The Complexity of Theorem-Proving Procedures

Algorithms - M.S. ΑΛΜΑ[1]

Andreas Panayi

School of Electrical and Computer Engineering NTUA

November 2024

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ
ΠΟΛΥΤΕΧΝΕΙΟ

[1] andreaspanayi8@gmail.com

Table of Contexts

Deterministic Turing Machine

A Deterministic Turing Machine (TM) is a tuple of seven elements:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}})$$

where:

1) Q, Σ, Γ are finite sets, and:
2) $q_0, q_{\text{yes}}, q_{\text{no}} \in Q$, such that $q_{\text{yes}} \neq q_{\text{no}}$.
   Here, q_{yes} and q_{no} are the final states.
3) $\Sigma \subset \Gamma$
4) $\triangleright, \sqcup \in \Gamma \setminus \Sigma$
5) $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function (a partial function) such that:
   - After the TM reaches q_{yes} or q_{no}, δ does not provide further transition (is not defined).
   - $\triangleright$ cannot be deleted and represents the beginning of the tape.

Non Deterministic Turing Machine

A Non Deterministic Turing Machine (NTM) is a tuple of seven elements:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}})$$

where:

1) Q, Σ, Γ are finite sets, and:
2) $q_0, q_{\text{yes}}, q_{\text{no}} \in Q$, such that $q_{\text{yes}} \neq q_{\text{no}}$.
   Here, q_{yes} and q_{no} are the final states.
3) $\Sigma \subset \Gamma$
4) $\triangleright, \sqcup \in \Gamma \setminus \Sigma$
5) $\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$ is the transition relation such that:
   - After the TM reaches q_{yes} or q_{no}, δ does not provide with further transition
   - $\triangleright$ cannot be deleted and represents the beginning of the tape.

Languages, Alphabets

We call alphabet a set of elements Σ (usually $\Sigma = \{0, 1\}$).

We call Language L every Set that contains elements constructed by (finite) selection of symbols form an alphabet Σ.

Every $L \subseteq \Sigma^*$ is called a language (where $*$ denotes the Kleene Star).

We are interested in decision problems, in other words if a string $w \in \Sigma^*$ belongs or not in a Language.

To do so we use algorithms to decide that.

Algorithm is defined to be everything that a Turing Machine (or an equivalent computation model) can do.

P and NP

$$\text{P} = \{L \subseteq \Sigma^* | \exists \text{ TM } M \text{ that given an input } w \text{ decides if } w \in L \text{ or } w \notin L \text{ within } O(Q(|w|)) \text{ steps, for some polynomial } Q\}$$

$$\text{NP} = \{L \subseteq \Sigma^* | \exists \text{ NTM } M \text{ that given an input } w \text{ decides if } w \in L \text{ or } w \notin L \text{ within } O(Q(|w|)) \text{ steps, for some polynomial } Q\}$$

Reduction

It is shown that any decision 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.

Notation

We will be talking about formulas in propositional calculus,which means we will need infinite propositional symbols (atoms). They will be represented as strings by a member of Σ, followed by the binary representation of a number.

We use the symbols $\neg, \vee, \wedge$ to denote negation, or, and respectively.

Definition $(\{\text{tautologies}\})$
The set of tautologies (denoted by tautologies) is a certain recursive set of strings on this alphabet.

Query TM and P-Reduction

Definition (Query TM)
A Query Machine is a Multitape Turing Machine with a distinguished tape called the query tape, and three distinguished states called the query state, yes state, and no state, respectively. If M is a query machine and T is a Language, then a T-computation of M is a computation of M in which initially M is in the initial state and has an input string w on its input tape, and each time M assumes the query state there is a string u on the query tape, and the next state M assumes is the yes state if $u \in T$ and the no state if $u \notin T$.
We think of an "oracle", which knows T, placing M in the yes state or no state.

Definition (P-Reduction)
A Language S of strings is P-reducible (P for polynomial) to a language T of strings iff there is some query machine M and a polynomial $Q(n)$ such that for each input string w, the T-computation of M with input w halts within $Q(|w|)$ steps ($|w|$ is the length of w) and ends in an accepting state iff $w \in S$.

Equivalence Classes

P-reducibility is a transitive relation.

Thus the relation E on Languages, given by $(S, T) \in E$ iff each of S, T is P-reducible to the other, is an equivalence relation.

The equivalence class containing a language S will be denoted by $\deg(S)$ (the polynomial degree of difficulty of S).

The notation we use is: (A is reduced to B) $\equiv$ ($A \leq_P B$)

Special Languages

We now define the following special Languages[2].

1. The subgraph problem, denoted by $\{$subgraph pairs$\}$
2. The graph isomorphism problem, denoted by $\{$isomorphic graphpairs$\}$
3. The language $\{$Primes$\}$
4. The language $\{$DNF tautologies$\}$
5. The language D_3

[2] DNF=Disjunctive Normal Form, CNF=Conjuctive Normal Form

Every Language in co-NP is reducible to $\{$DNF tautologies$\}$

Theorem ($\{$DNF tautologies$\}$ is co-NP Complete[3])
If a language L is decided by some nondeterministic Turing machine within polynomial time, then L is P-reducible to $\{$DNF tautologies$\}$ for some polynomial P.

Corollary:
All languages (1)-(5) from the previous slide are P-reducible to $\{$DNF tautologies$\}$.

[3] We will define this later on

Proof Step 0: $\{$CNF satisfiablity$\}$ to $\{$DNF tautologies$\}$

Case 1: $\varphi \in \{\text{CNF satisfiablity}\} \iff \neg\varphi \notin \{\text{DNF tautologies}\}$ and
Case 2: $\varphi \notin \{\text{CNF satisfiablity}\} \iff \neg\varphi \in \{\text{DNF tautologies}\}$[4].

So, it suffices to show that, every language in NP can be P-reduced to $\{$CNF satisfiablity$\}$.

[4] Using De Morgan's Law

Proof Step 1: Preliminaries

Let $L \in$ NP. Then exists a TM M that decides the language within $Q(n)$ steps where n is the size of the input, where Q is a polynomial.

For simplicity we assume that M has only one tape, which is infinite to the right but has a left-most square.

Let us number the squares from left to right 1, 2, . . .

Proof Step 2: Preliminaries

Let us fix an input w to M of length n. Then there is a computation of M with input w that ends within $T := Q(n)$ steps.

The formula $A(w)$ will be built from many different proposition symbols, whose intended meanings, listed below, refer to such a computation.

Suppose the tape alphabet for M is $\{\sigma_1, \ldots, \sigma_l\}$ and the set of states is $\{q_1, \ldots, q_r\}$.

Notice that since the computation has at most $T = Q(n)$ steps, no tape square beyond T is scanned.

Proof Step 3: Proposition Symbols

Proposition symbols:

$P^i_{s,t}$ for $1 \leq i \leq l, 1 \leq s,t \leq T$.
$P^i_{s,t}$ is true iff tape square number s at step t contains the symbol σ_i.

Q^i_t for $1 \leq i \leq r, 1 \leq t \leq T$.
Q^i_t is true iff at step t the machine is in state q_i.

$S_{s,t}$ for $1 \leq s,t \leq T$.
$S_{s,t}$ is true iff at time t square number s is scanned by the tape head.

The formula $A(w)$ is a conjunction $B \wedge C \wedge D \wedge E \wedge F \wedge G \wedge H \wedge I$ where each closed will be specified later on.

Notice $A(w)$ is in conjunctive normal form.

Proof Step 4: Construction of B

B is a conjunction

$$B := B_1 \wedge B_2 \wedge \ldots \wedge B_T$$

where, B_t asserts that at time t one and only one square is scanned:

$$B_t = (S_{1,t} \vee \ldots \vee S_{T,t}) \wedge \left(\bigwedge_{1 \leq i,j \leq T, i \neq j} (\neg s_{i,t} \vee \neg s_{j,t}) \right)$$

Proof Step 5: Construction of C

C is a conjunction

$$C := \bigwedge_{1 \leq s,t \leq T} C_{s,t}$$

where, $C_{s,t}$ asserts that at square s and time t there is one and only one symbol:

$$C_{s,t} = \left(\bigvee_{i=1}^{l} P_{s,t}^{i} \right) \wedge \left(\bigwedge_{1 \leq i_1, i_2 \leq l, i_1 \neq i_2} (\neg P_{s,t}^{i_1} \vee \neg P_{s,t}^{i_2}) \right)$$

Proof Step 6: Construction of D

D is a conjunction

$$D := D_1 \wedge D_2 \wedge \ldots \wedge D_T$$

where, D_t asserts that at time t one and only one state:

$$D_t = \left(\bigwedge_{i=1}^{r} Q_t^i \right) \wedge \left(\bigvee_{1 \leq i_1, i_2 \leq T, i_1 \neq i_2} (\neg Q_t^{i_1} \vee \neg Q_t^{i_2}) \right)$$

Proof Step 7: Construction of E

E asserts the initial conditions are satisfied:

$$E = Q_1^0 \wedge S_{1,1} \wedge P_{1,1}^{i_1} \wedge P_{2,1}^{i_2} \wedge \ldots \wedge P_{n,1}^{i_n} \wedge P_{n+1,1}^{1} \wedge \cdots \wedge P_{T,1}^{1}$$

where $w = \sigma_{i_1} \ldots \sigma_{i_n}$, q_0 is the initial state and σ_1 is the blank symbol.

Proof Step 8: Construction of F

F asserts that for each time t the value of P is determined properly.
F is the conjunction over all t, i, j of $G^t_{i,j}$

$$F = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} F^t_{i,j}$$

where $F^t_{i,j}$ asserts that if at time t the machine is in cell with symbol σ_i, then at time $t+1$ the same cell will have a new symbol σ_k, where σ_k is given by the transition function of M.

$$F^t_{i,j} = \bigwedge_{s=1}^{T} \left(\neg Q^t_i \vee \neg S_{s,t} \vee \neg P^j_{s,t} \vee P^k_{s,t+1} \right)$$

Proof Step 9: Construction of G

G asserts that for each time t the value of Q is determined properly.
G is the conjunction over all t, i, j of $G^t_{i,j}$

$$G = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} G^t_{i,j}$$

where, $G^t_{i,j}$ asserts that if at time t the machine is in state q_i scanning symbol σ_j , then at time $t+1$ the machine is in state q_k , where q_k is the new state given by the transition function of M.

$$G^t_{i,j} = \bigwedge_{s=1}^{T} \left(\neg Q^t_i \vee \neg S_{s,t} \vee \neg P^j_{s,t} \vee Q^k_{t+1} \right)$$

Proof Step 10: Construction of H

H asserts that for each time t the values of S is determined properly.
H is the conjunction over all t, i, j of $H_{i,j}^t$

$$H = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} H_{i,j}^t$$

where, $H_{i,j}^t$ asserts that if at time t the machine is scanning the cell with number s, then at time $t+1$ the machine will scan the cell with number k , where k is such, according direction given by the transition function of M.

$$H_{i,j}^t = \bigwedge_{s=1}^{T} \left(\neg Q_i^t \vee \neg S_{s,t} \vee \neg P_{s,t}^j \vee S_{k,t+1} \right)$$

Proof Step 11: Construction of I

Finally, the formula I asserts that the machine reaches an accepting state at some time.

$$I = \left(\bigvee_{i=1}^{T} Q_i^r \right)$$

Proof: Transition Relation

The same proof holds true for *Transition Relation* instead of transition function.

Also another idea is using a different (equivalent) definition of a Non Deterministic Turing machine that instead of having a transition relation it allows for multiple transition functions, we encode all those transition functions as shown in the steps 8-9-10.

Proof: Correctness

Constructing the logic formula takes polynomial time in terms of the encoding of a TM Machine.

It is now straightforward to verify that $w \in L \iff A(w) \in \{\text{DNF tautologies}\}$ because indeed:

$P^i_{s,t}$ is true $\iff$ tape square number s at step t contains the symbol σ_i

Q^i_t is true $\iff$ at step t the machine is in state q_i

$S_{s,t}$ is true $\iff$ at time t square number s is scanned by the tape head

Theorem 2

The following sets are P-reducible to each other in pairs (and hence each has the same polynomial degree of difficulty):
$\{$tautologies$\}$, $\{$DNF tautologies$\}$, D_3 , $\{$subgraph pairs$\}$.

Proof of Theorem 2: Step 1

By the corollary to the first Theorem: $\{$tautologies$\}$, D_3 , $\{$subgraph pairs$\}$ are P-reducible to $\{$DNF tautologies$\}$.

Obviously, $\{$DNF tautologies$\}$ is P-reducible to $\{$tautologies$\}$.
So, also D_3 , $\{$subgraph pairs$\}$ are P-reducible to $\{$DNF tautologies$\}$.

Proof of Theorem 2: Step 2

It is enough to show that: $\{$DNF tautologies$\}$ is P-reducible to D_3 and

D_3 is P-reducible to $\{$subgraph pairs$\}$.

Proof of Theorem 2: Step 3

$\{\text{DNF tautologies}\} \leq_P D_3$

Let A be a proposition formula in disjunctive normal form.
Say $A = B_1 \vee B2 \vee \ldots \vee B_k$, where $B_1 = R_1 \wedge \ldots \wedge R_s$, and each R_i is an atom or negation of an atom and $s > 3$.
Then A is a tautology if and only if A_0 is a tautology where,
$A_0 = (P \wedge R3 \wedge \ldots \wedge R_s) \vee (\neg P \wedge R_1 \wedge R_2) \vee B2 \vee \ldots \vee B_k$, where P is a new atom.

Since we have reduced the number of conjuncts in B_1, this process may be repeated until eventually a formula is found with at most three conjuncts per disjunct.

Clearly the entire process is bounded in time by a polynomial in the length of A.

Proof of Theorem 2: Step 4

It remains to show that D_3 is P-reducible to $\{\text{subgraph pairs}\}$.

Suppose A is a formula in disjunctive normal form with three conjuncts per disjunct.
Thus $A = C_1 \vee \ldots \vee C_k$, where $C_i = R_{i_1} \wedge R_{i_2} \wedge R_{i_3}$, and each R_{i_j} is an atom or a negation of an atom.

Now let G_1 be the complete graph with vertices $\{v_1, v_2, \ldots, v_k\}$, and let G_2 be the graph with vertices $\{u_{i,j}\}$, $1 \leq i \leq k, 1 \leq j \leq 3$, such that $u_{i,j}$ is connected by an edge to $u_{r,s}$ if and only if $i \neq r$ and the two literals $(R_{i,j}, R_{r,s})$ do not form an opposite pair (that is they are neither of the form $(P, \neg P)$ nor of the form $(\neg P, P)$). Thus there is a falsifying truth assignment to the formula A iff there is a graph homomorphism $\varphi : G_1 \to G_2$ such that for each i, $\varphi(i) = u_{i,j}$ for some j.

(The homomorphism tells for each i which of $R_{i,1}, R_{i_2}, R_{i_3}$ should be falsified, and the selective lack of edges in G_2 guarantees that the resulting truth assignment is consistently specified.)

Proof of Theorem 2: Step 5

In order to guarantee that a one-one homomorphism $\varphi : G_1 \rightarrow G_2$ has the property that for each i, $\varphi(i) = u_{i,j}$ for some j, we modify G_1 and G_2 as follows.

We select graphs $H_1, H_2, \ldots, H_k$ which are sufficiently distinct from each other that if G_1' is formed from G_1 by attaching H_i to v_i, $1 \leq i \leq k$, and G_2' is formed from G_2 by attaching H_i to each of $u_{i,1}$ and $u_{i,2}$ and $u_{i,3}$, $1 \leq i \leq k$, then every one-one homomorphism $\varphi : G_1' \rightarrow G_2'$ has the property just stated.
It is not hard to see such a construction can be carried out in polynomial time.
Then G_1' can be embedded in G_2' if and only if $A \notin D_3$.

This completes the proof of theorem 2.

Discussion

It had not been possible to add $\{$isomorphic graphpairs$\}$ and $\{$primes$\}$ to the list of the above Theorem.

NP & co-NP Completeness

Definition NP Complete

A language $L \subseteq \Sigma^*$ is called NP Complete iff:

1) $L \in$ NP

2) Every $L' \in$ NP is P-reducible to L.

Definition co-NP Complete

A language $L \subseteq \Sigma^*$ is called co-NP Complete iff:

1) $L \in$ co-NP

2) Every $L' \in$ co-NP is P-reducible to L.

Bibliography

[1] Stephen A. Cook: The Complexity of Theorem-Proving Procedures, 1971

Acknowledgments

Thanks for you attention ☺