Lecture Notes:
The Halting Problem; Reductions

COMS W3261
Columbia University

20 Mar 2012

1 Review

Key point. *Turing machines can be encoded as strings, and other Turing machines can read those strings to peform "simulations".*

Recall two definitions from last class:

Definition 1. *A language is* **Turing-recognizable** *if there exists a Turing machine which halts in an accepting state iff its input is in the language.*

Definition 2. *A language is* **Turing-decidable** *if it halits in an accepting state for every input in the language, and halts in a rejecting state for every other input.*

Intuitively, for recognizability we allow our TM to run forever on inputs that are not in the language, while for decidability we require that the TM halt on every input.

Now recall our two most important results:

Theorem 1. *There exist (uncountably many!) languages which are not Turing-recognizable.*

Proof. (intuitive) There are as many strings as natural numbers, because every (finite) string over a finite alphabet can be encoded as a binary number. There are as many TMs as strings, because every TM can be encoded as a string. Thus, both the strings and the TMs are countably infinite.

There are as many languages as real numbers. Every language is a subset of the set of strings; we can think of this subset as being encoded by an infinite binary sequence (i.e. a real number) with 1s at indices corresponding to strings in the language and 0s everywhere else.

Thus there are more languages than Turing machines; we conclude that some (indeed, "most") languages are not recognized by any TM. □

Now we will construct a specific undecidable language.

Definition 3. *The language* $X_{TM} = \{\langle M \rangle : M \text{ does not accept } \langle M \rangle\}$

Theorem 2. X_{TM} *is not Turing-decidable.*

Question. *Give the "paradox" proof.*

Proof 1. (by Epimenides' paradox) Suppose we had a Turing machine M deciding this language. What happens when we run M with input $\langle M \rangle$? If we claim it accepts, then by definition it ought to reject; if it rejects, then it ought to accept! Either way, we have a contradiction. □

Alternatively, here's a proof that it's not even Turing-recognizable.

Question. *Give the "diagonalization" proof.*

Proof 2. (by diagonalization) Suppose we constructed an (infinite) chart listing all the Turing machines and all encodings of Turing machines. We write 1 in cell $[i, j]$ if M_i accepts $\langle M_j \rangle$ and 0 otherwise:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\cdots$
M_1	0	1	1	
M_2	1	1	1	
M_3	1	0	1	
$\vdots$				$\ddots$

(The particular arrangement of 1s and 0s is unimportant—it will depend on our encoding scheme.) Note that by reading the 1s from the ith row, we get the language decided by M_i. Does X_{TM} appear in any row of this list? Not the first row: M_1 rejects $\langle M_1 \rangle$, so $\langle M_1 \rangle$ **must be in** X_{TM}. Not the second row: M_2 accepts $\langle M_2 \rangle$, so $\langle M_2 \rangle$ **must not be in** X_{TM}. Continuing with this procedure, we can show that X_{TM} is not the same as any row of this enumeration of TMs, and as a consequence is not decided by any TM. □

2 The Accepting Problem

X_{TM} is admittedly a rather strange language, and it's not obvious why we should really care that it's not recognizable. Let's look at a different one:

Definition 4. *The language* $A_{TM} = (\{\langle M \rangle, w) : M \text{ accepts } w\}$

The question of whether A_{TM} is decidable is precisely the question of whether one TM can predict the output of another TM!

Question. *Is this language Turing-recognizable?*

Yes. Given $\langle M \rangle, w$ as input, we can just simulate M on w.

Question. *Is this language Turing-decidable?*

No.

Theorem 3. *A_{TM} is undecidable.*

Proof. Suppose, towards contradiction, that we had a TM M deciding A_{TM}. Then we could construct a TM N deciding X_{TM}, as follows:

- Given input $\langle P \rangle$:
- Run M on $\langle P, P \rangle$.
- If M accepts, reject.
- If M rejects, accept.

Clearly, N is a TM deciding X_{TM}. But we know that there no such TM exists! Because the existence of M implies the existence of N, we conclude that M also does not exist. □

Clarification. *Understand why A_{TM} is recognizable but not decidable.*

Key point. *There is no Turing machine which can predict the output of all other Turing machines. However, there are Turing machines which can predict the output of* some *other Turing machines.*

3 The Halting Problem

Let's try a more modest goal: rather than actually attempting to predict output, let's just predict whether a Turing machine *halts* on its input, or runs forever.

Definition 5. *The language $HALT_{TM} = \{(\langle M \rangle, w) : M \text{ halts on input } w\}$*

Question. *Is this language Turing-recognizable?*

Yes: again, it just simulates (but it will run forever if the simulated machine does!).

Question. *Is this language Turing-decidable?*

No.

Theorem 4. *$HALT_{TM}$ is undecidable.*

Proof. This is very much like the previous proof. Suppose we had a TM M deciding $HALT_{TM}$. The we could construct a TM N deciding A_{TM}, as follows:

- Given input $(\langle P \rangle, w)$:
- Run M on $(\langle P \rangle, w)$.
- If M rejects, reject: P runs forever given input w, so does not accept w.

- Otherwise, we know that P terminates, and it is "safe" to simulate it on w. Do so.
- If P accepts, accept.
- If P rejects, reject.

Clearly, N decides A_{TM}. Similarly to the previous proof, we know that A_{TM} is undecidable, so M cannot exist. □

Key point. *There is no Turing machine which finds infinite loops in all other Turing machines (though again, it may do so for a restricted subset). This is a problem which would have real, practical benefits if it were soluble, but it's not.*

Clarification. *Is this clear?*

4 Other Undecidable Languages

Definition 6. *The language* $E_{TM} = \{\langle M \rangle : L(M) = \emptyset\}$

Question. *Is this language decidable?*

No: we can construct a decider for A_{TM} again.

Proof. Suppose M decides E_{TM}.

- Given input $\langle P, w \rangle$:
- Construct a new TM P' which rejects if its input is not w, and otherwise simulates P on w.
- Run M on $\langle P \rangle$.
- If M accepts, reject.
- If M rejects, accept.

We have a contradiction. □

5 Reductions

In working through these examples we've come across a very powerful proof technique: to prove that some language is undecidable, we assume that we have a decider, and show that this implies the existence of a decider for a known undecidable problem. This technique is called **reduction**.

There are several different kinds of reduction; the kind that we've discussed so far is called **Turing reduction** (for reasons that are hopefully obvious). The general idea is that

we assume we have a TM which computes the solution to one problem, and we use that to compute the solution to another problem.

To say that a solver for B allows us to solve A ("A reduces to B"), we write

$$A \leq_T B$$

(I've always found this notation weird. An easy way to remember it is that if A is reducible to B, the machine that computes B is at least as powerful as the machine that computes A: it is definitely powerful enough to solve A, and maybe other problems as well. So A is "easier than" B.)

Question. *$A_{TM} \leq_T X_{TM}$, or vice-versa?*

Vice-versa.

Let's formalize the proof technique we've been using so far:

Theorem 5. *$A \leq_T B$ and B decidable $\rightarrow$ A decidable.*

Corollary 1. *$A \leq_T B$ and A undecidable $\rightarrow$ B undecidable.*

6 Other Undecidable Languages, continued

Now let's try a few more reductions:

Definition 7. *The language $REG_{TM} = \{\langle M \rangle : L(M)$ is regular$\}$*

Question. *Prove that this is undecidable. (Hint: reduce from A_TM)*

Proof. Suppose we have a decider M for REG_{TM}.

- Given input $\langle P, w \rangle$:
- Construct a new TM P' as follows:
  - If the input is of the form 0^n1^n, accept.
  - Otherwise, ignore the input and run P on w
- Run M on P'

  If M doesn't accept w, $L(P') = \{0^n1^n\}$, and is not regular. If M accepts w, $L(P') = \Sigma^*$. Thus,
- If M accepts, accept.
- If M rejects, reject.

Once again, a contradiction. □

Definition 8. *The language* $EQ_{TM} = \{(\langle M \rangle, \langle N \rangle) : L(M) = L(N)\}$

Question. *Prove that this is undecidable. (Hint: reduce from* $E_T M$*)*

Proof. Suppose we have a decider M for EQ_{TM}. Construct the following decider for E_{TM}:

- Given input $\langle P \rangle$:
- Construct a TM Q which rejects immediately on all inputs.
- Run M on $(\langle P \rangle, \langle Q \rangle)$, and accept iff it accepts.

Once again, a contradiction. □

7 Next Class

It seems like every interesting language of Turing machines we've come up with is undecidable. Is there any property of Turing machines which can be decided?