CS 301

Lecture 20 – Reductions

Stephen Checkoway

April 9, 2018

Reductions

Reductions are a way of saying, "If problem B can be solved, then problem A can as well"

Reductions

Reductions are a way of saying, "If problem B can be solved, then problem A can as well"

Example:
A: Passing CS 301
B: Getting good grades on assignments, labs, and exams

We say that A reduces to B (i.e., the problem of passing CS 301 reduces to the problem of getting good grades) because

- If you get good grades, then you will pass
- If you fail, then you did not get good grades (contrapositive)

Reductions

Reductions are a way of saying, "If problem B can be solved, then problem A can as well"

Example:
A: Passing CS 301
B: Getting good grades on assignments, labs, and exams

We say that A reduces to B (i.e., the problem of passing CS 301 reduces to the problem of getting good grades) because

- If you get good grades, then you will pass
- If you fail, then you did not get good grades (contrapositive)

But note:

- Passing CS 301 doesn't say anything about your grade
- Getting bad grades doesn't mean you'll fail

Reduction of languages

We say language A reduces to language B (written $A \leq B$) to mean
"If B is decidable, then A is decidable"

We use a reduction $A \leq B$ in two different ways

- Proving that language A is decidable. "Good-news reduction." If B is decidable, then A is decidable
- Proving that language B is undecidable. "Bad-news reduction." If A is undecidable, then B is undecidable

"Good-news reduction"

To prove that language A is decidable, we need to build a TM D that decides it

If B is a decidable language, we can let R be a TM that decides B and use it as a subroutine in D

D = "On input __,

1. Using the input, construct some input for R
2. Run R on that input (it's possible we need to use R multiple times)
3. Make some decision to *accept* or *reject* based on the outcome of R"

Now we just need to prove that $L(D) = A$ and that D is a decider

In this way, we have reduced A to B (i.e., $A \leq B$)

"Bad-news reduction"

To prove that language B is undecidable, we need to pick an undecidable language A and show that $A \leq B$

We start by assuming that B is decidable

Just as with the good-news reduction, we let R be a decider for B and use it as subroutine to construct a decider for A

D = "On input __,

1. Using the input, construct some input for R
2. Run R on that input (it's possible we need to use R multiple times)
3. Make some decision to *accept* or *reject* based on the outcome of R"

Now we just need to prove that $L(D) = A$ and that D is a decider

Since A is undecidable and we were able to construct a decider for it, our assumption that B is decidable must be wrong

Good-news reductions we've already seen

- $A_{\textsf{NFA}} \leq A_{\textsf{DFA}}$
- $A_{\textsf{REX}} \leq A_{\textsf{NFA}}$
- $EQ_{\textsf{DFA}} \leq E_{\textsf{DFA}}$
- Every regular language $A \leq A_{\textsf{DFA}}$
- Every context-free language $A \leq A_{\textsf{CFG}}$

Bad-news reductions we've already seen

- $\textsc{Diag} \leq A_{\mathsf{TM}}$
- $A_{\mathsf{TM}} \leq \textsc{Halt}_{\mathsf{TM}}$
- $A_{\mathsf{TM}} \leq E_{\mathsf{TM}}$

Equality of TMs

Let's prove that

$$EQ_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

is undecidable

Let's perform a bad-news reduction from E_{TM}

Proof.

Assume that EQ_{TM} is decided by some TM R and build a TM to decide E_{TM}:
D = "On input $\langle M\rangle$,

Equality of TMs

Let's prove that

$$EQ_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

is undecidable

Let's perform a bad-news reduction from E_{TM}

Proof.

Assume that EQ_{TM} is decided by some TM R and build a TM to decide E_{TM}:
D = "On input $\langle M\rangle$,

1. Construct TM M' such that $L(M') = \varnothing$

Equality of TMs

Let's prove that

$$EQ_{\mathsf{TM}} = \{\langle M_1, M_2 \rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

is undecidable

Let's perform a bad-news reduction from E_{TM}

Proof.

Assume that EQ_{TM} is decided by some TM R and build a TM to decide E_{TM}:
D = "On input $\langle M \rangle$,

1. Construct TM M' such that $L(M') = \varnothing$
2. Run R on $\langle M, M' \rangle$

Equality of TMs

Let's prove that

$$EQ_{\mathsf{TM}} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

is undecidable

Let's perform a bad-news reduction from E_{TM}

Proof.

Assume that EQ_{TM} is decided by some TM R and build a TM to decide E_{TM}:
D = "On input $\langle M\rangle$,

1. Construct TM M' such that $L(M') = \varnothing$
2. Run R on $\langle M, M'\rangle$
3. If R accepts, then *accept*; otherwise *reject*"

Since R is a decider, D is a decider
Clearly D accepts $\langle M\rangle$ iff R accepts $\langle M, M'\rangle$ iff $L(M) = \varnothing$ so $L(D) = E_{\mathsf{TM}}$ □

Reducing decidable languages to regular languages

Prove that if A is decidable and B is regular, then $A \leq B$

How do we do this? Try to prove it

Reducing decidable languages to regular languages

Prove that if A is decidable and B is regular, then $A \leq B$

How do we do this? Try to prove it

Hint: You want to prove that the logical proposition "B is decidable implies A is decidable" is true

Reducing decidable languages to regular languages

Prove that if A is decidable and B is regular, then $A \leq B$

How do we do this? Try to prove it

Hint: You want to prove that the logical proposition "B is decidable implies A is decidable" is true

Hint 2: The proposition $P \implies \mathit{true}$ is true

Reducing decidable languages to regular languages

Prove that if A is decidable and B is regular, then $A \leq B$
How do we do this? Try to prove it

Hint: You want to prove that the logical proposition "B is decidable implies A is decidable" is true

Hint 2: The proposition $P \implies \mathit{true}$ is true

Proof.
Since A is decidable, then the implication "B is decidable implies A is decidable" is always true. □

More general statement: If A is decidable and B is arbitrary, then $A \leq B$. Same proof.

Checking if the language of a TM is regular

Theorem

$\textsc{Regular}_{TM} = \{\langle M\rangle \mid M$ *is a TM and* $L(M)$ *is regular*$\}$ *is undecidable*

To prove this, we want to perform a bad-news reduction from some undecidable language

A useful technique for languages involving properties of languages of TMs (here the property is that the language is regular) involves reducing from A_{TM}

Given a TM M and a string w, we want to construct a new TM M' such that the language of M' is regular if $w \in L(M)$ and is nonregular if $w \notin L(M)$

Proof

Let's construct a TM whose language is $\{0,1\}^*$ if $w \in L(M)$ and is $\{0^n 1^n \mid n \geq 0\}$ if $w \notin L(M)$

Proof.

Assume that $\text{REGULAR}_{\text{TM}}$ is decided by some TM R. Build D to decide A_{TM}

D = "On input $\langle M, w \rangle$,

1. Construct a new TM
   M' = "On input x,
   1. If $x = 0^n 1^n$ for some n, *accept*
   2. Otherwise, run M on w and if M accepts, *accept*; otherwise *reject*"
2. Run R on $\langle M' \rangle$ and if R accepts, then *accept*; otherwise *reject*"

Proof

Let's construct a TM whose language is $\{0,1\}^*$ if $w \in L(M)$ and is $\{0^n 1^n \mid n \geq 0\}$ if $w \notin L(M)$

Proof.

Assume that $\textsc{Regular}_{\mathsf{TM}}$ is decided by some TM R. Build D to decide A_{TM}

D = "On input $\langle M, w\rangle$,

1. Construct a new TM
   M' = "On input x,
   1. If $x = 0^n 1^n$ for some n, *accept*
   2. Otherwise, run M on w and if M accepts, *accept*; otherwise *reject*"
2. Run R on $\langle M'\rangle$ and if R accepts, then *accept*; otherwise *reject*"

We need to show that D is a decider and we need to show that $L(D) = A_{\mathsf{TM}}$

Why is D a decider?

Proof

Let's construct a TM whose language is $\{0, 1\}^*$ if $w \in L(M)$ and is $\{0^n 1^n \mid n \geq 0\}$ if $w \notin L(M)$

Proof.

Assume that $\textsc{Regular}_{\mathsf{TM}}$ is decided by some TM R. Build D to decide A_{TM}

D = "On input $\langle M, w\rangle$,

1. Construct a new TM
   M' = "On input x,
   1. If $x = 0^n 1^n$ for some n, *accept*
   2. Otherwise, run M on w and if M accepts, *accept*; otherwise *reject*"
2. Run R on $\langle M'\rangle$ and if R accepts, then *accept*; otherwise *reject*"

We need to show that D is a decider and we need to show that $L(D) = A_{\mathsf{TM}}$

Why is D a decider? Note that we never *run* M'. All D does is *construct* a new TM and then run a decider on its representation

Proof

Let's construct a TM whose language is $\{0,1\}^*$ if $w \in L(M)$ and is $\{0^n 1^n \mid n \geq 0\}$ if $w \notin L(M)$

Proof.

Assume that $\text{REGULAR}_{\text{TM}}$ is decided by some TM R. Build D to decide A_{TM}

D = "On input $\langle M, w \rangle$,

1. Construct a new TM
   M' = "On input x,
   1. If $x = 0^n 1^n$ for some n, *accept*
   2. Otherwise, run M on w and if M accepts, *accept*; otherwise *reject*"
2. Run R on $\langle M' \rangle$ and if R accepts, then *accept*; otherwise *reject*"

We need to show that D is a decider and we need to show that $L(D) = A_{\text{TM}}$

Why is D a decider? Note that we never *run* M'. All D does is *construct* a new TM and then run a decider on its representation

If $w \in L(M)$, then $L(M') = \{0,1\}^*$ which is regular so R and D accept. If $w \notin L(M)$, then $L(M')$ is not regular so R and D reject. Thus $L(D) = A_{\text{TM}}$ □

ALL_{CFG} is undecidable

Theorem

$ALL_{CFG} = \{\langle G\rangle \mid G$ is a CFG and $L(G) = \Sigma^*\}$ is undecidable.

Proof idea.

We want to reduce from A_{TM}

Given a TM M and a string w, we want to construct a CFG G such that if $w \in L(M)$, then G fails to generate some string and if $w \notin L(M)$, then $L(G) = \Sigma^*$

The string that G should fail to generate is an accepting computation of M on w

Recall, a configuration C of a TM is a string $C = uqv$ where $u \in \Gamma^*$ is the tape to the left of the tape head, $q \in Q$ is the current state, and $v \in \Gamma^*$ is the nonblank portion of the tape below and to the right of the tape head

Proof idea continued

An accepting computation is a sequence of configurations $C_1, C_2, \ldots, C_n$ such that

1. $C_1 = q_0 w$ is the initial configuration (where w is the input)
2. C_i follows from C_{i-1} according to the TM's transition; i.e., C_i is the same as C_{i-1} except for the symbols right around the states
3. $C_n = u q_{\text{accept}} v$ for some $u, v \in \Gamma^*$

We want to create a CFG G that generates all strings *except* for the string $h = \#C_1\#C_2^{\mathcal{R}}\#\cdots\#C_n\#$ where $C_1, C_2, \ldots, C_n$ is an accepting computation of M on w

For technical reasons, we need every other C_i to be reversed

$$h = \#\underbrace{\quad\rightarrow\quad}_{C_1}\#\underbrace{\quad\leftarrow\quad}_{C_2^{\mathcal{R}}}\#\underbrace{\quad\rightarrow\quad}_{C_3}\#\underbrace{\quad\leftarrow\quad}_{C_4^{\mathcal{R}}}\#\cdots\#\underbrace{\quad\rightarrow\quad}_{C_n}\#$$

If $w \notin L(M)$, then no such accepting computation exists and $L(G) = \Sigma^*$

If $w \in L(M)$, then $L(G) = \Sigma^* \smallsetminus \{h\}$

Proof idea continued

Rather than construct a CFG directly, we can construct a PDA P and then convert it to a CFG G

P should nondeterministically (i.e., using ε-transitions) check that one of the three conditions does not hold:

1. If C_1 is not the initial configuration (which is hard coded into P), *accept*; otherwise *reject*
2. If C_2 does not follow from C_{i-1}, *accept*; otherwise *reject*
3. If C_n is not an accepting configuration, *accept*; otherwise *reject*

Condition 1 is easy to check: this branch of the PDA just checks that the input does not start with #q_0w#

Condition 3 is likewise easy: this branch of the PDA just checks that the state that appears before the final # is not q_{accept}

Proof idea continued

Condition 2 is the hard one. P will nondeterministically pick a configuration C_i to check if it follows from C_{i-1}

P will push C_{i-1} onto its stack (or $C_{i-1}^{\mathcal{R}}$, depending on i being odd or even)

Then P will match C_i (or $C_i^{\mathcal{R}}$) by popping the stack. The symbols around the states and the states themselves need to change according to M's transition function (this is the slightly tricky part)

This branch rejects if C_i properly follows from C_{i-1} and accepts otherwise

Proof

Proof.

Assume ALL_{CFG} is decided by TM R and construct TM D to decide A_{TM}:
D = "On input $\langle M, w\rangle$,

1. Construct PDA P based on M and w
2. Convert P to an equivalent CFG G
3. Run R on $\langle G\rangle$ and if R rejects, *accept*; otherwise *reject*"

None of constructing the PDA, converting to a CFG, and running a decider loop so D is a decider

If $w \in L(M)$, then P rejects the string corresponding to the accepting computation so $L(G) \neq \Sigma^*$. Therefore, R rejects and D accepts

If $w \notin L(M)$, then P accepts every string so $L(G) = \Sigma^*$ and R accepts and D rejects

Since A_{TM} is undecidable and D decides it, our assumption must be wrong and ALL_{CFG} is undecidable □

EQ_{CFG} is undecidable

Homework: Prove that EQ_{CFG} is undecidable

Reduce from ALL_{CFG}