454 - Assignment 4

Decision Problem Π:

Key Idea:

Prove Π ∈ **NP** by presenting a nondeterministic algorithm that executes in polynomial time and produces a random set T and systematically ensure that |T ∩ Aj |≥ 1 and |T ∩ Bi|≤ 1 for j ≤ m and i ≤ n before outputting “Yes”.

∀Π’ ∈ **NP,** show that Π’ ∝ Π by showing that SAT ∝ Π and using Cook’s Theorem and Lemma 8.5 from the text book. Show SAT ∝ Π by finding a function 𝛤: SAT → Π such that 𝛤 can be computed in polynomial time and ∀π ∈ SAT, π is a yes-instance of SAT iff 𝛤(π) is a yes-instance of Π

Prove Π ∈ **NP**

**Nondeterministic-Algorithm:** VerifyT(A1, ... Am, B1… Bn)

**Input:** two collections of finite sets A1 … Am and B1… Bn

**Output:** yes, if there is a set T such that:

⊥, if there isn’t a set T with the above conditions

**1.** (the guessing phase)

write down a set T(|T| > 0) such that each element is in Aj or Bi, for 1 ≤ j ≤ m, 1 ≤ i ≤ n

**2.** (the verification phase)

**for** i = 1 to m **do:**

**if** |intersection(T, Aj)| < 1

**then:** STOP

**for** j = 1 to n **do:**

**if** |intersection(T, Bi)| > 1

**then:** STOP

output(“Yes”)

**Time Complexity:**

**Theorem 1:** VerifyT executed in polynomial time.

Phase 1 takes O(|T|) time

Phase 2 has two for-loops. The first finds the intersection of T and Aj­ by comparing each element in T with each element in Aj­. Thus, the second for-loop makes O(|T|M) comparisons, were M= in the first for loop. The second finds the intersection of T and Bi­ by comparing each element in T with each element in Bi. Thus, the second for-loop makes O(|T|N) comparisons, were N=

Therefore, non-deterministic algorithm VerifyT executes in O((M+N+1)|T|) time, VerifyT ∈ **P.**

**Correctness:**

In phase 1, a random set T is produced with elements from Aj or Bi, for 1 ≤ j ≤ m, 1 ≤ i ≤ n.

Next, in phase 2, control enters the first for-loop where, by lemma 1, control exits the first for-loop only if |T ∩ Aj| ≥ 1, for 1 ≤ j ≤ m, otherwise control stops. Then control enters the second for-loop where, by lemma 2, control exits the second for-loop only if |T ∩ Bi| ≤ 1, for 1 ≤ i ≤ m, otherwise control stops.

Finally control reaches the last line were ‘yes’ is outputted. Thus, algorithm VerifyT outputs yes if and only if |T ∩ Aj| ≥ 1, for 1 ≤ j ≤ m, and |T ∩ Bi| ≤ 1, for 1 ≤ i ≤ m which are the conditions for a yes instance of Π.

Therefore, algorithm VerifyT can verify a yes instance of Π if it is indeed a yes instance, and by Theorem 1, can do so in polynomial time. Moreover, there is thusly a nondeterministic algorithm that can solvef Π in polynomial time thus Π ∈ **NP.**

**lemma 1:** When the first for-loop in algorithm VerifyT iterates for the kth time, |T ∩ Aj| ≥ 1, for 1 ≤ j ≤ k.

proof by induction:

(*induction basis*) In the first iteration of the for-loop, j = 1, and control enters the then- block of the if-statement, where execution stops, only if |T ∩ A1| < 1, otherwise control moves to the next iteration. Therefore lemma 1 holds for k =1.

(*induction hypothesis*) Suppose Lemma 1 holds for k < m

(*inductive step*) When the for-loop iterates for the mth time, by the inductive hypothesis, |T ∩ Aj| ≥ 1, for 1 ≤ j < m, then control enters the then-block of the if-statement, where execution stops, only if |T ∩ Am| < 1, otherwise control moves to the next iteration, thus when the for-loop iterates again, |T ∩ Aj| ≥ 1, for 1 ≤ j ≤ m.

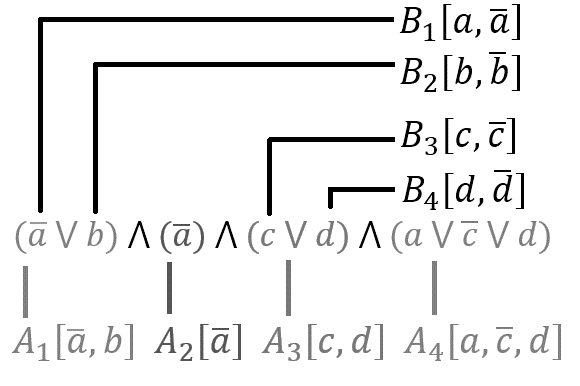
**lemma 2:** When the second for-loop in algorithm VerifyT iterates for the kth time, |T ∩ Bj| ≤ 1, for 1 ≤ j ≤ k.

proof by induction:

(*induction basis*) In the first iteration of the for-loop, i = 1, and control enters the then- block of the if-statement, where execution stops, only if |T ∩ B1| > 1, otherwise control moves to the next iteration. Therefore lemma 2 holds for k =1.

(*induction hypothesis*) Suppose Lemma 1 holds for k < n

(*inductive step*) When the for-loop iterates for the mth time, by the inductive hypothesis, |T ∩ Bj| ≤ 1, for 1 ≤ j < n, then control enters the then-block of the if-statement, where execution stops, only if |T ∩ Bn| > 1, otherwise control moves to the next iteration, thus when the for-loop iterates again, |T ∩ Bj| ≤ 1, for 1 ≤ j ≤ m.

∀Π’ ∈ NP, show that Π’ ∝ Π by showing that SAT ∝ Π

**Algorithm:** 𝛤(πSAT)

**Input:** A string, S, representing a CNF logical expression

**Output:** two collections of finite sets A1 … Am and B1… Bn

let c1, c2, … cm be the clauses in S.

Create lists A1 to Am­ such that Aj is populated with the literals in cj

Create lists B1 to Bn such that Bj = [ x̅, x ] for each variable, x, in the S

Output (A1 … Am, B1… Bn)

**Time Complexity:**

𝛤 executes in time proportional to |S| since it requires is only one read through of the string. 𝛤 ∈ **P**

**Lemma 3:**  ∀π ∈ SAT, π is a yes-instance of SAT ⇔ 𝛤 (π) is a yes-instance of Π

⇒) Assume π is a yes-instance of SAT, prove 𝛤 (π) is a yes-instance of Π

Since π is a yes instance of SAT then there is a truth assignment, TA, for the variables in π such that π evaluates to true.

For each variable x in π, if x is true in TA then add x to T, otherwise add x̅.

Since T satisfies π, and since each clause needs one literal that evaluates to “true” in it in order for π to be satisfied, and since T contains the literals that satisfy π, therefore, at least one literal in T must be in each of the clauses. Furthermore, since each clause is represented by exactly one set Aj which contains each literal in said clause, Therefore, at least one literal in T must be in Aj. 1 ≤ j ≤ m, that is, |T ∩ Aj| ≥ 1

Additionally, since T contains exactly one of x̅ or x for each variable x, |T ∩ Bi | = 1.

⇐) Assume 𝛤(π) is a yes-instance of Π, prove π is a yes-instance of SAT

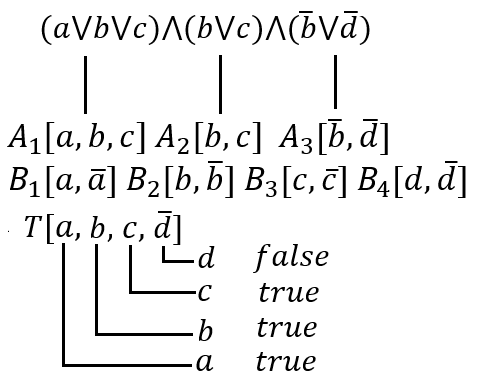
let C be the corresponding CNF logical expression that is constructed as follows. Create a clause in C for each Aj that contains each literal in Aj for 1 ≤ j ≤ m.

Since π ∈ SAT, we can assume that the elements in A and B are literals and that B contains only x and x̅ for each variable x in C.

Assign truth values to the variables in the C as follows. For each variable x in C, if x is in T then set x to “true”, if x̅ is in T set x to “false”. Otherwise set x arbitrarily.

Since (π) is a yes-instance of Π, we know that T contains at least one element from each Aj thus we know that by setting each variable in C to true or false so that each literal in T evaluates to true we will also set at least on literal in each clause in C to true, thus making each clause true. Furthermore, C is logically consistent since T contains at most one literal from each B so T thus cannot contain both x and x̅. (Illustrative example on next page)

Therefore, C is satisfiable.



Conclusion

Since we have shown that 𝛤 executes in polynomial time, and that by lemma 3, ∀π ∈ SAT, π is a yes-instance of SAT ⇔ 𝛤 (π) is a yes-instance of Π, therefore SAT is polynomial reducible to Π, that is SAT ∝ Π.

Since we have shown that SAT ∝ Π, and since by Cook’s theorem SAT is **NP**-complete, (i.e. ∀Π’ ∈ NP, Π’ ∝ SAT) thus by lemma 8.5 ∀Π’ ∈ NP, Π’ ∝ Π. Furthermore, since we have shown Π ∈ **NP** we thusly have Π is **NP-**complete by the definition on page 28 in chapter 8 of the courseware.

■