complexity to another). Indeed, we may say that the former is concerned with absolute answers regarding specific computational phenomena, whereas the latter is concerned with questions regarding the relation between computational phenomena.

Interestingly, so far complexity theory has been more successful in coping with goals of the latter ("relative") type. In fact, the failure to resolve questions of the "absolute" type led to the flourishing of methods for coping with questions of the "relative" type. Let us say that, in general, the difficulty of obtaining absolute answers may naturally lead to seeking conditional answers, which may in turn reveal interesting relations between phenomena. Furthermore, the lack of absolute understanding of individual phenomena seems to facilitate the development of methods for relating different phenomena. Anyhow, this is what happened in complexity theory.

Putting aside for a moment the frustration caused by the failure of obtaining absolute answers, there is something fascinating in the success to relate different phenomena: In some sense, relations between phenomena are more revealing than absolute statements about individual phenomena. Indeed, the first example that comes to mind is the theory of NP-completeness. Let us consider this theory, for a moment, from the perspective of these two types of goals.

P, NP, AND NP-COMPLETENESS

Complexity theory has failed to determine the intrinsic complexity of tasks such as finding a satisfying assignment to a given (satisfiable) propositional formula or finding a 3-coloring of a given (3-colorable) graph. But it has succeeded in establishing that these two seemingly different computational tasks are in some sense the same (or, more precisely, are computationally equivalent). This success is amazing and exciting; hopefully the reader shares these feelings. The same feeling of wonder and excitement is generated by many of the other discoveries of complexity theory. Indeed, the reader is invited to join a fast tour of some of the other questions and answers that make up the field of complexity theory.

Computational complexity is the general study of what can be achieved within natural limitations on natural computational resources.

We will start with the P versus NP question. Our daily experience is that it is harder to solve a problem than it is to check the correctness of a solution (e.g., think of either a puzzle or a homework assignment). Is this experience merely a coincidence or does it represent a fundamental fact of life (i.e., a property of the world)? Could you imagine a world in which solving any problem is not significantly harder than checking a solution to it? Would the term "solving a problem" not lose its meaning in such a hypothetical (and impossible in our opinion) world? The denial of the plausibility of such a hypothetical world (in which "solving" is not harder than "checking") is what "P different from NP" actually means, where P represents tasks that are efficiently solvable and NP represents tasks for which solutions can be efficiently checked.

The mathematically (or theoretically) inclined reader may also consider the task of proving theorems versus the task of verifying the validity of proofs. Indeed, finding proofs is a special type of the aforementioned task of "solving a problem" (and verifying the validity of proofs is a corresponding case of checking correctness). Again, "P different from NP" means that there are theorems that are harder to prove than to be convinced of their correctness when presented with a proof. This means that the notion of a "proof" is meaningful; that is, proofs do help when seeking to be convinced of the correctness of assertions. Here NP represents sets of assertions that can be efficiently verified with the help of adequate proofs, and P represents sets of assertions that can be efficiently verified from scratch (i.e., without proofs).

In light of the foregoing discussion, it is clear that the P versus NP question is a fundamental scientific question with far-reaching consequences. The fact that this question seems beyond our current reach led to the development of the theory of NP-completeness. Loosely speaking, this theory identifies a set of computational problems that are as hard as NP. That is, the fate of the P versus NP question lies with each of these problems: If any of these problems is easy to solve then so are all problems in NP. Thus, showing that a problem is NP-complete provides evidence to its intractability (assuming, of course, P different than NP). Indeed, demonstrating the NP-completeness of computational tasks is a central tool in indicating hardness of natural computational problems, and it has been used extensively both in computer science and in other disciplines. NP-completeness indicates not only the conjectured intractability of a problem but rather also its "richness," in the sense that the problem is rich enough to encode any other problem in NP. The use of the term "encoding" is justified by the exact meaning of NP-completeness, which in turn establishes relations between different computational problems (without referring to their absolute complexity).

SOME ADVANCED TOPICS

The foregoing discussion of NP-completeness hints to the importance of representation, since it referred to different problems that encode one another. Indeed, the importance of representation is a central aspect of complexity theory. In general, complexity theory is concerned with problems for which the solutions are implicit in the problem's statement (or rather in the instance). That is, the problem (or rather its instance) contains all necessary information, and one merely needs to process this information in order to supply the answer [1]. Thus, complexity theory is concerned with manipulation of information, and its transformation from one representation (in which the information is given) to another representation (which is the one desired).