CS 373 Summer 2011
Theory of Computation
Lucas Cook

Lecture 22: Rice's Theorem Proof

2011 July 20 at 2:00 PM

---

1 Definitions

In this lecture, we defined a *property* P of recognizable languages as a set of recognizable languages. A property is said to be *trivial* if it holds for all recognizable languages or for none. (i.e. $P = \varnothing$ and $P = ALLR$ are trivial.) Thus for a *nontrivial* property P, there must exist at least one language that has the property and one langauge that does not have the property (i.e. recognizable L_1 and L_2 such that $L_1 \in P$ and $L_2 \notin P$).

For a property P, we defined the *language of the property* P as the set of TMs that accept languages with property P:

$$L_P = \{\langle M\rangle \mid M \text{ is a } \mathsf{TM} \text{ and } \mathcal{L}(M) \text{ has the property } P\}$$

$$= \{\langle M\rangle \mid M \text{ is a } \mathsf{TM} \text{ and } \mathcal{L}(M) \in P\}$$

2 Theorem

Theorem 2.1 *(Rice's Theorem) For every nontrivial property P, L_P is undecidable.*

This is saying that TMs cannot decide interesting features about the languages of other TMs. Put in vague but intuitive terms, programs cannot analyze programs and definitively determine interesting properties about their purpose.

3 Proof

The proof is in the form of a reduction from the halting problem A_{TM} to L_P. We are able to make a 'generic' reduction for any P by taking one TM M_1 such that $\mathcal{L}(M_1) \in P$ and another TM M_2 such that $\mathcal{L}(M_2) \notin P$.

Proof:

Given nontrivial P, we will prove L_P is undecidable by cases: $\varnothing \notin P$ and $\varnothing \in P$.

First, assume $\varnothing$ is not in P. Since P is nontrivial, there is some TM M_1 such that $\mathcal{L}(M_1) \in P$. We will show $A_{TM} \leq_m L_P$. If some TM M_P decides L_P, then the following TM decides A_{TM}:

```
M_ATM: on input ⟨M, w⟩
  construct a TM N as follows:
    N: on input x
      run M on input w
      if M accepts
          run M_1 on input x
          if M_1 accepts, accept x
          else reject x
      else reject x
  run M_P on input ⟨N⟩.
  if M_P accepts, accept ⟨M, w⟩
  else reject ⟨M, w⟩
```

When M accepts w, N accepts strings exactly when M_1 would. Thus $\mathcal{L}(N) = \mathcal{L}(M_1) \in P$. When M rejects w, N rejects everything. Thus $\mathcal{L}(N) = \varnothing \notin P$, as assumed. When M loops on w, N never accepts anything, thus $\mathcal{L}(N) = \varnothing \notin P$. Therefore M accepts w exactly when $\mathcal{L}(N)$ has property P: $\langle M, w\rangle \in A_{TM} \Leftrightarrow \langle N\rangle \in L_P$.

Now, assume that $\varnothing$ is in P. There is some TM M_2 such that $\mathcal{L}(M_2) \notin P$ because P is nontrivial. Since $\varnothing \notin \overline{P}$, we can perform a similar reduction $A_{TM} \leq_m L_{\overline{P}}$ using M_2 in place of M_1. Thus $L_{\overline{P}}$ is undecidable.

Note that $L_{\overline{P}} = \overline{L_P}$, since the TMs in $\overline{L_P}$ recognize the exact set of languages that do not have P. To achieve contradiction, assume L_P is decided by TM M_P. Then we could flip the accept and reject states of M_P and decide $\overline{L_P}$, which has been shown to be undecidable. Thus L_P is also decidable.

■