计算机科学的探索

Computer Science Revisited

作者：vinton G.Cert

期刊：communications of the ACM

第55卷第七页十二号

By Vinton G. Cerf

Communications of the ACM, Vol. 55 No. 12, Page 7

在最近的几期专栏里，我常常质疑在计算科学中是否真正存在科学。这激起了很大的反应，好多人都提供了一些非常有价值的观点。在我们的研究领域中，某些方面的计算分析和建模程度都很公平的成为了一个个严谨的科学元素。可计算性，分析工作因素，估计计算的精确程度，计算成本等这些纯粹的 NP 问题等，都落入到这样一个我们认为非常强悍，被我们称之为“分析”的这个类上，比如如何计算一个产品成本的在一规模下的成长。在最近与 Alan Kay, Edward Feigenbaum, Leonard Kleinrock, 和 Judea Pearl 的探讨过程中，我开始相信存在者某些东西来帮助我们软件领域里分析和预测以后的变化情况。

In a recent column, I questioned whether there was any "science" in computer science. This provoked a great many responses that provided some very valuable perspective. To the degree that some aspects of computing are subject to analysis and modeling, it is fair to say there is a rigorous element of science in our field. Computability, analytical estimates of work to factor numbers, estimates of parsing complexity in languages, estimates of the precision of computations, computational costs of *NP*-complete problems, among others, fall into the category I call "analytic" in the sense we can say fairly strongly how computational cost grows with scale, for example. In recent conversations with Alan Kay, Edward Feigenbaum, Leonard Kleinrock, and Judea Pearl, I have come to believe that certain properties contribute to our ability to analyze and predict behaviors in software space.

Alan Kay 在最后的一次关于纪念犹太明珠图灵的活动中雄辩到计算不是一成不变的，相反，而是一个动态和在最好的特征之间相互作用的过程，尤其是在当代的网络环境中，了解计算过程的挑战是对现在基于输入输出和互动而产生的爆炸状态空间的管理。

Alan Kay spoke eloquently at a recent event honoring Judea Pearl's ACM Turing Award. Among the points that Alan made was the observation that computing is not static. Rather it is a dynamic process and often is best characterized as interactions among processes, especially in a networked environment. The challenge in understanding computational processes is managing the explosive state space arising from the interaction of processes with inputs, outputs, and with each other.

在最近与 Edward Feigenbaum 的讨论中，他指出那些应用在硬件分析方面的模型比那些应用在软件分析方面的模型更听话，因为在物理世界中，必然限制着对问题的寻求与解决。在逻辑的空间里，软件系统有少得多的结构可以在分析中被利用。但在今天，巨大的计算速度已经成为可能，再加之一些结构与优化的系统空间，可能会出现一些深入分析并且能够很有效的的分析软件，但他不可能使用的数学方法深入到人类的思想。他回忆到之前的一个东西，那是基于关于国际象棋的一些结构，那是场比赛，导致 IBM 的“深蓝”打败了国际象棋冠军——Kasparov，在这场比赛中，“深蓝”在每一步的走动中计算着、探索者数以百万计的的可能性。于是“深蓝”发现了一个最为优化的途径，然后一步步的计算下去，这是我们以前从未见过的，这场比赛也使的 Kasparov 在这场比赛中失败。

In a recent discussion with Edward Feigenbaum, he noted that models used in hardware analysis are more tractable than those used in software analysis because what is possible in the "physical" world necessarily constrains the search for solutions. Software systems are designed and analyzed in "logical" space, in which there is far less structure to exploit in the analysis. However, the enormous computing speeds possible today, combined with some structuring of the software design space, might allow deep and effective analyses of software as yet unavailable to mathematical methods or human thought. He recalled how very fast search, structured by some knowledge of chess, allowed IBM's Deep Blue to defeat the world's chess champion Kasparov. In one game, Deep Blue, exploring hundreds of millions of possibilities in the analysis of a move, found a brilliant solution path, probably never before seen, that led Kasparov to resign from the game and lose of the match.

被赋予抽象的空间时常约束着结构的发展。在这里，我们有得到了一个重要

的东西，它是有关研究计算科学与科学方法的潜力的问题。在某种程度上，抽象往往删除了某些不重要的东西，揭示出了事情的本质。抽象，本质上是一个强大的分析工具。这是我想起混沌理念说的一个事，在这个理念中模式的出现尽管有着明显的随机进程，但他的主旨却都是分析。如果程序和模型都进行的适当的抽象，那么他们就有可能出现足够有抽象度保证的模型，从而使模型更有分析化。这儿有一个在排列理论中的例子，即在复杂的流程中为自己的队列建模。Kleinrock 就通过网络的设计与分析，创作出了排列理论模型，这个模型的建设为社会作出了巨大的贡献。他最大的贡献是状态空间爆炸的认可，他由于他的独立性假设而成功。流经网络流量在每个字节处他都使用统计学对他进行了再生，但之前的变量仍独立，并没造成覆盖。

Structure that constrains state spaces is sometimes conferred through abstraction. Here we get to another key observation about the potential for scientific approaches to computer science. To the degree that abstraction eliminates unimportant details and reveals structure, abstraction is a powerful analytical tool. It strikes me that Chaos theory illustrates this notion because patterns emerge in this theory despite the apparent randomness of the processes it seeks to analyze. If programs and algorithms are subject to suitable abstraction, it may be possible to model them with adequate but abstract fidelity so as to render the model analyzable. An example is found in queueing theory in which complex processes are modeled as networks of queues. The models are analytic under the right conditions. Kleinrock made enormous contributions to network design and analysis through the construction of queueing theoretic models. His most famous contribution was the recognition that the state space explosion could be tamed by his Independence assumption in which the statistics of the traffic flowing through the network were regenerated at each node using statistically identical but independent variables.

建模是一个抽象的形式，并且，在计算机中是一个强大的工具。这导致我在犹太明珠的讲座中得出的结论涉及概率，而非一个固定的答案。珍珠理论在条件概率下的因果推论经常被图模型因果链增加，图使我们把构造解析方程成为了可能。这个量子相互作用模拟图是抽象的、复杂的，但他能帮助我们理解、分析和预测他们的行为。珍珠图可以证明强大的科学，而不仅仅是计算机科学、量子物理学和宇宙学。

Modeling is a form of abstraction and is a powerful tool in the lexicon of computer science. This leads me to a final observation illustrated by Judea Pearl's brilliant lecture on the use of Bayesian analysis to draw conclusions from problems involving probabilities rather than fixed values. Pearl's theories of causal reasoning in conditional probabilities are often aided by graph-like models linking the various conditional statements in chains of cause and effect. The diagrams make it possible to construct analytic equations that characterize the problem and make the solution computable. It struck me that Pearl's use of diagrams had an analogue in Richard Feynman's diagrams of quantum interactions. These diagrams are abstractions of complex processes that aid our ability to analyze and make predictions about their behavior. Pearl's diagrams may prove as powerful for science (not only computer science) as Feynman's have for quantum physics and cosmology.

我已经远离电脑，进军到科学这个问题。首先，计算机科学的确是科学；其次，抽象和建模是事情分析的关键；最后，计算机程序中有复杂细节和不同的交互方案，不承认抽象或建模使这些复杂的系统难以分析。现在，我相信我们有了解和预测软件行为的能力，发明更好的更高层次的编程语言，使那些被抑制的细节和模式的出现。我现在希望 Alan Kay 的炒作会严重改进我们的设计方式和计算机系统。

I have come away from this foray into computer 'science' with several conclusions. The first is there really is science to be found in computer science. The second is abstraction and modeling are key to making things analytic. The third is there is a lot of complex detail in computer programs and collections of interacting programs that has not admitted much in the way of abstraction or modeling, rendering these complex systems difficult to analyze. Finally, I believe our ability to understand and predict software behavior may rest in the invention of better high-level programming languages that allow details to be suppressed and models to emerge. I hope Alan Kay's speculation will lead to serious improvements in the way we design and program computer systems