Cook's Theorem

Papamakarios Theodoros

November 3, 2014

Reductions

Definition

A polynomial reduction from a language $L_1 \subseteq \Sigma_1^*$ *to a language* $L_2 \subseteq \Sigma_2^*$ *is a function* $f : \Sigma_1^* \to \Sigma_2^*$ *such that*

1. *There is a polynomial time DTM program that computes* f.
2. *For all* $x \in \Sigma_1^*$, $x \in L_1^*$ *if and only if* $f(x) \in L_2^*$.

Reductions

Definition

A polynomial reduction from a language $L_1 \subseteq \Sigma_1^*$ *to a language* $L_2 \subseteq \Sigma_2^*$ *is a function* $f : \Sigma_1^* \rightarrow \Sigma_2^*$ *such that*

1. *There is a polynomial time DTM program that computes* f.
2. *For all* $x \in \Sigma_1^*$, $x \in L_1^*$ *if and only if* $f(x) \in L_2^*$.

Lemma

If $L_1 \propto L_2$, *then* $L_2 \in \mathcal{P} \Rightarrow L_1 \in \mathcal{P}$.

Reductions

Definition

A language L is defined to be $\mathcal{NP}$-complete if $L \in \mathcal{NP}$ and, for all other languages $L' \in \mathcal{NP}$, $L' \propto L$.

Reductions

Definition

A language L is defined to be $\mathcal{NP}$-complete if $L \in \mathcal{NP}$ and, for all other languages $L' \in \mathcal{NP}$, $L' \propto L$.

If L is $\mathcal{NP}$-complete, $L \in \mathcal{P} \Leftrightarrow \mathcal{P} = \mathcal{NP}$.

Reductions

Definition

A language L is defined to be $\mathcal{NP}$-complete if $L \in \mathcal{NP}$ and, for all other languages $L' \in \mathcal{NP}$, $L' \propto L$.

If L is $\mathcal{NP}$-complete, $L \in \mathcal{P} \Leftrightarrow \mathcal{P} = \mathcal{NP}$.

Lemma

If L_1 and L_2 belong to $\mathcal{NP}$, L_1 is $\mathcal{NP}$-complete, and $L_1 \propto L_2$, then L_2 is $\mathcal{NP}$-complete.

SAT

SATISFIABILITY

INSTANCE: A set X of variables and a collection C of clauses over X (a CNF formula).

QUESTION: Is there a satisfying truth assignment for C?

SAT

SATISFIABILITY

INSTANCE: A set X of variables and a collection C of clauses over X (a CNF formula).

QUESTION: Is there a satisfying truth assignment for C?

$$SAT = \{\phi : \phi \textit{ a propositional formula in CNF such that } \phi \textit{ is satisfiable}\}$$

SAT

$$C = \{x_1 \vee \neg x_2, \neg x_1 \vee x_2\}$$

SAT

$$C = \{x_1 \vee \neg x_2, \neg x_1 \vee x_2\}$$

Satisfiable: $x_1 \longrightarrow$ **true**, $x_2 \longrightarrow$ **true**

SAT

$$C = \{x_1 \vee \neg x_2, \neg x_1 \vee x_2\}$$

Satisfiable: $x_1 \longrightarrow$ **true**, $x_2 \longrightarrow$ **true**

$$C' = \{x_1 \vee x_2, x_1 \vee \neg x_2, \neg x_1 \vee x_2, \neg x_1 \vee \neg x_2\}$$

SAT

$$C = \{x_1 \vee \neg x_2, \neg x_1 \vee x_2\}$$

Satisfiable: $x_1 \longrightarrow \textbf{true}$, $x_2 \longrightarrow \textbf{true}$

$$C' = \{x_1 \vee x_2, x_1 \vee \neg x_2, \neg x_1 \vee x_2, \neg x_1 \vee \neg x_2\}$$

Unsatisfiable: No satisfying truth assignment.

SAT

- We wish to show that SAT is $\mathcal{NP}$-complete,

SAT

- We wish to show that *SAT* is $\mathcal{NP}$-complete,
- i.e., for all $L \in \mathcal{NP}$, L is reduced in polynomial time to *SAT*.

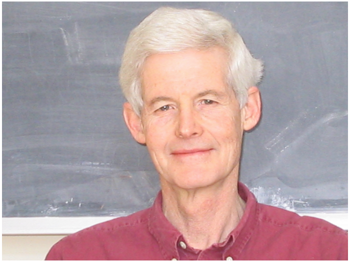
SAT

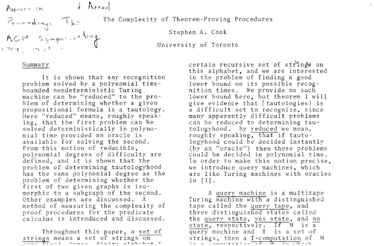
- We wish to show that *SAT* is $\mathcal{NP}$-complete,
- i.e., for all $L \in \mathcal{NP}$, L is reduced in polynomial time to *SAT*.
- *SAT* was the "first" $\mathcal{NP}$-complete problem.

SAT

- We wish to show that SAT is $\mathcal{NP}$-complete,
- i.e., for all $L \in \mathcal{NP}$, L is reduced in polynomial time to SAT.
- SAT was the "first" $\mathcal{NP}$-complete problem.
- But why SAT...?

Cook





Appears in ... Annual
Proceedings Th...
ACM Symposi... ...ting

The Complexity of Theorem-Proving Procedures

Stephen A. Cook

University of Toronto

<u>Summary</u>

It is shown that any recognition problem solved by a polynomial time-bounded nondeterministic Turing machine can be "reduced" to the problem of determining whether a given propositional formula is a tautology. Here "reduced" means, roughly speaking, that the first problem can be solved deterministically in polynomial time provided an oracle is available for solving the second. From this notion of reducible, polynomial degrees of difficulty are defined, and it is shown that the problem of determining tautologyhood has the same polynomial degree as the problem of determining whether the first of two given graphs is isomorphic to a subgraph of the second. Other examples are discussed. A method of measuring the complexity of proof procedures for the predicate calculus is introduced and discussed.

Throughout this paper, a <u>set of strings</u> means a set of strings on some fixed, large, finite alphabet Σ ... certain recursive set of strings on this alphabet, and we are interested in the problem of finding a good lower bound on its possible recognition times. We provide no such lower bound here, but theorem 1 will give evidence that {tautologies} is a difficult set to recognize, since many apparently difficult problems can be reduced to determining tautologyhood. By <u>reduced</u> we mean, roughly speaking, that if tautologyhood could be decided instantly (by an "oracle") then these problems could be decided in polynomial time. In order to make this notion precise, we introduce query machines, which are like Turing machines with oracles in [1].

A <u>query machine</u> is a multitape Turing machine with a distinguished tape called the <u>query tape</u>, and three distinguished states called the <u>query state</u>, <u>yes state</u>, and <u>no state</u>, respectively. If M is a query machine and T is a set of strings, then a <u>T-computation</u> of M ...

The Theorem

Proposition

$SAT \in \mathcal{NP}$.

Proof.

Trivial. $\square$

The Theorem

Proposition

$SAT \in \mathcal{NP}$.

Proof.

Trivial. $\square$

Proposition

For every $L \in \mathcal{NP}$, $L \propto SAT$.

Proof: Non trivial.

The Theorem

Let $L \in \mathcal{NP}$ and M a polynomial time NDTM which decides the language L. Let $p(n)$ be a polynomial that bounds the time complexity function $T_M(n)$.

The Theorem

Let $L \in \mathcal{NP}$ and M a polynomial time NDTM which decides the language L. Let $p(n)$ be a polynomial that bounds the time complexity function $T_M(n)$.

$$M, x \longrightarrow \phi(x),$$

such that ϕ is satisfiable if and only if there is a certificate for which M accepts input x (and $\phi(x)$ can be constructed in polynomial time).

The Theorem

Let $L \in \mathcal{NP}$ and M a polynomial time NDTM which decides the language L. Let $p(n)$ be a polynomial that bounds the time complexity function $T_M(n)$.

$$M, x \longrightarrow \phi(x),$$

such that ϕ is satisfiable if and only if there is a certificate for which M accepts input x (and $\phi(x)$ can be constructed in polynomial time).

Suppose that M's set of states is

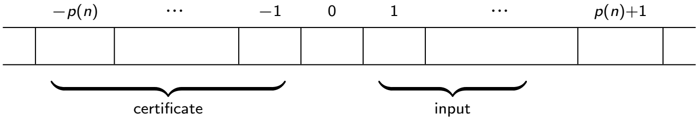
$$Q = \{q_0 = q_{start}, q_1 = q_{yes}, q_2 = q_{no}, \ldots, q_r\}$$

and M's alphabet is

$$\Gamma = \{s_0 = \sqcup, s_1, \ldots, s_v\}$$

The Theorem

Assume that the certificate is written in cells -1 to $-p(n)$ and the input x is written in cells 1 to $|x|$. Cell 0 always contains by convention the blank symbol $\sqcup$.



The computation is specified completely by giving the contents of these squares, the current state and the position of the head at each time 0 to $p(n)$.

Variables

ϕ's variables will be

Variable	Range	Intended Meaning
$Q[i,k]$	$0 \leq i \leq p(n)$ $0 \leq k \leq r$	At time i M is in state q_k.
$H[i,j]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n)+1$	At time i M's head is scanning cell j.
$S[i,j,k]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n)+1$ $0 \leq k \leq v$	At time i M's j's cell contains symbol s_k.

Clauses

Group 1

$$\{Q[i,0] \vee Q[i,1] \vee \cdots \vee Q[i,r]\}, \quad 0 \leq i \leq p(n)$$

$$\{\neg Q[i,j] \vee \neg Q[i,j']\}, \quad 0 \leq i \leq p(n),\ 0 \leq j < j' \leq r$$

$$\equiv \{\neg(Q[i,j] \wedge Q[i,j'])\}$$

Clauses

Group 1

$$\{Q[i,0] \vee Q[i,1] \vee \cdots \vee Q[i,r]\}, \quad 0 \leq i \leq p(n)$$
$$\{\neg Q[i,j] \vee \neg Q[i,j']\}, \quad 0 \leq i \leq p(n),\ 0 \leq j < j' \leq r$$
$$\equiv \{\neg(Q[i,j] \wedge Q[i,j'])\}$$

The machine must be at exactly one state at each time. We suppose that if M accepts before time $p(n)$, then it remains at this configuration until time $p(n)$.

Clauses

Group 1

$$\{Q[i,0] \vee Q[i,1] \vee \cdots \vee Q[i,r]\}, \quad 0 \leq i \leq p(n)$$
$$\{\neg Q[i,j] \vee \neg Q[i,j']\}, \quad 0 \leq i \leq p(n),\ 0 \leq j < j' \leq r$$
$$\equiv \{\neg(Q[i,j] \wedge Q[i,j'])\}$$

The machine must be at exactly one state at each time. We suppose that if M accepts before time $p(n)$, then it remains at this configuration until time $p(n)$.

$\mathcal{O}(p(n))$ such clauses.

Clauses

Group 2 $\mathcal{O}(p^3(n))$ clauses

$$\{H[i,-p(n)] \vee \cdots \vee H[i,p(n)+1]\},\ 0 \le i \le p(n)$$
$$\{\neg H[i,j] \vee \neg H[i,j']\},\qquad 0 \le i \le p(n),\ -p(n) \le j < j' \le p(n)+1$$

Clauses

Group 2 $\mathcal{O}(p^3(n))$ clauses

$$\{H[i,-p(n)] \vee \cdots \vee H[i,p(n)+1]\},\ 0 \leq i \leq p(n)$$
$$\{\neg H[i,j] \vee \neg H[i,j']\},\qquad 0 \leq i \leq p(n),\ -p(n) \leq j < j' \leq p(n)+1$$

Head must be reading exactly one cell at each time.

Clauses

Group 2 $\mathcal{O}(p^3(n))$ clauses

$$\{H[i,-p(n)] \vee \cdots \vee H[i,p(n)+1]\},\ 0 \leq i \leq p(n)$$
$$\{\neg H[i,j] \vee \neg H[i,j']\}, \qquad 0 \leq i \leq p(n),\ -p(n) \leq j < j' \leq p(n)+1$$

Head must be reading exactly one cell at each time.

Group 3 $\mathcal{O}(p^2(n))$ clauses

$$\{S[i,j,0] \vee \cdots \vee S[i,j,v]\}, \quad 0 \leq i \leq p(n),\ p(n) \leq j \leq p(n)+1$$
$$\{\neg S[i,j,k] \vee \neg S[i,j,k']\}, \quad 0 \leq i \leq p(n),\ -p(n) \leq j \leq p(n)+1,\ 0 \leq k < k' \leq v$$

Clauses

Group 2 $\mathcal{O}(p^3(n))$ clauses

$$\{H[i,-p(n)] \vee \cdots \vee H[i,p(n)+1]\},\ 0 \le i \le p(n)$$
$$\{\neg H[i,j] \vee \neg H[i,j']\},\qquad 0 \le i \le p(n),\ -p(n) \le j < j' \le p(n)+1$$

Head must be reading exactly one cell at each time.

Group 3 $\mathcal{O}(p^2(n))$ clauses

$$\{S[i,j,0] \vee \cdots \vee S[i,j,v]\},\qquad 0 \le i \le p(n),\ p(n) \le j \le p(n)+1$$
$$\{\neg S[i,j,k] \vee \neg S[i,j,k']\},\qquad 0 \le i \le p(n),\ -p(n) \le j \le p(n)+1,\ 0 \le k < k' \le v$$

For each time, there must be exactly one symbol at each cell.

Clauses

Group 4 $\mathcal{O}(p(n))$ clauses

$$\{Q[0,0]\}, \{H[0,1]\}, \{S[0,0,0]\},$$
$$\{S[0,1,k_1]\}, \{S[0,2,k_2\}, \ldots, \{S[0,n,k_n]\},$$
$$\{S[0,n+1,0]\}, \{S[0,n+2,0]\}, \ldots, \{S[0,p(n)+1,0]\},$$

where $x = s_{k_1} s_{k_2} \ldots s_{k_n}$

Clauses

Group 4 $\mathcal{O}(p(n))$ clauses

$$\{Q[0,0]\},\{H[0,1]\},\{S[0,0,0]\},$$
$$\{S[0,1,k_1]\},\{S[0,2,k_2\},\ldots,\{S[0,n,k_n]\},$$
$$\{S[0,n+1,0]\},\{S[0,n+2,0]\},\ldots,\{S[0,p(n)+1,0]\},$$
where $x = s_{k_1}s_{k_2}\ldots s_{k_n}$

At time 0, the computation is in the initial configuration for input x.

Clauses

Group 4 $\mathcal{O}(p(n))$ clauses

$$\{Q[0,0]\}, \{H[0,1]\}, \{S[0,0,0]\},$$
$$\{S[0,1,k_1]\}, \{S[0,2,k_2\}, \ldots, \{S[0,n,k_n]\},$$
$$\{S[0,n+1,0]\}, \{S[0,n+2,0]\}, \ldots, \{S[0,p(n)+1,0]\},$$
$$\text{where } x = s_{k_1} s_{k_2} \ldots s_{k_n}$$

At time 0, the computation is in the initial configuration for input x.

Group 5

$$\{Q[p(n),1]\}$$

Clauses

Group 4 $\mathcal{O}(p(n))$ clauses

$$\{Q[0,0]\}, \{H[0,1]\}, \{S[0,0,0]\},$$
$$\{S[0,1,k_1]\}, \{S[0,2,k_2\}, \ldots, \{S[0,n,k_n]\},$$
$$\{S[0,n+1,0]\}, \{S[0,n+2,0]\}, \ldots, \{S[0,p(n)+1,0]\},$$
where $x = s_{k_1} s_{k_2} \ldots s_{k_n}$

At time 0, the computation is in the initial configuration for input x.

Group 5

$$\{Q[p(n),1]\}$$

By time $p(n)$, M must enter state q_{yes} and hence accept x.

Clauses

Group 6 $\mathcal{O}(p^2(n))$ clauses

The first subgroup guarantees that if the head is not scanning tape square j at time i, then the symbol in cell j does not change between times i and $i+1$.

$$\begin{array}{ll} \{\neg S[i,j,l] \vee H[i,j] \vee S[i+1,j,l]\}, & 0 \leq i < p(n), \\ \quad \equiv \{(S[i,j,l] \wedge \neg H[i,j]) \Rightarrow S[i+1,j,l]\} & -p(n) \leq j \leq p(n)+1, \\ & 0 \leq l \leq v \end{array}$$

Group 6 $\mathcal{O}(p^2(n))$ clauses

The remaining subgroup guarantees that the changes from one configuration to the next are in accord with the transition function δ for M. For each quadruple (i, j, k, l), $0 \leq i \leq p(n)$, $-p(n) \leq j \leq p(n) + 1$, $0 \leq k \leq r$ and $0 \leq l \leq v$, this subgroup contains the following three clauses:

$$\{\neg H[i,j] \vee \neg Q[i,k] \vee \neg S[i,j,l] \vee H[i+1, j+\Delta]\}$$
$$\equiv \{(H[i,j] \wedge Q[i,k] \wedge S[i,j,l]) \Rightarrow H[i+1, j+\Delta]\}$$
$$\{\neg H[i,j] \vee \neg Q[i,k] \vee \neg S[i,j,l] \vee Q[i+1, k']\}$$
$$\{\neg H[i,j] \vee \neg Q[i,k] \vee \neg S[i,j,l] \vee S[i+1, j, l']\}$$

where if $q_k \in Q - \{q_{yes}, q_{no}\}$, then the values of Δ, k' and l' are such that $\delta(q_k, s_l) = (q_{k'}, s_{l'}, \Delta)$ and if $q_k \in \{q_{yes}, q_{no}\}$, then $\Delta = -$, $k' = k$ and $l' = l$.

Almost there

If $x \in L$, then there is a certificate for which M's computation on x will accept after at most $p(n)$ steps, and this computation, given the interpretation of the variables, imposes a truth assignment that satisfies all the clauses in $C = G_1 \cup G_2 \cup G_3 \cup G_4 \cup G_5 \cup G_6$.

Conversely, the construction of C is such that any satisfying truth assignment for C must correspond to an accepting computation of M on x for a certificate (the certificate constructed by the truth assignment).

Plus, the construction can be done in polynomial time.

Aftermath

REDUCIBILITY AMONG COMBINATORIAL PROBLEMS†

Richard M. Karp
University of California at Berkeley

Abstract: A large class of computational problems involve the determination of properties of graphs, digraphs, integers, arrays of integers, finite families of finite sets, boolean formulas and elements of other countable domains. Through simple encodings from such domains into the set of words over a finite alphabet these problems can be converted into language recognition problems, and we can inquire into their computational complexity. It is reasonable to consider such a problem satisfactorily solved when an algorithm for its solution is found which terminates within a number of steps bounded by a polynomial in the length of the input. We show that a large number of classic unsolved problems of covering, matching, packing, routing, assignment and sequencing are equivalent, in the sense that either each of them possesses a polynomial-bounded algorithm or none of them does.

Cook's paper was published in 1971. In 1972 Karp showed in the above paper 21 $\mathcal{NP}$-complete problems. And so on...

An aside: Satisfiability variants

3-SAT

INSTANCE: A CNF formula C such that every clause has three literals.

QUESTION: Is there a satisfying truth assignment for C?

Proposition

$3 - SAT$ is $\mathcal{NP}$-complete.

An aside: Satisfiability variants

3-SAT

INSTANCE: A CNF formula C such that every clause has three literals.

QUESTION: Is there a satisfying truth assignment for C?

Proposition

$3 - SAT$ is $\mathcal{NP}$-complete.

MON3-SAT

INSTANCE: A CNF formula C such that every clause has three variables all negated or all not negated.

QUESTION: Is there a satisfying truth assignment for C?

Proposition

$MON3 - SAT$ is $\mathcal{NP}$-complete.

An aside: Satisfiability variants

Proposition

$2-SAT$ is $\mathcal{NL}$-complete.

An aside: Satisfiability variants

Proposition

$2-SAT$ is $\mathcal{NL}$-complete.

A Horn clause is a clause such that all variables in it are negated except (maybe) one. Many Horn clauses make up a Horn formula.

HORN-SAT

INSTANCE: A Horn formula C.

QUESTION: Is there a satisfying truth assignment for C?

Proposition

$HORN-SAT$ is $\mathcal{P}$-complete.