Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 124 (2005) 17–28
www.elsevier.com/locate/entcs

Invariant-Driven Strategies for Maude[1]

Francisco Durán[2] Manuel Roldán[3] Antonio Vallecillo[4]

Dpto. de Lenguajes y Ciencias de la Computación
Universidad de Málaga
Málaga, Spain

Abstract

We propose generic invariant-driven strategies that control the execution of systems by guaranteeing that the given invariants are satisfied. Our strategies are generic in the sense that they are parameterized by the system whose execution they control, by the logic in which the invariants are expressed, and by the invariants themselves. We illustrate the use of the strategies in the case of invariants expressed in propositional logic. However, the good properties of Maude as a logical and semantic framework, in which many different logics and formalisms can be expressed and executed allow us to use other logics as parameter of our strategies.

Keywords: Execution strategies, Maude, rewriting logic, reflection.

1 Introduction

To deal with nonterminating and nonconfluent systems, we need good ways of controlling the rewriting inference process. In this line, different languages offer different mechanisms, including approaches based on metaprogramming, like Maude [2,3], or on strategy languages, like ELAN [1]. However, although the separation of logic and control greatly simplifies such a task, these mechanisms are sometimes hard to use, specially for beginners, and usually compromise fundamental properties like extensibility, reusability, and maintainability.

[1] Partially supported by projects TIC 2001-2705-C03-02 and 2002-04309-C02-02
[2] Email: duran@lcc.uma.es
[3] Email: mrc@lcc.uma.es
[4] Email: av@lcc.uma.es

doi:10.1016/j.entcs.2004.11.018

Formalisms like Z and UML suggest an interesting alternative, since they allow to define invariants or constraints as part of the system specifications. Although executing or simulating Z specifications may be hard, we can still find tools like Possum [9] or Jaza [12], which can do a reasonable simulation of such specifications. We find something somehow similar in UML, where, by specifying OCL constraints on our specifications, they can be made executable [13].

The execution or simulation of specifications with constraining invariants is typically based on integrating somehow the invariants into the system code. However, such an integration is clearly unsatisfactory: the invariants get lost amidst the code, and become difficult to locate, trace, and maintain. Moreover, the programs and the invariants to be satisfied on them are usually expressed in different formalisms, and *live at different levels of abstraction*: invariants are defined *on* programs. Therefore, it is interesting to have some way of expressing them separately, thus avoiding the mixing of invariants and code.

Maude does not provide direct support for expressing execution invariants. However, it does provide reflective capabilities and support to control the execution process, being also an excellent tool in which to create executable environments for various logics and models of computation [4]. Thus, it turns out to be a very good candidate for giving support to different types of invariants, which may be expressed in different formalisms.

In this paper we propose generic invariant-driven strategies to control the execution of systems by guaranteeing that the given invariants are always satisfied. Our strategies are generic in the sense that they are parameterized by the system whose execution they control, by the logic in which the invariants are expressed, and by the invariants themselves. The good properties of Maude as a logical and semantic framework [8], in which many different logics and formalisms can be expressed and executed, allow us to say that other logics and formalisms may be used as parameters of our strategies. We will use in this paper the case of propositional logic, although we have also experimented with future time linear temporal logic.

The paper is structured as follows. Section 2 serves as a brief introduction to rewriting logic and Maude. Section 3 introduces the definition of strategies in Maude, and serves as a basis for the introduction of invariant-guided strategies in Section 4. Section 5 describes as an example the case of invariants expressed using propositional calculus. Finally, Section 6 draws some conclusions.