Sariel Har-Peled and Madhusudan Parthasarathy

Lecture 23: Rice Theorem and Turing machine behavior properties

21 April 2009

This lecture covers Rice's theorem, as well as decidability of TM behavior properties.

1 Outline & Previous lecture

1.1 Forward outline of lectures

This week and next, we'll see three major techniques for proving undecidability:

- Rice's Theorem (today): generalize a lot of simple reductions with common outline.
- Linear Bounded automata (Thursday): Allow us to show that ALL_{CFG}, EQ_{CFG} are undecidable. Also, LBAs illustrate a useful compromise in machine power: much of the flexibility of a TM but enough resource limits to be more analyzable.
- Post's Correspondence problem (a week from Thursday): allows us to show that $AMBIG_{\mathsf{CFG}}$ is undecidable.

1.2 Recap of previous class

In the previous class, we proved that the following language is undecidable.

$$\text{Regular}_{\mathsf{TM}} = \left\{ \langle M \rangle \;\middle|\; M \text{ is a } \mathsf{TM} \text{ and } L(M) \text{ is regular} \right\}.$$

To do this, we assume that $\text{Regular}_{\mathsf{TM}}$ was decided by some TM S. We then used this to build a decider for A_{TM} (which can not exist)

Decider for A_{TM}

(i) Input $= \langle M, w \rangle$
(ii) Construct $\langle M_w \rangle$ (see below).
(iii) Feed $\langle M_w \rangle$ to S and return the result.

Our auxiliary TM M_w looked like:

TM M_w:

(i) Input $= x$

(ii) If x has the form $\mathtt{a}^n\mathtt{b}^n$, halt and accept.

(iii) Otherwise, simulate M on w.

(iv) If the simulation accepts, then accept.

(v) If the simulation rejects, then reject.

The language of M_w was either Σ^* or $\mathtt{a}^n\mathtt{b}^n$, depending on whether M accepts w.

2 Rice's Theorem

2.1 Another Example - The language L_3

Let us consider another reduction with a very similar outline. Suppose we have the following language

$$\mathsf{L}_3 = \left\{ \langle M \rangle \;\middle|\; |L(M)| = 3 \right\}.$$

That is L_3 contains all Turing machines whose languages contain exactly three strings.

Lemma 2.1 *The language L_3 is undecidable.*

Proof: Proof by reduction from A_{TM}. Assume, for the sake of contradiction, that L_3 was decidable and let $\mathtt{decider}_{\mathsf{L}_3}$ be a TM deciding it. We use $\mathtt{decider}_{\mathsf{L}_3}$ to construct a Turing machine $\mathtt{decider}_9\text{-}\mathrm{A}_{\mathsf{TM}}$ deciding A_{TM}. The decider TM$\mathtt{decider}_9\text{-}\mathrm{A}_{\mathsf{TM}}$ is constructed as follows:

```
decider_9-A_TM ( ⟨M, w⟩ )
        Construct a new Turing machine M_w:

            M_w( x ): // x:  input
                res ← Run M on w
                if (res = reject) then
                    reject
                if x = UIUC or x = Iowa or x = Michigan then
                    accept

                reject

        return decider_L3 (⟨M_w⟩).
```

(We emphasize here again, that constructing M_w involve taking the encoding of $\langle M \rangle$ and w, and generating the encoding of $\langle M_w \rangle$.)

Notice that the language of M_w has only two possible values. If M loops or rejects w, then $L(M_w) = \emptyset$. If M accepts w, then th the language of M_w contains exactly three strings: "UIUC", "Iowa", and "Michigan".

So $\mathtt{decider}_9\text{-}\mathrm{A}_{\mathsf{TM}}\big(\langle M_w \rangle\big)$ accepts exactly when M accepts w. Thus, $\mathtt{decider}_9\text{-}\mathrm{A}_{\mathsf{TM}}$ is a decider for A_{TM} But we know that A_{TM} is undecidable. A contradiction. As such, our assumption that L_3 is decidable is false. ∎

2.2 Rice's theorem

Notice that these two reductions have very similar outlines. Our hypothetical decider **decider** looks for some property P. The auxiliary TM's tests x for membership in an example set with property P. The big difference is whether we simulate M on w before or after testing x and, consequently, whether the second possibility for $L(M_w)$ is $\emptyset$ or Σ^*.

It's easy to cook up many examples of reductions similar to this one, all involving sets of TM's whose *languages* share some property (e.g. they are regular, they have size three). Rice's Theorem generalizes all these reductions into a common result.

Theorem 2.2 (Rice's Theorem.) *Suppose that* L *is a language of Turing machines; that is, each word in* L *encodes a TM. Furthermore, assume that the following two properties hold.*

(a) Membership in L *depends only on the Turing machine's language, i.e. if* $L(M) = L(N)$ *then* $\langle M \rangle \in \mathsf{L} \Leftrightarrow \langle N \rangle \in \mathsf{L}$.

(b) The set L *is "non-trivial," i.e.* $\mathsf{L} \neq \emptyset$ *and* L *does not contain all Turing machines.*

Then L *is a undecidable.*

Proof: Assume, for the sake of contradiction, that L is decided by TM**deciderForL**. We will construct a TM`Decider`$_4$-A_{TM} that decides A_{TM}. Since `Decider`$_4$-A_{TM} does not exist, we will have a contradiction, implying that **deciderForL** does not exist.

Remember from last class that $\mathsf{TM}_\emptyset$ is a TM (pick your favorite) which rejects all input strings. Assume, for the time being, that $\mathsf{TM}_\emptyset \notin \mathsf{L}$. This assumption will be removed shortly.

Since L is non-trivial, also choose some other TM $Z \in \mathsf{L}$. Now, given $\langle M, w \rangle$ `Decider`$_4$-A_{TM} will construct the encoding of the following TM M_w.

TM M_w:

(1) Input $= x$.

(2) Simulate M on w.

(3) If the simulation rejects, halt and reject.

(4) If the simulation accepts, simulate Z on x and accept if and only if T halts and accepts.

If M loops or rejects w, then M_w will get stuck on line (2) or stop at line (3). So $L(M_w)$ is $\emptyset$. Because membership in L depends only on a Turing machine's language and $\langle \mathsf{TM}_\emptyset \rangle$ is not in L, this means that M_w is not in L. So M_w will be rejected by N.

If M accepts w, then M_w will proceed to line (4), where it simulates the behavior of Z. So $L(M_w)$ will be $L(Z)$. Because membership in L depends only on a Turing machine's language and T is L, this means that M_w is in L. So M_w will be accepted by N.

As usual, our decider for A_{TM} looks like:

```
Decider_4-A_TM (⟨M, w⟩)
        Construct ⟨M_w⟩ from ⟨M, w⟩
        return deciderForL (⟨M_w⟩)
```

So $\texttt{Decider}_4\texttt{-}\mathrm{A}_{\mathsf{TM}}$ $(\langle M, w\rangle)$ will accept $\langle M, w\rangle$ iff **deciderFor**L accepts M_w. But we saw above that **deciderFor**L accepts M_w iff M accepts w. So $\texttt{Decider}_4\texttt{-}\mathrm{A}_{\mathsf{TM}}$ is a decider for A_{TM}. Since such a decider cannot exist, we must have been wrong in our assumption that there was a decider for L.

Now, let us remove the assumption that $\mathsf{TM}_\emptyset \notin \mathsf{L}$. The above proof showed that L is undecidable, assuming that $\langle \mathsf{TM}_\emptyset \rangle$ was not in L. If $\mathsf{TM}_\emptyset \in \mathsf{L}$, then we run the above proof using $\overline{\mathsf{L}}$ in place of L. At the end, we note that $\overline{\mathsf{L}}$ is decidable iff L is decidable. ■

3 TM decidability by behavior

3.1 TM behavior properties

One thinking about TMs there are three kind of properties one might consider:

(1) The language accepted by the TM's, e.g. the TM accepts the string "`UIUC`". In this case, such a property is very likely undecidable by Rice's theorem.

(2) The TM's structure, e.g. the TM has 13 states. In this case, the property can probably be checked directly on the given description of the TM, and as such this is (probably) decidable.

(3) The TM's behavior, e.g. the TM never moves left on input "UIUC". This kind properties can be either decidable or not depending on the behavior under consideration, and this classification might be non-trivial.

3.2 A decidable behavior property

For example, consider the following set of Turing machines:

$$L_R = \left\{ \langle M \rangle \;\middle|\; M \text{never moves left for the input } x \text{, where } x \text{ is the empty word} \right\}.$$

Surprising, the language L_R is decidable because never moving left (equivalently: always moving right) destroys the Turing machine's ability to do random access into its tape. It is effectively made into a DFA.

Specifically, if a Turing machine M never moves left, it reads through the whole input, then starts looking at blank tape cells. Once it is on the blank part of the tape, it can cycle through its set of states. But after $|Q|$ moves, it has run out of distinct states and must be in a loop. So, if you watch M for four moves (the length of the string "UIUC") plus $|Q| + 1$ moves, it has either halted or its in an infinite loop.

Therefore, to decide L_R, you simulate the input Turing machine for $|Q| + 5$ moves. After that many moves, it has either

- moved left (in which case you reject), or
- has halted or gone into an infinite loop without ever moving left (in which case you accept).

This algorithm is a decider (not just a recognizer) for L, because it definitely halts on any input Turing machine M.

3.3 An undecidable behavior property

By contract, consider the following language:

$$L_x = \left\{ \langle M \rangle \;\middle|\; M \text{ writes an } x \text{ at some point, when started on blank input} \right\}.$$

This language L_x is undecidable. The reason is that a Turing machine with this restriction (no writing x's) can simulate a Turing machine without the restriction.

Proof: Suppose that L_x were decidable. Let R be a Turing machine deciding L_x. We will now construct a Turing machine S that decides A_{TM}.

S is constructed as follows:

- Input is $\langle M, w \rangle$, where M is the code for a Turing Machine and w is a string.
- Construct the code for a new Turing machine M_w as follows
  - (a) On input y (which will be ignored).
  - (b) Substitute X for x every where in $< M >$ and w, creating new versions $< M' >$ and w'.
  - (c) Simulate M' on w'
  - (d) If M' rejects w', reject.
  - (e) If M' accepts w', print x on the tape and then accept.
- Run R on $\langle M_w \rangle$. If R accepts, then accept. If R rejects, then reject.

If M accepts w, then M_w will print x on any input (and thus on a blank input). If M rejects w or loops on w, then M_w is guaranteed never to print x accidently. So R will accept $\langle M_w \rangle$ exactly when M accepts w. Therefore, S decides A_{TM}.

But we know that A_{TM} is undecidable. So S can not exist. Therefore we have a contradiction. So L_x must have been undecidable. ∎

A More examples

The following examples weren't presented in lecture, but may be helpful to students.

A.1 The language L_{UIUC}

Here's another example of a reduction that fits the Rice's Theorem outline.

Let

$$L_{\text{UIUC}} = \left\{ \langle M \rangle \;\middle|\; L(M) \text{ contains the string "UIUC"} \right\}.$$

Lemma A.1 *L_{UIUC} is undecidable.*

Proof: Proof by reduction from A_{TM}. Suppose that L_{UIUC} were decidable and let R be a Turing machine deciding it. We use R to construct a Turing machine deciding A_{TM}. S is constructed as follows:

- Input is $\langle M, w \rangle$, where M is the code for a Turing Machine and w is a string.
- Construct code for a new Turing machine M_w as follows:
  - Input is a string x.
  - Erase the input x and replace it with the constant string w.
  - Simulate M on w.
- Feed $\langle M_w \rangle$ to R. If R accepts, accept. If R rejects, reject.

If M accepts w, the language of M_w contains all strings and, thus, the string "UIUC". If M does not accept w, the language of M_w is the empty set and, thus, does not contain the string "UIUC". So R($\langle M_w \rangle$) accepts exactly when M accepts w. Thus, S decides A_{TM}

But we know that A_{TM} is undecidable. So S does not exist. Therefore we have a contradiction. So L_{UIUC} must have been undecidable. ■

A.2 The language Halt_Empty_TM

Here's another example which isn't technically an instance of Rice's Theorem, but has a very similar structure.

Let

$$\mathsf{Halt\_Empty\_TM} = \left\{ \langle M \rangle \;\middle|\; M \text{ halts on blank input} \right\}.$$

Lemma A.2 Halt_Empty_ *TM is undecidable.*

Proof: By reduction from A_{TM}. Suppose that Halt_Empty_TM were decidable and let R be a Turing machine deciding it. We use R to construct a Turing machine deciding A_{TM}. S is constructed as follows:

- Input is $\langle M, w \rangle$, where M is the code for a Turing Machine and w is a string.
- Construct code for a new Turing machine M_w as follows:
  - Input is a string x.
  - Ignore the value of x.
  - Simulate M on w.
- Feed $\langle M_w \rangle$ to R. If R accepts, then accept. If R rejects, then reject.

If M accepts w, the language of M_w contains all strings and, thus, in particular the empty string. If M does not accept w, the language of M_w is the empty set and, thus, does not contain the empty string. So $R\big(\langle M_w \rangle \big)$ accepts exactly when M accepts w. Thus, S decides A_{TM}

But we know that A_{TM} is undecidable. So S can not exist. Therefore we have a contradiction. So Halt_Empty_TM must have been undecidable. ■

A.3 The language $L_{\texttt{111}}$

Here is another example of an undecidable language defined by a Turing machine's behavior, to which Rice's Theorem does not apply.

Let

$$L_{\texttt{111}} = \left\{ \langle M \rangle \,\middle|\, M \text{ prints three one's in a row on blank input} \right\}.$$

Lemma A.3 *The language $L_{\texttt{111}}$ is undecidable.*

Proof: Suppose that $L_{\texttt{111}}$ were decidable. Let R be a Turing machine deciding $L_{\texttt{111}}$. We will now construct a Turing machine S that decides A_{TM}.

The decider S for A_{TM} is constructed as follows:

- Input is $\langle M, w\rangle$, where M is the code for a Turing Machine and w is a string.
- Construct the code for a new Turing machine M', which is just like M except that
  - every use of the character `1` is replaced by a new character $\texttt{1}'$ which M does not use.
  - when M would accept, M' first prints `111` and then accepts
- Similarly, create a string w' in which every character `1` has been replaced by $\texttt{1}'$.
- Create a second new Turing machine M'_w which simulates M' on the hard-coded string w'.
- Run R on $\langle M'_w\rangle$. If R accepts, accept. If R rejects, then reject.

If M accepts w, then M'_w will print `111` on any input (and thus on a blank input). If M does not accept w, then M'_w is guaranteed never to print `111` accidently. So R will accept $\langle M'_w\rangle$ exactly when M accepts w. Therefore, S decides A_{TM}.

But we know that A_{TM} is undecidable. So S can not exist. Therefore we have a contradiction. So $L_{\texttt{111}}$ must have been undecidable. ■