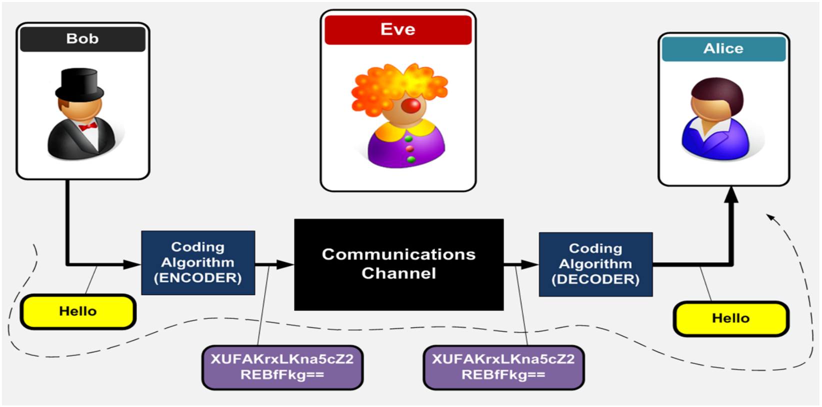
How Useful is the Word Problem?

Robert Gilman
Stevens Institute of Technology

Jahrestagung der Deutschen Mathematiker Vereinigung
Hamburg 21-25.9.2015

Introduction


Bob
Eve
Alice
Coding Algorithm (ENCODER)
Communications Channel
Coding Algorithm (DECODER)
Hello
XUFAKrxLKna5cZ2 REBfFkg==
XUFAKrxLKna5cZ2 REBfFkg==
Hello

RSA depends on the difficulty of factoring certain products of two primes.

22601385262034057849416540486101975135080389157197767183
21197768109445641817966676608593121306582577250631562886
67697044807000181114971186300211248792819948748206607013
1066586646083327982803560379205391980139946496955261

1. Factoring is in **NP**; is it in **P**?
2. If no, how do we sample the difficult instances?
3. If yes, might there be sufficiently difficult instances anyhow?

Public key systems rely on underlying algebraic problems.

A public key – private key pair is equivalent to an instance of this problem.

Problem instances must be feasible to generate, and with overwhelming probability hard to solve without the private key. Standard complexity does not help much here.

Thus we need a theory of complexity that will enable us to state and prove that a certain computation is intractable in virtually every case. For example, a block-encoding system is safe if any algorithm for key determination will terminate in practical time only on $O(2^{-n})$ of the cases. We are very far from creation of such a theory, especially at the present stage when $P = NP$ is not yet settled.

Michael Rabin
Turing Award Lecture
1976

Complexity cores are collections of hard instances. Can we sample them to get hard instances?

The question of whether or not complexity cores of practical computational problems can be efficiently sampled seems to have been open for forty years.

Decision problems involving enumeration of all Turing machines are not practical.

Word problems for finitely presented groups are practical.

Word problems for recursively presented groups which are not finitely presented are impractical.

Theorem

There is a computational problem with a computable complexity core which can be efficiently sampled. Namely a finitely presented group whose word problem has a complexity core C such that

- *Any algorithm succeeds only on an exponentially negligible subset of C;*
- *C is computable and can be sampled efficiently.*

Hard Instances

How rare are hard instances of practical problems? How do we sample them?

The simplex algorithm is exponential time but the hard instances do not show up in real life.

The word search problem of any finitely presented group is polynomial time except on a negligible set of words.

It might be that $\mathbf{P} \neq \mathbf{NP}$, but hard instances are rare.

Challenger–Solver Games

Challenger repeatedly picks instances of a given problem Q problem, and Solver tries to solve them.

Even if Q has very hard instances, Solver may have easy life: It may take a very long time to find a hard instance.

If Solver almost always succeeds in time polynomial in the time it takes challenger to generate an instance, then Solver wins. Otherwise Challenger wins.

Complexity Cores

Theorem (Lynch 1975)

If Q is a computable problem not in **P***, then there is an infinite computable set X such that any* **P***-time algorithm for Q fails on all but finitely many instances of Q.*

Let $A_1, A_2, \ldots$ be the (partial) algorithms for Q and $I = i_1, i_2, i_3, \ldots$ the inputs to Q.

Initialize X to be empty.

Let i be the first input for which A_1 fails to converge in time $n = |i|$, and add i to X.

Let i' be the first subsequent input for which A_1 and A_2 both fail to converge in time $2n^2$, and add i' to X.

Etc.

If Q is computable, then X can be computable too.

Every recursive Q not in **P** has a core recognizable in subexponential time. Every $Q \in \mathbf{NP} - \mathbf{P}$ has a core with super polynomial growth.

P can be replaced by other complexity classes.

The construction of X is impractical; Lynch raises the question of practical construction.

Genericity

Consider formal languages, i.e., sets of words over a finite alphabet Σ, i.e., subsets of Σ^*.

B_n is the set of all words in Σ^* of length at most n.

A language $L \subset \Sigma^*$ is

1. Generic if $\lim_{n\to\infty} \frac{|L\cap B_n|}{|B_n|} = 1$.
2. Exponentially generic if the convergence is exponentially fast. In other words $\frac{|L\cap B_n|}{|B_n|} \geq 1 - cr^{-n}$ for some $c > 0$ and $r > 1$.
3. Negligible if its complement is generic.
4. Exponentially negligible if its complement is exponentially generic.

Generic Properties of Groups

If G has an infinite cyclic quotient, then the word problem is solvable in linear time on a generic set of inputs.

If G has a free quotient of rank at least 2, then the word problem is solvable in linear time on an exponentially generic subset.

If G is finitely presented, then a generic van Kampen diagram has diameter proportional to the log of the length of the word it defines. A generic word defining the identity can be checked in polynomial time. (Here genericity is defined using van Kampen diagrams.)

If G is hyperbolic, then for each k, the set of k-tuples in Σ^* which generate undistorted free subgroups is exponentially generic, and the corresponding membership problem is solvable in cubic time.

If G is any amenable group with unsolvable word problem then the word problem is not solvable on any exponentially generic subset of words.

Definition

A modified complexity core for a computational problem Q is an infinite set of inputs such that any algorithm for Q fails on all but an exponentially negligible set of inputs.

Algorithmically finite groups

Let G be a finitely generated group, Σ a finite alphabet, and $\Sigma \to G$ a choice of generators. Σ^* is the free monoid of all group words.

Definition

G is algorithmically finite if no infinite recursively enumerable subset of Σ^* projects injectively to G.

G is a torsion group.

If the word problem is decidable on $L \subset \Sigma^*$, then the projection of L is finite.

If G is non-amenable, then any algorithm for the word problem succeeds only on an exponentially negligible subset of Σ^*.

Theorem (Myasnikov, Osin)

There exists an infinite finitely generated recursively presented non-amenable algorithmically finite group.

Enumerate recursively enumerable languages $S \subset \Sigma^*$.

From each infinite S pick two words v, w and impose the relation $v = w$.

Do this in such a way that the Golod-Shafarevich Theorem applies.

Are there any finitely presented infinite algorithmically finite groups?

Conclusion

Let G be an infinite recursively presented algorithmically finite group. G is a subgroup of a finitely presented group K.

Let Σ be a set of generators for K with generators $\Delta \subset \Sigma$ for G.

Lemma

Δ^* is a modified complexity core for the word problem of K.

Proof: Let A be an algorithm for the word problem. The restriction of A to Δ^* succeeds only on an exponentially negligible subset.

Corollary

Challenger wins.

For a given n Challenger picks words from Δ^* uniformly at random in linear time. Solver succeeds with probability at most cr^{-n} for constants $c > 0$ and $r > 1$.

Question: Are there analogous results at lower complexity levels?