A TERMINOLOGICAL PROPOSAL

D. E. Knuth

While preparing a book on combinatorial algorithms, I felt a strong need for a new technical term, a word which is essentially a one-sided version of polynomial complete. A great many problems of practical interest have the property that they are at least as difficult to solve in polynomial time as those of the Cook-Karp class NP. I needed an adjective to convey such a degree of difficulty, both formally and informally; and since the range of practical applications is so broad, I felt it would be best to establish such a term as soon as possible.

The goal is to find an adjective x that sounds good in sentences like this:

The covering problem is x .
It is x to decide whether a given graph has a Hamiltonian circuit.
It is unknown whether or not primality testing is an x problem.

We also probably need the associated noun "x-ness" or "x-hood" as appropriate. Here x is not necessarily to imply that a problem is in NP, merely that everything in NP can reduce to x . For example, let's imagine the situation a few months ago before Pratt showed that primality testing is in NP; the third sentence above does not call in question whether or not primality testing is in NP, while if I said "It's unknown whether or not primality testing is polynomial complete" I imply that there is uncertainty either about primality in NP or about NP reducing to primality.

In my lectures at Oslo last year I used x = "hard". But this turned out to be unsatisfactory because the word "hard" is so common it is unclear when it is being used in a technical sense. I thought of "tough" because it is informal enough that it is pretty clear when technical usage is intended; however, it conflicts with some graph-theoretic terminology and doesn't sound quite right. When non-specialists talk about difficult computational problems, they unfortunately call them "combinatorial", which of course is a completely unwelcome usage.

I think many people are interested in how terminology gets started, and it seems that it often happens by accident, with the originator having no idea that he is fixing a name for all time. That certainly was the case with me when I defined the term " LR(k) ", I had no idea that anyone else would ever use it; and I imagine that misleading terms like 'statement' (in programming languages), 'context-free languages', 'AVL trees', etc., were never considered very seriously for their appropriateness when they were first proposed.

In this case I wanted to try to do something better, to get a large number of qualified people helping to decide on a name before the first publication. So I did two things: (1) I got out my copy of Roget and my unabridged dictionary, and found a set of candidates for x .
(2) I wrote to about 30 people asking them to vote on these choices. [My sincere apologies to all readers who I forgot to include in the balloting.]

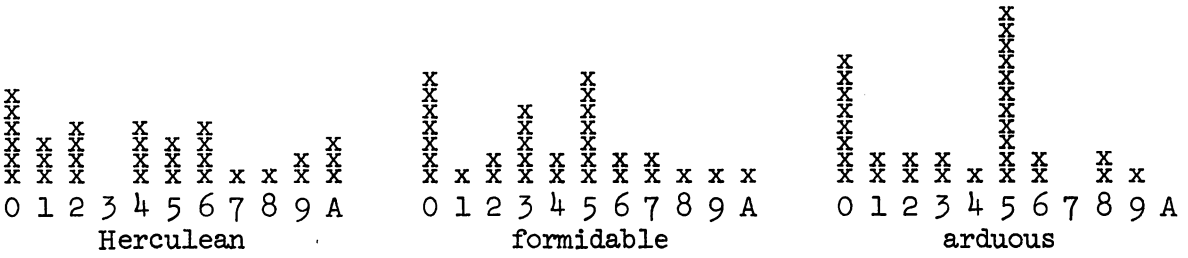
The three choices I listed were "Herculean", "formidable", and "arduous". I asked all voters to assign a real number between 0 and 1 to each term, indicating the degree to which they approved of it. (Thus,

0 meant complete disagreement, 1 meant complete agreement, and 1/2 or more meant "would use if it became standard".) I also left space for write-in votes.

After mailing the ballots, I personally came to the conclusion that "Herculean" would be best; I tried using it for a week, and found that it felt comfortable in all the necessary contexts, and in fact I began to like it. Then a week later the returns began to come in, and I was reminded of the fact that nobody but me has my tastes; I had forgotten that everybody else in the world is hopeless when it comes to terminology (and that they think the same about me). The first week's returns were comparatively few, but there was a strong preference for "formidable" and only very weak responses for "Herculean". I had favored Herculean partly because it translates immediately into all the prominent languages, but I found that even my foreign-speaking correspondents didn't like it. That night I met Dick Karp socially, and we decided to start saying "formidable". This worked well in conversation and seemed to be a good solution. I went home and replaced 'Herculean' by 'formidable' in my files.

The next week, many more ballots arrived, and I found to my chagrin that "formidable" was losing its early popularity. I began to wonder how chemists ever got any of their horrible terms adopted, especially their names for the four constituents of DNA code. And I also began to wonder whether my task of assessing the results of votes was going to be Herculean, formidable, or merely arduous.

The final results are shown on the histograms below, showing how many of the 31 respondents assigned numbers in various ranges for each word. [I've multiplied the range by 10, so that the number of x's above digit n indicates the number of votes in the range $n/10 \leq v < (n+1)/10$. It wasn't trivial to make these charts, since Mike Harrison used $(\sqrt{5}-1)/2$ as one choice, and Al Aho used ε and ε^2. I presume Al meant ε to be a small positive real, not a large ordinal.]



The distributions are remarkably different, 'Herculean' being rather uniform and 'arduous' very peaked. But one thing was perfectly clear: all three words fared rather badly. Only 'arduous' was able to get $\geq .5$ from a majority of the voters, and this majority (16 votes to 15) was hardly conclusive.

Then I applied a secret weighting factor to all the ballots, based on approximately how many papers related to this subject I felt each particular voter would be writing, in the next few years, and how much influence on computer science students he has, etc. [Naturally I assigned the weights

before looking at the ballots.] It's preposterous to do such a thing in a democracy, but I did it. The resulting weighted average scores were

Herculean	.369
formidable	.373
arduous	.353

In other words, very low. [I'll bet that the term 'polynomial complete' would have fared even worse in the early days; but I'm just trying to heal my wounded feelings when I say this.]

Fortunately, there was a ray of hope remaining, namely the space for write-in votes. I received very many ingenious suggestions; indeed, the write-ins proved conclusively that creative research workers are as full of ideas for new terminology as they are empty of enthusiasm for adopting it.

The write-in votes were so interesting, I'd like to discuss them here at some length. First, there were several other English words suggested:

impractical	intractable
bad	costly
heavy	obdurate
tricky	obstinate
intricate	exorbitant
prodigious	interminable
difficult	

Also, Ken Steiglitz suggested "hard-boiled", in honor of Cook who originated this subject. Al Meyer tried "hard-ass" (hard as satisfiability). [You can see what I mean about creative researchers.]

Bob Floyd suggested Sisyphean instead of Herculean, since the problem of Sisyphus was time-consuming while Hercules needed great strength. The difficult NP problems seem to be more time-consuming than energy-spending, so this may be a better term. On the other hand, Sisyphus never finished his task, so we could use this more appropriately for unsolvable problems. A similar remark applies to Tithonian. I prefer Ulyssean to these, because Ulysses was noted for his persistence and he also finished. Incidentally, Al Aho said he lauds my intent in "trying to clean the theory of computing of its Augean terminology". So, we find classicists amongst us.

The next group of suggestions was based on acronyms. Shen Lin thought of calling them PET problems, as he likes to work on them (e.g. the traveling salesman problem) in spite of their difficulty. He points out that PET stands for "probably exponential time". But if this gets proved, PET stands for "provably exponential time". And if the proof goes the other way, it stands for "previously exponential time".

There's also another Al Meyerism, to let x = GNP (greater than or equal to NP in difficulty, with the possibility of costing more than the GNP to resolve). Or, x = XS (seeming to demand an exhaustive search which requires an excess of time).

The most tantalizing suggestions I received were based on newly-coined words, taken from appropriate classical roots. This, after all, is the way biologists, etc., get nearly all of their high-faluting terms. Mike Paterson contributed two nice ones:

exparent (literally, seeming outside; also joc. fr. exponential + apparent)
and perarduous (since 'per' means 'through, in space or time' and/or 'completely, extremely').

Al Meyer tried also supersat, meaning greater than or equal to satisfiability. Another excellent one comes from Ed Reingold and his classicist friend Howard Jacobson:

polychronious

This pleasant word curiously appears in Webster's 2nd unabridged, but not the 3rd or in the Oxford English or apparently any other dictionary. The definition given in Webster's is quite appropriate: "enduringly long; chronic (rare)". However, to my ears the word polychronious actually implies polynomial time rather than the contrary.

I can see many words in the above list that I wouldn't mind using, but none that I can see becoming standard. There was, however, one further class of write-in votes, based on hyphenated compound words that relate strongly to the presently entrenched terminology, but which clearly dominate what we now are saying. Since these words were proposed as write-in candidates by quite a few people, apparently acting independently of each other, I believe that the solution to the problem lies here.

The "winning" write-in vote is the term NP-hard, which was put forward mainly by several people at Bell Labs, after what I understand was considerable discussion. Similar if not identical proposals were made by Steve Cook, by Ron Rivest, and in earlier publications by Sakti Sahni. This term is intended for use together with another new one, NP-complete, which abbreviates 'polynomial complete' and is at the same time more exact.

Motivation for these terms is easy to describe to a novice, once he understands NP, namely

NP-hard means as hard as the most difficult problem in NP.
NP-complete means representative of the complete class NP with respect to difficulty.

In recent weeks, Karp and I have tried this terminology, and it seems to stand up well in practice. As Jeff Ullman remarked in his letter, "The natural thing to do is substitute 'hard' or some other word for 'complete' so that 'blah hard' means 'blah complete or worse'."

The strength of support for this write-in vote makes it reasonable to propose it for immediate adoption by all workers in the field. I will conclude this note by examining such a proposal critically and concluding that it survives all the attacks I can muster.

First, are the terms well-defined? Answer: Well, not by the above discussion, but we can make them so. One of the things I learned from the letters received was that Cook's original definition of polynomial reducibility is not known to be the same as the one Karp used to relate

so many combinatorial problems to each other. Since the latter is simpler to deal with and supports all the constructions which I believe are of interest to real-world programmers, I propose to make the following explicit definitions (following Karp):

A problem L is a subset of the strings on some finite alphabet (i.e., a problem is a language).

A polynomial-bounded transduction f is a function $\Sigma^* \to \Sigma'^*$, where Σ and Σ' are finite alphabets, such that, for some integer k, the output f(x) is computable in at most $(|x|+2)^k$ steps for all x, on (say) a one-tape Turing machine.

If L and L′ are problems, we say L reduces to L′ if there is polynomial-bounded transduction f such that $x \in L$ if and only if $f(x) \in L'$.

The satisfiability problem S is the set of all strings α in (say) the context-free language A over the alphabet $\{(,),\wedge,\vee,\neg,v,'\}$ defined by syntax and integer-valued semantics

$A \to (C)$	$m(A) = m(C)$
$A_1 \to A_2 \wedge (C)$	$m(A_1) = m(A_2)m(C)$
$C \to L$	$m(C) = m(L)$
$C_1 \to C_2 \vee L$	$m(C_1) = m(C_2) + m(L)$
$L \to A$	$m(L) = f(m(A))$
$L \to \neg A$	$m(L) = 1 - f(m(A))$
$A \to v$	$m(A) = 1$
$A_1 \to A_2'$	$m(A_1) = m(A_2) + 1$

such that there exists a function $f: N \to \{0,1\}$ which makes $m(A) > 0$.

A problem L is NP-hard iff S reduces to L; it is in NP iff L reduces to S; it is NP-complete iff both conditions hold.

Thus, there exists a way to define the terminology precisely. Note that the definition relies on Cook's theorem to relate NP to 'nondeterministic polynomial time', so that S-hard and S-complete look like better terms in the sense of this definition.

A more general definition would say that, for any class C of problems, we define C-hard to mean " L′ reduces to L for all $L' \in C$ "; and C-complete is C-hard plus " L reduces to L′ for some $L' \in C$ ". We get immediate metatheorems from the fact that "reduces to" is transitive and reflexive, e.g.

If L is C-hard and L reduces to M then M is C-hard.

If L is C-complete then L reduces to M if and only if M is C-hard.

The class C of context-sensitive languages is just one interesting example.

Cook's definition of reducibility was different; he said that L reduces to L′ iff L is accepted in polynomial time by a Turing machine extended with the capability of deciding membership in L′ in one step. This definition corresponds of course to similar definitions with respect to recursive functions and unsolvable problems. We might call this Turing-reducibility or Cook-reducibility. However, I must admit that I don't see a critical distinction here. Unless I'm mistaken, it is possible to prove the following theorem, by extending Cook's original construction slightly:

If L reduces to S and L′ Cook-reduces to L on a nondeterministic Turing machine, then L′ reduces to S .

(The clauses generated for those instants of time when L problems are to be solved are replaced by clauses corresponding to the reduction of L to S .) If we set L = S we get "Cook-reducibility to S implies nondeterministic Cook-reducibility to S implies reducibility to S ". But reducibility obviously implies Cook-reducibility, so the three concepts are the same at this level of the hierarchy unless I've missed something.

Even if my supposed proof of the above theorem breaks down when I get around to writing the details, I would argue that it is best to use the simpler definitions in connection with notions that will be used by non-automata-theorists. There is an enormous literature on combinatorial algorithms applied to practical problems, and a large body of practical people who get excited about them but not about the technical details of what can happen in weird cases on curious abstract machines. For more technical discussions, the terminology issue is comparatively unimportant, even though the concepts are likely to be important in the total theory, since fewer people are involved; but NP-hard problems hit lots of people, and that's why I began searching for a special term.

In other words, I don't consider it a major goal to invent completely descriptive terminology for every conceivably interesting question of the type considered here; the major goal is to have a good term to use for the masses, in the one case which experience shows is almost omnipresent. To say NP-hard actually smacks of being a little too technical for a mass audience, but it's not so bad as to be unusable; fortunately the term does easily generalize to a lot of other cases considered by automata theorists, so it appears to be a good compromise. When more technicalities are introduced, there is less need for a universally accepted term or notation. Although it's very interesting to consider, e.g., log-space reductions, I don't think it's necessary to come up with a special short name for them.

John Hopcroft's suggestion was "NP-time" instead of NP-hard, with the corresponding "NP-space" (which equals P-space). I considered this seriously but decided that it was not satisfactory (mostly because 'time' and 'space' are nouns). The words NP-long or NP-big might be OK; but really, as I have said, the practical problems of mass interest are all associated with one case, and terminology should be optimized for that case.

'NP-hard' meets Mike Fischer's objection to my original words which gave an absolute meaning to a relative quantity; he said that calling some problems formidable, is as bad as deciding to say that "big" means "greater than 17". [I don't agree here; after all, the word "positive" has merit, and NP-hard problems are just those above the 0-th level of difficulty.] Mike was afraid of running out of terms; he said that problems of double exponential difficulty might be called "hopeless", and triple exponential "disastrous", but then he was at a loss for words.

One final criticism (which applies to all the terms suggested) was stated nicely by Vaughan Pratt: "If the Martians know that P = NP for Turing Machines and they kidnap me, I would lose face calling these problems 'formidable'." Yes; if P = NP , there's no need for any term at all. But I'm willing to risk such an embarrassment, and in fact I'm willing to give a prize of one live turkey to the first person who proves that P = NP .

****

PAPERS FORM THE HIGH TATRAS CONFERENCE (Cont'd)

M. Nekvinda (Liberec)
On the complexity of countable functions

P. Ng, R. T. Yeh (Austin, Texas)
Tree transformations via finite recursive transition machines

I. Peák (Salgótarján), Nguyen Quy Khang (Hanoi)
On endomorphism semigroup of nilpotent automata

V. Rajlich (Prague)
Relational structures and dynamics of certain discrete systems

W. P. de Roever (Amsterdam)
Operational and mathematical semantics for recursive polyadic program schemata

B. Rovan (Bratislava)
Necessary conditions for containment of principal (semi-) AFL with bounded generators

Л. Г. Самойленко (Киев)
О методе построения и свойствах контекстнозависимых грамматик и языков
(Construction and properties of context-sensitive grammars and languages)

D. A. Simovici (Iaşi)
On some measures on free semigroups induced by semiautomata

A. Skowron (Warsaw)
Machines with input and output

O. Štěpánková, I. M. Havel (Prague)
Some results concerning the situation calculus

H. G. Stork (Darmstadt)
A note on improving input strings for paging machines

P. Strnad (Liberec)
Turing machine recognition

I. H. Sudborough, A. Zalcberg (Evanston, Illinois)
On families of languages defined by time bounded random access machines