1.1 Three areas of ToC (Sipser 0.1)
Theory of Computation is to study the fundamental capabilities and limitations of computers. It contains three areas.

- Automata theory: Models of computation. Seeking a precise but concise definition of a computer.
- Computability theory: What can and cannot a computer do? Computationally unsolvable versus computationally solvable problems. Determining whether a mathematical statement is true or false is unsolvable by any computer algorithms.
- Complexity theory: What can a computer do efficiently? Computationally hard versus computationally easy problems. For example, factorization and sorting. Cryptography needs hard problems such as factorization to ensure security.
1.2 Basic math *(Sipser 0.2)*

Sets, sequences, functions, relations, graphs, and logic. Here is a discussion on strings and languages, which are central concepts in automata theory.

- **Alphabet** $\Sigma$: Finite and nonempty, e.g., $\Sigma = \{0, 1\}$ and $\Sigma = \{a, b, \ldots, z\}$.
- **String** (or word), e.g., $w = 01110$, the empty string $\varepsilon$, the length of the string, $|w|$, the concatenation of two strings $w_1 w_2$, the reverse of a string $w^R$, and the substring of a string.
- **Language**: A language is a set of strings, e.g., $\{\varepsilon\}$, $\emptyset$, $\Sigma$, $A = \{w|w \text{ has an equal number of 0s and 1s}\}$, and $B = \{0^n1^n|n \geq 1\}$.
Regular operators: Let $A$ and $B$ be two languages.

- **Union:** $A \cup B = \{ x | x \in A \text{ or } x \in B \}$.
- **Concatenation:** $AB = \{ xy | x \in A \text{ and } y \in B \}$.
- **Star:** $A^* = \{ x_1 x_2 \cdots x_k | \text{ all } k \geq 0 \text{ and each } x_i \in A \}$. $A^*$ is infinite unless $A = \emptyset$ or $A = \{ \varepsilon \}$.

Power of a language: Let $A$ be a language.

- $A^k = \{ x_1 x_2 \cdots x_k | x_i \in A \text{ for } i = 1, 2, \cdots, k \}$
- $A^0 = \{ \varepsilon \}$
- $A^* = A^0 \cup A^1 \cup A^2 \cup \cdots$
- $A^+ = A^1 \cup A^2 \cup \cdots$
- $A^* = A^+ \cup \{ \varepsilon \}$

Why languages, not problems:

- Decision problems: Given an input $x$, does $x$ satisfy property $P$, or is $x \in \{ y | y \text{ satisfies } P \}$?
- Membership in a language: Given a language $A$ and a string $w \in \Sigma^*$, is $w$ a member of $A$, or is $w \in A$?
1.3 Common forms of theorems *(Sipser 0.3)*

- If $H$ (hypothesis) then $C$ (conclusion). Also equivalently, $H$ implies $C$, $H$ only if $C$, $C$ if $H$, whenever $H$ holds, $C$ follows, or $H \rightarrow C$.
- $A$ if and only if $B$. Also equivalently, $A$ iff $B$, $A$ is equivalent to $B$, $A$ exactly when $B$, $A \equiv B$, or $A \leftrightarrow B$.
- $S$ (a statement without any hypothesis).
- For all $x$, $S(x)$ holds. Also equivalently, $\forall x S(x)$.
- There is $x$ such that $S(x)$ holds. Also equivalently, $\exists x S(x)$. 
1.4 Proof techniques (*Sipser 0.4*)

- By definition: Convert terms in the hypothesis to their definitions.
- By construction: To prove the existence of $X$, construct it.
- By counterexample: Used to disprove something seemingly true but actually not true.
  **Example:** All primes are odd.
  **Example:** There is no integers $a$ and $b$ such that $a \mod b = b \mod a$.
- By contradiction: $H \rightarrow C$ is equivalent to $\neg C \rightarrow \neg H$ or $\neg C \land H \rightarrow \neg T$, where $T$ is an axiom, a known truth, or a proven fact.
  **Example:** The number of primes is infinite.
  **Example:** $\sqrt{2}$ is irrational.
By induction: Used to prove a statement \( S(n) \) about an integer \( n \geq c \). There are several forms of inductions:

- Simple integer induction: \( S(n) \) for \( n \geq c \) iff \( S(c) \land \forall k (S(k) \rightarrow S(k + 1)) \). (What are the basis step, inductive hypothesis, and inductive step?)
  
  **Example**: For \( n \geq 1 \), \( \sum_{i=1}^{n} i^2 = \frac{1}{6} n(n+1)(2n+1) \).

- General (strong) integer induction: \( S(n) \) for \( n \geq c \) iff \( S(c) \land \forall k (S(c) \land S(c + 1) \land \cdots \land S(k) \rightarrow S(k + 1)) \).
  
  **Example**: For any \( n \geq 8 \), \( n \) can be written as a sum of 3s and 5s.

- Structural induction: To prove a property \( S(X) \) of a recursively defined structure \( X \), it is suffice to prove that \( S(X) \) is true for the basis structures \( X \) and \( S(Y_1) \land S(Y_2) \land \cdots \land S(Y_k) \rightarrow S(X) \), where \( X \) is constructed from \( Y_1, Y_2, \ldots, Y_k \).
  
  **Example**: Every tree has one more node than it has edges.
2.1 Example of FA (Sipser 1.1 (pp. 31-34))

- A farmer (F) with a cabbage (C), a dog (D), and a goat (G) wants to cross a river. There is a small boat which can only carry the farmer plus one of the three things. At any time, the cabbage and the goat cannot be left alone neither can the dog and the goat. How can the farmer cross the river? A finite automaton with start state FCDG-∅ and accept state ∅-FCDG can solve the puzzle.
2.2 DFA (Sipser 1.1 (pp. 35-44))

- DFA \( M = (Q, \Sigma, \delta, q_0, F) \), where
  - \( Q \) is a finite set of states,
  - \( \Sigma \) is an alphabet,
  - \( q_0 \in Q \) is the start state,
  - \( F \subseteq Q \) is a set of accept/final states, and
  - \( \delta : Q \times \Sigma \rightarrow Q \) is a transition function, where \( \delta(q, a) = p \) is the next state of \( M \) if the current state is \( q \) and the current symbol is \( a \).
Extending $\delta$ to $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$: For any $q \in Q$ and any $w = xa \in \Sigma^*$, define $\hat{\delta}(q, x)$ recursively as below:

- $\hat{\delta}(q, \varepsilon) = q$ and
- $\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$

Language of a DFA $M$ (or language recognized by $M$) is $L(M) = \{w | \hat{\delta}(q_0, w) \in F\}$. A language is called a regular language if some DFA recognizes it.

How does a DFA accept/reject a string?
Example: A DFA that accepts strings over alphabet \{0, 1\} that have even numbers of 0s and 1s. (Given a language \(A\), what is the DFA \(M\) such that \(L(M) = A\)?)
Example (Sipser p. 36): A DFA that accepts strings over the alphabet \{0, 1\} that have at least one 1 and an even number of 0s after the last 1. (Given a DFA \( M \), what is \( L(M) \)?)
Remarks:

- State diagrams and transition tables are other representations of DFAs.
- Although rigorously, $\delta(q, a)$ should be defined $\forall q \in Q$ and $\forall a \in \Sigma$, all inaccessible and dead-end states with associated arcs may be removed to simplify the DFA.
2.3 NFA (Sipser 1.2 (pp. 47-54))

- NFA $N = (Q, \Sigma, \delta, q_0, F)$, where
  - $Q$ is a finite set of states,
  - $\Sigma$ is an alphabet,
  - $q_0 \in Q$ is the start state,
  - $F \subseteq Q$ is a set of accept/final states, and
  - $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ is a transition function, where
    $\delta(q, a) = P$ is the set of states that $N$ may enter if the current state is $q$ and the current symbol is $a$. In the case of $\delta(q, \varepsilon) = P$, $N$ ignores the current input symbol and transitions from the current state $q$ to any state in $P$. 
\(\varepsilon\)-closure: For any \(P \subseteq Q\), \(E(P)\) is the set of all states reachable from any state in \(P\) via zero or more \(\varepsilon\)-transitions.

Extending \(\delta\) to \(\hat{\delta}: Q \times \Sigma^* \rightarrow 2^Q\): For any \(q \in Q\) and any \(w = xa \in \Sigma^*\), define

- \(\hat{\delta}(q, \varepsilon) = E(\{q\})\) and
- \(\hat{\delta}(q, w) = E(\bigcup_{i=1}^{k} \delta(p_i, a))\) if \(w = xa\) and \(\hat{\delta}(q, x) = \{p_1, \ldots, p_k\}\).

Language of an NFA \(N\) (or language recognized by \(N\)) is

\[L(N) = \{w | \hat{\delta}(q_0, w) \cap F \neq \emptyset\}.\]

How does an NFA accept/reject a string? The meaning of nondeterminism.
**Example:** An NFA that accepts decimal numbers (a number that may have + or − preceding it, but must have a decimal point).
2.4 DFAs ⇔ NFAs (*Sipser 1.2 (pp.54-58)*)
Subset construction method: Given NFA $N = (Q, \Sigma, \delta, q_0, F)$, construct a DFA $M = (Q', \Sigma, \delta', q'_0, F')$ such that $L(M) = L(N)$.

- $Q' = 2^Q$ (power set), i.e., $Q'$ contains all subsets of $Q$. Note that if $|Q| = n$ then $|Q'| = 2^n$. This is just the worst case. Since many states in $M$ are inaccessible or dead-end states and thus may be thrown away, so in practice, $|Q'|$ may be much less than $2^n$.
- $q'_0 = E(\{q_0\})$.
- $F' = \{R \in Q' | R \cap F \neq \emptyset\}$.
- For each $R \in Q'$ and each $a \in \Sigma$, $\delta'(R, a) = E(\bigcup_{p \in R} \delta(p, a))$.

**Theorem:** The equivalence of DFAs, NFAs, and RLs.
**Example:** A DFA that accepts decimal numbers (converted from the previous NFA).

**Example** (Sipser p. 51): A four-state NFA and an eight-state DFA that recognize the language consisting of strings over \{0, 1\} with a 1 in the third position from the end.

**Example:** A bad case for the subset construction: \(|Q_N| = n + 1\) and \(|Q_D| = 2^n\).
2.5 Closure Properties of RL’s (Sisper 1.1 (pp. 44-47) and 1.2 (pp. 58-63))

- Union: If A and B are regular, so is \( A \cup B \).
- Concatenation: If A and B are regular, so is \( AB \).
- Star: If A is regular, so is \( A^* \). (Need a new start state.)
- Complementation: If A is regular, so is \( \overline{A} \) (which is \( \Sigma^* - A \)).
- Intersection: If A and B are regular, so is \( A \cap B \).
- Difference: If A and B are regular, so is \( A - B \).
- Reverse: If A is regular, so is \( A^R \).
- Homomorphism: If A is regular, so is \( h(A) \) (which is \( \{ h(w) | w \in A \} \) for a homomorphism \( h : \Sigma \rightarrow (\Sigma’)^* \)).
- Inverse homomorphism: If A is regular, so is \( h^{-1}(A) \) (where \( h^{-1}(A) = \{ w | h(w) \in A \} \)).
Example: Prove that \( A = \{ w \in \{ a, b \}^* | w \) is of odd length and contains an even number of \( a \)'s \} is regular. (\( A \) is the intersection of two regular languages.)
3.1 Definition of REs (Sipser 1.3 (pp. 63-66))

Regular expressions (REs) are to represent regular languages. Let $L(R)$ be the language that regular expression $R$ represents. A recursive definition is given below:

- **Basis**: $\varepsilon$ and $\emptyset$ are REs, and $L(\varepsilon) = \{\varepsilon\}$ and $L(\emptyset) = \emptyset$. For any $a \in \Sigma$, $a$ is an RE and $L(a) = \{a\}$.
- **Induction**: If $R_1$ and $R_2$ are REs, then
  - $R_1 \cup R_2$ is an RE, with $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$,
  - $R_1 R_2$ is an RE, with $L(R_1 R_2) = L(R_1)L(R_2)$,
  - $R_1^*$ is an RE, with $L(R_1^*) = (L(R_1))^*$, and
  - $(R_1)$ is an RE, with $L((R_1)) = L(R_1)$. 
Remark:

- Precedence order for regular-expression operators: Star, concatenation, and finally union. () may override this order.
- Use of $R^+$ and $R^k$.
- Algebraic laws:
  - $R_1 \cup R_2 = R_2 \cup R_1$, $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$, and $(R_1 R_2) R_3 = R_1 (R_2 R_3)$.
  - $\emptyset \cup R = R \cup \emptyset = R$, $\varepsilon R = R \varepsilon = R$, $\emptyset R = R \emptyset = \emptyset$, and $R \cup R = R$.
  - $R_1 (R_2 \cup R_3) = R_1 R_2 \cup R_1 R_3$ and $(R_1 \cup R_2) R_3 = R_1 R_3 \cup R_2 R_3$.
  - $(R^*)^* = R^*$, $\emptyset^* = \varepsilon$, $R^+ = RR^* = R^* R$, and $R^* = R^+ \cup \varepsilon$. 
Examples:
1. A regular expression for the language of strings that consist of alternating 0s and 1s: \((01)^* \cup (10)^* \cup 0(10)^* \cup 1(01)^*\).
2. A regular expression for the language of strings with a 1 in the third position from the end: \((0 + 1)^* 1(0 + 1)(0 + 1)\).
3.2 RE $\Rightarrow$ NFA (*Sipser 1.3 (pp. 67-69)*)
Since REs are defined recursively, it is suitable to construct the equivalent NFAs recursively.

- **Basis**: The finite automata for regular expressions $\varepsilon$, $\emptyset$, and $a$ for $a \in \Sigma$.

- **Induction**: Given the NFAs for REs $R_1$ and $R_2$, what are the NFAs for $R_1 \cup R_2$, $R_1 R_2$, and $R_1^*$?

**Example**: Convert regular expression $(0 \cup 1)^*1(0 \cup 1)$ to a finite automaton.
3.3 DFA ⇒ RE (Sipser 1.3 (pp. 69-76))
A generalized NFA (GNFA) is an NFA with REs (not symbols) on its transition arcs.
Assume that the given finite automaton is a DFA $M = (Q, \Sigma, \delta, q_0, F)$. We first convert the DFA to a GNFA by

1. adding a new start state $s$ that goes to the old start state $q_0$ via an $\varepsilon$-transition,

2. adding a new accept state $a$ to which there is an $\varepsilon$-transition from each old accept state in $F$, and

3. converting symbols to regular expressions on all arcs.

Then this GNFA with $|Q| + 2$ states will be converted to an equivalent GNFA with $|Q| + 1$ states by eliminating a state that is neither $s$ nor $a$. This state elimination step will be applied a total of $|Q|$ times until there are only states $s$ and $a$ left in the resulting GNFA. The regular expression on the arc from $s$ to $a$ is the regular expression for the original DFA.
Given a GNFA with $k$ states, how can one convert it to an equivalent GNFA with $k - 1$ states by eliminating a state that neither $s$ nor $a$? (Sipser Figure 1.63 on p.72)
Example (Siper p. 76): A three-state DFA to be converted to a regular expression.
Example: A language of all strings over \{0, 1\} with one 1 either two or three positions from the end.
Theorem: The equivalence of RLs, REs, and FAs.
4.1 Regular versus nonregular languages

- $A = \{0^*1^*\}$
- $B = \{0^n1^n | n \geq 0\}$
- $C = \{w \in \{0, 1\}^* | w \text{ has an equal number of 0s and 1s}\}$
- $D = \{w \in \{0, 1\}^* | w \text{ has an equal \# of substrings 01 and 10}\}$
4.2 Proving nonregularity by pumping lemma (Sisper 1.4 (pp. 77-82))

**Theorem** (The pumping lemma for regular languages): Let $A$ be a regular language. Then there exists a constant $p$ (the pumping length which depends on $A$) such that $\forall s \in A$ with $|s| \geq p$, we can break $s$ into three substrings $s = xyz$ such that

1. $|y| > 0$;
2. $|xy| \leq p$; and
3. $\forall i \geq 0$, string $xy^i z \in A$. 

**Proof:** Let $A = L(M)$ for some DFA $M$ with $p$ states. Consider any $s \in A$ with $s = s_1s_2 \cdots s_n$, where $n \geq p$ and $s_i \in \Sigma$ for $i = 1, 2, \ldots, n$. Assume that $\delta(q_0, s_1 \cdots s_i) = q_i$. On the path $q_0 \rightarrow q_1 \rightarrow \cdots \rightarrow q_n$ that accepts $s$, there are $n + 1 \geq p + 1$ states. By the pigeonhole principle, there are at least two identical states on the path. Let $q_i = q_j$ for some $0 \leq i < j \leq n$ be the first such pair on the path. Now we can break $s = xyz$ as follows:

- $x = s_1 \cdots s_i$;
- $y = s_{i+1} \cdots s_j$; and
- $z = s_{j+1} \cdots s_n$.

We can then easily verify that this partition of $s$ satisfies all three requirements stated in the theorem, i.e., $|y| > 0$, $|xy| \leq p$, and $xy^iz \in A$ for any $i \geq 0$. This completes the proof.
How to use the pumping lemma to prove that a language is not regular:

- Assume that $A$ was regular by contradiction. Then the pumping lemma applies to $A$. Let $p$ be the constant in the pumping lemma.
- Select $s \in A$ with $|s| = f(p) \geq p$.
- By the pumping lemma, $\exists x, y, z$ such that $s = xyz$ with $|y| > 0$, $|xy| \leq p$ and $xy^i z \in A$ for any $i \geq 0$.
- For any $x, y, z$ such that $s = xyz$, $|y| > 0$, and $|xy| \leq p$, find $i \geq 0$ such that $xy^i z \not\in A$. A contradiction!
Example (Sipser p. 80): Prove that $B = \{0^n1^n | n \geq 0\}$ is not regular.

Example: Prove that $A = \{1^r | r \text{ is a prime}\}$ is not regular.

Example (Sipser p. 81): Prove that $A = \{ww | w \in \{0,1\}^*\}$ is not regular.

Example: Prove that $A = \{(01)^a0^b | a > b \geq 0\}$ is not regular.

Example: Prove that $A = \{0^m0^n | m \neq n\}$ is not regular.

Example: Prove that $L = \{1^{n^2} | n \geq 1\}$ is not regular.
4.3 Prove nonregularity by closure properties
To prove that $A$ is not regular, assume it was. Find a regular language $B$ and a language operator that preserves regularity, and then apply the operator on $A$ and $B$ to get a regular language $C$. If $C$ is known to be nonregular, a contradiction is found.

**Example:** Prove that
$C = \{ w \in \{0, 1\}^* | w \text{ has an equal } \# \text{ of 0s and 1s} \}$ is not regular.
5.1 Context-free grammars (Sipser 2.1 (pp. 100-105))

- **CFG** \( G = (V, \Sigma, R, S) \), where \( V \) is the set of variables, \( \Sigma \) is the set of terminals (alphabet), \( R \) is the set of rules in the form of \( V \rightarrow (V \cup \Sigma)^* \) (head→body), and \( S \in V \) is the start variable.

- The CFG that generates all palindromes (strings that read the same forward and backward) over \( \{0, 1\} \) is \( G = (\{S\}, \{0, 1\}, R, S) \), where \( R \) contains \( S \rightarrow \varepsilon | 0 | 1 | 0S0 | 1S1 \).

- Let \( u, v, w \) be strings in \( (V \cup \Sigma)^* \). If \( A \rightarrow w \) is a rule, then \( uAv \) yields \( uwv \), written \( uAv \Rightarrow uwv \). We say \( u \) derives \( v \), written \( u \Rightarrow^* v \), if \( \exists u_1, \ldots, u_k \in (V \cup \Sigma)^* \) such that \( u \Rightarrow u_1 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v \). Here, \( \Rightarrow \) means one step and \( \Rightarrow^* \) means zero or more steps.

- **Leftmost and rightmost derivations:** \( \Rightarrow_{lm} \), \( \Rightarrow_{rm} \), \( \Rightarrow^*_{lm} \), \( \Rightarrow^*_{rm} \).

- The language of a **CFG** \( G \), \( L(G) = \{ w \in \Sigma^* | S \Rightarrow^* w \} \). \( L(G) \) is said to be a CFL.

- Why called “context-free”?
Example: Is \( L = \{0^n1^n | n \geq 0\} \) a context-free language? Yes since it is the language of the context-free grammar \( S \to 0S1|\varepsilon \).

Example: What is the language for grammar \( G \) with \( S \to AA \) and \( A \to AAA|bA|Ab|a \)? \( L(G) = \{w \in \{a, b\}^* | w \) has an even (nonzero) number of as\}.

Example: A CFG for simple expressions in programming languages: \( E \to E + E | E * E | (E) | I \) and \( I \to Ia|Ib|I0|I1|a|b \).
5.2 Parse trees

- A parse tree is a tree representation for a derivation, in which each interior node is a variable, each leaf node is either a variable, a terminal, or $\varepsilon$, and if an interior node is a variable $A$ and its children are $X_1, \ldots, X_k$, then there must be a rule $A \rightarrow X_1 \cdots X_k$.

- Yield of a parse tree: Concatenation of the leaf nodes in a parse tree rooted at the start variable.

- Four equivalent notions:
  1. $A \Rightarrow^* w$;
  2. $A \Rightarrow_{lm}^* w$;
  3. $A \Rightarrow_{rm}^* w$; and
  4. A parse tree with root $A$ and yield $w$. 
5.3 Regular grammars

- A regular grammar (RG) is a special CFG where the body of each rule contains at most one variable which, if present, must be the last symbol in the body.

- For each regular language $L$ there is a RG $G$ such that $L = L(G)$.

- Given RG $G = (V, \Sigma, R, S)$, construct an equivalent NFA $N = (Q, \Sigma, \delta, q_0, F)$, where $Q = V \cup \{f\}$, $q_0 = S$, $F = \{f\}$, and for each rule $A \rightarrow wB$, $\delta(A, w) = B$ and for each rule $A \rightarrow w$, $\delta(A, w) = f$.

- Given DFA $A = (Q, \Sigma, \delta, q_0, F)$, define an equivalent RG $G = (V, \Sigma, R, S)$, where $V = Q$, $S = q_0$, and for each transition $\delta(q, a) = p$, $q \rightarrow ap$ and for each final state $q \in F$, $q \rightarrow \varepsilon$.

**Example**: Given RG: $S \rightarrow aS|bA|aB$, $A \rightarrow aS|\varepsilon$, and $B \rightarrow bS|\varepsilon$, what is its NFA?

**Example**: Given a four-state DFA for language $aa^*bb^*a$, what is its regular grammar?
5.4 Ambiguity in grammars and languages \(\textit{(Sipser 2.1 (pp. 105-106))}\)

- A CFG \(G = (V, \Sigma, R, S)\) is ambiguous if there is \(w \in \Sigma^*\) for which there are at least two parse trees (or leftmost derivations).

- Grammar \(G\): \(E \rightarrow E + E | E \cdot E | (E) | I\) and \(I \rightarrow Ia | Ib | I0 | I1 | a | b\) is ambiguous since \(a + b \cdot a\) has two parse trees.

- Some ambiguous grammars have an equivalent unambiguous grammar. For example, an unambiguous grammar for the simple expressions is \(G'\): \(E \rightarrow E + T | T\), \(T \rightarrow T \cdot F | F\), \(F \rightarrow (E) | I\), and \(I \rightarrow Ia | Ib | I0 | I1 | a | b\).
A context-free language is said to be inherently ambiguous if all its grammars are ambiguous. For example, \( \{a^n b^n c^m d^m \mid m, n \geq 1 \} \cup \{a^n b^m c^m d^n \mid m, n \geq 1 \} \). Its grammar is \( S \rightarrow S_1 \mid S_2 \), \( S_1 \rightarrow AB \), \( A \rightarrow aAb \mid ab \), \( B \rightarrow cBd \mid cd \), \( S_2 \rightarrow aS_2d \mid aCd \), and \( C \rightarrow bCc \mid bc \). The grammar is ambiguous (considering \( abcd \)). There is a not so easy proof that all grammars for the language are ambiguous, thus it is inherently ambiguous.

There is no algorithm to determine whether a given CFG is ambiguous. There is no algorithm to remove ambiguity from an ambiguous CFG. There is no algorithm to determine whether a given CFL is inherently ambiguous.
5.5 Chomsky normal form (Sipser 2.1 (pp. 106-109))
For a CFL, different people may come up with different but equivalent CFGs. There are several steps to simplify a CFG.

▶ Eliminating $\varepsilon$ rules for $L(G) - \{\varepsilon\}$ by finding nullable variables ($A \Rightarrow^* \varepsilon$). For example, $S \rightarrow AB$, $A \rightarrow aAA|\varepsilon$, and $B \rightarrow bBB|\varepsilon$ can be changed to $S \rightarrow AB|A|B$, $A \rightarrow aAA|aA|a$, and $B \rightarrow bBB|bB|b$.

▶ Eliminating unit rules. For example, $E \rightarrow T|E + T$, $T \rightarrow F|T*F$, $F \rightarrow l|(E)$, and $l \rightarrow la|lb|l0|l1|a|b$ can be changed to $E \rightarrow E + T|T*F|(E)|la|lb|l0|l1|a|b$, $T \rightarrow T*F|(E)|la|lb|l0|l1|a|b$, $F \rightarrow (E)|la|lb|l0|l1|a|b$, and $l \rightarrow la|lb|l0|l1|a|b$.

▶ Eliminating useless variables (and thus associated rules) by finding nongenerating and unreachable variables. For example, $S \rightarrow AB|a$ and $A \rightarrow b$ can be simplified to $S \rightarrow a$. 
The Chomsky Normal Form (CNF): Any nonempty CFL without ε has a CFG $G$ in which all rules are in one of the following two forms: $A \rightarrow BC$ and $A \rightarrow a$, where $A, B, C$ are variables, and $a$ is a terminal. Note that one of the uses of CNF is to turn parse trees into binary trees.

To convert a CFG to a grammar in CNF:

- Add a new start variable $S_0$ in the case when the old start variable $S$ appears in the body of some rules.
- Simplify the grammar by removing ε rules, unit rules, and useless variables.
- Convert the rules in the simplified grammar into the proper forms of CNF by adding additional variables and rules.

**Example** (Sipser p. 108): Given a CFG $G$, construct a CNF $G'$ such that $L(G) - \{ε\} = L(G')$. 