Theorem

Automata Theory - Complete Theorems & Lemmas

Theorem 1: Closure of Regular Languages under Union

Statement: If L_1 and L_2 are regular languages, then $L_1 \cup L_2$ is also regular.

Proof Method: Construct a new DFA/NFA by combining the automata for L_1 and L_2 using product construction or by creating a new start state with ε-transitions.

Theorem 2: Closure of Regular Languages under Intersection

Statement: If L_1 and L_2 are regular languages, then $L_1 \cap L_2$ is also regular.

Proof Method: Use product construction of DFAs. The accepting states are pairs (q_1, q_2) where both q_1 and q_2 are accepting states in their respective automata.

Theorem 3: Closure of Regular Languages under Complement

Statement: If L is a regular language, then $\bar{L}$ (complement of L) is also regular.

Proof Method: Convert to DFA and swap accepting and non-accepting states.

Theorem 4: Closure of Regular Languages under Concatenation

Statement: If L_1 and L_2 are regular languages, then $L_1 \cdot L_2$ is also regular.

Proof Method: Use NFA construction with ε-transitions from accepting states of L_1 to the start state of L_2.

Theorem 5: Closure of Regular Languages under Kleene Star

Statement: If L is a regular language, then L^* is also regular.

Proof Method: Add a new start state with ε-transitions to the original start state and from all accepting states back to the start state.

Theorem 6: Myhill-Nerode Theorem

Statement: A language L is regular if and only if the equivalence relation $\equiv_L$ has finitely many equivalence classes.

Definition: $x \equiv_L y$ if for all strings z, $xz \in L \iff yz \in L$

Applications: Used for proving non-regularity and DFA minimization.

Theorem 7: Equivalence of DFA and NFA

Statement: For every NFA, there exists an equivalent DFA that accepts the same language.

Proof Method: Subset construction (powerset construction) where each state in the DFA corresponds to a subset of states in the NFA.

Theorem 8: DFA Minimization Theorem

Statement: Every regular language has a unique minimal DFA (up to isomorphism), and this minimal DFA can be constructed from any DFA accepting the language.

Construction: Use table-filling algorithm or partition refinement to identify and merge equivalent states.

Theorem 9: Kleene's Theorem (Regular Expression ⇔ Finite Automaton)

Statement: A language is regular if and only if it can be described by a regular expression.

Two Directions:

- **RE → NFA:** Thompson's construction
- **NFA → RE:** State elimination method or algebraic method

Lemma 10: Pumping Lemma for Regular Languages

Statement: If L is a regular language, then there exists a constant $p > 0$ (pumping length) such that for any string $w \in L$ with $|w| \geq p$, w can be written as $w = xyz$ where:

1. $|xy| \leq p$
2. $|y| > 0$
3. For all $i \geq 0$, $xy^i z \in L$

Use: To prove that a language is not regular (contrapositive).

Theorem 11: Closure of Context-Free Languages under Union

Statement: If L_1 and L_2 are context-free languages, then $L_1 \cup L_2$ is context-free.

Proof: Construct CFG with new start symbol S and productions $S \to S_1 \mid S_2$.

Theorem 12: Closure of Context-Free Languages under Concatenation

Statement: If L_1 and L_2 are context-free languages, then $L_1 \cdot L_2$ is context-free.

Proof: Construct CFG with new start symbol S and production $S \to S_1S_2$.

Theorem 13: Closure of Context-Free Languages under Kleene Star

Statement: If L is a context-free language, then L^* is context-free.

Proof: Construct CFG with new start symbol S and productions $S \to \varepsilon \mid SS_1$.

Theorem 14: Non-Closure of CFLs under Intersection

Statement: Context-free languages are NOT closed under intersection.

Counterexample: $L_1 = \{a^nb^nc^m \mid n,m \geq 1\}$ and $L_2 = \{a^mb^nc^n \mid n,m \geq 1\}$ are both CFL, but $L_1 \cap L_2 = \{a^nb^nc^n \mid n \geq 1\}$ is not CFL.

Theorem 15: Non-Closure of CFLs under Complement

Statement: Context-free languages are NOT closed under complement.

Proof: Follows from non-closure under intersection and De Morgan's laws.

Theorem 16: Chomsky Normal Form Theorem

Statement: Every context-free grammar can be converted to an equivalent grammar in Chomsky Normal Form, where all productions are of the form $A \to BC$ or $A \to a$ (except $S \to \varepsilon$ if $\varepsilon \in L$).

Construction: Eliminate ε-productions, unit productions, and useless symbols, then convert remaining productions.

Theorem 17: Greibach Normal Form Theorem

Statement: Every context-free grammar can be converted to an equivalent grammar in Greibach Normal Form, where all productions are of the form $A \rightarrow a\alpha$ where a is a terminal and α is a string of variables.

Theorem 18: Equivalence of CFG and PDA

Statement: A language is context-free if and only if it is accepted by some pushdown automaton.

Two Directions:

- **CFG → PDA:** Construct PDA that simulates leftmost derivations
- **PDA → CFG:** Construct CFG from PDA transitions using dynamic programming

---

Lemma 19: Pumping Lemma for Context-Free Languages

Statement: If L is a context-free language, then there exists a constant $p > 0$ such that any string $s \in L$ with $|s| \geq p$ can be written as $s = uvwxy$ where:

1. $|vwx| \leq p$
2. $|vx| \geq 1$
3. For all $i \geq 0$, $uv^i wx^i y \in L$

Use: To prove that a language is not context-free.

Theorem 20: Decidability of CFL Membership

Statement: For any context-free language L and string w, it is decidable whether $w \in L$.

Algorithm: CYK algorithm (requires CNF) with time complexity $O(n^3)$.

Theorem 21: Undecidability of CFL Emptiness Intersection

Statement: It is undecidable whether the intersection of two context-free languages is empty.

Theorem 22: Church-Turing Thesis

Statement: The class of functions computable by a Turing machine exactly corresponds to the class of functions that are algorithmically computable.

Note: This is a thesis (philosophical statement), not a mathematical theorem.

Theorem 23: Equivalence of Turing Machine Variants

Statement: The following are all equivalent in computational power:

- Single-tape Turing machines
- Multi-tape Turing machines
- Non-deterministic Turing machines
- Two-way finite automata with two pushdown stacks

Theorem 24: Universal Turing Machine Theorem

Statement: There exists a Turing machine U (Universal TM) that can simulate any other Turing machine M on input w, given an encoding ⟨M,w⟩.

Theorem 25: Undecidability of the Halting Problem

Statement: The language H = {⟨M,w⟩ | M is a TM that halts on input w} is undecidable.

Proof: Diagonalization argument showing that no TM can decide H.

Theorem 26: Rice's Theorem

Statement: Any non-trivial property of recursively enumerable languages is undecidable.

Definition: A property P is non-trivial if some r.e. languages have property P and others don't.

Theorem 27: Closure Properties of Recursive Languages

Statement: The class of recursive (decidable) languages is closed under:

- Union, intersection, complement
- Concatenation, Kleene star
- Homomorphism and inverse homomorphism

Theorem 28: Closure Properties of R.E. Languages

Statement: The class of recursively enumerable languages is closed under:

- Union, intersection, concatenation, Kleene star
- Homomorphism and inverse homomorphism
- But NOT closed under complement

Theorem 29: Post's Theorem

Statement: A language L is recursive if and only if both L and $\bar{L}$ are recursively enumerable.

Theorem 30: Hierarchy Theorems

Statement:

- **Space Hierarchy:** For space-constructible functions f(n) and g(n), if f(n) = o(g(n)), then SPACE(f(n)) $\subsetneq$ SPACE(g(n))
- **Time Hierarchy:** Similar statement holds for time complexity classes

---

Additional Important Results

Theorem 31: Context-Sensitive Languages and Linear Bounded Automata

Statement: A language is context-sensitive if and only if it is accepted by some linear bounded automaton.

Theorem 32: Savitch's Theorem

Statement: For any function $f(n) \geq \log n$, NSPACE(f(n)) $\subseteq$ SPACE($f(n)^2$).

Theorem 33: Immerman-Szelepcsényi Theorem

Statement: NSPACE(s(n)) = co-NSPACE(s(n)) for $s(n) \geq \log n$.