Automated Reasoning Systems and Molecular Computing

Carmen Graciani-Díaz, Mario J. Pérez-Jiménez

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. Ingeniería Informática, Universidad de Sevilla
Avda. Reina Mercedes, s/n – 41012 Sevilla, Spain
E-mail: cgdiaz, marper@us.es

Abstract

This work was intended as an attempt to show the possible advantages provided by bringing together two areas, automated reasoning systems and DNA computing. The former as theoretical and formal devices to study the correctness of a program. The latter as practical devices to handle DNA strands to solve classical hard problems using laboratory techniques. To illustrate this approximation we present how we have obtained in the PVS proof checker the correctness of a program in a sticker based model for DNA computation. This result required a previous work: the formalization of the elected model within the PVS language. Also, in order to deal with imperative programs, we have studied a formalization of the Floyd–Hoare logic in the same system, PVS.

1 Introduction

The study and use of formal methods is among the most active areas in Computer Science. By formal methods we understand the study of how primarily mathematics and its techniques can be useful to solve software engineering problems. The widely development of this area and the increasing complexity of interesting problems have given rise to the use of informatics tools in order to "automate" logic reasoning.

In automate reasoning one of the main attacked problems is the correctness one [1]: developing specifications and proofs in a formal way to ensure that a program meets its specifications. That is, to prove that a program really solves the problem for which it was designed. This aim requires a previous work of formalization: expressing all definitions and results in the formal language of the chosen system to avoid semantic ambiguity. One of the most important advantage of using formal methods lies in the fact that the obtained proofs can support rigorous revisions.

The development of provers that use as their specification language more and more expressive logics cause difficulties to automate the process of deduction. The majority of systems constructed within this spirit are interactive. An overview to the "*Database of Existing Mechanized Reasoning System*", `http://www-formal.stanford.edu/clt/ARS/systems.html` shows how many initial system have disappeared, other have evolved and many other new ones have appeared.

Formal methods and automated reasoning systems acquired special relevance in new computing paradigms such as Natural Computing in order to prevent the appearance of numerous results whose veracity is assume but, however, a careful revision of them shows their lacks.

We have focused our attention in *DNA based molecular computing* and the *Prototype Verification System, PVS* [7]. To illustrate our work we present how the correctness of a program, in a *sticker based model* for DNA computation, designed to solve the exact cover problem can be established with the usual techniques for imperative programs due to Floyd–Hoare using the PVS proof checker.

This article is organized as follows. First, a presentation of PVS and of the sticker model is given. The former with the purpose to introduce the system used and to make easier the reading of the rest of the article. The latter to present the molecular model based on stickers due to S. Roweis, E. Winfree et al. [12] in 1998. Along with the description of the computational model a possible formalization using the specification language of PVS in given. In the third section we review imperative programs and the common tools based on specifications of partial correctness from Floyd–Hoare logic used to deal with their verification. Also, a possible implementation of them in PVS is given. To conclude, in order to illustrate the behavior of the previous presented work we include two subroutines and a program designed in the sticker model that solve the exact cover problem. We provide a correctness result obtained with PVS to demonstrate this assert.

The achievement of this work has included the developing of a set of PVS *theories* that allows us to represent and operate in the system with different data structures (finite sequences, multisets, etc) in a natural way. Moreover, following the treatment given by P. Y Gloess [5] a construction of a first order logic and a Floyd–Hoare calculus to handle imperative programs over two types states is included. The well-foundness of the logic has also been established.

The description of different used elements are illustrated with the tcorresponding declaration in PVS. Each declaration is included in a box with a label on the upper right corner indicating the theory which it belongs to. The complete set of theories built in PVS 3.1 for this article are available on the web at `http://www.cs.us.es/ cgdiaz/investigacion`.

References

[1] Boyer, R. S.; Moore, J. S. *The correctness problem in computer science.* Academic Press, 1981.

[2] Crow, J.; Owre, S.; Rushby, J.; Shankar, N.; Srivas, M. *A tutorial introduction to PVS*, presented at Workshop on Industrial-Strength Formal Specification Techniques (1995). Available at `http://www.csl.sri.com/wift-tutorial.html`.

[3] Graciani-Díaz, C. *Especificación y verificación de programas moleculares en PVS.* PhD thesis, University of Seville, 2003, `http://www.cs.us.es/ cgdiaz/investigacion`.

[4] Floyd, R. W. Assigninig meaning to programs, in Schwartz, J. T. (ed.), *American Mathematical Society Symposium on Applied Mathematics*, **19** (1967), Providence, R.I., 19–32.

[5] Gloess, P. Y. *Imperative program verification in PVS*, 1999, `http://dept-info.labri.u-bordeaux.fr/ gloess/pvs/imperative/`.

[6] Hoare, C. A. R. An axiomatic basis for computer programming, *Communications of the ACM*, **12**, 10 (1969), 576–583.

[7] Owre, S.; Rushby, J. M.; Shankar, N. PVS: A prototype verification system, in Kapur, D. (ed.), *11th International Conference on Automated Deduction (CADE)*, LNCS **607**, Springer-Verlag (1992), 748–752.

[8] Owre, S.; Rushby, J. M.; Shankar, N.; Stringer-Calvert, D. W. J. *PVS Language Reference*, Computer Science Laboratory, SRI International, Menlo Park, CA, 1999.

[9] Owre, S.; Rushby, J. M.; Shankar, N.; Stringer-Calvert, D. W. J. *PVS Prover Guide*, Computer Science Laboratory, SRI International, Menlo Park, CA, 1999.

[10] Owre, S.; Shankar. N. *The formal semantics of PVS*, Technical Report **SRI-CSL-97-2**, Computer Science Laboratory, SRI International, Menlo Park, CA, 1997.

[11] Owre, S.; Shankar, N.; Rushby, J. M.; Stringer-Calvert, D. W. J. *PVS System Guide*, Computer Science Laboratory, SRI International, Menlo Park, CA, 1999.

[12] Roweis, S.; Winfree, E.; Burgoyne, R.; Chelyapov, N. V.; Goodman, M. F.; Rothemund, P. W. K.; Adleman, L. M. A sticker based model for DNA computation, in *Proceedings of the Second Annual Meeting on DNA Based Computers*, DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society (1996), 1–27.