15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

UNDECIDABILITY II: REDUCTIONS

TUESDAY Feb 18

A_{TM} = { (M,w) | M is a TM that accepts string w }

A_{TM} is undecidable: (constructive proof & subtle)

Assume machine H semi-decides A_{TM} (such exist, why?)

$$H(\ (M,w)\ ) = \begin{cases} \text{Accept if M accepts w} \\ \text{Rejects or loops otherwise} \end{cases}$$

Construct a new TM D_H as follows: on input M, run H on (M,M) and output the "opposite" of H whenever possible.

$$D_H(\ M\ ) = \begin{cases} \text{Reject if M accepts M} \\ \text{(i.e. if H(M , M) = Accept)} \\ \text{Accept if M rejects M} \\ \text{(i.e. if H(M , M) = Reject)} \\ \text{loops if M loops on M} \\ \text{(i.e. if H(M , M) loops)} \end{cases}$$

$$D_H(\ D_H\ ) = \begin{cases} \text{Reject if } D_H \text{ accepts } D_H \\ \text{(i.e. if H(} D_H , D_H \text{) = Accept)} \\ \text{Accept if } D \text{ rejects } D_H \\ \text{(i.e. if H(} D_H , D_H \text{) = Reject)} \\ \text{loops if } D_H \text{ loops on } D_H \\ \text{(i.e. if H(} D_H , D_H \text{) loops)} \end{cases}$$

Note: It must be the case that D_H loops on D_H

There is no contradiction here!

Thus we effectively constructed an instance which does not belong to A_{TM} (namely, (D_H, D_H)) but H fails to tell us that.

That is:

Given any semi-decision machine H for A_{TM}

(and thus a potential decision machine for A_{TM}),

we can effectively construct an instance which does not belong to A_{TM} (namely, (D_H, D_H))

but H fails to tell us that.

So H cannot be a decision machine for A_{TM}

In most cases, we will show that a language L is undecidable by showing that if it is decidable, then so is A_{TM}

We reduce deciding A_{TM} to deciding the language in question

A_{TM} "<" L

So, A_{TM} "<" $Halt_{TM}$
Is $Halt_{TM}$ "<" A_{TM} ?

A_{TM} = { (M,w) | M is a TM that accepts string w }

$HALT_{TM}$ = { (M,w) | M is a TM that halts on string w } (*)

E_{TM} = { M | M is a TM and L(M) = ∅ } (*)

REG_{TM} = { M | M is a TM and L(M) is regular} (*)

EQ_{TM} = {(M, N) | M, N are TMs and L(M) = L(N)} (*)

ALL_{PDA} = { P | P is a PDA and L(P) = Σ* } (*)

ALL UNDECIDABLE

(*) Use Reductions to Prove

Which are SEMI-DECIDABLE?

E_{TM} = { M | M is a TM and L(M) = ∅ }

Theorem: E_{TM} is undecidable

Proof: Assume, for a contradiction, that TM Z decides E_{TM}. Use **Z** as a subroutine to decide A_{TM}

Algorithm for deciding A_{TM}: On input (M,w):

1. Create M_w

s → [M_w — If s ≠ w, REJECT; If s = w, run M(w)] →

So, L (M_w) = ∅ ⇔ M(w) does not accept
L (M_w) ≠ ∅ ⇔ M(w) accepts

2. Run Z on M_w

$REGULAR_{TM}$ = { M | M is a TM and L(M) is regular}

Theorem: $REGULAR_{TM}$ is undecidable

Proof: Assume, for a contradiction, that TM R decides $REGULAR_{TM}$

Use **R** as a subroutine to decide A_{TM}

1. Create M'_w

s → [M'_w — If s = 0^n1^n, accept; Else run M(w)] →

So, L (M'_w) = Σ* ⇔ M(w) accepts
L (M'_w) = {0^n1^n} ⇔ M(w) does not accept

2. Run R on M'_w

---

s → [M_w' — If s = 0^n1^n, accept; Else run M(w)] →

L(M_w') = Σ* if M(w) accepts
{0^n1^n} otherwise

L(M_w') is regular ⇔ M(w) accepts

N → R [Is L(N) regular?] →

---

s → [M_w' — If s = 0^n1^n, accept; Else run M(w)] →

L(M_w') = Σ* if M(w) accepts
{0^n1^n} otherwise

L(M_w') is regular ⇔ M(w) accepts

M_w' → R [Is L(M_w') regular?] →

Yes ⇔ M accepts w

MAPPING REDUCIBILITY

$f : \Sigma^* \rightarrow \Sigma^*$ is a computable function if some Turing machine M, on every input w, halts with just f(w) on its tape

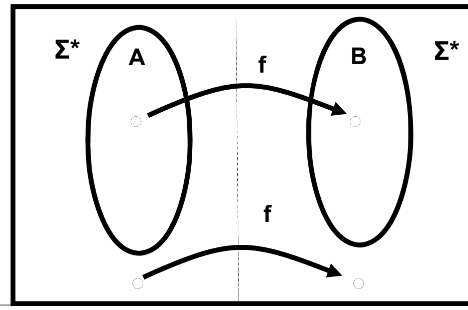
A language A is *mapping reducible* to language B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, where for every w,

$$w \in A \Leftrightarrow f(w) \in B$$

f is called a *reduction* from A to B

Think of f as a "computable coding" from A to B

A is mapping reducible to B, $A \leq_m B$, if there is a computable $f : \Sigma^* \rightarrow \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$



Also, $\neg A \leq_m \neg B$, **why?**

Theorem: If $A \leq_m B$ and B is decidable, then A is decidable

Proof: Let M decide B and let f be a reduction from A to B

We build a machine N that decides A as follows:

On input w:

1. Compute f(w)

2. Run M on f(w)

Theorem: If $A \leq_m B$ and B is (semi) decidable, then A is (semi) decidable

Proof: Let M (semi) decide B and let f be a reduction from A to B

We build a machine N that (semi) decides A as follows:

On input w:

1. Compute f(w)

2. Run M on f(w)

All undecidability proofs from today can be seen as constructing an f that reduces A_{TM} to the proper language

(Sometimes you have to consider the complement of the language.)

All undecidability proofs from today can be seen as constructing an f that reduces A_{TM} to the proper language

$A_{TM} \leq_m HALT_{TM}$ (So also, $\neg A_{TM} \leq_m \neg HALT_{TM}$):

Map (M, w) → (M', w)
where M'(w) = M(w) if M(w) accepts
loops otherwise

So $(M, w) \in A_{TM} \Leftrightarrow (M', w) \in HALT_{TM}$

A_{TM} = { (M,w) | M is a TM that accepts string w }

E_{TM} = { M | M is a TM and L(M) = ∅ }

CLAIM: $A_{TM} \leq_m \neg E_{TM}$ $\boxed{\neg A_{TM} \leq_m E_{TM}}$

CONSTRUCT f : Σ* → Σ*

f: (M,w) → M_w where M_w (s) = M(w) if s = w
loops otherwise

So, M(w) accepts ⇔ L (M_w) ≠ ∅

So, (M, w) ∈ A_{TM} ⇔ $M_w \in \neg E_{TM}$

So ¬ E_{TM} is NOT DECIDABLE, but it is SEMI-DECIDABLE (why?) Is E_{TM} SEMI-DECIDABLE?

A_{TM} = { (M,w) | M is a TM that accepts string w }

REG_{TM} = { M | M is a TM and L(M) is regular}

CLAIM: $A_{TM} \leq_m REG_{TM}$ So REG_{TM} is UNDECIDABLE

CONSTRUCT f : Σ* → Σ*

f: (M,w) → M'_w where M'_w (s) = accept if s = 0^n1^n
M(w) otherwise

So, L (M'_w) = Σ* if M(w) accepts
{0^n1^n} if not

So, (M, w) ∈ A_{TM} ⇔ $M'_w \in REG_{TM}$

Is REG SEMI-DECIDABLE?

A_{TM} = { (M,w) | M is a TM that accepts string w }

REG_{TM} = { M | M is a TM and L(M) is regular}

CLAIM: $\neg A_{TM} \leq_m REG_{TM}$ So REG_{TM} is NOT SEMI-DECIDABLE

CONSTRUCT f : Σ* → Σ*

f: (M,w) → M''_w where M''_w (s) = accept if s = 0^n1^n
and M(w) accepts
Loop otherwise

So, L (M''_w) = {0^n1^n} if M(w) accepts
∅ if not

So, (M, w) ∉ A_{TM} ⇔ $M''_w \in REG_{TM}$

So, REG NOT SEMI-DECIDABLE

A_{TM} = { (M,w) | M is a TM that accepts string w }

$HALT_{TM}$ = { (M,w) | M is a TM that halts on string w }

E_{TM} = { M | M is a TM and L(M) = ∅ }

REG_{TM} = { M | M is a TM and L(M) is regular}

EQ_{TM} = {(M, N) | M, N are TMs and L(M) = L(N)}

ALL_{PDA} = { P | P is a PDA and L(P) = Σ* }

ALL UNDECIDABLE

Which are SEMI-DECIDABLE?

E_{TM} = { M | M is a TM and L(M) = ∅ }

EQ_{TM} = {(M, N) | M, N are TMs and L(M) =L(N)}

CLAIM: $E_{TM} \leq_m EQ_{TM}$ So EQ_{TM} is UNDECIDABLE

CONSTRUCT f : Σ* → Σ*

f: M → (M, $M_\emptyset$) where $M_\emptyset$ (s) = Loops

So, M∈ E_{TM} ⇔ (M, $M_\emptyset$) ∈ EQ_{TM}

Is EQ_{TM} SEMI-DECIDABLE? NO, since,

$\boxed{\neg A_{TM} \leq_m E_{TM} \leq_m EQ_{TM}}$ What about $\neg EQ_{TM}$?

A_{TM} = { (M,w) | M is a TM that accepts string w }

ALL_{PDA} = { P | P is a PDA and L(P) = Σ* }

CLAIM: $A_{TM} \leq_m \neg ALL_{PDA}$ $\boxed{\neg A_{TM} \leq_m ALL_{PDA}}$

CONSTRUCT f : Σ* → Σ*

Idea! More subtle construction

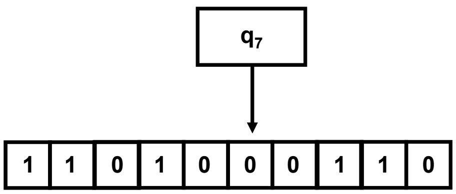
Map (M,w) to a PDA $P_{M,w}$ that recognizes Σ* if and only if M does not accept w

So, (M, w) ∉ A_{TM} ⇔ $P_{M,w} \in ALL_{PDA}$

$P_{M,w}$ will recognize all (and only those) strings that are NOT accepting computation histories for M on w

CONFIGURATIONS

$11010q_700110$



COMPUTATION HISTORIES

An accepting computation history is a sequence of configurations $C_1, C_2, \ldots, C_k$, where

1. **C_1 is the start configuration,**
2. **C_k is an accepting configuration,**
3. **Each C_i follows from C_{i-1}**

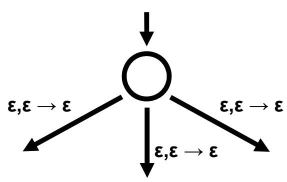
An rejecting computation history is a sequence of configurations $C_1, C_2, \ldots, C_k$, where

1. **C_1 is the start configuration,**
2. **C_k is a rejecting configuration,**
3. **Each C_i follows from C_{i-1}**

M accepts w if and only if there exists an accepting computation history that starts with $C_1 = q_0w$

$P_{M,w}$ will recognize all strings (read as sequences of configurations) that:

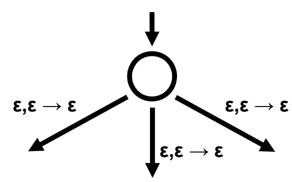
1. **Do not start with C_1 or**
2. **Do not end with an accepting configuration or**
3. **Where some C_i does not properly yield C_{i+1}**



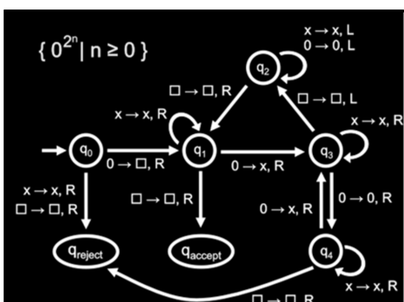
Non-deterministic checks for 1, 2, and 3.

$P_{M,w}$ will reject all strings (read as sequences of configurations) that:

1. **Start with C_1 and**
2. **End with an accepting configuration and**
3. **Where each C_i properly yields C_{i+1}**



Non-deterministic checks for 1, 2, and 3.



q_00000
$\square q_1000$
$\square x q_3 00$
$\square x 0 q_4 0$
$\square x 0 x q_3$
$\square x 0 q_2 x$
$\square x q_2 0 x$
$\square q_2 x 0 x$
$q_2 \square x 0 x$

P recognizes all strings except accepting computation histories :

$\#C_1\# C_2^R \#C_3 \#C_4^R \#C_5 \#C_6^R \#\ldots.\# C_k$

If i is odd, put C_i on stack and see if C_{i+1}^R follows properly:

For example,

If $=uaq_ibv$ and $\delta(q_i,b) = (q_j,c,R)$,

then C_i properly yields $C_{i+1} \Leftrightarrow C_{i+1} = uacq_jv$

P recognizes all strings except accepting computation histories :

$\#C_1\# C_2^R \#C_3 \#C_4^R \#C_5 \#C_6^R \#\ldots.\# C_k$

If i is odd, put C_i on stack and see if C_{i+1}^R follows properly:

For example,

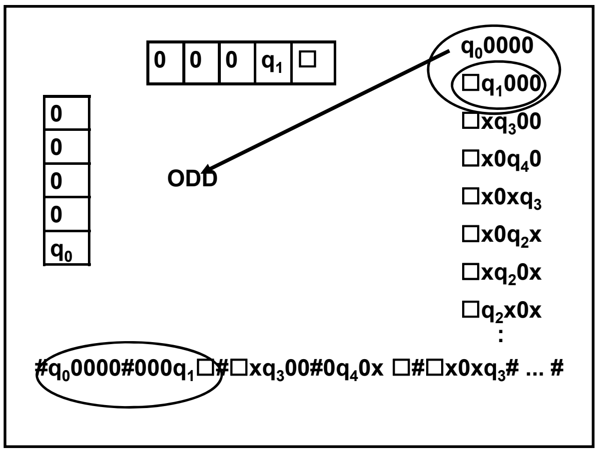
If =uaq_ibv and $\delta(q_i,b) = (q_j,c,L)$,

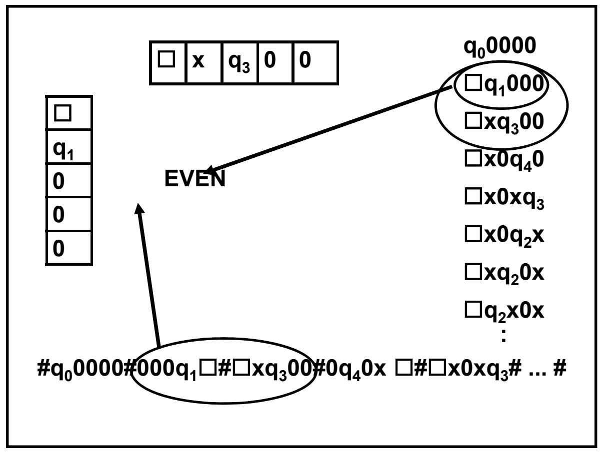
then C_k properly yields $C_{k+1} \Leftrightarrow C_{k+1} = uq_jacv$

P recognizes all strings except accepting computation histories :

$\#C_1\# C_2^R \#C_3 \#C_4^R \#C_5 \#C_6^R \#\ldots.\# C_k$

If i is even, put C_i^R on stack and see if C_{i+1} follows properly.





A_{TM} = { (M,w) | M is a TM that accepts string w }

ALL_{PDA} = { P | P is a PDA and L(P) = Σ* }

CLAIM: $A_{TM} \leq_m \neg ALL_{PDA}$ $\boxed{\neg A_{TM} \leq_m ALL_{PDA}}$

CONSTRUCT f : Σ* → Σ*

f: (M,w) → $P_{M,w}$ where

$P_{M,w}$ (s) = accept iff **s** is **NOT** an accepting computation of **M(w)**

$\boxed{\text{So, } (M, w) \notin A_{TM} \Leftrightarrow P_{M,w} \in ALL_{PDA}}$

$\boxed{\text{So, } (M, w) \in A_{TM} \Leftrightarrow P_{M,w} \in \neg ALL_{PDA}}$

EXPLAIN THE PROOF TO YOUR NEIGHBOR

WWW.FLAC.WS

Read chapter 5.1-5.3 of the book for next time