NP-Completeness: and life with it

Models of Computation:

- Random Access Machine
- Turing Machine, multitape if you like, or for that matter nondeterministic
- Recursive functions
- Type 0 Grammars
- Other bounded action machines

Function computable on one is computable on the other.

To simplify things we often deal with simple Boolean functions, or recognizing inputs.

The Class P

If we are just a bit more careful about the models:

- Random Access Machine with fixed sized words
- Turing Machine, multitape if you like, but deterministic

Then the runtimes differ by "at most" a "squaring or so", so a polynomial time algorithm on one machine gives one on the other.

P is the class of recognition problems for which we have algorithms that take worst case time $O(n^k)$ on a TM or RAM with fixed sized word, where n is the number of bits of input and k is a specific constant for each problem.

The Class NP

NP is the class of recognition problems for which we have *procedures* that take time $O(n^k)$ *on a nondeterministic Turing machine*, where n is the number of bits of input and k is a specific constant for each problem.

This translates to "there exists a proof of polynomial length, and a polynomial time algorithm to check that proof"

There are a lot of problems in NP that may not appear to be in P

Cook's Theorem

"Theorem 1: If a set S of strings is accepted by some nondeterministic Turing machine within polynomial time, then S is P-reducible to {DNF tautologies}" "The Complexity of Theorem-Proving Procedures" Proc 3rd Annual ACM Symp. on Theory of Computing (1971) 151-158

P-reducible: P2 P-reducible to P1 Given poly time alg. For P1 then we can use it to solve P2 in poly time

Tautology: Boolean formula that is true for all assignments of variables. (Similarly "Satisfiable")

DNF: Disjunctive Normal Form .. "or" together clauses consisting of "and" of symbols and their negations

Similarly CNF: Conjunctive Normal Form

Cook's Proof

Reformulate: Map a computation in a nondet. TM (of known maximum runtime) to a Boolean formula in CNF. The formula is satisfiable iff the TM accepts its input

The proof is mechanical … great insight to get it … but no technically hard parts.

Sketch: TM input w, of length n

Runtime p(n), a specific polynomial

If TM accepts w ∃ at least one sequence of TM id's

Q_0, Q_1, … Q_q ($q \leq p(n)$) describing

- Tape contents
- Head Position
- State

Cook's Proof- Definitions

A few variables help:

C<i,j,t> is 1 iff i^{th} tape cell contains x_j at time t $\{1 \le i \le p(n), 1 \le j \le m, 0 \le t \le p(n)\}$

S<k,t> is 1 iff TM in state q_k at time t $\{1 \le k \le s, 0 \le t \le p(n)\}$

H<i,t> is 1 iff head is scanning tape square t at time i $\{ 0 \le t \le p(n)\}$

So $O(p(n)^2)$ variables.

Picky point: times and tape squares represented in binary, so an extra factor of lg n in description size.

Useful formula: Exactly one true

$U(x_1,\ldots x_r) = (x_1+x_2+\ldots x_r)\ (\Pi_{i\neq j}(\neg x_i+\neg x_j))$ {size $O(r^2)$}

All Seven parts

We now construct one big formula, ABCDEFG, that is true iff the TM accepts the input:

First 4 says "it looks like a Turing machine to me"

A asserts that TM scans exactly one square at time t, $\forall$t. So A = ΠA_t

$$A_t = U(H<1,t>, H<2,t> .., H<p(n),t>)$$

B asserts each tape square has exactly 1 symbol in it at time t, $\forall$t.

C asserts TM is in 1 state at time t, $\forall$t.

D asserts at most one tape square (the correct one) changes from 1 step to the next

$$D = \Pi_{i,j,t} [C<i,j,t> \equiv C<i,j,t+1> + H<i,t>]$$

And It's the Right TM

E asserts the moves of the TM from one ID to the next are allowed by the next move function

E_{ijkt} asserts at least one of:

- i^{th} cell does not contain symbol j at time t
- Head is not on i^{th} cell at time t
- M is not in state k at time t
- ID of TM obtained from previous ID by legal transformation

F asserts initial conditions are OK, F=

$S\langle 1,0\rangle\ H\langle 1,0\rangle\ \Pi_{1\le i\le n}\ C\langle i,j_i,0\rangle\ \Pi_{n<i\le p(n)}\ C\langle i,1,0\rangle$

j_i indicate i^{th} input symbol

G asserts TM eventually enters accept state, i.e. $G=S\langle s,p(n)\rangle$ where q_s is accept state.

Indeed entire formula size is $O(p(n)^3)$

Then

Subgraph isomorphism also as hard as SAT.

Karp (’72) showed about a dozen other problems were in the same class

Then … the term NP-Complete

Note: NP and CoNP, hence NP-Complete and Co-NP-Complete

NP-hard

Then and 1000’s of problems

And what to do with them