NP-Completeness

1. Satisfiable boolean formula
2. Polinomial time reducibility
3. (3)CNF-formula
4. NP-complete
5. Cook-Levin theorem
6. 3SAT is NP-complete
7. Satisfiable Boolean Formula

* Variables that can take on the values TRUE and FALSE are called Boolean variables
* The Boolean operations are AND (∧), OR(V), NOT (¬).
* A Boolean formula is an expression involving Boolean variables and operations. For ex: φ = (x ∧ y) ∨ (x ∧ z)
* A Boolean formula is satisfiable if some assignment of 0s and 1s to the variables makes the formula evaluate to 1. The preceding formula is satisfiable because the assignment of x=0, y=1 and z=0 makes it evaluate to 1.
* We define SAT = {⟨φ⟩| φ is a satisfiable Boolean formula}

1. Polynomial time reductibility

* Definition: A function f : Σ∗→Σ∗ is a polynomial time computable function if some polynomial time Turing machine M exists that halts with just f(w) on its tape, when started on any input w
* Language A is polynomial time reducible, to language B, written A ≤P B, if a polynomial time computable function f : Σ∗−→Σ∗ exists, where for every w, w ∈ A ⇐⇒ f(w) ∈ B. The function f is called the polynomial time reduction of A to B.
* If A <=p B and B is in P, then A is in P

Proof: Let M be the polynomial time algorithm deciding B and f be the polynomial time reduction from A to B. We describe a polynomial time algorithm N deciding A as follows:

N = “On input w:

a. Compute f(w).

b. Run M on input f(w) and output whatever M outputs.”

Because f is a reduction from A to B => w ∈ A whenever f(w) ∈ B thus M accepts f(w) whenever w ∈ A.

1. (3)CNF-formula

* a literal is a Boolean variable or a negated Boolean variable. A clause is several literals connected with Vs. A Boolean formula is in conjunctive normal form called a cnf-formula, if it comprises several clauses connected with ∧s. It is a 3cnf-formula if all the clauses have exactly three literals.
* Let 3SAT = {⟨φ⟩| φ is a satisfiable 3cnf-formula}.

1. NP-Complete

* Definition: A language B is NP-Complete if it satisfies two conditions:

1. B is in NP, and
2. Every A in NP is polynomial time reducible to B

* Theorem: If B is in NP-Complete and B is in P, then P =NP

Proof: This theorem follows directly from the definition of polynomial time reducibility

* Theorem: If B is NP-Complete and B <=P C for C in NP, then C is NP-Complete

Proof:

For any A in NP:

A ≤P B (because B is NP-complete)

B ≤P C (given)

Therefore, A ≤P C (by composition)

1. Cook-Levin Theorem

It states that SAT is NP-Complete

Proof:

1. First SAT ∈ NP:

Nondeterministic TM can guess assignment and verify in polynomial time

1. Show every A ∈ NP reduces to SAT:

-Let N be NTM deciding A in time nk

- For input w, construct formula φ that simulates N on w

- φ = φcell ∧ φstart ∧ φmove ∧ φaccept where:

Variables: xi,j,s for position (i,j) and symbol s

φcell = ensures one symbol per cell

φstart = encodes starting configuration on w

φmove = ensures valid transitions between configs

φaccept = ensures accepting state appears

1. Key properties

Φ is satisfiable => N accepts w

Φ’s size is polynomial in |w|

Construction takes polynomial time

Therefore, SAT is NP-Complete as:

1. SAT is in NP
2. Every language in NP reduces to SAT in polynomial time
3. 3SAT is NP-Complete

Proof:

1. 3SAT ∈ NP (obvious)
2. Show NP-Completeness by modifying Cook-Levin proof:

* Start with formula φ = φcell ∧ φstart ∧ φmove ∧ φaccept
* Convert each part to CNF:

φcell : already in CNF (AND of clauses)

φstart: already CNF (AND of single variables)

φaccept: single clause (OR of variables)

φmove : convert using distributive law:

(OR of ANDs) → (AND of ORs)

1. Convert to 3CNF:

For clauses < 3 literals:

- Replicate literals until 3

For clauses > 3 literals:

(a₁ ∨ a₂ ∨ a₃ ∨ a₄) →

(a₁ ∨ a₂ ∨ z) ∧ (z ∨ a₃ ∨ a₄)

General case for l literals:

(a₁ ∨ ... ∨ aₗ) →

(a₁ ∨ a₂ ∨ z₁) ∧ (z₁ ∨ a₃ ∨ z₂) ∧ ... ∧ (zₗ₋₃ ∨ aₗ₋₁ ∨ aₗ)

1. Key points:

* Transformation preserves satisfiability
* Size increases only by constant factor
* Process takes polynomial time

Therefore, 3SAT is NP-Complete.