Reductions and Computational Histories

Mark Greenstreet, CpSc 421, Term 1, 2006/07

- The Idea of Using Reductions
- Proving Undecidable Problems Using Reductions

Anatomy of a Reduction Proof (1/2)

Want to show that $A \preceq \mathcal{B}$.

- We can reduce the problem of deciding whether or not a string w is in A to deciding whether or not string w' is in B if for every string w, we can derive a string w' such that $(w \in A) \Leftrightarrow (w' \in B)$.
  - In this case, we say that A reduces to B, that B is at least as hard as A, and that A is no harder than B.
  - In general, we can talk about what we are allowed to do to transform w to w'. In this class, we will typically allow any transformation that can be performed by a Turing machine. In this case, we say that A is Turing reducible to B.
  - In other contexts, we could talk about "polynomial time reductions", "log-space reduction", etc.

Anatomy of a Reduction Proof (2/2)

- For problems in this class, reduction typically involves a bunch of Turing Machines:
  - M_B a machine that decides (recognizes, etc.) B.
  - Often, A is defined for strings that include TM descriptions. e.g.

$$A = \{M\#w \mid \text{where } M \text{ is a TM that} \ldots\}$$

  - We define a TM, M_A that decides (recognizes, etc.) A by:
    - Constructing a new machine, M' based on M and possibly w.
    - Runs M_B on an input that includes a description of M'.
    - M_A accepts if M_B accepts and M_A rejects if M_B rejects (and loops if M_B loops).
    - NOTE: we never actually run M' on anything!

Anatomy of a Reduction Proof (2/2)

- For problems in this class, reduction typically involves a bunch of Turing Machines:
  - We define a TM, M_A that decides (recognizes, etc.) A by:
    - Constructing a new machine, M' based on M and possibly w.
    - Runs M_B on an input that includes a description of M'.
    - M_A accepts if M_B accepts and M_A rejects if M_B rejects (and loops if M_B loops).
    - NOTE: we never actually run M' on anything!
  - Thus we have 4 TM's:
    - M_B: A machine that we are given that decides (recognizes, etc.) B.
    - M_A: A machine that we construct that decides (recognizes, etc.) A by using M_B.
    - M: A machine description that is part of the input to A.
    - M': A machine description that M_A derives from its input and includes in the input to M_B.
    - Note that to test $w \in A$, we run machine M_A which in turn runs machine M_B on string w'.
    - w includes a description of M, and w' includes a description of M'. We don't actually run these machines, we just manipulate their descriptions. In particular, M_B may or may not simulate M'.

Three Java Programs (1/3)

Program 1:

```
class hello {
  public static void main(String[] args) {
    System.out.println("Hello, world.");
  }
}
```

Three Java Programs (2/3)

Program 2:

```
class H {
  public static void main(String[] args) {
    System.out.println(
      "class hello {");
    System.out.println(
      "  public static void main(String[] args) {");
    System.out.println(
      "    System.out.println(\"Hello, world.\");");
    System.out.println(
      "  }");
    System.out.println(
      "}");
  }
}
```

Three Java Programs (3/3)

Program 3:

```
class X {
  public static void main(String[] args) {
    System.out.println(
      "class hello {");
    System.out.println(
      "  public static void main(String[] args) {");
    System.out.println(
      "    while(true);");
    System.out.println(
      "  }");
    System.out.println(
      "}");
  }
}
```

A Typical Reduction

```
Outcome = { accept, reject, loop}
Outcome M_A(String M#w) {
  M' = construct_new_TM(M and w');
  w' = construct_new_String(M and w');
  if(M_B(M'#w') == accept) return(accept);
  else return(reject);
}
```

$REGULAR$ is Undecidable

- Let $REGULAR = \{M \mid L(M) \text{ is regular}\}$.
- We'll show that $REGULAR$ is undecidable by reducing A_{TM} to $REGULAR$.
  - Assume that we have $M_{REGULAR}$ with $L(M_{REGULAR}) = REGULAR$.
  - Define $M_{A_{TM}}$ such that on input $M\#w$:
    - $M_{A_{TM}}$ constructs the description of Turing machine M'
      - On input w', M' checks to see if $w' \in 0^n1^n$.
      - If $w' \in 0^n1^n$, then M' accepts w'.
      - Otherwise M' runs M on input w.
      - If M accepts w, then M' accepts w'.
    - $M_{A_{TM}}$ runs $M_{REGULAR}$ with the description of M' as its input.
    - If $M_{REGULAR}$ accepts M', then $M_{A_{TM}}$ accepts $M\#w$.
    - If $M_{REGULAR}$ rejects M', then $M_{A_{TM}}$ rejects $M\#w$.
  - M' accepts Σ^* if M accepts w
  - M' accepts 0^n1^n if M does not accept w.
  - Thus, $L(MATM) = \{M\#w \mid M \text{ accepts } w\} = A_{TM}$.
- We have shown $A_{TM} \preceq REGULAR$.

$E_{TM} \preceq REGULAR$

- Define $M_{E_{TM}}$ such that on input M:
  - $M_{E_{TM}}$ constructs the description of Turing machine M'
    - On input w', M' checks if $M \in \overline{E_{TM}}$.
    - If $M \in overlineE_{TM}$ and $w' \in 0^n1^n$, then M' accepts w'.
    - Otherwise M' loops.
  - $M_{E_{TM}}$ runs $M_{REGULAR}$ with the description of M' as its input.
  - If $M_{REGULAR}$ accepts M', then $M_{E_{TM}}$ accepts M.
  - If $M_{REGULAR}$ rejects M', then $M_{E_{TM}}$ rejects M.
- If $L(M) = \emptyset$, then $L(M') = \emptyset$, and $M_{E_{TM}}$ accepts.
- If $L(M) \neq \emptyset$, then $L(M') = 0^n1^n$, and $M_{E_{TM}}$ rejects.
- This shows $E_{TM} \preceq REGULAR$.
- Thus, both $A_{TM} \preceq REGULAR$, and $E_{TM} \preceq REGULAR$.
- Neither $A_{TM} \preceq E_{TM}$ nor $E_{TM} \preceq A_{TM}$.
- $\therefore REGULAR$ is harder than A_{TM} (Turing recognizable) and harder then E_{TM} (Turing corecognizable).

Linear Bounded Automata

- A linear bounded automaton (LBA) is like a Turing Machine except that the input starts with a $\vdash$ and ends with a $\dashv$.
  - When an LBA reads $\vdash$ it must write $\vdash$ and move its head to the right.
  - When an LBA reads $\dashv$ it must write $\dashv$ and move its head to the left.
  - Thus, an LBA only uses as much tape as the size of its input.
- Let $A_{LBA} = \{B\#w \mid \text{LBA } B \text{ accepts } w\}$. A_{LBA} is Turing decidable. Proof:
  - Let w be an input to an LBA with tape alphabet Γ.
  - The LBA has at most $(|w| + 2) * |\Gamma|^{|w|}$ possible configurations:
    - $|w| + 2$ tape squares (including the $\vdash$ and $\dashv$).
    - $|w|$ of them have $|\Gamma|$ possible values each.
  - Just simulate B. If it within after $(|w| + 2) * |\Gamma|^{|w|}$ steps, you know if it accepts or rejects. Otherwise, it must be looping.

E_{LBA} is Not Decidable

- We'll reduce A_{TM} to $\overline{E_{LBA}}$.
- Assume that $M_{\overline{ELBA}}$ is a TM with $L(M_{\overline{ELBA}}) = \overline{E_{LBA}}$.
- Let $M_{A_{TM}}$ be a TM that on input string $M\#w$:
  - constructs an LBA, B' that on input w':
    - Checks to make sure that w' is of the form $\vdash w\Box^* \dashv$.
    - Runs M on w'.
    - If M accepts, then B' accepts.
    - If M reaches the $\dashv$, then B' rejects.
  - $M_{A_{TM}}$ runs $M_{\overline{ELBA}}$ on B'.
  - If $M_{\overline{ELBA}}$ accepts, $M_{A_{TM}}$ accepts.
  - If $M_{\overline{ELBA}}$ rejects, $M_{A_{TM}}$ rejects.
- If $w \in L(M)$, then there is some integer n such that M accepts w after n moves. M moves at most n squares to the right when accepting w. Thus, B' accepts $\vdash w\Box^n \dashv$.
- $(w \in L(M)) \Leftrightarrow (L(B') \neq \emptyset)$.
- $\therefore A_{TM} \preceq \overline{E_{LBA}}$.

Computational Histories

E_{LBA} is Not Decidable (again)

Reading List:

- Today: Sipser, 5.1
- Nov. 13: Remembrance Day (no lecture)
- Nov. 15: Midterm 2
- Nov. 17: Sipser, 5.2
- Nov. 20: Sipser, 5.3
- Nov. 22: Sipser, 6.1
- Nov. 24: Sipser, 6.2
- Nov. 27: Sipser, 6.2 (cont., final exam cut-off)
- Nov. 29: The GHz race is over, and what it means for you
- Dec. 1: Everything else about complexity theory