Languages and Computation (COMP2049/AE2LAC)

Enumerability, Decidability, and the Rest

Dr. Tianxiang Cui

tianxiang.cui@nottingham.edu.cn

The Language of A TM

- Recall, saying that a language L is accepted by a Turing machine M means that for any $x \in \sum^*$, $x \in L$ if and only if x is accepted by M
- This **does not** imply that if $x \notin L$, then x is rejected by M
- Two possibilities for a string x not in $L(M)$:

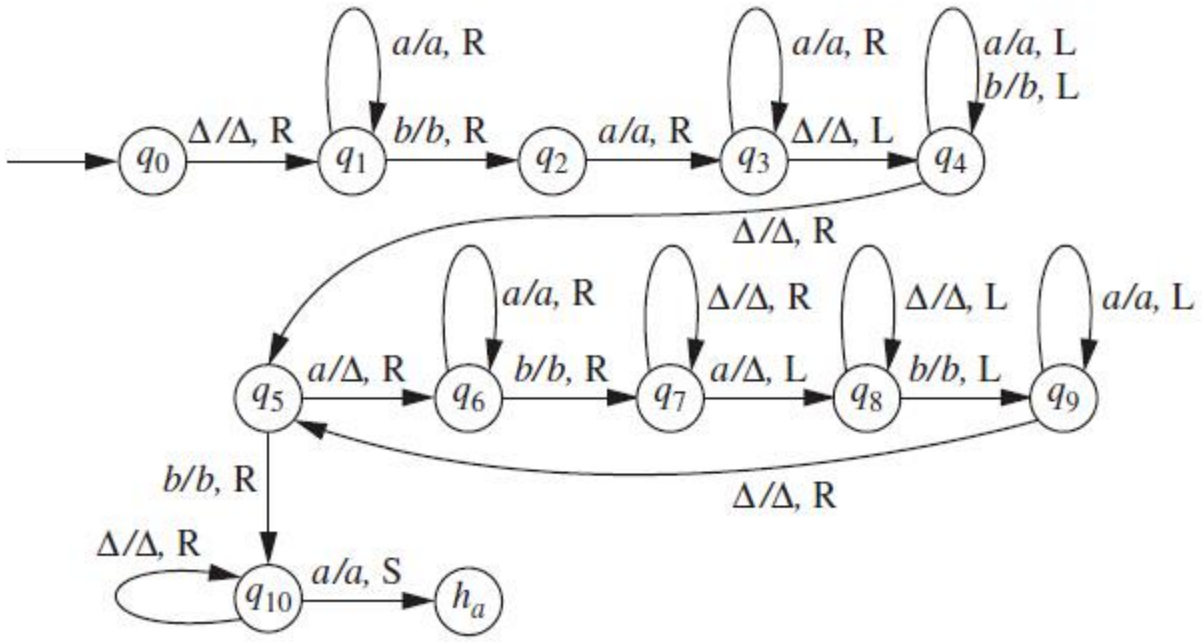
1. M rejects x
2. M never halts, or loops forever, on input x

- A TM **may never** stop
  - This is unlike the machines we have encountered before

The Language of A TM

- This TM accepting

$$\{a^i b a^j \mid 0 \le i < j\}$$



The Language of A TM

- If we try the input string *aba* with the previous TM, TM will be in an infinite loop

$$\begin{array}{lllllllll}
q_0\Delta aba & \vdash & \Delta q_1 aba & \vdash & \Delta a q_1 ba & \vdash & \Delta ab q_2 a & \vdash & \Delta aba q_3 \Delta \\
 & \vdash & \Delta ab q_4 a & \vdash^* & q_4 \Delta aba & \vdash & \Delta q_5 aba & \vdash & \Delta\Delta q_6 ba \\
 & \vdash & \Delta\Delta b q_7 a & \vdash & \Delta\Delta q_8 b & \vdash & \Delta q_9 \Delta b & \vdash & \Delta\Delta q_5 b \\
 & \vdash & \Delta\Delta b q_{10} \Delta & \vdash & \Delta\Delta b \Delta q_{10} \Delta & \vdash & \Delta\Delta b \Delta\Delta q_{10} \Delta & \vdash & \ldots
\end{array}$$

- But, the TM still works. Since it **does not** accept any string that **is not** in the language
- However, if we have some TM *M'* that will **always reject** a string *x* **not in** *L*(*M'*), it may tell us something more about the language *L*(*M'*)

Recursively Enumerable and Recursive

- **Recursively enumerable** languages are those that can be **accepted** by a TM
- **Recursive languages** are those that can be **decided** by a TM
- A TM M with input alphabet Σ accepts a language $L \subseteq \Sigma^*$ if it **accepts** the strings in L and no others
- M **decides** L if M computes the characteristic function $\chi_L : \Sigma^* \to \{0,1\}$ that has the value 1 at strings in L and the value 0 otherwise

Recursively Enumerable and Recursive

- Recursively enumerable languages are sometimes referred to as **Turing-acceptable**
  - The TM will only ever tell you that it accepts that a string is a member of the language
- Recursive languages are sometimes referred to as **Turing-decidable**
  - The TM will decide whether or not the string is a member of the language - if it isn't, it will tell you
- In both cases, the issue is whether the input string is an element of L, recursive languages may be more **informative**, as for recursively enumerable languages, it may not return an answer if the string is not in the language L

Recursively Enumerable and Recursive

- **Theorem**: Every recursive language is recursively enumerable
  - If L is recursive there is a TM M that decides L (returns 1 for strings in L, and 0 for strings not in L)
  - Given an input string x, simply execute M on input x, M will halt and produce an output
  - If output is 1, accept, otherwise reject
- Recursive languages are a **subset** of recursively enumerable languages

Undecidable Languages

- There exist languages that can be **accepted** by TMs but not **decided**
- These are the same languages that can be accepted by TMs but whose complements cannot
- They are languages that can be accepted, but only by TMs that may loop forever on some inputs that are not in the language
- These are **non-recursive languages**, and their existence implies that there are also languages that are **not recursively enumerable**

Unrestricted Grammars

- Unrestricted grammars are more general than CFG
- They correspond to **recursively enumerable languages**, in the same way that:
  - CFGs correspond to languages accepted by PDAs
  - Regular grammars correspond to languages accepted by FA
- Recall, for CFG, the productions are of the form:

$$nonterminal \rightarrow terminals\ and\ nonterminals$$

- Because they are context free, we can use any production rule regardless of context

Unrestricted Grammars

- **Definition**
- An unrestricted grammar is a 4-tuple $G=(N, T, S, P)$, where
  - N is a finite set of nonterminals
    - Or variables, and consequently sometimes it is denoted by V
  - T is a finite set of terminals
    - The terminals are the alphabet of the language defined by a context-free grammar, and for that reason the set of terminals is sometimes denoted by Σ
  - $N \cap T = \varnothing$ (N and T are disjoint finite sets)
  - $S \in N$ is the start symbol
  - P is a finite set of productions (or grammar rules) of the form $\alpha \rightarrow \beta$, where $\alpha, \beta \in (N \cup T)^*$ and **α contains at least one variable**

Unrestricted Grammars

- For context free grammars, production rules have to **be a variable** (non-terminal) on the left hand side
- For unrestricted grammars, production rules must **have at least one variable** on the left hand side
  - So we can have multiple variables
  - And we can have terminals
- This is much **less restrictive** than CFG rules

Unrestricted Grammars: Example

Not context free

A Grammar Generating $\{a^{2^k} \mid k \in \mathcal{N}\}$

$$S \to LaR \qquad L \to LD \qquad Da \to aaD \qquad DR \to R \qquad L \to \Lambda \qquad R \to \Lambda$$

- So if we have the input string *aaaa*, the derivation is

$$\begin{aligned} S \;\Rightarrow\; LaR \;\Rightarrow\; LDaR \;\Rightarrow\; LaaDR \;\Rightarrow\; LaaR \;\Rightarrow\; LDaaR \\ \Rightarrow\; LaaDaR \;\Rightarrow\; LaaaaDR \;\Rightarrow\; LaaaaR \;\Rightarrow\; aaaaR \;\Rightarrow\; aaaa \end{aligned}$$

- Use variable D to act as a "doubling operator". D replaces each a with two a's using production $Da \to aaD$. D is introduced on the left of the string and each application moves past an a, doubling it each time

Unrestricted Grammars And TMs

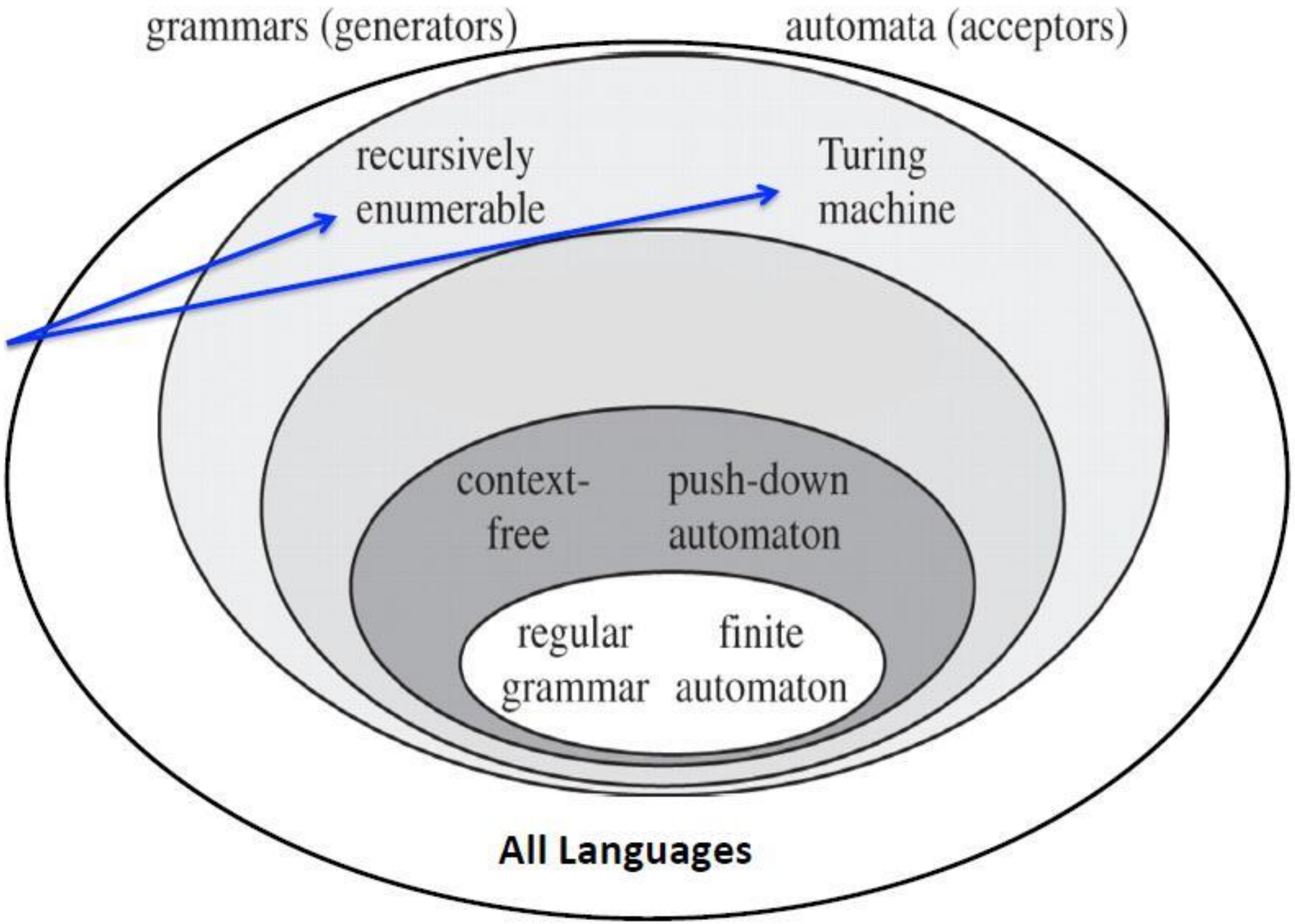
- Every **recursively enumerable** language can be generated by an **unrestricted grammar**

- **Theorem**: For every unrestricted grammar G, there is a Turing machine M with $L(M) = L(G)$
  - For every unrestricted grammar, G, there is a Turing machine that accepts it

- **Theorem**: For every TM M with input alphabet Σ, there is an unrestricted grammar generating the language $L(M) \subseteq \Sigma^*$
  - For every Turing Machine, M, there is an unstructured grammar G that generates the language of M

Chomsky Hierarchy

Recursively enumerable languages are generated by unrestricted grammars and accepted by Turing Machines



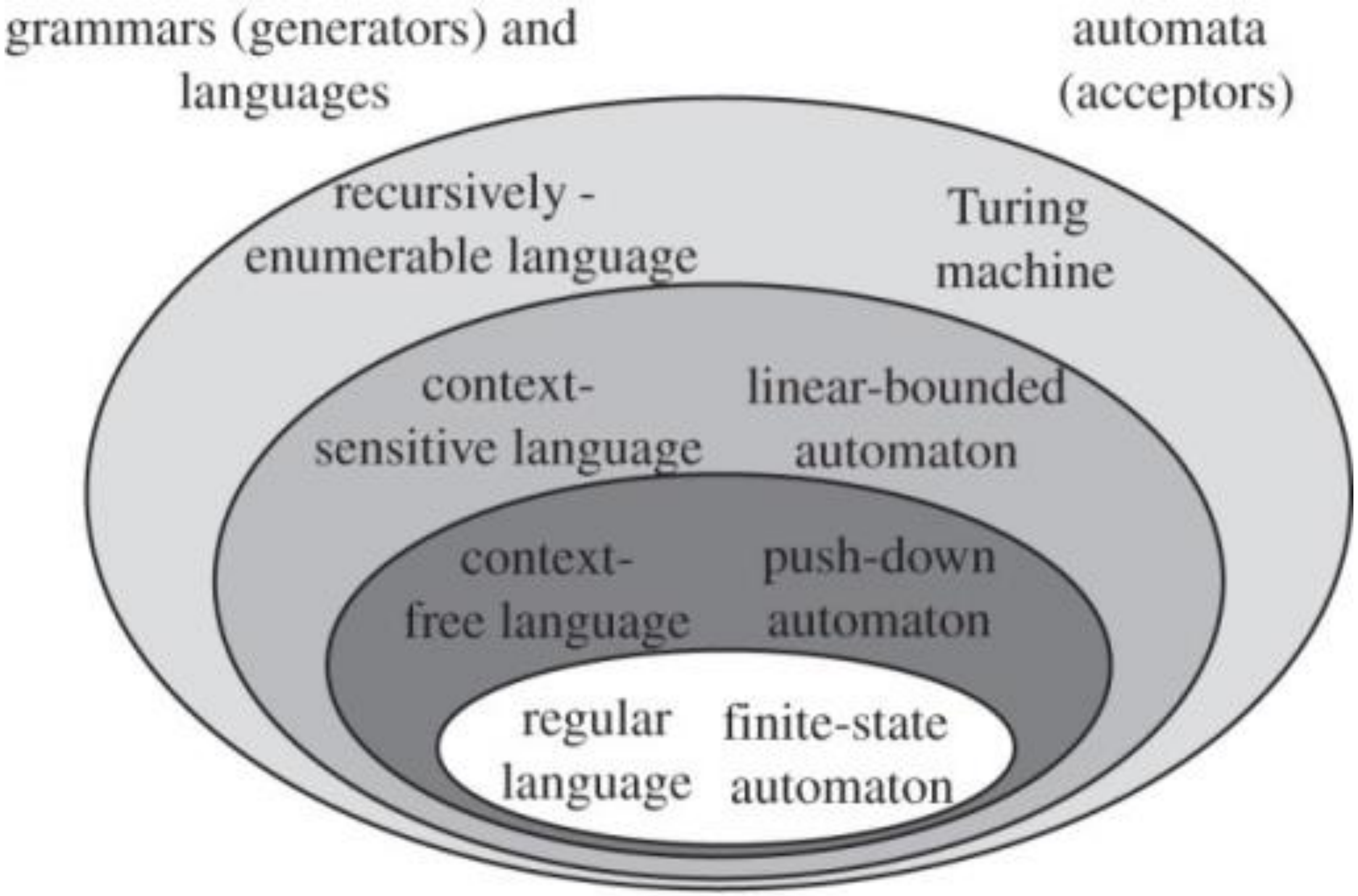
Context-Sensitive Grammar (CSG)

- **Definition**: A **context-sensitive grammar** (**CSG**) is an unrestricted grammar in which no production is length-decreasing
  - In other words, every production is of the form $\alpha \rightarrow \beta$, where $|\beta| \geq |\alpha|$
- CFG has an additional restriction over unrestricted grammar:
  - Length of RHS of production must be greater than or equal to the length of LHS
- A language is a **context-sensitive language** (**CSL**) if it can be generated by a CSG
- CSGs cannot have ε-productions, and CSLs cannot include ε
- We think of CSLs as a generalization of CFLs

Linear-Bounded Automata (LBA)

- **Definition**: A linear-bounded automaton (LBA) is a nondeterministic TM with this exception:
  - There are two extra tape symbols, [and]
  - The initial configuration of M corresponding to input x is $q_0[x]$
  - During its computation, M is not permitted to replace either of these brackets or to move its tape head to the left of the "[" or to the right of the "]"
- **Theorem**: If $L \subseteq \Sigma^*$ is a CSL, then there is an LBA that accepts L
- A LBA can simulate the computation of a TM, provided the proportion of the tape used by the TM is bounded by some linear function of input length
  - Since the tape is bounded, LBA will always halt

The Chomsky Hierarchy



the traditional Chomsky hierarchy

Chomsky Hierarchy

Type	Languages (Grammars)	Form of Productions	Accepting Device
3	Regular	$A \to aB, \quad A \to \Lambda$	Finite Automaton
2	Context-free	$A \to \alpha$	Pushdown Automaton
1	Context-sensitive	$\alpha \to \beta$ with $\lvert\beta\rvert \geq \lvert\alpha\rvert$	LBA
0	Unrestricted	$\alpha \to \beta$	Turing machine

Concluding Remarks

- Recursively enumerable languages and Recursive languages
- Unrestricted grammar
- Context-sensitive grammar
- Linear-bounded automata