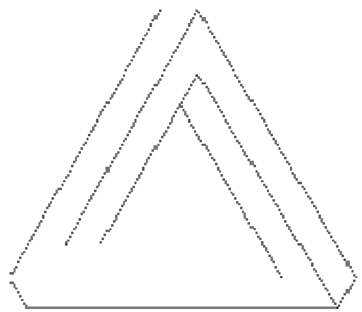
UNDECIDABILITY

Syrgkanis Vasileios



Outline

- Universal Turing Machine (UMT)
- Recursive(R) and Recursively Enumerable(RE)
- Undecidability
- The Halting Problem
- R and RE Theorems
- Rice Theorem

Universal

T	u	r	i	n	g	_	M	a	c	h	i	n	e

Why a Universal Turing Machine

- Proving Undecidability Theorems has at its essence the action of giving a Turing Machine as input to another
- The above needs a formal method for encoding a Turing Machine as an input
- And making another TM (the Universal) simulate the first

Turing Machines (TM) Notation

- Given a TM, M=(K, Σ, δ, s)
  - K = Set of States
  - Σ = Set of Symbols
  - δ = Transition Function, $K \cup \Sigma \rightarrow (K \cup \{h, \text{"}yes\text{"}, \text{"}no\text{"}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$
  - s = Initial State

- If a TM halts on input x, we define the output of M on x as M(x)
  - If M accepts or rejects x, then M(x)="yes" or "no"
  - If h state was reached then M(x) is the string of M at the time of halting

TM Binary Encoding(1)

- $\Sigma = \{1, 2, \ldots, |\Sigma|\}$
- $K = \{|\Sigma|+1, |\Sigma|+2, \ldots, |\Sigma|+|K|\}$
- $s = |\Sigma|+1$
- $|K|+|\Sigma|+1, \ldots, |K|+|\Sigma|+6 = \leftarrow, \rightarrow, -, h,$ "yes", "no"

- $\lceil \log(|K|+|\Sigma|+6) \rceil$ bits to encode each of the above entities

TM Binary Encoding(2)

- Encode the transition function as ((q,σ),(p,ρ,D))

$$q, p \in K \cup \{h, "yes", "no"\}$$

$$\sigma, \rho \in \Sigma$$

$$D \in \{\leftarrow, \rightarrow, -\}$$

A Simple Example(1)

- Suppose the following TM

$$\Sigma = \{a, b\} \cup \{\triangleright, \_\}$$

$$K = \{s\}$$

$$\delta(s, a) = (s, b, \rightarrow)$$

$$\delta(s, b) = (s, a, \rightarrow)$$

$$\delta(s, \_) = (h, \_, -)$$

A Simple Example(2)

- $|K|=1$
- $|\Sigma|=4$
- 4 bits for each entity
- Construct encoding according to previous formal description

$$
\begin{array}{lll}
a & \rightarrow & 0001 \\
b & \rightarrow & 0010 \\
\triangleright & \rightarrow & 0011 \\
\_ & \rightarrow & 0100 \\
s & \rightarrow & 0101 \\
\rightarrow & \rightarrow & 0110 \\
\leftarrow & \rightarrow & 0111 \\
- & \rightarrow & 1000 \\
h & \rightarrow & 1001 \\
\text{"} yes \text{"} & \rightarrow & 1010 \\
\text{"} no \text{"} & \rightarrow & 1011
\end{array}
$$

A Simple Example(2)

- The binary encoding of the TM is:

$$\boxed{\begin{array}{l} 0001,0010, \\ ((0101,0001),(0101,0010,0110)) \\ ((0101,0010),(0101,0001,0110)) \\ ((0101,0100),(1001,0000,1000)) \end{array}}$$

$$\begin{array}{l} a \to 0001 \\ b \to 0010 \\ \triangleright \to 0011 \\ \_ \to 0100 \\ s \to 0101 \\ \rightarrow \to 0110 \\ \leftarrow \to 0111 \\ - \to 1000 \\ h \to 1001 \\ \text{"}yes\text{"} \to 1010 \\ \text{"}no\text{"} \to 1011 \end{array}$$

Universal TM (UTM)

- A TM U that interprets each input as a concatenation of a description of another TM and a description of an input for that TM
  - The binary description of x is the binary description of each symbol of x separated by “,”

$$U(M; x)=M(x)$$

- Introduced by Turing
- Resembles the von Neumann architecture

An Implementation

- 2-string TM
- 1^{st} string contains the binary description *M*
- 2^{nd} string contains the binary description of current configuration of simulation (w,q,u)

UTM Description(1)

- Initially the 2 strings have the following content:

Binary Description of M

▷	,	Initial State (s) binary description	,	Input (x) binary description

UTM Description(2)

Search 1^{st} **string** to find an integer corresponding to State (between $|\Sigma|+1$ and $|\Sigma|+|K|$)

Search 2^{nd} **string** to find the same integer in the place of q of some rule $((q,\sigma),(p,\rho,D))$

Move to the next $\log(|\Sigma|+|K|+6)$ bit word of the 1^{st} **string**

Does that word match that of the symbol that triggers the rule

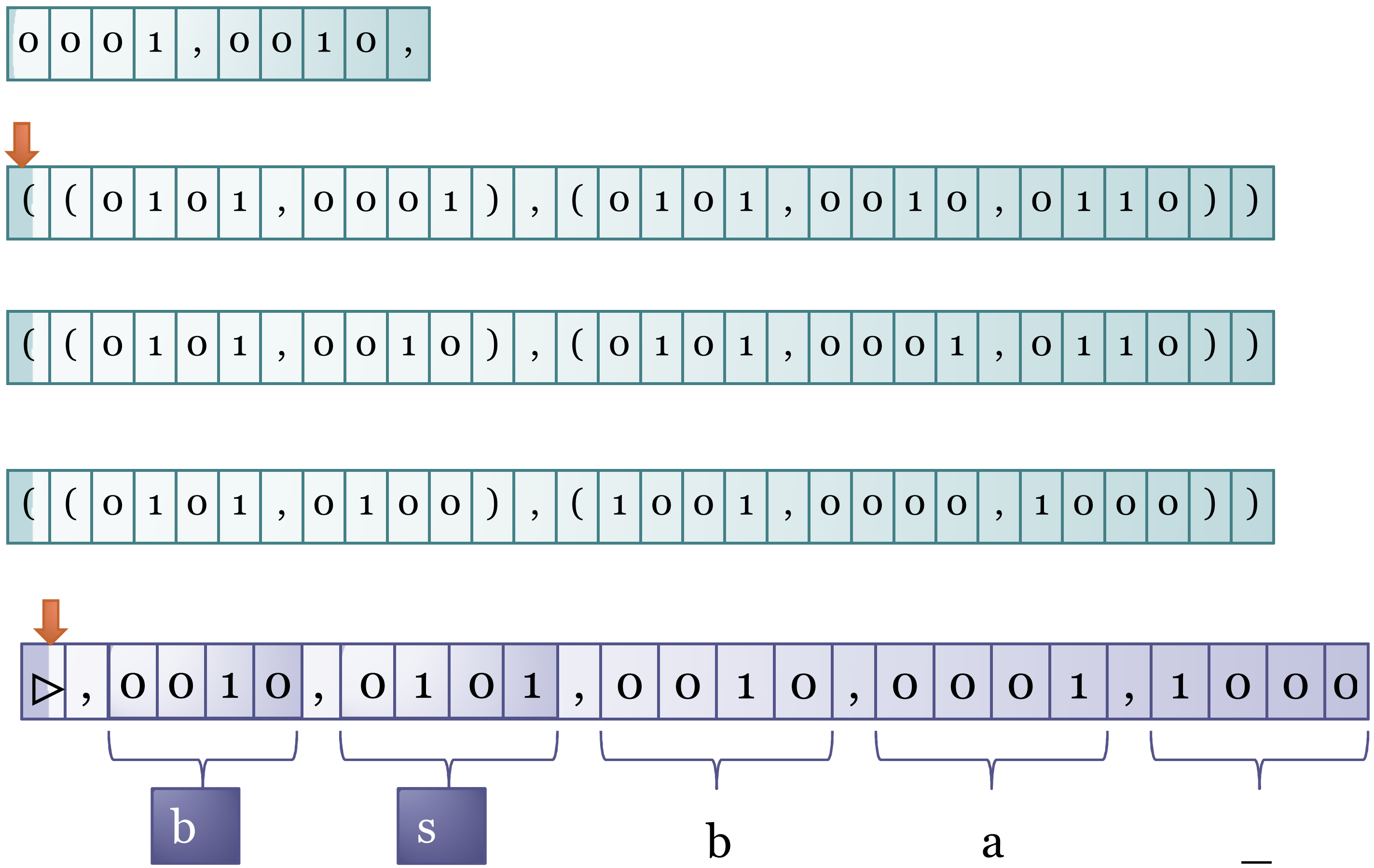
No

Yes

Activate Rule:

1. Change the current state in the 2^{nd} string
2. If Next State is $|K|+|\Sigma|+4$ = h then halt
3. Change the current symbol in 2^{nd} string
4. Move the binary description of the state on word left or right or not according to D symbol of rule

Simple Example Simulation

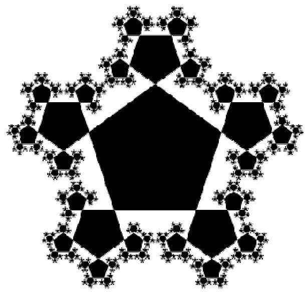


$a \rightarrow$	0001
$b \rightarrow$	0010
$\triangleright \rightarrow$	0011
$\_ \rightarrow$	0100
$s \rightarrow$	0101
$\rightarrow \rightarrow$	0110
$\leftarrow \rightarrow$	0111
$- \rightarrow$	1000
$h \rightarrow$	1001
" *yes* " $\rightarrow$	1010
" *no* " $\rightarrow$	1011

TM encoding

Recursive and Recursively Enumerable

Basic Definitions



Recursive (R) Language

- L is a recursive language if there exist a TM M that **decides** L.
- That is: for any string x:
  - If x is in L then M(x)="yes" (TM *halts* at the "yes" state)
  - If x is not in L then M(x)="no" (TM *halts* at the "no" state)

- Hence, Not Recursive means Undecidable

Recursively Enumerable (RE) Language

- L is a recursive language if there exist a TM M that **accepts** L.
- That is: for any string x:
  - If x is in L then M(x)="yes" (TM *halts* at the "yes" state)
  - If x is not in L then M(x) doesn't halt

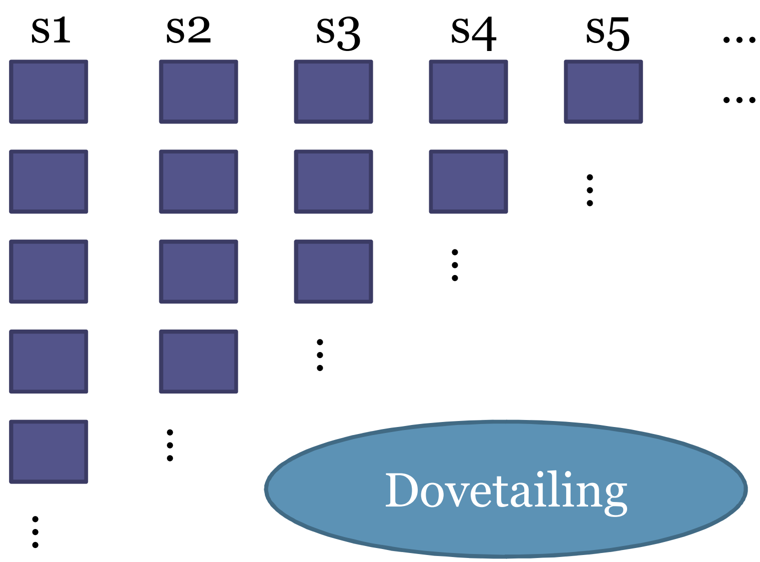
- Only useful for categorizing problems, not an algorithmic concept

RE Language(2)

- If L is in RE then there is TM that enumerates all its elements without repeating any of them
- Let M_L the TM that accepts L
- Run M for all possible strings of the symbols of L (e.g. in lexicographic order)
- When a string is accepted output it

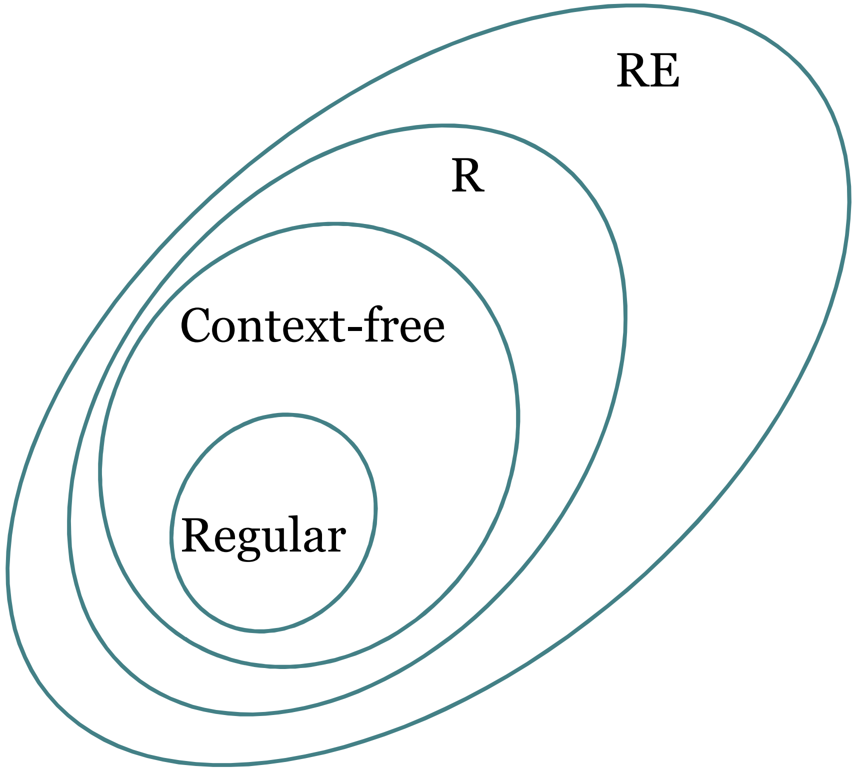
RE Language(3)

- Do it the following way:



- Eventually all s_i in L would be enumerated

Language Classes



Undecidability

Undecidability(1)

- Undecidable problem
  - A problem with no algorithm
- Undecidable language
  - A language that is not recursive

- Universal TM immediately led to prove that some problems are undecidable

Undecidability(2)

- It is an immediate consequence of the following two facts
  - Languages are not enumerable (using diagonalization)
  - Turing machines are enumerable (binary encoding described in first part is a valid encoding from TMs to natural numbers)

- Hence, there must be languages that cannot be decided by a TM

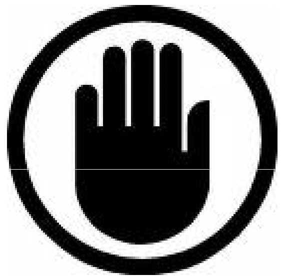
Undecidability(3)

- First undecidable problems/languages introduced in 1936
  - April: Church introduced an undecidable problem in lambda calculus
  - May: Turing introduced the halting problem

- Strong connection with Godel's incompleteness theorems (1931)
  - Similar proofs used in both theories
  - A weaker form of First Incompleteness Theorem is an immediate consequence of the Halting Problem

The Halting Problem

Not just any recursively enumerable language



HALTING (H)

- Given the description of a TM M and its input x

  Will M halt on x ?

- H is a language on the alphabet of UTM

$$\{M\,;x : M(x) \neq \uparrow\}$$

- H is **Recursively Enumerable**
  - **Proof #1 Outline: The UMT accepts H with a slight modification**

Recursively Enumerable Complete

- Suppose a TM M_H could decide HALTING
- Then deciding any recursively enumerable language L accepted by a TM M could be reduced to M_H
- Just check if M;x is accepted by M_H
- Similar concept to NP-Completeness though here we have a proof that H is not in R so we know that $R \neq RE$

H is Not Recursive (Undecidable)

- H is not recursive
- Proof Outline: Diagonalization
- Use the program
  - D(M): **if** M_H(M;M)="yes" **then** ↑ **else** "yes"
- And produce a contradiction
- Hence there is no M_H that decides H

Diagonalization

	$<M_1>$	$<M_2>$	$<M_3>$	$<M_4>$	…	$<D>$	…
M_1	**accepts**	rejects	accepts	accepts		accepts	
M_2	rejects	**accepts**	rejects	rejects		accepts	
M_3	accepts	rejects	**rejects**	accepts			
M_4	accepts	rejects	accepts	**accepts**		accepts	
…							
D	accepts	rejects	accepts	accepts		**?**	
…							

D(M): **if** $M_H(M;M)$="yes" **then** $\uparrow$ **else** "yes"

Other Non Recursive Languages

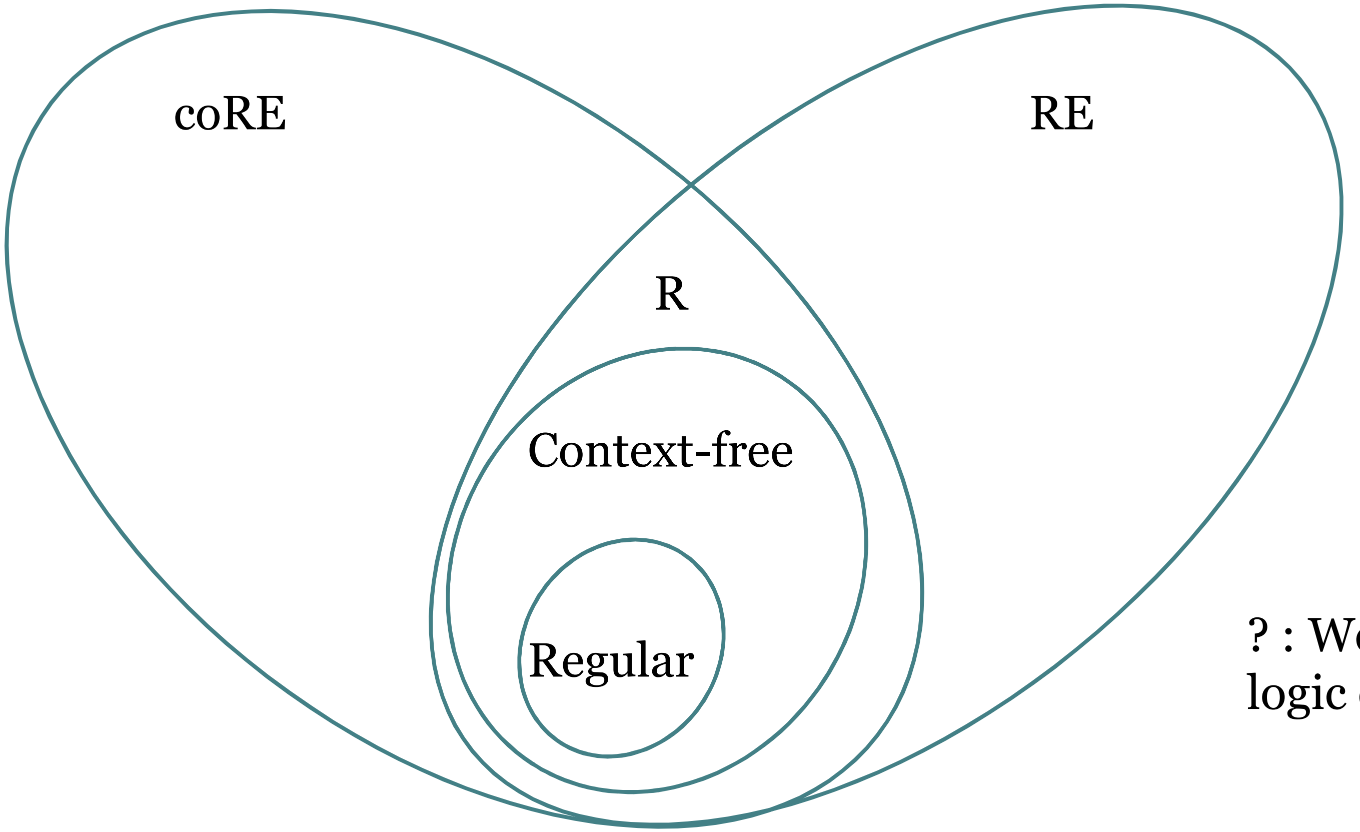
- $L1 = \{M: M$ halts on all inputs$\}$
- $L2 = \{M;x$: there is a y such that $M(x)=y\}$
- $L3 = \{M;x$: the computation M on input x uses all states of $M\}$
- $L4 = \{M;x;y: M(x)=y\}$

R and RE Theorems

R and RE Theorems(1)

- If L is in R, then so is ¬L
- L is in R if and only if both L and ¬L are in RE
  - Corollary: ¬H is not even in RE
- L is in RE iff there is a TM M such that L = E(M)
  - L is enumerated by M (dovetailing)

Revision of Language Classes



? : We'll see in logic chapters

Rice's Theorem

It isn't just HALTING the problem

Rice Theorem

- Suppose C is a proper, non-empty subset of the set of all RE languages.
- Then the following problem is undecidable:
  - Given a TM M, is L(M) in C
- In other words:
  - Any non-trivial property of TMs represents an undecidable problem

Proof

Fix Point in C

- **main(){char q=34, n=10,*a="main() {char q=34,n=10,*a=%c%s%c; printf(a,q,a,q,n);}%c";printf(a,q,a,q,n);}**