Computational Models — Lecture 10[1]

Handout Mode

Ronitt Rubinfeld and Iftach Haitner.

Tel Aviv University.

May 25/27, 2015

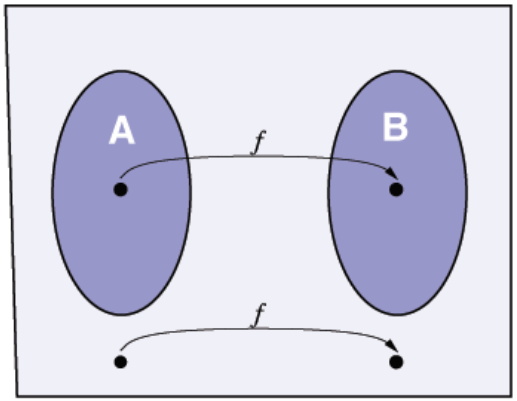
[1] Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University. Also with modifications of Yishay Mansour.

Talk Outline

- Rice's theorem revisited
- $\mathcal{RE}$-Completeness
- Reductions via controlled executions

- Sipser's book, Chapter 5, Sections 5.1, 5.3

Mapping Reductions (Review)



A mapping reduction converts questions about membership in A to membership in B.

Theorem 1

If $A \leq_m B$ and B is decidable, then A is decidable.

Corollary 2

If $A \leq_m B$ and A is undecidable, then B is undecidable.

In fact, this has been our principal tool for proving undecidability of languages other than A_{TM}

Section 1

Rice's Theorem

Non Trivial Properties of $\mathcal{RE}$ Languages

A few examples

- L is finite.
- L is infinite.
- L contains the empty string.
- L contains no prime number.
- L is co-finite.
- . . .

All these are non-trivial properties of enumerable languages, since for each of them there is $L_1, L_2 \in \mathcal{RE}$ such that L_1 satisfies the property but L_2 does not.

Rice's Theorem

Theorem 3

Let $\mathcal{C}$ be a proper non-empty subset of the set of $\mathcal{RE}$ and let $L_{\mathcal{C}} = \{\langle M \rangle : L(M) \in \mathcal{C}\}$. Then $L_{\mathcal{C}}$ is undecidable.

Proof's idea: Reduction from H_{TM}.

Given M and w, we construct M_0 such that:

- If M halts on w, then $\langle M_0 \rangle \in L_{\mathcal{C}}$.
- If M does not halt on w, then $\langle M_0 \rangle \notin L_{\mathcal{C}}$.

Before the proof: Primes

- $\text{Primes} = \{p \in \mathbb{N} : p \text{ is a prime}\}$
- $\text{Primes}_{\text{TM}} = \{\langle M \rangle : M \text{ is a TM and } L(M) = \text{Primes}\}$

Theorem 4

$\text{Primes} \in \mathcal{R}$

Let $\mathcal{C}$ contain exactly ONE set, Primes, and so $L_{\mathcal{C}}$ is exactly $\text{Primes}_{\text{TM}}$.

Theorem 5

$L_{\mathcal{C}} = \text{Primes}_{\text{TM}} \notin \mathcal{R}$.

Proof's idea: We define a computable function f with

$$\langle M, w \rangle \in H_{\text{TM}} \iff \langle B_{M,w} \rangle \in \text{Primes}_{\text{TM}}$$

Hence, $H_{\text{TM}} \leq_m \text{Primes}_{\text{TM}}$

$H_{TM} \leq_m Primes_{TM}$

Let P be a decider for Primes. (How?)

Definition 6 ($B_{M,w}$)

On input x

- Emulate $M(w)$
- Emulate $P(x)$; Accept if $P(x)$ accepts and reject if $P(x)$ rejects.

- f is computable
- f does a correct mapping reduction:
  - $\langle M, w\rangle \in H_{TM} \implies L(B_{M,w}) = Primes \in \mathcal{C}$
  - $\langle M, w\rangle \notin H_{TM} \implies L(B_{M,w}) = \emptyset \notin \mathcal{C}$

  Hence $H_{TM} \leq_m Primes_{TM} \implies Primes_{TM} \notin \mathcal{R}$

  What did we use about our choice of $\mathcal{C}$?
  - $\emptyset \notin \mathcal{C}$
  - $Primes \in \mathcal{C}$
  - There is an acceptor for Primes (we don't need it to halt on reject inputs, that is just an extra bonus that we have for Primes...).

Proving Rice's Theorem

We assume wlg. that $\emptyset \notin \mathcal{C}$ (otherwise, look at $\overline{\mathcal{C}}$, also proper and non-empty).
Fix $\mathsf{L} \in \mathcal{C}$ and let M_{L} be a TM accepting it (recall $\mathcal{C} \subseteq \mathcal{RE}$).

Algorithm 7 (M_0)

On input y:

1. Emulate $M(w)$.
2. Emulate $M_{\mathsf{L}}(y)$:

   Accept if M_{L} accepts; reject if M_{L} rejects.

Let $f(\langle M, w \rangle) := \langle M_0 \rangle$, and let $f(x) = \emptyset$ if x is not of the form $\langle M, w \rangle$.

Claim 8

f is a mapping reduction from H_{TM} to $\mathsf{L}_{\mathcal{C}}$

f is Computable

Claim 9

f is computable.

Proof: On a valid pair $\langle M, w \rangle$, the TM $M_0 = f(\langle M, w \rangle)$ is simply a concatenation of two known TMs: the universal machine and M_L. ♣

$\langle M, w \rangle \in \mathrm{H_{TM}} \iff f(\langle M, w \rangle) \in \mathrm{L}_{\mathcal{C}}$

Claim 10

$\langle M, w \rangle \in \mathrm{H_{TM}} \iff f(\langle M, w \rangle) \in \mathrm{L}_{\mathcal{C}}$

(Hence, f is a mapping reduction from $\mathrm{H_{TM}}$ to $\mathrm{L}_{\mathcal{C}}$)

Proof:

- If $\langle M, w \rangle \in \mathrm{H_{TM}}$, then M_0 gets to Step 2, and emulates $M_{\mathrm{L}}(y)$. Hence $\mathrm{L}(M_0) = \mathrm{L} \in \mathcal{C}$.
- Otherwise (i.e., $\langle M, w \rangle \notin \mathrm{H_{TM}}$), M_0 never gets to Step 2. Hence $\mathrm{L}(M_0) = \emptyset \notin \mathcal{C}$.
- Thus, $\langle M, w \rangle \in \mathrm{H_{TM}}$ iff $\langle M_0 \rangle \in \mathrm{L}_{\mathcal{C}}$.

♣

We proved that $\mathrm{H_{TM}} \leq_m L_{\mathcal{C}}$, thus $\mathrm{L}_{\mathcal{C}}$ is undecidable.

Rice's Theorem, Reflections

- Rice's theorem can be used to show undecidability of properties like
  - *Does* $\mathsf{L}(M)$ *contain infinitely many primes*
  - *Does* $\mathsf{L}(M)$ *contain an arithmetic progression of length 15*
  - *Is* $\mathsf{L}(M)$ *empty*
- Decidability of properties related to the encoding itself cannot be inferred from Rice.
  - The question *does* $\langle M \rangle$ *has an even number of states* is decidable.
  - The question *does* M *reaches state* q_6 *on the empty input string* is undecidable, but this does not follow from Rice's theorem.
- Rice does not say anything on membership in $\mathcal{RE}$.
- **Rice's Theorem is a powerful tool, but use it with care!**

Section 2

More on Primes

Primes

- $\text{Primes} = \{p \in \mathbb{N} : p \text{ is a prime}\}$
- $\text{Primes}_{\text{TM}} = \{\langle M \rangle : M \text{ is a TM and } L(M) = \text{Primes}\}$

Theorem 11

$\text{Primes}_{\text{TM}} \notin \mathcal{RE}$.

Proof's idea: We define a computable function f with

$$\langle M, w \rangle \in \overline{A_{\text{TM}}} \iff \langle B_{M,w} \rangle \in \text{Primes}_{\text{TM}}$$

Hence, $\overline{A_{\text{TM}}} \leq_m \text{Primes}_{\text{TM}}$

$\overline{A_{TM}} \leq_m \text{Primes}_{TM}$

Let P be a decider for Primes (how?).

Definition 12 ($B_{M,w}$)

On input x

- Emulate $P(x)$ and Accept if P accepts, otherwise continue
- Emulate $M(w)$ and Accept if M accepts, otherwise reject

- f is computable
- f does a correct mapping reduction:
  - $\langle M, w \rangle \in \overline{A_{TM}} \implies L(B_{M,w}) = \text{Primes}$
  - $\langle M, w \rangle \notin \overline{A_{TM}} \implies L(B_{M,w}) = \mathbb{N} \neq \text{Primes}$

  Hence $\overline{A_{TM}} \leq_m \text{Primes}_{TM} \implies \text{Primes}_{TM} \notin \mathcal{RE}$

Generalizing Primes

Question 13

What property of Primes have we used?

Theorem 14

Let $\Sigma^* \notin \mathcal{C} \subset \mathcal{R}$ *and* $\mathcal{C} \neq \emptyset$.
Let $L_{\mathcal{C}} = \{\langle M \rangle : L(M) \in \mathcal{C}\}$.
Then $L_{\mathcal{C}} \notin \mathcal{RE}$.

Section 3

RE-Completeness

$\mathcal{RE}$-Completeness

Question 15

Is there a language L that is hardest in the class $\mathcal{RE}$?

Answer: Well, you have to define what you mean by "hardest language"...

Definition 16 ($\mathcal{RE}$-complete)

A language $L_0 \subseteq \Sigma^*$ is called $\mathcal{RE}$-complete, if the following holds

- $L_0 \in \mathcal{RE}$ (membership).
- for every $L \in \mathcal{RE}$ we have $L \leq_m L_0$ (hardness).

- The second item means that $\forall L \in \mathcal{RE}$, there is a mapping reduction f_L from L to L_0.
- The reduction f_L depends on L and will typically differ from one language to another.

A_{TM} is $\mathcal{RE}$-Complete.

Question 17

Are there $\mathcal{RE}$-complete languages?

Theorem 18

A_{TM} is $\mathcal{RE}$-Complete.

Proof:

- Clearly $A_{TM} \in \mathcal{RE}$.
- Let $L \in \mathcal{RE}$, and let M_L be a TM accepting it.
  Then $f_L(w) = \langle M_L, w \rangle$ is a mapping reduction from L to A_{TM} (why?).

♣

Other $\mathcal{RE}$-Complete problems

Question 19

Are there other $\mathcal{RE}$-complete languages?

Observations 20

Reductions are transitive:

$$A \leq_m B, \quad B \leq_m C \quad \Rightarrow \quad A \leq_m C$$

Theorem 21

$\mathrm{H_{TM}}$ *is* $\mathcal{RE}$*-Complete.*

Proof:

- Clearly $\mathrm{H_{TM}} \in \mathcal{RE}$.
- We have $\mathrm{A_{TM}} \leq_m \mathrm{H_{TM}}$ and $L \leq_m \mathrm{A_{TM}}$ for $L \in \mathcal{RE}$.

♣

Other $\mathcal{RE}$-Complete problems

Question 22

Are there other $\mathcal{RE}$-complete languages?

Observations 23

Reductions are transitive:

$$A \leq_m B, \quad B \leq_m C \quad \Rightarrow \quad A \leq_m C$$

Theorem 24

Let L be a language that is (1) in $\mathcal{RE}$ and (2) such that $A_{TM} \leq_m L$, then L is $\mathcal{RE}$-Complete.

Note that $\mathcal{RE}$-Complete languages form an equivalence class!

Between $\mathcal{R}$ and $\mathcal{RE}$-complete

Are there problems not in $\mathcal{R}$ but not $\mathcal{RE}$-complete?

YES! but not natural problems, see work of Emil Post.

Section 4

Controlled Executions

Bounded Acceptance – CET is Decidable

Assume all TMs are deterministic!

Definition 25

CET $:= \{\langle M, w, k\rangle : \ M$ accepts w within k steps$\}$.

Theorem 26

CET *is decidable.*

Proof?
What about space?

Definition 27

CES $:= \{\langle M, w, k\rangle : \ M$ accepts w using k cells$\}$.

Theorem 28

CES *is decidable.*

Bounded Acceptance – CET is Decidable, 2

Theorem 29

CES *is decidable.*

Proof: How to check that the computation will not terminate?

Proof uses *TM configurations.*
Let $m = |Q| \cdot |\Gamma|^k \cdot k$ be the number of configurations.

- Wait until a configuration repeats.
- Run for $m + 1$ steps.
- Consider a graph with nodes corresponding to TM configurations, and with an edge if M moves from one configuration to another. Each node has out-degree 1. Is there is a cycle reachable from the start state?

Note that proof uses that TM is deterministic. ♣

Alternate proof

Algorithm 30

On input $\langle M, w, k \rangle$

1. Emulate $M(w)$ while maintaining a step counter
2. Counter incremented by 1 per each simulated step (of M).
3. Keep emulating M for $m+1$ steps, or until it halts (whichever comes first)
4. Accept if M has halted and accepted; otherwise, Reject

By Pigeon-hole-principle, must have seen some configuration twice, which means there is a loop

Reductions via Controlled Executions

$L_\infty = \{\langle M \rangle : \ L(M) \text{ is infinite}\}$

- By Rice Theorem: $L_\infty \notin \mathcal{R}$ (why?)

Theorem 31

$L_\infty \notin \mathcal{RE}$.

Proof's idea:

Reduction from $\overline{H_{TM}}$.

- We are after a reduction $f(\langle M, w \rangle) = \langle B_{M,w} \rangle$ such that
  - If M halts on $w \implies L(B_{M,w})$ is finite.
  - If M does not halt on $w \implies L(B_{M,w})$ is infinite.
- It will follow that $x \in \overline{H_{TM}} \iff f(x) \in L_\infty$
- Hence, $\overline{H_{TM}} \leq_m L_\infty$.
- Since $\overline{H_{TM}} \notin \mathcal{RE}$, this will imply $L_\infty \notin \mathcal{RE}$.

The TM $B_{M,w}$

Definition 32 ($B_{M,w}$)

On input y

1. Emulate $M(w)$ for $|y|$ steps.
2. Accept, if $M(w)$ did not halt in that many steps; Otherwise, Reject.

- $M(w)$ does not halt $\implies$ $B_{M,w}$ accepts all y's $\implies$ $\mathrm{L}(B_{M,w}) = \Sigma^* \implies \langle B_{M,w} \rangle \in \mathrm{L}_\infty$.
- $M(w)$ halts after k steps $\implies$ $B_{M,w}$ accepts only y's of length smaller than k $\implies$ $\mathrm{L}(B_{M,w})$ is finite $\implies$ $\langle B_{M,w} \rangle \notin \mathrm{L}_\infty$.

  Hence, $x \in \overline{\mathrm{H_{TM}}} \iff f(x) \in \mathrm{L}_\infty \implies \overline{\mathrm{H_{TM}}} \leq_m \mathrm{L}_\infty \implies \mathrm{L}_\infty \notin \mathcal{RE}$

Section 5

Computation Histories

Reduction via Computation Histories

Important technique for proving undecidability. Examples

- Basis for proof of undecidability in Hilbert's tenth problem (given a polynomial with integer coefficients, does it have a solution over the integers?).
- Another example: Does a context free grammar generate Σ^*?

Reminder: Configurations

Configuration: $1011q_70111$, means:

- state is q_7
- LHS of tape is 1011
- RHS of tape is 0111
- head is on RHS 0

- (configuration) uaq_ibv yields (i.e., $\Longrightarrow$) uq_jacv, if $\delta(q_i, b) = (q_j, c, \mathrm{L})$
- uaq_ibv yields $uacq_jv$ if $\delta(q_i, b) = (q_j, c, \mathrm{R})$
- Special case (left end of tape): q_ibv yields q_jcv if $\delta(q_i, b) = (q_j, c, \mathrm{L})$.

Emptiness of CFGs

We have seen an algorithm to check whether a CFG is empty.

Algorithm 33

On input $\langle G \rangle$ (where G is a CFG):

1. Mark all terminal symbols in G.
2. Repeat until no new variables become marked:
3. Mark any A where $A \to U_1 U_2 \ldots U_k$, and each U_i has already been marked.
4. Accept, if start symbol marked; otherwise Reject.

So $\text{EMPTY}_{\text{CFG}}$ is decidable.

Question 34

What about the complementary question: does a CFG generate all strings? Namely, does $\text{All}_{\text{CFG}} := \{\langle G \rangle : \ G \text{ is a CFG and } L(G) = \Sigma^*\} \in \mathcal{R}$

All_{CFG} is Undecidable

Theorem 35

All_{CFG} *is undecidable.*

Proof's idea: Reduction from A_{TM} to $\overline{\text{All}_{\text{CFG}}}$:

1. Given $\langle M, w \rangle$, construct a coding of a CFG, $\langle G \rangle$, that generates all strings that are not accepting computation histories for M on w
2. if M does not accept w, G generates all strings
3. if M does accept w, then G does not generate the accepting computation history.

The PDA

Instead of a CFG, we construct a PDA (recall equivalence) that "guesses" which condition is violated, and verifies the guessed violation.

Algorithm 36 (D)

On input $h = C_1 \# C_2 \ldots \# C_\ell$, check

1. Is there some C_i that is not a configuration of M (i.e., number of q symbols $\neq 1$)?
2. Is C_1 not the starting configuration of M on w?
3. Is C_ℓ not an accepting configuration of M?
4. $\exists i \in [\ell]$ s.t. $C_i \not\Rightarrow C_{i+1}$ according to δ – the transition function of M?

The last condition is the tricky one to check.

Checking $C_i \not\Rightarrow C_{i+1}$

Algorithm 37 (Checking $C_i \not\Rightarrow C_{i+1}$)

1. Push C_i onto the stack till $\#$.
2. Scan C_{i+1} and pop matching symbols of C_i

   Check if C_i and C_{i+1} match everywhere, except around the head position where difference dictated by transition function for M.

Problem

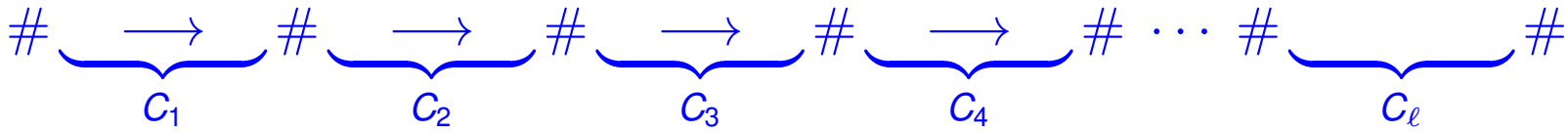
When C_i is popped from stack, it is in reverse order.

But we only trying to identify (ignoring the local changes around head position) the language $x\#y$, with $x \neq y$.

This can be done a PDA (see Lecture 5), but here we give a simpler solution.

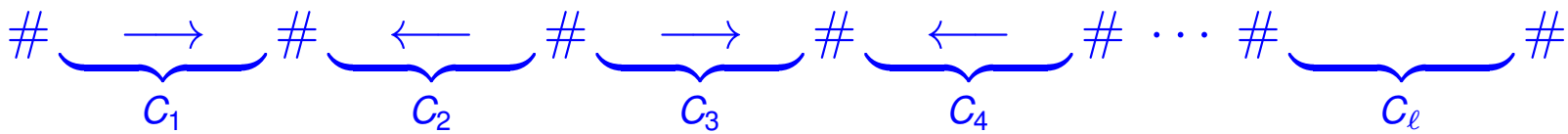
Checking $C_i \Rightarrow C_{i+1}$, take 2

- So far, we used a "straight" notion of accepting computation histories



- But why not employ an alternative notion of accepting computation history, one that will make the life of our PDA much easier?

  A solution: write the accepting computation history so that every other configuration is in reverse order.

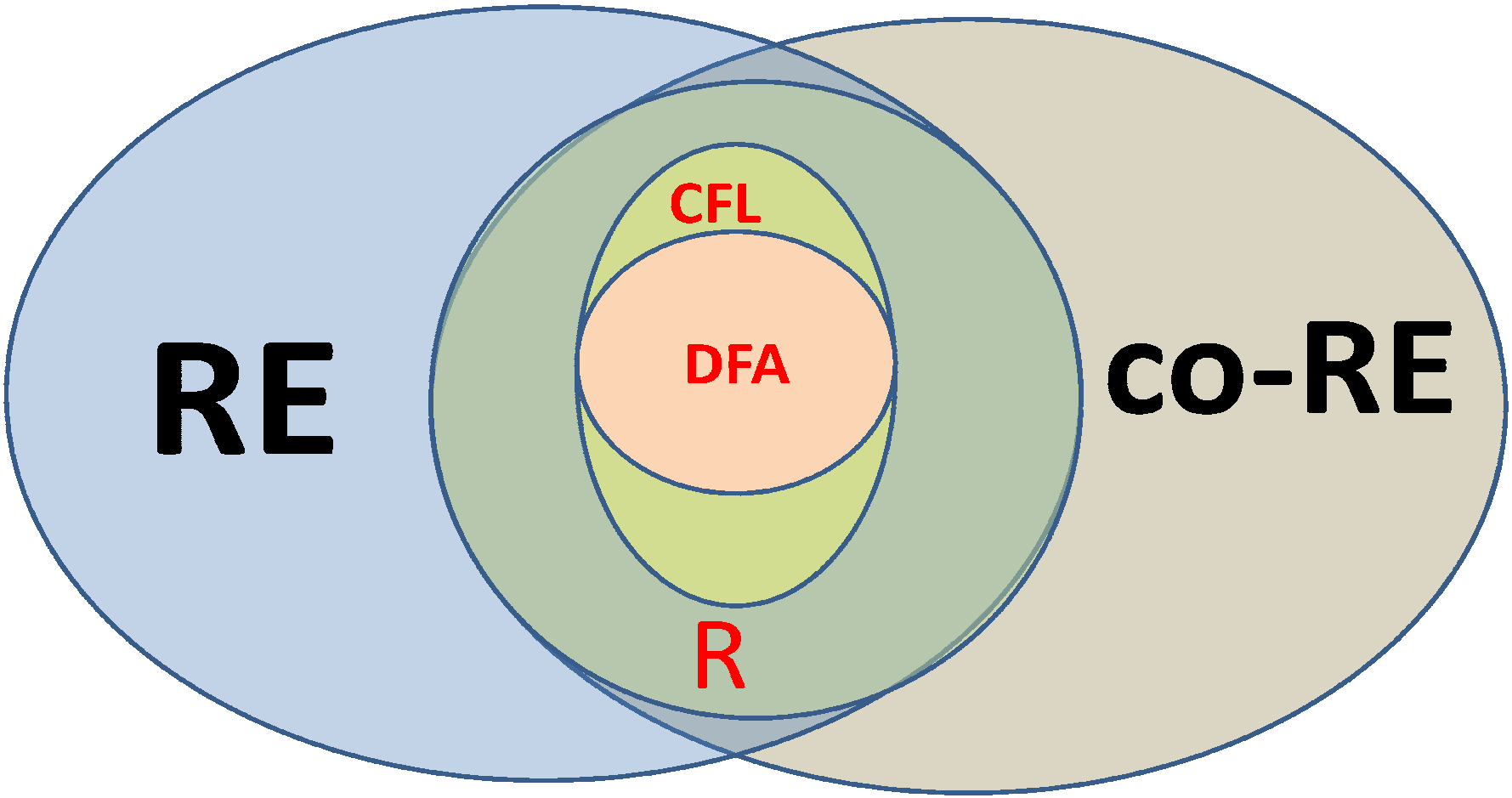


- This resolves the difficulty in the proof.

Putting It Together

- Given $\langle M, w \rangle$, we constructed (algorithmically) a PDA, D, that rejects the string z if and only if z equals an accepting computation history of M on w, written in the "alternating format".
- Therefore $\mathsf{L}(D)$ is either Σ^* or $\Sigma^* \setminus \{z\}$.
- This D has an equivalent (and efficiently described) CFG, G, namely $\mathsf{L}(D) = \mathsf{L}(G)$. So $\mathsf{L}(G)$ is either Σ^* or $\Sigma^* \setminus \{z\}$. The mapping $\langle M, w \rangle \mapsto \langle G \rangle$ is thus a reduction from A_{TM} to $\overline{\mathsf{All}_{\mathsf{CFG}}}$.
- (Since $\mathsf{A}_{\mathsf{TM}} \notin \mathcal{R}$) $\implies$ $\overline{\mathsf{All}_{\mathsf{CFG}}} \notin \mathcal{R}$.
- (Since $L \in \mathcal{R} \Leftrightarrow \overline{L} \in \mathcal{R}$) $\implies$ $\mathsf{All}_{\mathsf{CFG}} \notin \mathcal{R}$.
- Actually, this is also a mapping from $\overline{\mathsf{A}_{\mathsf{TM}}}$ to $\mathsf{All}_{\mathsf{CFG}}$, so $\mathsf{All}_{\mathsf{CFG}}$ is not in $\mathcal{RE}$. ♠

Revised View of the World of Languages



Decidability, Summary

- Turing Machine - a universal computational model
- Language classes RE, co-RE, and R.
- Not Decidable
  - Acceptance/Halting problem
  - Any non-trivial property of a program (Rice Theorem)
  - Questions with respect to Grammars.
  - much more exists ...
- Not in RE (what does it mean?)
- What does this imply to verification of software and hardware? (useless ???)