Rice's Theorem

Robert M. Keller
Harvey Mudd College
13 April 2005

What is Rice’s Theorem?

- Could be considered a “meta-theorem” due to its sweeping character.

- Applies to ***functional*** questions: languages and functions computed by Turing machines in the abstract;

  not to TM ***structural*** questions.

Questions about languages

- We can’t simply hand an **entire language** to a program and ask a question about it. A language can, in general, be infinite.
- We need a way to give the language in a **finite representation**.
- One way of doing this is to use a **machine** that recognizes the language as the representation.

Questions about languages

- So when we speak of properties of languages, we will assess that property through the representation, e.g. the machine description. This immediately means that we are restricting ourselves to the universe **recursively-enumerable languages**.

- But we must make sure that the property in question is a property of the language and **not of a specific machine**.

- In other words, if we perform a test on one machine recognizing the language, we get the same answer for another machine recognizing the same language.

Functional vs. Structural

- Functional questions:
  - Is there an algorithm for determining whether a TM recognizes **the empty language**?
  - Is there an algorithm for determining whether a TM recognizes **all strings**?
  - Is there an algorithm for determining whether a TM recognizes **a finite language**?

- Structural questions:
  - Is there an algorithm for determining whether a TM has **more than 500 reachable control states**?
  - Is there an algorithm for determining whether a TM ever uses **more than 500 reachable cells of tape on a given input**?

Trivial Functional Properties

- A functional property is called "**trivial**" if it is true for all machine-recognizable (i.e. recursively-enumerable) languages or none of them.

- A **non-trivial property**, then, holds for **some** languages, but **not all**.

Non-Trivial Functional Properties

- Whether the language recognized by a machine M
  - includes the empty string Λ.
  - is a finite set.
  - is the empty set.
  - is the set of all strings.
  - is recursive.
  - etc.
- In each case, some machines do and some don't have the property. But the properties can be viewed as properties of the **language** recognized by the machine.

Rice’s Theorem

- Any non-trivial functional property of recursively-enumerable languages is not recursive.

- Put another way:

  For any non-trivial functional property, there is no algorithm that will determine whether or not the language recognized by a given TM has the property.

Observations about Non-Trivial Properties P

- Any given language has property P, or it does not.
- A language L has property P iff L does not have $\neg P$.
- To decide P, it is adequate to decide the complementary property $\neg P$, and vice-versa, since yes/no answers are demanded in both cases.

Proof of Rice’s Theorem (1 of 3)

- Suppose **P** is a non-trivial property.
- If the empty set ∅ has property P, then interchange P and ¬P and proceed under the assumption that ∅ does not have P. **This assumption is critical**.
- Let $\mathbf{L_P}$ be some arbitrary language with property P (which must exist, because P is non-trivial). We know that $\mathbf{L_P}$ is distinct from ∅, by the assumption above.
- The plan is to reduce the halting problem to that of deciding property P, which will imply that there is no algorithm for the latter. This will hinge on having the P decider differentiate between $\mathbf{L_P}$, which has property P, and ∅, which does not.

Proof of Rice's Theorem (2 of 3)

- Let $\mathbf{M_P}$ be a machine recognizing the chosen language $\mathbf{L_P}$.
- The reduction works as follows: Suppose we have an algorithm for testing property P for *any* input <M>.
- We can then test whether an arbitrary Turing machine T halts on an input x by constructing a machine M'(<T, x>) with the specifications on the following page.
- Note: <T, x> are **not** the **input** to M', but rather are part of the construction of M'.

Proof of Rice’s Theorem (3 of 3)

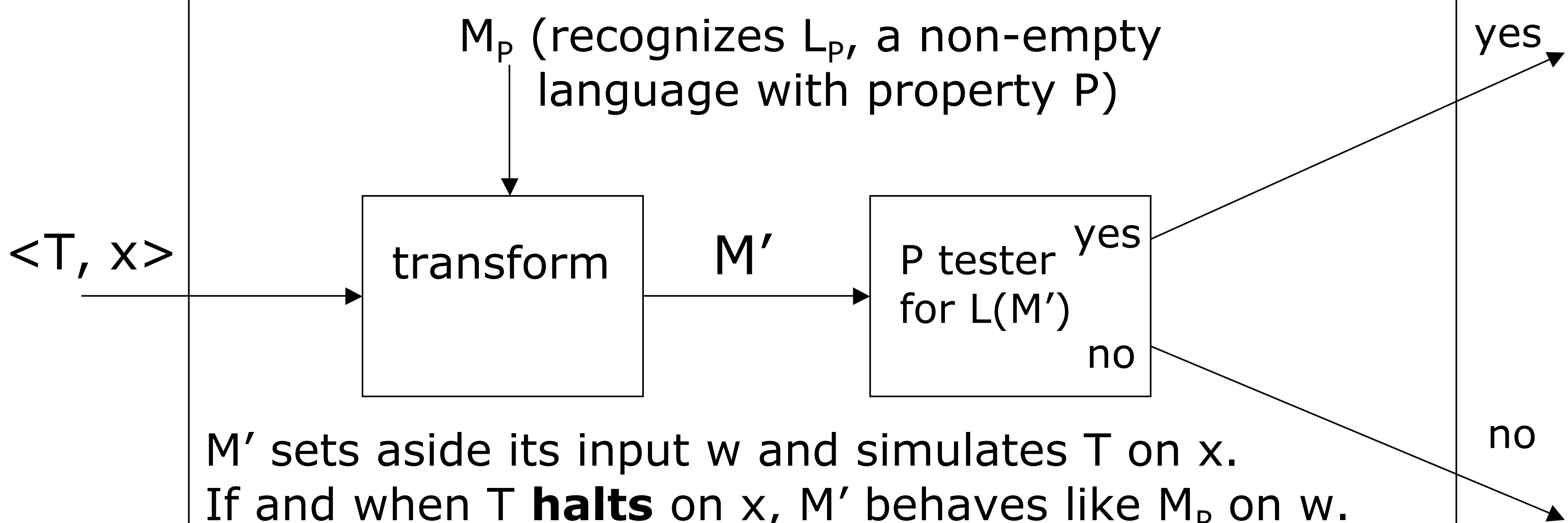
- M’(<T, x>): with input **w**, temporarily set aside w and **simulate** T on x.

- If the simulation of T on x **terminates**, then simulate $\mathbf{M_P}$ on the original input w. Thus if and when $\mathbf{M_P}$ terminates, M’ accepts w iff $\mathbf{M_P}$ accepts w, i.e. L(M’) = L($\mathbf{M_P}$). So in this case, L(M’) **has** property P, because $\mathbf{M_P}$ was selected to have it.

- If the above simulation **does not terminate**, then L(M’) = $\varnothing$ which **does not** have property P.

- So if we can test an arbitrary machine for its language having property P, we can test L(M’) in particular. But the answer to this test determines whether or not T halts on x.

T halts on x?



M' sets aside its input w and simulates T on x.
If and when T **halts** on x, M' behaves like M_P on w.
L_P has property P, so L(M') **has property P** in this case.

If T does **not halt** on x, neither does M', so M' recognizes the empty language, and thus L(M') **does not have property P**.

Thus L(M') has property P iff T halts on x.

Note

- Although P was discussed as a property of **language** recognized by a Turing machine, the same proof works in the case that:

  P is a property of the **partial function** computed by a Turing machine.

- This means there is no algorithm that will decide equivalence of an arbitrary machine's partial function to that of a given machine.

Note: Functional Properties of Languages are representable as Languages Themselves

- Consider some functional property P.
- We can define a property precisely as the language of descriptions of TMs that have the property:
- Example:

  $L_{AcceptsBlank} = \{<M> \mid M \text{ accepts the blank tape}\}.$