CPSC 313: Introduction to Computability **Fall 2016**

Lecture 30: Reductions

Lecturer: Dr. Catalin Dohotaru *Scribe: Shane Sims*

Add some intro for the notes here about the general method of reduction and how this fits into what we learned in the last two lectures

30.1 Reduction: A first example

Theorem 30.1 *The language Halt* $= \{\langle M, w\rangle \mid M$ *halts on input* $w\}$ *is undecidable.*

Proof: Suppose for contradiction, that the language Halt, is not undecidable. Then there is a Turing Machine, denoted H, deciding Halt.

Proof Idea: From this reductio assumption, it will be shown that we can can use H to build another TM, SH that decides the language SelfHalt (seen last lecture). But since we have proved SelfHalt to be undecidable, we will have discovered a contradiction. This contradiction negates our assumption and proves that in fact, Halt is also undecidable.

We use H to build a machine SH as follows: Given any string w, the machine SH should first verify that $w = \langle M\rangle$ for some Turing Machine M (and rejects otherwise). Then SH writes the string $ww = \langle M, M\rangle$ and passes it to H. Thus we have described a machine that decides SelfHalt, using our assumption that Halt is decidable. But this is impossible as shown in last lecture. So we have a contradiction and have proven that Halt is undecidable, as required.

■

30.1.1 Reduction by Formal Definition

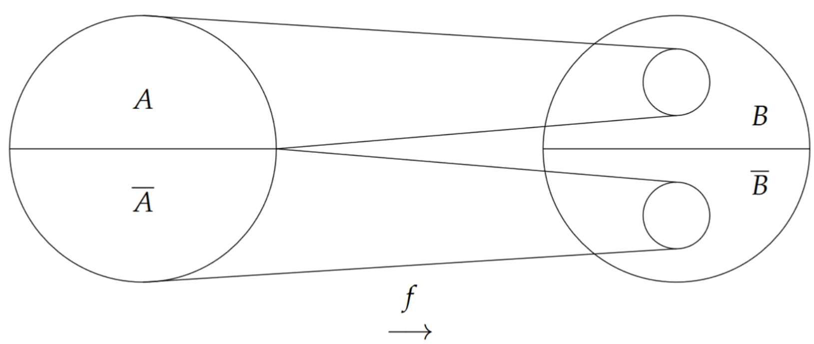

Figure 30.1: A reduction, f, from A to B

Definition 30.2 *A function* $f : \Sigma^* \to \Sigma^*$ *is a* ***computable function*** *if there exists a Turing Machine that, for every input* $w \in \Sigma^*$*, halts with* $f(w)$ *written on its tape.*

Note: We say that language A ***reduces*** *to language B, denoted* $A \leq_m B$ *if there exists a computable reduction from **A** to **B**.*

Theorem 30.3 *If* $A \leq_m B$ *and A is undecidable, then so is B.*

Proof: Assume that B is decidable and let $f : \Sigma^* \to \Sigma^*$ such that $w \in A \leftrightarrow f(w) \in B$ for every $w \in \Sigma^*$. Let M_B be a Turing Macine that decides B and M_f be a Turing Machine that computes f. Then the following machine decides the language A:
Operation of M on winput w:

1. Simulate M_f on w to compute $f(w)$.
2. Simulate M_B on $f(w)$.
3. Accept if M_B accepts and reject if M_B rejects.

This contradicts the fact that A is not decidable.
In general, to prove a that a language L is undecidable, reduce a known undecidable language to L. ■

Theorem 30.4 *The language* $ANY = \{\langle M \rangle | L(M)$ *contains at least one string*$\}$ *is undecidable.*

Proof Idea: We know that HALT is undecidable (shown in 30.1). So if we can reduce HALT to ANY (i.e. $HALT \leq_m ANY$), then we will have shown taht ANY is undecidable by appealing to Theorem 30.3, without having to do any further work.

Proof: Definition 30.2 tells us that HALT reduces to ANY if there exists a function f such that $w \in HALT \leftrightarrow f(w) \in ANY$. That is:
If a string w is an element of the language HALT, then its transformation under f is an element of the language ANY. If $w \notin HALT$ then $f(w) \notin ANY$.

It remains to find f as described in Definition 30.2:
First we define a Turing Machine, M' as follows:

On input $\langle M, w \rangle$, M' will

1. Run M on input w.
2. If M accepts w, then M' accepts w.
3. If M rejects w, then M' **accepts** w.
4. If M loops on w, then M' loops on w.

Now we define the function $f : \Sigma^* \to \Sigma^*$ as follows:

$$f(x) = \begin{cases} \langle M', w \rangle & : x = \langle M, w \rangle \\ x & \quad otherwise \end{cases}$$

Now we can check to see if f is a reduction from HALT to ANY.
Suppose we have input $\langle M, w \rangle \in HALT$. Then:

$\langle M, w\rangle \in HALT \to$ M halts on $w \to M'$ accepts $w \to M' \in ANY \to f(\langle M, w\rangle) \in ANY$.

Then we have that $\langle M, w\rangle \in HALT$, then $f(\langle M, w\rangle) \in ANY$.
We must now show that If $f(\langle M, w\rangle) \in ANY$, then $\langle M, w\rangle \in HALT$. But this is equivalent to $\langle M, w\rangle \notin HALT \to f(\langle M, w\rangle) \notin ANY$, which is more suitable to our purpose.

Suppose we have $\langle M, w\rangle \notin HALT \to$ M does not halt on $w \to M'$ does not halt on $w \to M' \notin ANY \to f(\langle M, w\rangle) \notin ANY$.

Then f is a reduction and $HALT \leq_m Any$, proving that ANY is undecidable, as required. ∎