Enoch Hwang

✶ ehwang@cs.ucr.edu ✶

CS215A Theory of Computation
Notes
February 15, 1995

2.1 Turing Machines TM

A Turing machine is a quadruple $M = (K, \Sigma, \delta, s)$, where K is a finite set of states, $s \in K$ is the initial state, Σ is a finite set of symbols, and δ is the transition function.

Church's Thesis: Anything that is "computable in reality" can be computed by a Turing Machine.

1. $L(M)$ means "the language accepted by M" or "the language recognized by M"
2. Given a machine M, define $L(M) = \{x \mid x \text{ is accepted by } M\}$

2.2 Recursive and Recursive Enumerable (p 24)

We say that M **decides** L whenever, for any string $x \in (\Sigma\text{-}\{\text{blank}\})^*$, if $x \in L$, then $M(x)$ = "yes" (i.e., M on input x halts at the "yes" state), and, if $x \notin L$, then $M(x)$ = "no". If L is **decided** by some Turing machine M, then L is called a **recursive language**.

We say that M simply **accepts** L whenever, for any string $x \in (\Sigma\text{-}\{\text{blank}\})^*$, if $x \in L$, then $M(x)$ = "yes" (i.e., M on input x halts at the "yes" state if x is in L), however, if $x \notin L$, then $M(x) = \nearrow$ (i.e. the machine never halts if x is not in L). If L is **accepted** by some Turing machine M, then L is called **recursive enumerable**.

We say that M **computes** function f if, for any string $x \in (\Sigma\text{-}\{\text{blank}\})^*$, $M(x) = f(x)$. That is, $f(x)$ is the output string when M halts. If such an M exists, f is called a **recursive function**.

1. If L is recursive, then it is recursively enumerable.
2. A decision problem is **decidable** if its corresponding language is recursive. There exists an algorithm that solves it.
3. An algorithm is just a TM that always halts.

2.3 TM with Multiple Strings

1. We say that M accepts x in time t if M accepts x after making t transitions.
2. M accepts L in time $f(n)$ if for all inputs x of length n, M accepts x in time $\leq f(n)$.
3. **TIME**$(f(n))$ = the set of all languages that can be accepted in time $f(n)$.

3b. If L is a polynomially decidable language, it is in TIME(n^k) for some integer $k > 0$. The union of all these classes, that is, the set of all languages decidable in polynomial time by Turing machines, is denoted by ***P***.

4. Off-line TM is a TM with one read-only input tape.

4b. $L \in$ **SPACE**($f(n)$) if there is an off-line TM M that accepts L and uses $\leq f(n)$ cells on the scratch tapes for all inputs of length n.

Tape Reduction Theorem: If M is a k-tape TM, then there is a 1-tape TM M' that is equivalent to M. That is $L(M') = L(M)$, and if M runs in time $f(n)$ then M' runs in time $O(f^2(n))$.

2.4 Linear Speedup Theorem

No matter how fast a machine is, you can always build another one that is faster. If $L \in$ Time($f(n)$) then $L \in$ Time($\varepsilon f(n) + n + 2$) for any $\varepsilon > 0$.

2.5 Space Bounds

2.7 NonDeterministic Machines (p 45)

A nondeterministic Turing machine is a quadruple $N = (K, \Sigma, \Delta, s)$. e.g. $\vartheta(q, a) = \{(p, b, \rightarrow), (p, a, \rightarrow), (r, b, \leftarrow)\}$. That is, for each state-symbol combination, there may be more than one appropriate next steps—or none at all.

An input is accepted if there is some sequence of nondeterministic choices that results in "yes". Other choices may result in rejection; just one accepting computation is enough.

The set of languages decided by nondeterministic Turing machines within time $f(n)$ is **NTIME**($f(n)$). The union of all these classes is denoted by ***NP***. $\mathbf{P} \subseteq \mathbf{NP}$

Theorem: If M is a NTM that runs in time $f(n)$ then there is a TM M' equivalent to M (i.e. $L(M) = L(M')$) that runs in time $c^{f(n)}$ for some $c > 1$. i.e. an exponential loss of efficiency. In other words,

$$\mathbf{NTIME}(f(n)) \subseteq \bigcup_{c>0} \mathbf{TIME}(c^{f(n)}) \qquad \text{N.B. TIME = DTIME}$$

Sawitch Theorem: If M is a NTM that uses space $f(n)$ then there is a TM M' equivalent to M (i.e. $L(M) = L(M')$) that uses space $f^2(n)$. Assuming $f(n) \geq \log n$ and $f(n)$ can be computed in space $f(n)$ (i.e. proper). In other words,

$$\mathbf{NSPACE}(f(n)) \subseteq \mathbf{SPACE}(f^2(n)) \qquad \text{N.B. SPACE = DSPACE}$$

3.1 A Universal Turing Machine U (p 57)

U's function is to simulate the behavior of M presented with input x. We can write $U(M;x) = M(x)$. That is, the universal Turing machine U on input $M;x$ simulates M on input x.

The Universal Language

$L_u = \{M; x \mid M \text{ accepts } x\}$, i.e. Given the description of a Turing machine M and its input x, does M accept the input x?

1. L_u is recursively enumerable but is not recursive.
2. L_d, the diagonal language, is not recursively enumerable and is not recursive.

3. $\overline{L}_d$, the complement of L_d, is recursively enumerable but is not recursive.

3.2 The Halting Problem H (p 59)

Given the description of a Turing machine M and its input x, will M halt on x?

1. HALTING (H) is recursively enumerable but is not recursive.
2. HALTING (H) is undecidable.
3. If we had an algorithm for deciding HALTING, then we would be able to derive an algorithm for deciding **any** recursively enumerable language.
   In other words, all recursively enumerable languages can be reduced to HALTING.

Post Theorem: If L is recursively enumerable and $\overline{L}$ is recursively enumerable then L is recursive.

Rice Theorem: All non-trivial properties of recursively enumerable languages are undecidable, where trivial properties are:

- "Is $L(M)$ recursively enumerable?"
- "Is $L(M)$ not recursively enumerable?"

In other words, if C is a set of recursively enumerable languages and $C \neq \varnothing$ and $\overline{C} \neq \varnothing$, then the following problem is undecidable

"Is $L(M) \in C$?"

Ex. The problems "Is $L(M) = \varnothing$?" and "Is $L(M) = \Sigma^*$?" are undecidable.

3.3 Reduction (p 60)

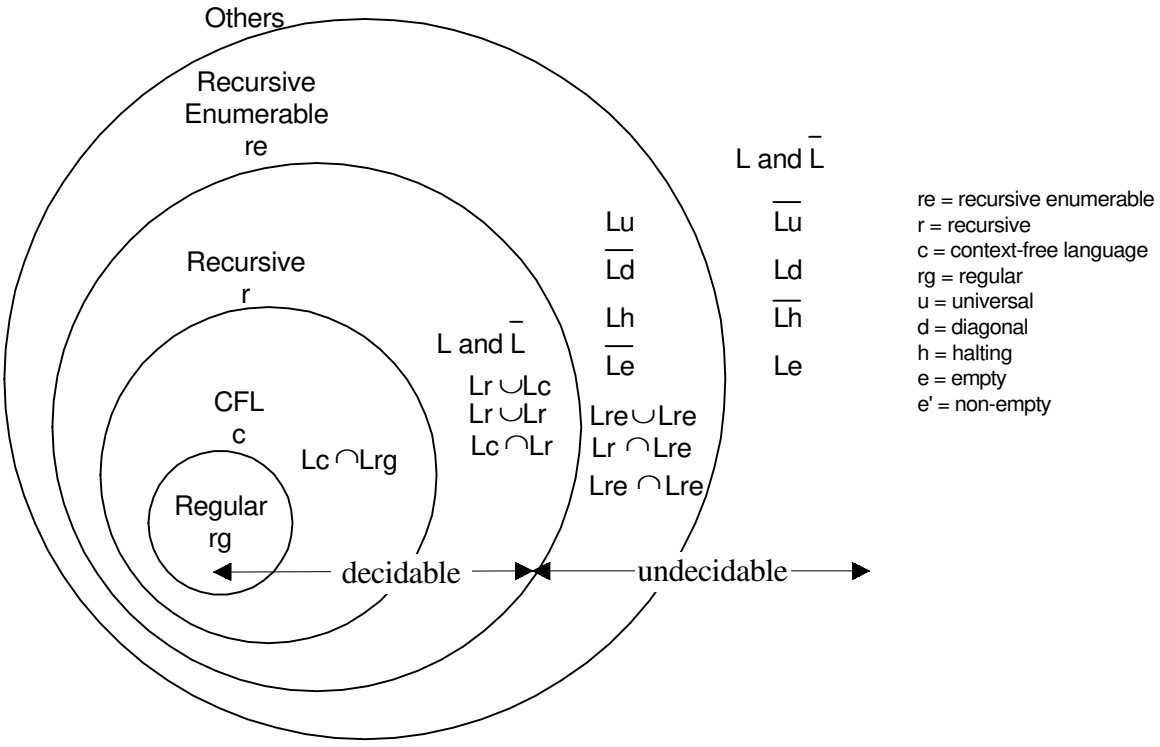
To show that problem A is undecidable we establish that, if there were an algorithm for problem A, then there would be an algorithm for HALTING, which is absurd. To do this, we reduce HALTING to A. That is, first, we assume that there is a machine M that solves HALTING. Then we build M' from M to solve problem A.

The process whereby we construct an algorithm for HALTING, using a supposed algorithm for problem A, is called a **reduction** of HALTING to A. In general, when we reduce HALTING to problem A we are showing that A is at least as hard as HALTING. The following statements are equivalent:

HALTING $\leq A$
HALTING reduces to A
HALTING is easier than A
HALTING can be simulated by A
HALTING is an instance of A
HALTING is a subset of A
If HALTING is undecidable then so is A

1. Recursive languages constitute a proper subset of recursively enumerables ones.
2. If L is recursive, then so is $\overline{L}$. (That is, the class of recursive languages is closed under complement.)

3. $\overline{H}$ is not recursively enumerable. (That is, the class of recursively enumerable languages is not closed under complement.)
4. L is recursive if and only if both L and $\overline{L}$ are recursively enumerable.
5. $L_1 \cup L_2$ is always in the outer class. e.g. $L_{cfl} \cup L_r$ is recursive.
6. $L_1 \cap L_2$ is always in the outer class except if both L_1 & $L_2 \in$ cfl. e.g. $L_{rg} \cap L_{re}$ is re.
7. Σ^* (everything) is regular.
8. $\varnothing$ (empty set) is regular.



7.1 Complexity Classes (p 139)

TIME(f) = DTIME(f)	problems (languages) that can be solved in time $f(n)$ deterministically.
SPACE(f) = DSPACE(f)	problems (languages) that can be solved in space $f(n)$ deterministically.
NTIME(f)	problems that can be solved in time $f(n)$ nondeterministically.
NSPACE(f)	problems that can be solved in space $f(n)$ nondeterministically.

1. Assumption is that f is proper, i.e. 1) f is nondecreasing, and 2) f can be computed using not more than $f(n)$ time and space.
2. Most of "reasonable" complexity functions are proper. e.g. $\log n, n^2, \sqrt{n}, n \log n, 2^n ...$

L = LOG = DLOG = SPACE(log n)	logspace
NL = NLOG = NSPACE(log n)	nondeterministic logspace
P = PTIME = $\bigcup_i$ TIME(n^i)	polynomial time
NP = NPTIME = $\bigcup_i$ NTIME(n^i)	nondeterministic polynomial time

$\text{PSPACE} = \bigcup_i \text{SPACE}(n^i)$ polynomial space

$\text{NPSPACE} = \bigcup_i \text{NSPACE}(n^i)$ nondeterministic polynomial space

$\text{EXP} = \text{TIME}(2^{n^k})$ exponential

Closure Properties:

1. If $L_1, L_2 \in \text{TIME}(f)$ then $L_1 \cup L_2 \in \text{TIME}(f)$ and $L_1 \cap L_2 \in \text{TIME}(f)$.

2. If $L_1, L_2 \in \text{NTIME}(f)$ then $L_1 \cup L_2 \in \text{NTIME}(f)$ and $L_1 \cap L_2 \in \text{NTIME}(f)$.

3. If $L_1, L_2 \in \text{SPACE}(f)$ then $L_1 \cup L_2 \in \text{SPACE}(f)$ and $L_1 \cap L_2 \in \text{SPACE}(f)$.

4. If $L_1, L_2 \in \text{NSPACE}(f)$ then $L_1 \cup L_2 \in \text{NSPACE}(f)$ and $L_1 \cap L_2 \in \text{NSPACE}(f)$.

Theorem:

1. If $L \in \text{TIME}(f)$ then $\overline{L} \in \text{TIME}(f)$.
   If $L \in \text{SPACE}(f)$ then $\overline{L} \in \text{SPACE}(f)$.

2. If $L \in \text{NSPACE}(f)$ and $f \geq \log n$ and proper then $\overline{L} \in \text{NSPACE}(f)$

3. open problem for NTIME...???

Relations between different complexity classes

1. $\text{TIME}(f) \subseteq \text{SPACE}(f)$, $\text{NTIME}(f) \subseteq \text{NSPACE}(f)$.

2. if $f \leq g$ then $\text{TIME}(f) \leq \text{TIME}(g)$, $\text{NTIME}(f) \leq \text{NTIME}(g)$,
   $\text{SPACE}(f) \leq \text{SPACE}(g)$, $\text{NSPACE}(f) \leq \text{NSPACE}(g)$.

3. $\text{TIME}(f) \subseteq \text{NTIME}(f)$, and $\text{SPACE}(f) \subseteq \text{NSPACE}(f)$.
   recall, however, that $\text{NTIME}(f) \subseteq \bigcup_c \text{TIME}(c^f)$, and $\text{NSPACE}(f) \subseteq \text{SPACE}(f^2)$.

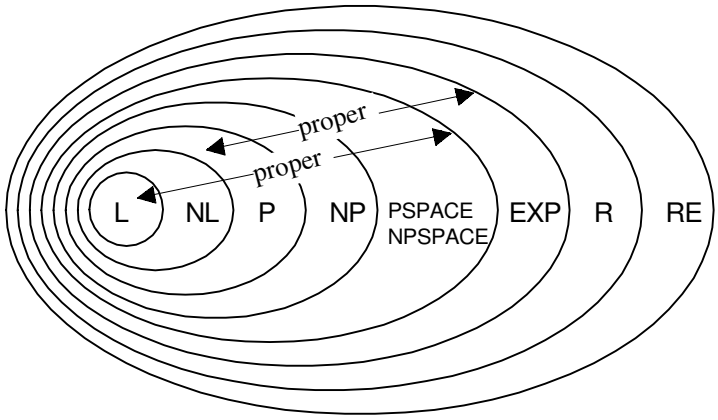
4. therefore, $\text{NTIME}(f) \subseteq \text{NSPACE}(f) \subseteq \bigcup_c \text{TIME}(c^f)$

Theorem:

$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXP}$ and

1. at least one of inclusions $\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE}$ is proper.

2. at least one of inclusions $\text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXP}$ is proper.


proper
proper
L
NL
P
NP
PSPACE
NPSPACE
EXP
R
RE