The Satisfiability Problem

Cook's Theorem: An NP-Complete Problem

Restricted SAT: CSAT, 3SAT

Boolean Expressions

- Boolean, or propositional-logic expressions are built from variables and constants using the operators AND, OR, and NOT.
  - Constants are true and false, represented by 1 and 0, respectively.
  - We'll use concatenation (juxtaposition) for AND, + for OR, - for NOT, unlike the text.

Example: Boolean expression

- (x+y)(-x + -y) is true only when variables x and y have opposite truth values.
- Note: parentheses can be used at will, and are needed to modify the precedence order NOT (highest), AND, OR.

The Satisfiability Problem (*SAT*)

- Study of boolean functions generally is concerned with the set of *truth assignments* (assignments of 0 or 1 to each of the variables) that make the function true.
- NP-completeness needs only a simpler question (SAT): does there exist a truth assignment making the function true?

Example: SAT

- (x+y)(-x + -y) is satisfiable.
- There are, in fact, two satisfying truth assignments:
  1. x=0; y=1.
  2. x=1; y=0.
- x(-x) is not satisfiable.

SAT as a Language/Problem

- An instance of SAT is a boolean function.
- Must be coded in a finite alphabet.
- Use special symbols (,), +, - as themselves.
- Represent the i-th variable by symbol x followed by integer i in binary.

SAT is in **NP**

- There is a multitape NTM that can decide if a Boolean formula of length n is satisfiable.
- The NTM takes $O(n^2)$ time along any path.
- Use nondeterminism to guess a truth assignment on a second tape.
- Replace all variables by guessed truth values.
- Evaluate the formula for this assignment.
- Accept if true.

Cook's Theorem

- SAT is NP-complete.
  - Really a stronger result: formulas may be in conjunctive normal form (CSAT) – later.
- To prove, we must show how to construct a polytime reduction from each language L in **NP** to SAT.
- [Details omitted – see book, or slides for CIRCUIT-SAT.]

Conjunctive Normal Form

- A Boolean formula is in *Conjunctive Normal Form* (CNF) if it is the AND of *clauses*.
- Each clause is the OR of *literals*.
- A literal is either a variable or the negation of a variable.
- Problem *CSAT* : is a Boolean formula in CNF satisfiable?
- Example: (x + -y + z)(-x)(-w + -x + y + z)

NP-Completeness of CSAT

- Cook's proof (from 1971) can be modified to produce a formula in CNF.

The Complexity of Theorem-Proving Procedures

Stephen A. Cook

University of Toronto

Summary

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.

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 reduced 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 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

Other NP-Complete Problems

- There are a lot of other problems that can be shown to be NP-complete via a reduction from SAT. E.g., [Karp, 1972]:

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.

Richard Karp's Original Set

- A set of polynomial-time reductions.

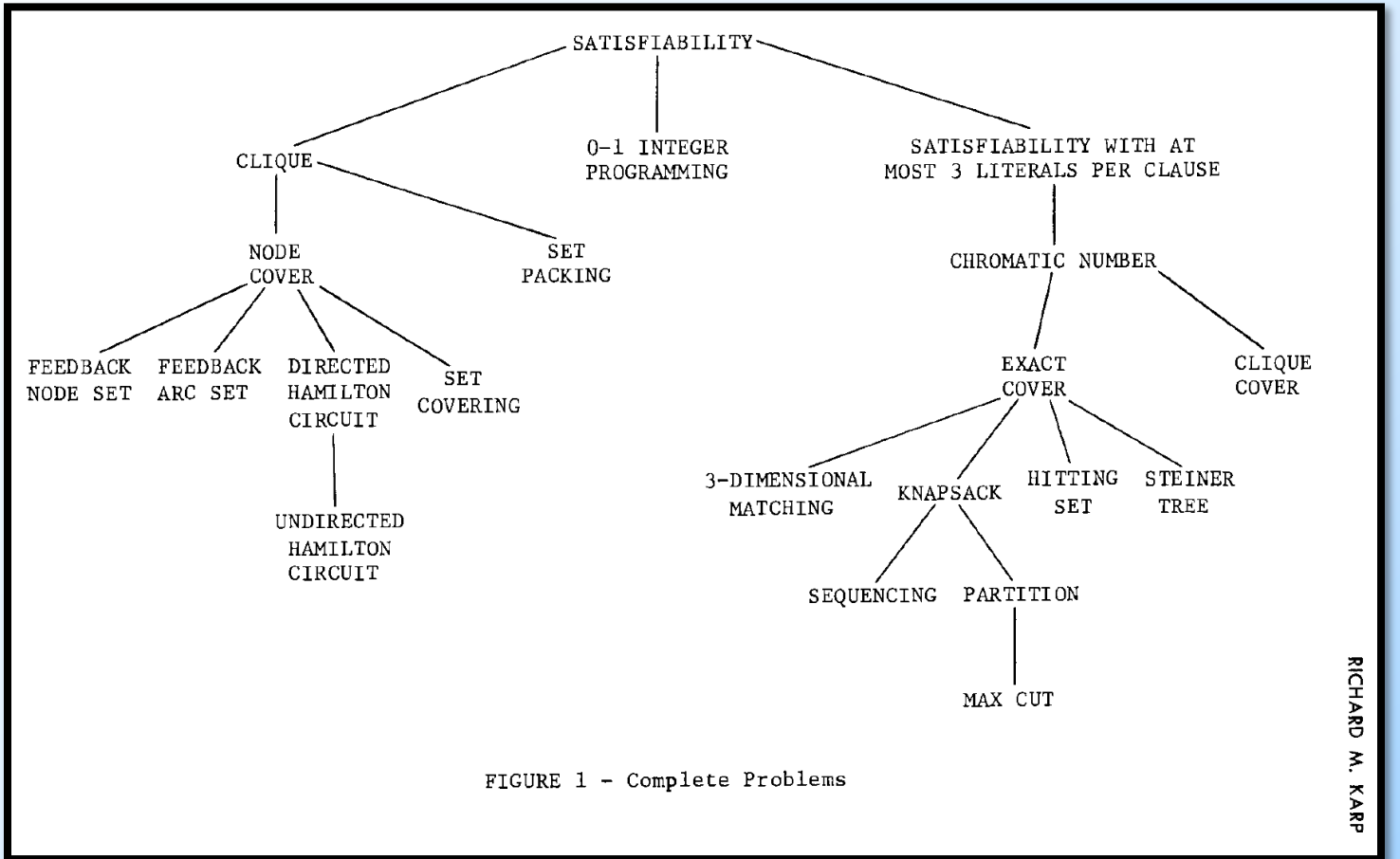


FIGURE 1 - Complete Problems

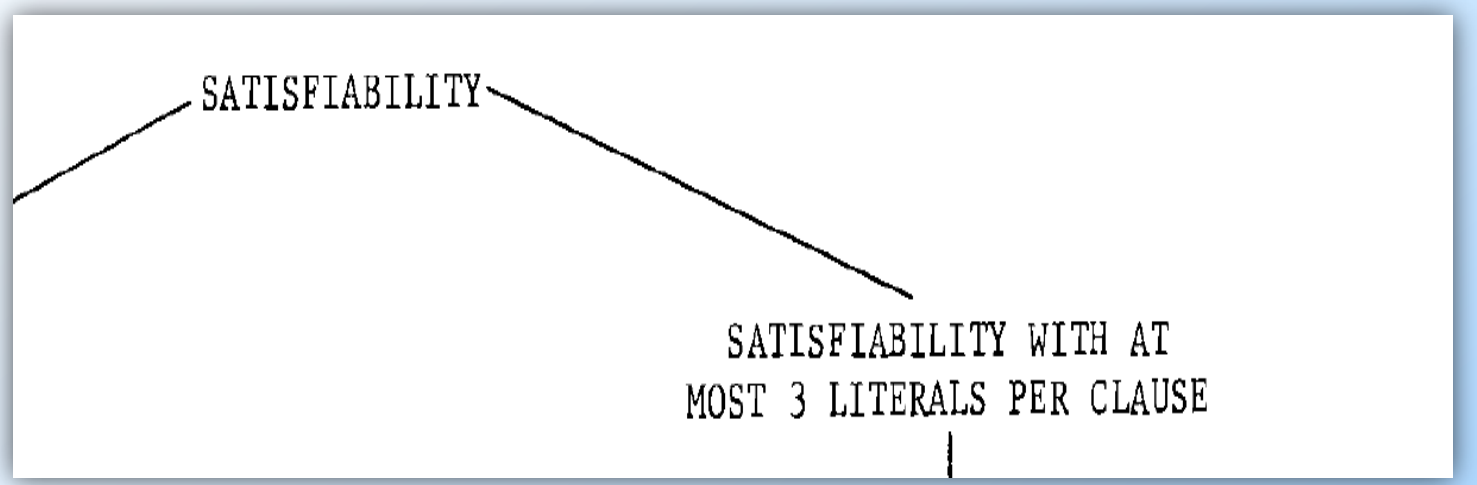
k-SAT

- If a boolean formula is in CNF and every clause consists of exactly k literals, we say the boolean formula is an instance of *k-SAT*.
  - Say the formula is in *k-CNF*.
- Example: 3-SAT formula

(x + y + z)(x + -y + z)(x + y + -z)(x + -y + -z)

k-SAT Facts

- 2-SAT is in **P**.
- 3-SAT is NP-complete.



Proof: 2SAT is in **P** (Sketch)

- Pick an assignment for some variable, say x = true.
- Any clause with –x forces the other literal to be true.
  - Example: (-x + -y) forces y to be false.
- Keep seeing what other truth values are forced by variables with known truth values.

Proof – (2)

- One of three things can happen:
  1. You reach a contradiction (e.g., z is forced to be both true and false).
  2. You reach a point where no more variables have their truth value forced, but some clauses are not yet made true.
  3. You reach a satisfying truth assignment.

Proof – (3)

- Case 1: (Contradiction) There can only be a satisfying assignment if you use the other truth value for x.
  - Simplify the formula by replacing x by this truth value and repeat the process.
- Case 3: You found a satisfying assignment, so answer "yes."

Proof – (4)

- Case 2: (You force values for some variables, but other variables and clauses are not affected).
  - Adopt these truth values, eliminate the clauses that they satisfy, and repeat.
- In Cases 1 and 2 you have spent $O(n^2)$ time and have reduced the length of the formula by ≥ 1, so $O(n^3)$ total.

3SAT

- This problem is NP-complete.
- Clearly it is in **NP:**
  1. Guess an assignment of true and false to the variables
  2. Test whether the Boolean formula is true.

3SAT – (2)

- We need to reduce every CNF formula F to some 3-CNF formula that is satisfiable if and only if F is.
- Reduction involves introducing new variables into long clauses, so we can split them apart.

Reduction of CSAT to 3SAT

- Let $(x_1+\ldots+x_n)$ be a clause in some CSAT instance, with $n \geq 4$.
  - Note: the x's are literals, not variables; any of them could be negated variables.
- Introduce new variables $y_1,\ldots,y_{n-3}$ that appear in no other clause.

CSAT to 3SAT – (2)

- Replace $(x_1+\ldots+x_n)$ by
  $(x_1+x_2+y_1)(x_3+y_2+ -y_1) \ldots (x_i+y_{i-1}+ -y_{i-2})$
  $\ldots (x_{n-2}+y_{n-3}+ -y_{n-4})(x_{n-1}+x_n+ -y_{n-3})$
- If there is a satisfying assignment of the x's for the CSAT instance, then one of the literals x_i must be made true.
- Assign y_j = true if $j < i-1$ and y_j = false for larger j.

CSAT to 3SAT – (3)

- We are not done.
- We also need to show that if the resulting 3SAT instance is satisfiable, then the original CSAT instance was satisfiable.

CSAT to 3SAT – (4)

- Suppose $(x_1+x_2+y_1)(x_3+y_2+ -y_1)$ … $(x_{n-2}+y_{n-3}+ -y_{n-4})(x_{n-1}+x_n+ -y_{n-3})$ is satisfiable, but none of the x's is true.
- The first clause forces y_1 = true.
- Then the second clause forces y_2 = true.
- And so on … all the y's must be true.
- But then the last clause is false.

CSAT to 3SAT – (5)

- There is a little more to the reduction, for handling clauses of 1 or 2 literals.
- Replace (x) by $(x+y_1+y_2)$ $(x+y_1+ -y_2)$ $(x+ -y_1+y_2)$ $(x+ -y_1+ -y_2)$.
- Replace (w+x) by (w+x+y)(w+x+ -y).
- Remember: the y's are different variables for each CNF clause.

CSAT to 3SAT Running Time

- This reduction is surely polynomial.
- In fact it is linear in the length of the CSAT instance.
- Thus, we have polytime-reduced CSAT to 3SAT.
- Since CSAT is NP-complete, so is 3SAT.