On the development of Emil Post's ideas in structural complexity theory

Zenon Sadowski

Institute of Mathematics,
University at Białystok,
15-257 Białystok, ul. Akademicka 2, Poland
e-mail: sadowski@math.uw.bialystok.pl

Abstract. Structural complexity theory is one of the branches of computer science. In this survey we will show how notions of a complete r.e. set and many-one reducibility from Post's 1944 paper influenced the study of NP-complete languages in structural complexity theory.

1 Introduction

Emil Post's 1944 address to the American Mathematical Society, "Recursively enumerable sets of positive integers and their decision problems" [2], had a great impact on later research in recursive function theory and theoretical computer science. The line of research which started with this paper culminated in the invention of **NP**-completeness, the central notion of structural complexity theory and the favorite paradigm of the whole field of computer science.

In Post's paper, in recursive function theory and structural complexity theory the concept of decision problem is the main subject of research. Connected with every decision problem are an infinite family of instances and a question. For any instance we ask this question and expect the answer YES or NO.

An example of decision problem is the Hamilton cycle problem (see [1]).
Instance: A graph G=(V,E) with a finite set of nodes and a set E of edges, which are pairs of nodes.
Question: Is there a cycle that visits all nodes of the graph G once?

Every problem is fully determined by the set of its YES instances. Coding these YES instances by strings or by natural numbers we can treat problems as sets of words (formal languages) or as subsets of natural numbers. The first approach appears in Post's paper and in recursive function theory, the second one is typical for complexity theory.

2 Classification of recursively enumerable sets

In his paper Emil Post introduced the concept of reductions between problems, defined complete problems and started the classification of recursively enumerable sets. In this section we will consequently code problems, following the approach from Post's paper, as subsets of natural numbers.

A nonempty set of natural numbers is recursively enumerable provided there is an automatic method (a recursive function) for listing out its members, one after the other. This does not imply that there is a decision method for determining membership in the set. Sets possessing such a decision method are called recursive.

There is another equivalent definition of recursively enumerable sets. A set of natural numbers is recursively enumerable if and only if every number belonging to this set has a certificate witnessing its membership in the set. Numbers which are not elements of the set possess no such certificates.

Let A, B be two sets of natural numbers. The set A is reducible to the set B if there exists an effective method which determines for each natural number n a natural number m such that m is or is not in B according as n is or is not in A. Emil Post studied reductions between recursively enumerable sets. He considered five types of reducibility: 1) many-one reducibility, 2) one-one reducibility, 3) truth-tables reducibility, 4) bounded truth-tables reducibility and 5) Turing reducibility.

The notion of reducibility orders recursively enumerable sets. The maximal elements of this partial order Post named complete recursively enumerable sets. Every recursively enumerable set is reducible to a complete set. Post proved the following theorem:

Theorem 1. (Post)
There exist complete recursively enumerable sets with respect to one-one reducibility (many-one, truth-tables, bounded truth-tables, Turing reducibility).

Whether there exist recursively enumerable sets which are not recursive and which are not complete was the main question considered by Post in paper [2]. He proved that with respect to many-one, one-one, truth-tables, and bounded truth-tables reducibility such sets exist. The problem whether there exists a recursively enumerable set which is not recursive and which is not complete with respect to Turing reducibility was solved over ten years later by R. Friedberg and independently by A. Muchnik.

3 Classification of NP-languages

By an alphabet we mean any finite set of symbols, as denoted by Σ. A finite sequence of symbols from Σ is named a word. By a formal language we mean any set of words over Σ. In this section decision problems are coded as formal languages (sets of codes of their YES-instances).

In structural complexity theory the basic models of computation are deterministic and nondeterministic Turing machines (see [1]). In the deterministic Turing machine model every move is completely defined by the current situation. The state of the machine and the symbol currently scanned by the tape head completely determine the next state and the move of the tape head. Nondeterministic Turing machines may have choices in selecting the next move. Such a machine accepts input x if and only if there is a sequence of choices of the allowable moves which lead from the starting configuration to an accepting state.

In structural complexity theory decision problems are classified as tractable and intractable. Tractable problems are those which are feasibly computable (practically solvable). A problem is generally considered to be tractable if it can be solved by an algorithm with a time-complexity function which is bounded by a polynomial. Since problems are encoded as formal languages, the class **P** consisting of all languages accepted by deterministic Turing machines in polynomial time may be considered as a formal equivalent of tractable problems.

The class of problems which can be solved in polynomial time using a nondeterministic procedure is the second class central to structural complexity theory. Problems from this class have one common property: the certificate property. In each case, if a given input of the problem is a YES instance, then there is a short argument (a succint certificate) which convinces us that the input is indeed a YES instance. Naturally NO instances possess no such certificates. Certificates are small which means that their length is bounded by a polynomial in the length of the input. The formal counterpart of these problems is the class **NP** consisting of all languages accepted by a nondeterministic Turing machine in polynomial time. At the moment we do not know whether **NP**=**P**, and this open question is the central problem in structural complexity theory.

The polynomial time analogue of many-one reducibility from Post's paper is the notion of polynomial time reducibility (Karp reducibility) intensively studied in complexity theory. Just as the family of recursively enumerable sets possesses complete sets with respect to many-one reducibility, **NP**-languages possess complete languages with respect to Karp reducibility.

NP-complete languages capture the essence and difficulty of the whole **NP** class. They enjoy the following property: one of **NP**-complete languages belongs to **P** only if they all do, only if the whole **NP** class comes down to **P**.

The most important **NP**-complete language was discovered by Stephen Cook.

Theorem 2. (Cook)
SAT, the set of satisfiable Boolean formulas, is **NP**-*complete.*

The importance of **NP**-completeness for combinatorial optimization was revealed by Richard Karp. He proved that the following graph theoretic problems are **NP**-complete: HAMILTONIAN CYCLE, CLIQUE, VERTEX COVER, TSP - the travelling salesman problem (for exact definitions see [1]). His results opened the floodgate to proofs that hundreds of different problems are **NP**-complete. A proof that a language is **NP**-complete is now considered strong evidence that the problem encoded by this language is not feasibly solvable.

Post's problem was considered in polynomial setting by Richard Ladner. He finished the classification of **NP** languages by proving the following theorem:

Theorem 3. (Ladner)
If $\mathbf{P} \neq \mathbf{NP}$, *then there is a language in* **NP** *which is neither in* **P** *nor is it* **NP**-*complete.*

4 Isomorphism Conjecture

It was proved by John Myhill that all many-one complete recursively enumerable sets are recursively isomorphic, i.e., are identical up to a recursive permutation of their elements. **NP**-complete languages correspond to many-one complete recursively enumerable sets and this analogy together with Myhill's result led to the Isomorphism Conjecture: all **NP**-complete languages are isomorphic under polynomial time computable permutations (polynomial time bijections).This conjecture was supported by Leonard Berman and Juris Hartmanis' result that all known-up-to-now **NP**-complete languages are polynomially isomorphic.

One way of disproving the Isomorphism Conjecture would be by showing that there exist **NP**-complete languages with sufficiently different densities, because if two languages are polynomially isomorphic, then their densities are polynomially related.

We say that a language is sparse if the number of its words up to length n is bounded by a polynomial in n. A sparse language can not be polynomially isomorphic to SAT, since SAT is too dense to be mapped by a bijection onto a sparse set. Thus sparse **NP**-complete languages are natural candidates to refute the Isomorphism Conjecture.

The possibility of finding sparse **NP**-complete languages was resolved by Stephen Mahaney's result.

Theorem 4. (Mahaney)
There exist sparse **NP**-*complete languages if and only if* **P**=**NP**.

It is worth mentioning that the first step in this direction was done by Polish mathematician Piotr Berman. He proved that there exist sparse **NP**-complete languages over a single letter alphabet if and only if **P**=**NP**.

References

1. Aho A., Hopcroft J. and Ullman D. (1977) *The Design and Analysis of Computer Algorithms,* Addison-Wesley.
2. Post E. (1944) *Recursively Enumerable Sets of Positive Integers and their Decision Problems,* Bulletin AMS, 50, pp. 284–316.

This article was processed using the LaTeX macro package with LLNCS style