development involving simultaneous targeting of several proteins, it then becomes a different ball game requiring large organization and funds. That phase does not come under the purview of the present discussion. In any case, drug development is not the only use of inhibitors. They are indispensable tools in biological research. Therefore, designing of inhibitors is an intrinsically worthwhile exercise, quite apart from its utility in drug development.

Conclusion

As indicated earlier, detailed structural information on a large number of proteins from different pathogens is now available in Indian laboratories. This is particularly true in relation to *M. tuberculosis*. The time is now propitious to initiate concerted efforts in the area of structure-based inhibitor design. Efforts with emphasis on validated targets should certainly be encouraged. In addition, it is also important to support the more holistic approaches of the type outlined above. Apart from other things, there is a crying need to develop drugs for infectious diseases, including TB. India now has the competence to contribute substantially to addressing this need. In this note, I have focused on structure-based efforts, as I am particularly familiar with them. Our efforts in this area should involve proven paradigms as well as modified or new paradigms. In the present context, the adage 'let a hundred flowers bloom and let a hundred ideas contend', should guide us.

1. Vijayan, M., *Curr. Sci.*, 2003, **85**, 878–885.
2. Kumar, R. A., Vaze, M. B., Chandra, N. R., Vijayan, M. and Muniyappa, K., *Biochemistry*, 1996, **35**, 1793–1802.
3. Bachhawat, N. and Mande, S. C., *J. Mol. Biol.*, 1999, **291**, 531–536.
4. Datta, S. *et al.*, *Nucleic Acids Res.*, 2000, **28**, 4964–4973.
5. Arora, A. *et al.*, *Tuberculosis*, 2011, **91**, 456–468.
6. Chidambaram, R., *Curr. Sci.*, 2007, **92**, 1229–1233.
7. Ramakrishnan, T. and Chandrasekhar, P., *J. Biosci.*, 1999, **24**, 143–152.

M. Vijayan is in the Molecular Biophysics Unit, Indian Institute of Science, Bengaluru 560 012, India.
e-mail: mv@mbu.iisc.ernet.in

Is 'compiler construction' a dead subject?

Pinaki Chakraborty

It will not be an exaggeration if I say that today we live in the age of computers. We are surrounded by computers and other programable devices. Almost all software programs that run on these devices are written in high-level programing languages like C, C++ and Java. These programing languages allow software developers to specify what the program is supposed to do in a human intelligible form. This property of the high-level programing languages makes it convenient for software developers to write and debug programs. Unfortunately, computers understand none of these high-level programing languages. A computer can only run a program written in its machine language. A machine language is a machine-specific low-level language, and is difficult to understand and use by software developers. So, a special type of software program called compilers is used to bridge the gap between the high-level programing languages and the machine languages. A compiler translates a program written by a software developer in a high-level programing language into machine language.

The first realistic compiler was developed by a team led by John W. Backus in 1957. That compiler translated programs written in the FORTRAN (FORmula TRANslation) programing language into the machine language of the then latest IBM 704 computers. When the developers were commissioned to develop that compiler, they hardly had any idea of the difficulty of the project which they expected to complete within six months. However, they ended up consuming two and half years of time and 18 man-years of effort to complete the project. Their experience taught two important lessons to the computer science community. First, compilers are complex programs and a subject called 'compiler construction' should be formally established. Second, and more importantly, compilers are useful software programs that can revolutionize the art of computer programing. Serious research and repeated development activities over the years have by now standardized the structure and the internal working of compilers. However, both high-level programing languages and computer architectures have been evolving continuously since 1957. Consequently, compilers have been forced to evolve too.

A few years ago while studying at Jawaharlal Nehru University, I once heard a senior professor from another premier university in India, who was delivering an invited talk in a conference in our university, make a passing remark that nobody works on compilers anymore. So, is compiler construction a dead subject? Courses on compiler construction are taught in both undergraduate and postgraduate-level computer engineering programs in most universities in India. However, these courses are often taught in a dry and uninteresting manner with either little or absolutely no laboratory support. Moreover, hardly anybody does research on compilers in India. However, things are quite different abroad, especially in the top universities. There are active research groups working on compilers. Courses on compiler construction are taught based on a programing exercise. This programing exercise is often the largest and the most sophisticated program that computer engineers write in their student life.

There are quite a few reasons for studying compiler construction and researching compilers. A decent knowledge of compilers helps software developers to write programs with desired characteristics like small size when translated into machine language, less running time, better fault tolerance and low power consumption. For programs that will be used many times by multiple

Table 1. Scientists and new programing languages developed by them

Scientist	Programing language	Year of receiving Turing Award
John McCarthy	Lisp	1971
Allen Newell and Herbert A. Simon	IPL	1975
John W. Backus	Fortran	1977
Kenneth E. Iverson	APL	1979
Dennis M. Ritchie	C	1983
Niklaus E. Wirth	Pascal	1984
A. J. R. G. Milner	ML	1991
O.-J. Dahl and Kristen Nygaard	Simula	2001
Alan C. Kay	Smalltalk	2003
Peter Naur	Algol	2005

users, a small percentage of decrease in either running time or power consumption is quite an achievement. When it comes to research, a thorough knowledge of compilers is essential for developing a new programing language. In fact, new programing languages are developed and new features are added to existing programing languages regularly. Knowledge of compilers is also useful in developing new computer hardware. It is more beneficial to develop hardware utilities which can be used efficiently by the machine language programs generated by the compilers. This holds true for all types of programable devices, including personal computers, embedded systems and supercomputers. This concept was actually used in developing the reduced instruction set computers (RISC), which were much simpler but almost as powerful as their predecessors. As a result, there are some journals and conferences dedicated to compilers and related topics.

I will like to discuss some interesting facts and figures. The A. M. Turing Award is considered to be the most prestigious award in the field of computer science. This award is given to an individual or a small clique of collaborators for pioneering research in computer science every year since 1966. Till date, five scientists have received this award primarily for their work on compilers – Alan J. Perlis (1966), John W. Bakus (1977), John Cocke (1987), Peter Naur (2005) and Frances E. Allen (2006). In fact, Allen is the first woman to receive this award. There are some other recipients of this award who have also worked on compilers among other things, like Edsger W. Dijkstra (1972), Donald E. Knuth (1974), Robert W. Floyd (1978), C. A. R. Hoare (1980) and Richard E. Stearns (1993). Some others have received this award for developing new high-level programing languages (Table 1). The fact that so many prominent scientists chose to work on compilers shows its importance in computer science.

At the international level, there are several objectives of research on compilers today. Three such objectives warrant special mention. First, the most important objective is to develop compilers that can efficiently translate new programing language features which can better represent the thought processes of the software developers into machine language. Designing new programing language features is a continuous process helping in the evolution of programing languages. Second, is to develop compilers that will be able to generate machine language programs which will take maximum advantage of the parallel processing capabilities of multi-core processors in personal computers and multiple processors in supercomputers. Third, developing compilers which will be able to generate low power consuming machine language programs for mobile phones and other battery-powered programable devices.

It is important to start working on these topics in India too. Research groups devoted to compilers should be established here. Both short- and long-term research and development projects should be initiated. Partnerships between academia and industry should be encouraged. The Indian IT industry, in contrast to its international counterparts, has not made any major contribution to compiler research. Research on compilers and programing languages, although resource-consuming, will prove beneficial in the long run. However, to support research on compilers, it is first necessary to improve the way 'compiler construction' is taught in India.

Pinaki Chakraborty is in the Division of Computer Engineering, Netaji Subhas Institute of Technology, New Delhi 110 078, India.
e-mail: pinaki_chakraborty_163@yahoo.com