전산학특강 수업정리

2018320161 송대선

November 30, 2020

1 10월 27일

input : 2 binary decision trees A, B
output : if A and B represent the same boolean function, then YES
otherwise, NO

This problem belongs to coNP and BPP

There exists the probabilistic efficient algorithm that decides whether 2 binary decision trees represent the same boolean function

n : number of variables in a boolean function
use a random assignment of integers in S=[0,2n-1]
p : a prime number $\geq$ —S—

1. assign value i ($0 \leq i \leq 2n-1$) to a variables x for the negation of $x(\backsim x)$
assign 1-i (mod p)
2. For each leaf of the tree labeled "T",
compute the product of the values of the values of variables along the path
(mod p)
3. Add all the values computed in step 2 (mod p)
4. if the values computed in step 3 are different, then return NO,
otherwise, then return YES

$A \vDash \psi$ iff ψ is always true in A

NP is the set of structures T such that
there exists a SOE s such that T is a set of structure A such that s is true in A

2 10월 30일

circuit family :
circuit family C is an infinite list of circuit, $(C_0, C_1, ...)$,

where C_n has n input variables.

We say that C decides a language A over 0, 1 if, for every string w,

$$w \in A \; iff \; C_n(w) = 1$$

where n is the length of w

a circuit is minimal if no smaller circuit is equivalent to it.
the depth of a circuit is the length of the longest path from an input variable to the one output gate
The circuit size complexity of a language is the size of complexity of a minimal circuit family for that language.
The circuit depth complexity of a language is the depth of complexity of a depth minimal circuit family for that language.

Let $T : N \to N$ be a function.
a T(n)-size circuit family is a sequence $\{C_n\}_{n \in N}$ of Boolean circuit,
where C_n has n inputs and a single output, and its size $|C_n| \leq T(n)$ for every n
We say that a language L is in SIZE(T(n))
if there exists a T(n)-size circuit family $\{C_n\}_{n \in N}$ such that for every $x \in \{0,1\}^n, \in L \leftrightarrow C_n(x) = 1$

$P_{/poly}$ is the class of languages that are decidable by polynomial-sized circuit families. That is

$$P_{/poly} = \cup_c SIZE(n^c)$$

P is a subset of $P_{/poly}$
$P_{/poly}$ is not a subset of P.
because there are unary languages that are not undecidable, but in $P_{/poly}$

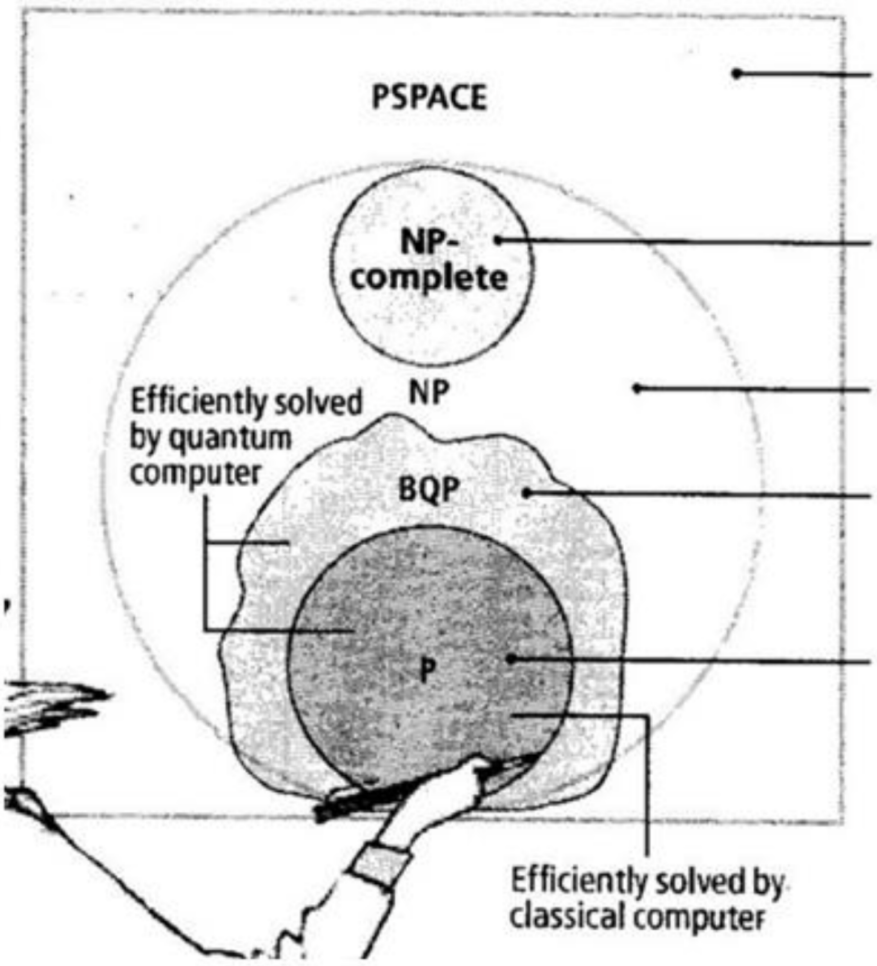
$|0>$ ket vector, $|1>$ bra vector
superposition :

$$\alpha|0> + \beta|1>$$

$$where \; |\alpha|^2 \leq 1, |\beta|^2 \leq 1, |\alpha|^2 + |\beta|^2 = 1$$

$$where \; \alpha^2 \leq 1, \beta^2 \leq 1, \alpha^2 + \beta^2 = 1$$

α and β are "amplitudes"



Deutsch's algorithm
Given $f : \{0, 1\} \to \{0, 1\}$
balanced function : $f(0) = 0, f(1) = 1 | f(0) = 1, f(1) = 0$
constant function : $f(0) = 0, f(1) = 0 | f(0) = 1, f(1) = 1$

1. Preparation

$$|\psi_1 >= |0 > |1 >$$

2. Use Hadamard transformation defined
$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ for each qubit

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0\\ 1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ -1\end{bmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix} = \frac{1}{\sqrt{2}}(|0> + |1>)$$

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\ -1\end{bmatrix} = \frac{1}{\sqrt{2}}(|0> - |1>)$$

$$|\psi_2> = \frac{1}{\sqrt{2}}(|0> + |1>) \otimes \frac{1}{\sqrt{2}}(|0> - |1>)$$

...기타 과정이 더 있다. 시험기간에 더 찾아볼 것

3 11월 03일

a quine : a program whose output is itself when executed
1. program self copy
2. L = ip - 1
3. loop until line[L] = ”end”
4. {
5. print(line[L])
6. L = L+1
7. }
8. print(”end”)
9. end

Recursive Theorem
Let T be a Turing machine that computes a function $t : \Sigma^* \times \Sigma^* \to \Sigma^*$
There is a Turing machine R that computes a function $r : \Sigma^* \to \Sigma^*$ where for every w,

$$r(w) = t(< R >, w)$$

Rice theorem
Let F be any non-trivial subset of the set of all computable partial functions, and let Sf be the set of strings
that describe machines that compute function in F.
Then deciding membership in Sf cannot be solved by an algorithm.

simulated by uniform circuit of size O(t(n)).
a t(n) time-bounded Turing machine can be simulated by uniform circuit of size O(t(n)log t(n)).
an s(n) space-bounded Turing machine can be simulated by uniform circuit of

depth $O(s^*(n)^2)$, where $s^*(n) := max\{s(n), [logn]\}$

Polynomial circuit

Definition :
A language $L \in \{0,1\}^*$ has polynomial circuits
if there is a family of circuits $C = (C_0, C_1, ...)$ such that the following are true :
1. the size of C_n is at most p(n) for some fixed polynomial p.
2. $\forall_x \{0,1\}^*, x \in L$ iff
the output of $C_{|x|}$ is true,
when the i-th input variable is true if x=1, and false otherwise

3 Theorems about polynomial circuit

1. All languages in P have polynomial circuits
2. All languages in BPP have polynomial circuits
3. A language L has uniformly polynomial circuit
if and only if L is a member of P

2 open problems

let LINEAR-SIZE be the class of languages over $\{0,1\}$ that can be computed by family $< B_n >$ of Boolean circuit of size O(n).
PROPOSITION 2: if $P \subset LINEAR - SIZE$, then $P \neq NP$
PROPOSITION 3: if A then BPP=P. if not A then $P \neq NP$

4 11월 06일

NP : a game between a prover and verifier
Prover(Alice) : has unlimited power,
wants to convince verifier(Bob) by giving YES-instance to Bob

Verifier(Bob) : has only deterministic polynomial-time power
that an input X to the problem is a yes input

Interactive proof system and IP
An interactive proof system of a set S is a two-party game.
between a verifier executing a probabilistic polynomial-time strategy(denoted V),
and a prover which executes a computationally unbounded strategy(denoted P),
satisfying Completeness and Soundness

Completeness : For every $x \in S$ the verifier V always accepts

after with the prover P on common input x
Soundness : For some polynomial p,
it holds that for every $x \notin S$ and every potential strategy P^*,
the verifier V rejects with probability at least $1/p(|x|)$,
after interacting with P^* on common input x

The class of problems having interactive proof systems is denoted IP

By repeating a proof system for $O(p(|x|)^2)$ times,
we may decrease the probability that V accepts a false statement
from $1 - (1/p(|x|))$ to $2^{-p(|x|)}$

GRAPH ISOMORPHISM belongs to NP
Given a certificate c, an efficient verifier V(G, H, c) verifies
that c is "really" an isomorphism between G and H.
1. verify c is a permutation of nodes in G
2. permute nodes of G as given in c - check whether the resulting graph is H

step1 : O(number of nodes in G)
step2 : O(number of nodes and edges in G)
$\rightarrow$ Polynomial time (in the size of the graph)

graph isomorphism belongs to NP
Graph nonisomorphism is not known to be in NP
If G_1 and G_2 were indeed nonisomorphic,
the Prover could not have no better than a 50-50 chances

$$IP = PSPACE$$

The power of probabilistic polynomial-time interaction is
exactly the same as the power of power of space!

MIP is multiple-prover interactive proofs.(provers can't communicate)
NEXPTIME is the class of problems computable with an expoenetial time using a magic coin
NEXPTIME is known to be strictly larger than P and NP.
NP is a proper subset of NEXPTIME

Zero Knowledge protocol
This is a paradox

5 11월 10일

Given an undirected graph G = (V,E),
1. a vertex cover is a subset of V such that
for each edge $(u, v) \in E$, contains at least one of the adjacent vertices u and v

2. an independent set S is a subset of V such that
no two nodes in S are joined by an edge

3. a matching M is a subset E such that
for all vertices $v \in V$, at most one edge of M is incident on v
M에 incident한 v를 "matched"되었다고 하고,
그렇지 않으면 "unmatched"하다고 한다.
maximum matching M은 $|M| \geq |M`|$을 의미한다.

S is a vertex cover iff V-S is an independent set

the number of elements in a minimum vertex cover CANNOT be smaller than the number of elements in a maximum matching

Given a maximum matching M,
the set of all endpoints of edges in M is a vertex cover

if $P = NP$,
then solving과 verifying이 "hardness"관점에서 not significantly different하다.
if $P \neq NP$,
solving과 verifying이 "hardness"관점에서 significantly different하다.

solving과 verifying은 사실 proving theorem과 verifying the validity of proof와 같다.
if $P = NP$,
then proving과 verifying이 "hardness"관점에서 not significantly different하다.
if $P \neq NP$,
proving이 verifying보다 더 harder하다.

world를 표현하는 방법음올 formal system을 사용한다.
formal system은 다음 세가지 요소로 이루어진다.
(1) A decidable set of expressions called well-formed formulas(wffs)
(2) A decidable set of axioms which are wffs that are assumed to be true
(3) A set of truth-preserving transformations, called inference rules

Theorem is a wff that is either an axiom or (axiom and theorem) recursively applied inference rules

if we have an oracle that can decides SAT in constant time,
then $P^{SAT} = P^{NP}$, $P = NP$

There is two version of (imaginable) P
(1) our programming language is augmented with a magical new command, which solves any instance of A in a single step
(2) Where A is in P, there is a program that solves A in polynomial time, written using just ordinary commands of our programming language.

Zero knowledge

(1) Bob does not know how to 3-color G and,
(2) Bob "believes" that G is 3-colorable한다고 해도,
he CANNOT convince someone-else the G is 3-colorable

If ONE-way functions exist,
then we can guarantee that all members of NP admit zero-knowledge proofs

One-way functions

A function $f : \{0,1\}^* \to \{0,1\}^*$ is called one-way
if the following two conditions hold:
1. Easy to evaluate : There exists a polynomial-time algorithm A such that $A(x) = f(x)$ for every $x \in \{0,1\}^*$
2. Hard to invert : For every family of polynomial-size circuits $\{C_n\}$, every polynomial p, and all sufficiently large n,

$$Pr[C_n(f(x)) \in f^{-1}(f(x))] < \frac{1}{p(n)}$$

where the probability is taken uniformly
over all the possible choices of $x \in \{0,1\}^*$

SAT is p-selective
a set A is p-selective iff
there exists a polynomial time computable function f
such that for any two strings x and y, $f(x,y) \in \{x,y\}$
and if x or y is in A then f(x,y) is in A.

proof)
let F a SAT formula that is $F(x_1, X_2, \cdots, x_n)$
then there exists polynomial time computable function f such that
if $f(F(True, x_2, \cdots, x_n), F(False, x_2, \cdots, x_n)) = F(True, x_2, \cdots, x_n)$
then $F(x_1, x_2, \cdots, x_n)$ is satisfiable iff $F(True, x_2, \cdots, x_n)$ is satisfiable
if $f(F(True, x_2, \cdots, x_n), F(False, x_2, \cdots, x_n))(True, x_2, \cdots, x_n)$
then $F(x_1, x_2, \cdots, x_n)$ is satisfiable iff $F(False, x_2, \cdots, x_n)$ is satisfiable

6 11월 17일

proof of halting problem)

if there exists a program A(P) such that
if P(P) run forever then halt
if P(P) halts then run forever

if A(A) run forever, then A(A) halts → contradiction!
if A(A) halts, then A(A) run forever → contradiction!

algorithmically verifiable mathematics(AV-mathematics)

any consistent and sufficiently powerful formal theory
that enables to algorithmically verify some text is a proof
is algorithmically verifiable mathematics(AV-mathematics)

Let ψ be a logic (formal system)
that is powerful enough to specify any language in NP

P_{ver}={ L(M)|α is a specification from ψ and M is an algorithm
and there exists a proof in AV-mathematics
that M works in polynomial time and recognizes L(α)}

NP_{ver}={ L(M)|α is a specification from ψ and M is a non-deterministic TM
that provably in AV-mathematics works in polynomial time and accepts L(α)}

7 11월 20일

for SAT, The parameter *alpha* is the ratio:

$$\alpha = \frac{number\ of\ clauses}{number\ of\ variables}$$

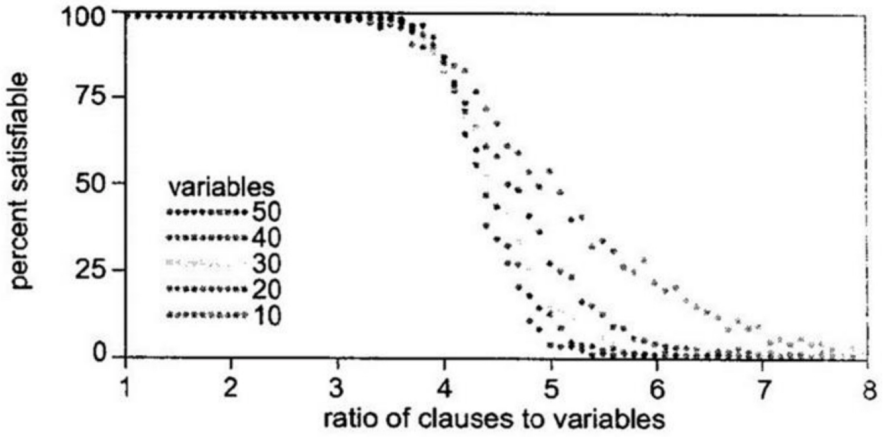
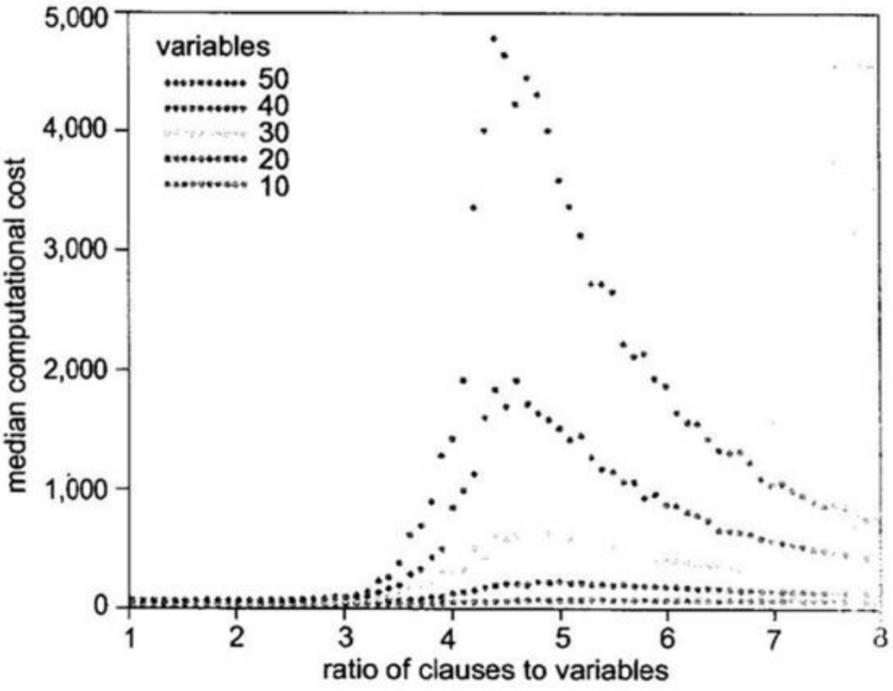


Figure 3. Transition from satisfiable to unsatisfiable gets steeper as the number of variables increases. Each dot is the average of 300 instances.



Where $|V|$ is the number of vertices in the graph,
and p is the edge probability.

$$p = \frac{number\ of\ edges}{number\ of\ possible\ edges} = \frac{|E|}{\frac{|V|(|V|-1)}{2}}$$

$$c = |V|p$$

$$c = \frac{2|E|}{|V|-1}$$

5 possible worlds

(1) Algorithmica
P = NP or some moral equivalent, e.g,$NP \subseteq BPP$
From an efficient verifier → efficient solver can be constructed
This is computational utopia
암호학은 쓸모가 없어진다.

In Algorithmica, 2-step algorithm solves everything
(VLSI circuit minimization problem)
a. you could use your solution to an NPC problem that will recognize.
b. you can use solve it using NPC algorithm

(2) Heuristica
거의 대부분의 NP-problem을 efficient하게 solve하는 마법의 알고리즘이 존재한다.
NP=P는 유효하나, 이 알고리즘을 오작동시키는 몇가지 예외적인 input들이 있다.
그 input들을 알아내는 것은 힘들다.

(3) Pessiland
가장 부정적인 세상
There is no one-way functions.
It is easy to generate many hard instances of NP-problems
one-way function은 계산은 쉬우나, 그 역을 구하는 것이 어려운 함수를 뜻한다.

(4) Minicrypt
one-way function이 존재하는 world이다.

(5) Cryptomania
Cryptomania is the world where the problem of factoring large integers
is exponentially hard on the average case.
아마도 우리가 사는 세상이 이것이다.

RSA

1. 두 소수 p, q를 정한다.
2. r=p*q로 r을 구한다. (r은 공개키가 된다.)
3. (p-1)(q-1)과 서로소인 e를 고른다.
4. d is the unique multiplicative inverse of e mod (p-1)(q-1)
(e*d) mod (p-1)(q-1) = 1이여야 한다.
5. r, e를 공개키로 사용한다.

8 11월 24일

Godel's 1^{st} incompleteness theorem:
if F is a sound formal system,
(soundness : if it is proved, then it comes true.)
(unsound : if it is proved, then it comes false.)
then there exists a natural number m such that
(1) m is not a member of K
(2) the fact "m is not a member of K" is not provable in F

F is in which all wffs are form "x is not a member of K".
x는 자연수, K is a non-computable set
whose members are function values of a certain computable function

즉, 참이면서 증명이 불가능한 명제가 F안에 존재한다는 것이다.

a fuction $f : N \to N$ is computable function
if there exists an algorithm that compute f(x) for given $x \in N$.

set $S \subseteq N$ is computable
if its characteristic function

$$C_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{otherwise} \end{cases}$$

is computable
즉, 어떤 자연수가 그 집합에 속하는지 아닌지를
판별하는 알고리즘이 존재하는 자연수 집합을 computable set이라고 한다.

Theorem :
There is a computable function f whose range

$$K = \{f(0), f(1), f(2), ...\}$$

is not computable

MIU system

wff은 any positive length string over M, U, I으로 이루어진다.
MI는 유일한 axiom이다.
4가지 inference rule이 있다.
(1) wff의 끝에 I,U가 덧붙을 수 있다.
(2) Mx의 형태인 wff가 있고, x도 wff이면, Mxx인 wff가 만들어질 수 있다.
(3) III은 U으로 대체 가능하다.
(4) UU는 삭제될 수 있다.

pq- system

wff is any positive length over p,q,-
infinite axioms : xp-qx-
only one inference rule :
if xpyqz is wff, then xpy-qz- is wff
p를 +로, q를 =로 해석가능하다.

Completeness theorem

In formal system, each proposition is represented by
a corresponding string of symbol "P_n"

P_n is provable in $\mathcal{F} \equiv \vdash_{\mathcal{F}} P_n$

$\mathcal{F}$ is sound if
Whenever $\vdash_{\mathcal{F}} P_n$ for a particular n,
it will also be the case that $n \notin K$

soundness means that the provable statement are true

Incompleteness Theorem

Let $\mathcal{F}$ be a sound formal system.
Then there is a number n_0 such that $n_0 \notin K$,
but it is not the case that $\vdash_{\mathcal{F}} P_{n_0}$

Proof of the Incompleteness Theorem

Incompleteness Theorem을 만족하는 n_0이 존재하지 않는다고 가정하면,
$\vdash_{\mathcal{F}} P_n$ for particular n, iff $n \notin K$을 얻는다.

Theorem of F의 관점으로 보면
f(0), f(1), f(2)을 계산하기 위해서 $n \in K$이어야하고,
따라서 $C_K(n) = 1$이 된다.
반면, P_n의 관점으로 볼때, $C_K(n) = 0$이 된다.
이는 모순이므로, n_0은 반드시 존재한다.

algorithmically verifiable mathematics(AV-mathematics)

주어진 any text가 proof인지 아닌지를
algorithmically verify가능한 mathematics이다.

$x \in \Sigma^*$에 대하여, AV-mathematics안에서
$x \in L$ or $x \notin L$으로 증명이 가능하다면,
decision problem $L \subseteq \Sigma^*$은 solvable하다.

formal Theory

formal Theory is consistent and sound
and in which proof deciding problem is computable

input : an arbitrary text T
output :
yes, if T is a proof
no, otherwise

(1) 어떤 membership 문제는 "almost everywhere solvable하다"고
증명이 가능하다.
(2) language of AV-mathematics:
1st order logic, logical connectives이다.

Theorem
If the claim P=NP is not provable in AV-mathematics,
then $P_{ver} \neq NP_{ver}$