Theory of Computation
Chapter 5

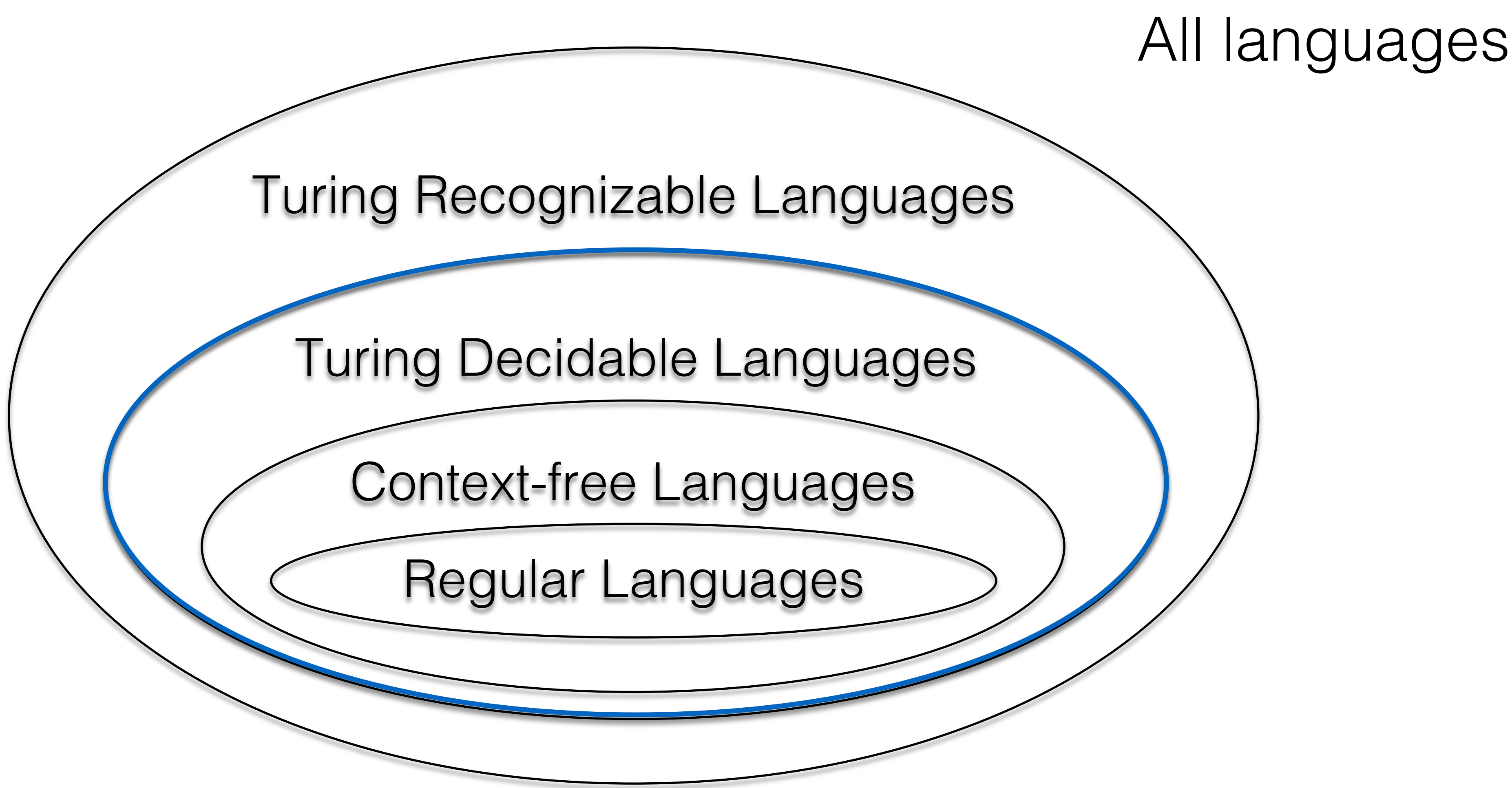
Reductions and Rice's Theorem

School of Engineering | Computer Science

Henry Rice
1920 - 2003

- American logician and mathematician best known as the author of [Rice's theorem]
- Doctoral dissertation at Syracuse University in 1951 with advisor Paul Rosenbloom
- Professor of Mathematics at the University of New Hampshire

Decidability



Theorem 5.3: $REGULAR_{TM}$

- Remember A_{TM} = {<M, w> | M is a TM and M accepts w} is undecidable
- <u>Theorem 5.3</u>: $REGULAR_{TM}$ = {<M> | M is TM and L(M) is a regular language} is undecidable
- <u>Proof Idea</u>: Assume a TM R decides $REGULAR_{TM}$. We can show R can be used to decide A_{TM} ($A_{TM} \leq REGULAR_{TM}$), which is a contradiction
  - If S decides A_{TM}, S takes input <M, w> and modifies M so that:
    1. If M accepts w, then M_w accepts any string Σ^* (This is a regular language, we randomly selected Σ^*)
    2. If M does not accept w, then M_w accepts $\{0^n 1^n \mid n \geq 0\}$ (This is not a regular language, again it was randomly selected)

Theorem 5.3: $REGULAR_{TM}$

- Theorem 5.3: $REGULAR_{TM}$ = {<M> | M is TM and L(M) is a regular language} is undecidable
- Proof Idea cont.: M_w is the modified TM. This TM is constructed only for the purpose of feeding its description into the deciders for $REGULAR_{TM}$ that we assume exists.
- How to design M_w:
  - M_w = "On input x:
    1. If x has form 0^n1^n, then accept.
    2. If x does not have this form, run M on w, and accept if M accepts w."
  - (Notice that if x does not match 0^n1^n, we move on to see if w is accepted. If w is accepted, then it does not matter what x is, we still reach the accept state. If w is not accepted, x is rejected.

Theorem 5.3: $REGULAR_{TM}$

- What do we now know about M_w?
  1. $L(M_w) \supseteq \{0^n1^n \mid n \geq 0\}$
  2. Can add more strings to $L(M_w)$ if M accepts w (Step 2 on previous slide)

Theorem 5.3: $REGULAR_{TM}$

- Theorem 5.3: $REGULAR_{TM}$ = {<M> | M is TM and L(M) is a regular language} is undecidable
- Proof: Let R be a TM to decide $REGULAR_{TM}$ and S to be a TM to decide A_{TM}.
- How to design M_w:
  - S = "On input <M, w>, where M is a TM and w is a string:
    1. Construct the following TM M_w.

       M_w = "On input x:

       1. If x has the form 0^n1^n, accept.
       2. If x does not have this form, run M on input w and accept if M accepts w."
    2. Run R on input <M_w>.
    3. If R accepts, accept; if R rejects, reject."

We cannot have this decider for A_{TM}

School of Engineering | Computer Science

Mapping Reductions

- The idea is that we set up some intermediate machine, M_w. We then use R, which is a ”black box” to distinguish between M_w's two possible cases.
- Ex: Analogy – We want to determine if someone drank poison. We create a machine M_w with two cases:
  - Face is purple
  - Face is not purple
  - We use R as a doctor that runs M_w and can determine the status of accept or reject. We then take R's advice.

Mapping Reductions 5.3

- What does it mean for a machine to be computable?
- Definition 5.17 Computable Function: A function f: $\Sigma^* \rightarrow \Sigma^*$ is computable if some Turing Machine M on every input w $\in \Sigma^*$ can halt with just f(w) on its tape.
- Examples of Computable Functions:
  - Addition, subtraction, the usual arithmetic operations

Mapping Reductions 5.3

- Definition 5.20 Mapping / Many-one Reduction: A language “A” is mapping-reducible to language ”B” (denoted $A \leq_M B$) if there exists a computable function $f: \Sigma^* \rightarrow \Sigma^*$ such that for all $w \in \Sigma^*$ $w \in A \Leftrightarrow f(w) \in B$. The function f is the **reduction** from A to B.
- Simple version: Take an input w, $w \in A$, compute $f(w)$, then plug $f(w)$ into a ”black box” solving B, and return B’s answer of membership for A.

Mapping Reductions 5.3

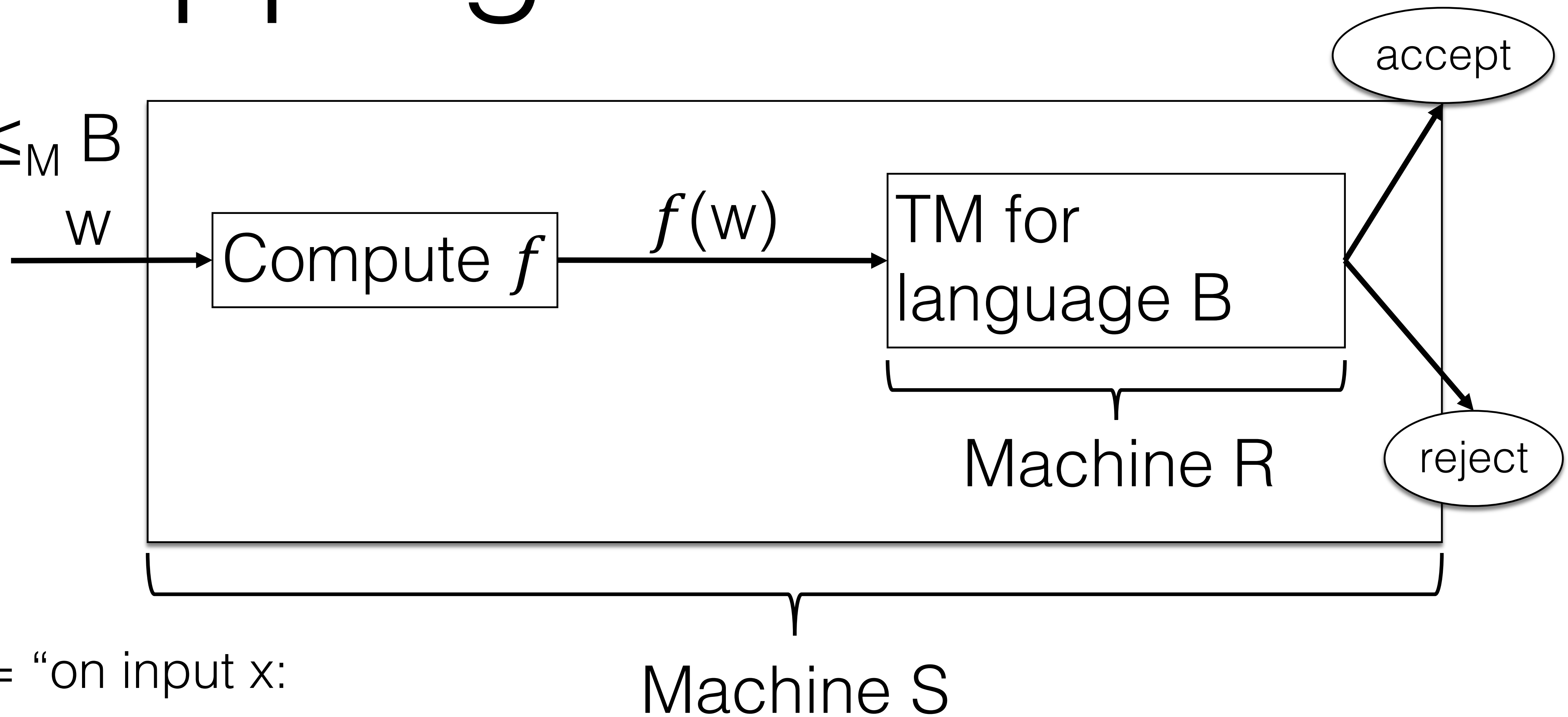
- Recall that $HALT_{TM}$ = {<M, w> | M is a TM such that M halts on w}
- In Theorem 5.1 we showed that $A_{TM} \leq HALT_{TM}$, thus $HALT_{TM}$ is undecidable
  - The steps we took were:
    1. Run TM R on input <M, w> to see if M would halt on w ("Plug <M, w> into a black box for $HALT_{TM}$)
    2. If R rejects, reject (If "black box" rejects, reject)
    3. Else run M on w for A_{TM} and return M's answer
  - Is this a mapping reduction?
    - No. Step 3 does post-processing after the "black box" is called.

$HALT_{TM}$ as a Mapping Reduction

- We can turn $HALT_{TM}$ = {<M, w> | M is a TM such that M halts on w} into a mapping reduction ($A_{TM} \leq_M HALT_{TM}$)
  - Make a TM F, which upon receiving an input <M, w> for A_{TM} it outputs <M′, w′> for $HALT_{TM}$ such that: <M, w> ∈ A_{TM} if and only if <M′, w′> ∈ $HALT_{TM}$
  - Define F = "on input <M, w>:
    1. Construct TM M′.
       - M′ = "on input x:
         a. Run M on x
         b. If M accepts, accept
         c. If M rejects, enter an infinite loop."
    2. Output <M′, w′> where w′ = w."

Mapping Reductions

- $A \leq_M B$



- S = "on input x:
  a. Compute f(w)
  b. Run Machine R on f(w)
  c. If R accepts f(w), then S accepts w
  d. If R rejects f(w), then S rejects w."

Mapping Reductions

- Implications:
  - If R is a decider for B, then S is a decider for A.
  - If R is a recognizer for B, then S is a recognizer for A.
  - $A \leq_M B$ means that "A is not harder than B".
- We can use the contrapositive to flip this around:
  - If S is not a decider for A, then R is not a decider for B.
  - If S is not a recognizer for A, then R is not a recognizer for B.
  - $A \leq_M B$ means that "B is at least as hard as A".

$E_{TM} \leq_M EQ_{TM}$

- Theorem 5.4: $E_{TM} \leq_M EQ_{TM}$, where $EQ_{TM} = \{<M_1, M_2> \mid M_1$ and M_2 are TMs and $L(M_1) = L(M_2)\}$
- Proof by Contradiction: Suppose we have a TM R deciding EQ_{TM}. We can show that $E_{TM} = \{<M> \mid M$ is a TM and $L(M) = \emptyset\}$ is decidable using R.
  - S = "On input <M>, where M is a TM:
    1. Run R on input $<M, M_1>$, where M_1 is a TM that rejects all inputs
    2. If R accepts, accept. If R rejects, reject."
  - Because of step 2, we have a mapping reduction from E_{TM} to EQ_{TM}. EQ_{TM} is undecidable since E_{TM} is undecidable.

Language Properties

- P is a property of the language of Turing Machines if, given TMs M_1 and M_2 with $L(M_1)$ and $L(M_2)$, machine M_1 has property P if and only if machine M_2 has property P. A property P is <u>non-trivia</u>l if some TM has P and some TM does not have P.
- Non-trivial properties of enumerable languages can include:
  - The language is finite, infinite, contains the empty string, contains no prime number, etc.
  - These are non-trivial properties since for each of them there is $L_1, L_2 \in$ Recursively Enumerable Languages, where L_1 satisfies the property, but L_2 does not.

Rice's Theorem

- Rice's Theorem (Problem 5.28): Let P be a non-trivial property of the language of TMs. Determining if a given TM satisfies P is undecidable
  - Let P be a language consisting of some set of TM descriptions where P fulfills two conditions:
    1. P is "non-trivial" (It contains some, but not all, TMs. If P was empty or all inclusive, it is easy to decide and does not tell us much. We want the cases in between.)
    2. For all TMs M_1 and M_2, if $L(M_1) = L(M_2)$ then $\langle M_1 \rangle \in P$ if and only if $\langle M_2 \rangle \in P$ (P is a property of $L(M_1)$ and $L(M_2)$.)
  - If these two conditions hold, then P is undecidable.

Rice's Theorem Proof

- <u>Rice's Theorem Proof</u>:
  - Assume, to the contrary, that there is a TM R_P that decides P. We use R_P to decide A_{TM} since $A_{TM} \leq_M P$.
  - Preliminary Steps:
    1. Let $T_\emptyset$ be a TM that always rejects ($L(T_\emptyset) = \emptyset$). We can assume that $< T_\emptyset> \notin P$.
    2. By Condition 1 of Rice's Theorem (Slide above), P is non-trivial, so we can say that L contains some TM T where $<T> \in P$.
  - R_P should now have the ability to distinguish between $T_\emptyset$ and T.

Rice's Theorem Proof

- <u>Rice's Theorem Proof cont.</u>:
  - Here is a TM S which decides input <M, w> to A_{TM} given access to R_P
    - S = "on input <M, w>
      1. Construct the following TM

         M_w = "on input x:

         1. Simulate M on w. If it halts and rejects w, reject x. If it accepts, go to Step 2.
         2. Simulate T on x. Accept if and only if T accepts x."
      2. Use TM R_P to decide if $\langle M_w \rangle \in P$. If yes, accept. If no, reject.

Rice's Theorem Proof

- Rice's Theorem Proof cont.:
  - What does the previous slide mean:
    - Case 1: M does not accept w. (For any input x, M_w will reject or get stuck in an infinite loop. Language of M_w is the empty set since $L(M_w) = \emptyset$ and $L(T_\emptyset) = \emptyset$, by Condition 2 (Rice's Theorem Slide) $M_w \in P$ if and only if $T_\emptyset \in P$. We know $T_\emptyset \notin P$, so $M_w \notin P$.)
    - Case 2: M accepts w. (We always run Step 2 of M_w and output T's answer on x. The language of M_w, $L(M_w) = L(T)$. (T and $T_\emptyset$ are not the same language). Therefore, by Condition 2 (Rice's Theorem Slide) $M_w \in P$ if and only if $T \in P$. Since $T \in P$, then $M_w \in P$.)
  - Again, this is not possible since A_{TM} is undecidable.

Try It

1. If I said $A \leq_M B$ and B is undecidable, what does this say about A?
2. If I said $A \leq_M B$ and A is undecidable, what does this say about B?
3. Discuss what the following means:

   $A \leq_M B$ and $B \leq_M C$

Try It

1. If I said $A \leq_M B$ and B is undecidable, what does this say about A?
    - Nothing. A is not harder than B, so A could be any undecidable, Turing Recognizable or decidable.
2. If I said $A \leq_M B$ and A is undecidable, what does this say about B?
    - B is undecidable. B is at least as hard as A, so B is undecidable as well.
3. Discuss what the following means:

    $A \leq_M B$ and $B \leq_M C$

    - This implies $A \leq_M C$, we will see an example of this in the next lecture.