CS81 Spring 2012: Les Undecidables

Earlier, we focus on languages that are either *regular* or *context-free*. Here, we will be interested in languages that are *decidable*, *semidecidable*, or neither (*not semidecidable*). Recall that a language is decidable if there is a Turing machine that always halts on any input and outputs "Yes" (by entering the halting state y) or "No" (by entering the halting state n). A language is *semidecidable* if there is a Turing machine that halts and accepts (by entering the state y) if the input is an element of the language.

We define some variants of HALT. Let $w \in \Sigma^\star$ be a string and let $S \subseteq \Sigma^\star$ be a set of strings.

- $\mathsf{HALT}_w = \{\langle M \rangle : M \text{ halts on } w\}$.
- $\mathsf{HALT}_S = \{\langle M \rangle : M \text{ halts on each } w \in S\}$.

Note the languages we consider range over strings which are valid encoding of Turing machines. Thus, the complement operation is also defined with respect to this set of strings. We use $\overline{L}$ or L^c interchangeably for the complement of L.

We list and describe some natural languages and their undecidability levels.

1. $\boxed{\mathsf{HALT} = \{(\langle M \rangle, w) : M \text{ halts on } w\} \in \mathcal{SD} \setminus \mathcal{D}.}$

   Proof. HALT is semidecidable since there is a Turing machine that, on input $(\langle M \rangle, w)$, runs $M(w)$ and accept if this halts.

   To show HALT is not decidable, we use a **diagonalization** argument. Assume for contradiction that HALT is decidable by a Turing machine U. We define another Turing machine called D:

   $D(\langle M \rangle)$:
   (a) If $U(\langle M \rangle, \langle M \rangle)$ accepts, then run forever.
   (b) Else halt.

   We arrive at a contradiction by asking if $D(\langle D \rangle)$ halts or not. □

   Remark: The second part of the above proof is based on Cantor's famous *diagonalization* argument (the same proof that shows the set of real numbers in $[0, 1)$ is not countable).

2. $\boxed{\overline{\mathsf{HALT}} = \{(\langle M\rangle, w) : M \text{ does not halt on } w\} \notin \mathcal{SD}.}$

Proof. Assume for contradiction that $\overline{\mathsf{HALT}}$ is semidecidable. Then, since HALT is also semidecidable, we have HALT is decidable. This is a contradiction. □

Remark: The above proof illustrates the main technique for showing that a language is not even semidecidable. It relies on the fact that if both L and $\overline{L}$ are semidecidable, then L (and hence also $\overline{L}$) is decidable. If we know that $\overline{L}$ is semidecidable but *not* decidable, then L is cannot be semidecidable.

3. $\boxed{\mathsf{HALT}_\epsilon = \{\langle M\rangle :\ M \text{ halts on } \epsilon\} \in \mathcal{SD} \setminus \mathcal{D}.}$

Proof. HALT_ϵ is semidecidable since a Turing machine, on input $\langle M\rangle$, can simulate M on ϵ and accepts if this halts.

To show HALT_ϵ is not decidable, we show $\mathsf{HALT} \preceq \mathsf{HALT}_\epsilon$; that is, we reduce HALT to HALT_ϵ. Assume for contradiction that HALT_ϵ is decidable by a Turing machine E. Consider the following Turing machine U for HALT:

$U(\langle M\rangle, w)$:

(a) Define a Turing machine D where:

$D(x)$:

i. If $x = \epsilon$ then run $M(w)$.

ii. Else run forever.

(b) Run $E(\langle D\rangle)$.

Note $D(\epsilon)$ halts iff $M(w)$ halts. So, U decides HALT, a contradiction. □

Remark: The second part of the above proof shows a **reduction** from HALT to HALT_ϵ (which we denote $\mathsf{HALT} \preceq \mathsf{HALT}_\epsilon$). The idea of a reduction $A \preceq B$ is to show that any "algorithm" for B can be helpful in constructing an "algorithm" for A.

4. $\boxed{\mathsf{HALT}_\exists = \{\langle M\rangle : M \text{ halts on some string}\} \in \mathcal{SD} \setminus \mathcal{D}.}$

Proof. $\mathsf{HALT}_\exists$ is semidecidable since there is a Turing machine A that accepts it:

$A(\langle M \rangle)$:

(a) Let $w_1, w_2, \ldots$ be a lexicographic enumeration of all strings.

(b) For an enumeration of pairs (j, k) of positive integers:

  i. Run M on w_j for k steps.

  ii. If this halts, then accept.

  iii. Else continue.

We show $\mathsf{HALT}_\exists$ is not decidable by showing $\mathsf{HALT} \preceq \mathsf{HALT}_\exists$. Assume for contradiction that $\mathsf{HALT}_\exists$ is decidable by a Turing machine E. We construct a Turing machine U for HALT:

$U(\langle M \rangle, w)$:

(a) Define a Turing machine D:

  $D(x)$:

  i. Run M on w.

  ii. If this halts, then halt.

(b) Ask $E(\langle D \rangle)$.

Note $M(w)$ halts iff D halts on some (actually all) string. $\square$

5. $\boxed{\mathsf{HALT}_\forall = \{\langle M \rangle : M \text{ halts on all inputs}\} \notin \mathcal{SD}.}$

Proof. $\mathsf{HALT}_\forall$ is not decidable since $\mathsf{HALT} \preceq \mathsf{HALT}_\forall$ (by using the same proof as $\mathsf{HALT} \preceq \mathsf{HALT}_\exists$). But to show $\mathsf{HALT}_\forall \notin \mathcal{SD}$, we need a different reduction. Assume for contradiction that $\mathsf{HALT}_\forall \in \mathcal{SD}$. We construct a Turing machine U for $\overline{\mathsf{HALT}}$ (the complement of HALT):

$U(\langle M \rangle, w)$:

(a) Define a Turing machine A:

  $A(x)$:

  i. If $M(w)$ halts in $|x|$ steps, then run forever.

  ii. Else halt.

(b) Ask $E(\langle A \rangle)$.

Note A halts on all strings iff $M(w)$ does not halt. $\square$

Remark: The above reduction is interesting in that it uses the inputs to the Turing machine A as *timers* for the simulation of M on its input w.

6. $\boxed{\mathsf{ACCEPT} = \{(\langle M \rangle, w) : M \text{ accepts } w\} \in \mathcal{SD} \setminus \mathcal{D}.}$

Proof. ACCEPT is semidecidable by a Turing machine that, on input $(\langle M \rangle, w)$, simulates M on w and accepts if this accepts.

To show ACCEPT is not decidable, we show HALT $\preceq$ ACCEPT. Assume for contradiction that ACCEPT is decided by a Turing machine E. We construct a Turing machine U for HALT:

$U(\langle M \rangle, w)$:

(a) Define a Turing machine A:

$A(x)$:

i. If $x = w$, then run M on w.

ii. If this halts, then accept.

(b) Ask $E(\langle A \rangle, w)$.

Note A accepts w iff M halts on w. $\square$

The following variants of ACCEPT are similar to the ones for HALT. The proofs are similar to the above reduction of HALT $\preceq$ ACCEPT.

7. $\boxed{\mathsf{ACCEPT} = \{(\langle M \rangle, w) : M \text{ accepts } w\} \in \mathcal{SD} \setminus \mathcal{D}.}$

8. $\boxed{\mathsf{ACCEPT}_\epsilon = \{\langle M \rangle : M \text{ accepts } \epsilon\} \in \mathcal{SD} \setminus \mathcal{D}.}$

9. $\boxed{\mathsf{ACCEPT}_\exists = \{\langle M \rangle : M \text{ accepts some string}\} \in \mathcal{SD} \setminus \mathcal{D}.}$

10. $\boxed{\mathsf{ACCEPT}_\forall = \{\langle M \rangle : M \text{ accepts all strings}\} \notin \mathcal{SD}.}$

11. $\boxed{\mathsf{EQUAL}_{TM} = \{(\langle M_1 \rangle, \langle M_2 \rangle) : L(M_1) = L(M_2)\} \notin \mathcal{SD}.}$

Proof. We show $\mathsf{ACCEPT}_\forall \preceq \mathsf{EQUAL}_{TM}$. Suppose EQUAL_{TM} is semidecidable by a Turing machine E. We construct a Turing machine U for $\mathsf{ACCEPT}_\forall$:

$U(\langle M \rangle)$:

(a) Let A be a Turing machine that accepts all strings.
(b) If $E(\langle M \rangle, \langle A \rangle)$ accepts, then accept.
(c) Else run forever.

Thus, M accepts all strings iff $E(\langle M \rangle, \langle A \rangle)$ accepts (since $L(A) = \Sigma^\star$). □

12. $\boxed{\mathsf{TM}_{MIN} = \{\langle M \rangle : M \text{ is the smallest Turing machine for } L(M)\} \notin \mathcal{SD}.}$

Proof. Assume for contradiction that TM_{MIN} is semidecidable. Any semidecidable language L has a Turing machine *enumerator* E which outputs elements of L. If L is accepted by a Turing machine A, the enumerator E for L may be constructed by simulating A on all inputs w_j and all running times k (using a dovetailing enumeration of (j, k)). Consider the following Turing machine U for L:

$U(x)$:

(a) Run E until it outputs $\langle M \rangle$ so that $|\langle M \rangle| > |\langle U \rangle|$.
(b) Run M on x.

By the property of E, M is the smallest Turing machine that accepts $L(M)$, but U accepts $L(M)$ and it has a smaller description than M. □

Remark: The above proof shows another method for showing that a language is not semidecidable. It uses the fact that semidecidable languages are *enumerable*.

13. $\mathsf{ACCEPT}_{\overline{REG}} = \{\langle M \rangle : L(M) = \{a^n b^n : n \geq 0\}\} \notin \mathcal{SD}$.

Proof. To show $\mathsf{ACCEPT}_{\overline{REG}}$ is not semidecidable, we show $\overline{\mathsf{HALT}} \preceq \mathsf{ACCEPT}_{\overline{REG}}$. Assume for contradiction that $\mathsf{ACCEPT}_{\overline{REG}}$ is semidecidable by a Turing machine E. We construct a Turing machine U that accepts $\overline{\mathsf{HALT}}$.

$U(\langle M \rangle, w)$:

(a) Define a Turing machine A:

$A(x)$:

i. If $x = a^n b^n$ for some $n \geq 0$, then accept.

ii. Else run M on w. If this halts, then accept.

(b) If $E(\langle A \rangle)$ accepts, then accept.

Note, if M halts on w then $L(A) = \Sigma^\star$, else $L(A) = \{a^n b^n : n \geq 0\}$. So, E accepts $\langle A \rangle$ iff M does not halt on w. Hence, U accepts $\overline{\mathsf{HALT}}$. $\square$