Available online at www.sciencedirect.com

SCIENCE DIRECT

ELSEVIER Electronic Notes in Theoretical Computer Science 133 (2005) 295–313

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

An Abstract Interpretation Toolkit for μCRL

Jaco van de Pol Miguel Valero Espada[1],[2]

Centrum voor Wiskunde en Informatica
Amsterdam, The Netherlands

Abstract

This paper describes a toolkit that assists in the task of generating abstract approximations of process algebraic specifications written in the language μCRL. Abstractions are represented by *Modal Labelled Transition Systems*, which are mixed transition systems with *may* and *must* modalities. The approach permits to infer the satisfaction or refutation of *safety* and *liveness* properties expressed in the (action-based) μ-calculus. The tool supports the abstraction of states and action labels which allows to deal with infinitely branching systems.

Keywords: Abstract Interpretation, Modal Transition Systems, Abstract Model Checking, μCRL Toolset.

1 Introduction

The automatic verification of distributed systems is limited by the well known state explosion problem. Abstraction is a useful approach to reduce the complexity of such systems. From a *concrete* specification, it is possible to extract an *abstract* approximation that preserves some interesting properties of the original. In [32], we have presented the theoretical framework for abstracting μCRL [16] specifications. μCRL is a language that combines ACP style process algebra [3] with abstract data types. In this paper, we will describe the toolkit that implements the theory.

[1] Email: {`vdpol, miguel`}`@cwi.nl`

[2] Partially supported by PROGRESS, the embedded systems research program of the Dutch organisation for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW, grant CES.5009.

doi:10.1016/j.entcs.2004.08.070

Semantically, abstractions are represented by *Modal Labelled Transition Systems* [23], which are *mixed* transition systems in which transitions are labelled with actions and with two modalities: *may* and *must*. *May* transitions determine the actions that are part of some refinements of the system while *must* transitions denote the ones that necessarily appear in all refinements. The use of the two modalities allows to infer the satisfaction or refutation of formulas written in (action-based) μ-calculus [22] from the abstract to the concrete system.

The implementation of the previously developed theory is an indispensable step in order to apply abstract interpretation techniques to realistic systems. There exist different abstraction approaches that can be applied within the verification methodology. For example, *variable hiding* or *pointwise* abstraction in which, first, the value of some variables of the specification is considered as unknown, subsequently, extra non-determinism is added to the system when there are predicates over the abstracted variables. Another automated abstraction technique is the so-called *predicate abstraction* in which only the value of some conditions is retained and propagated over the dependent predicates of the specification. *Program slicing* is a technique that tries to eliminate all parts of the specification that are not relevant for the current verification.

The most common abstraction technique consists in interpreting the concrete specification over a smaller data domain. The user selects the set of variables to abstract and provides a new abstract domain that reflects some aspects of the original. This technique requires creative human interaction in order to select the parts of the system that are suitable to abstract and to provide the corresponding data domains. Furthermore, the user must ensure that the abstract interpretation satisfies some so-called safety requirements.

Our tool implements the automatic *pointwise* abstraction and, moreover, assists the user to create his own abstractions. The tool supports the use of two mainstream techniques for data abstraction. One proposed by Long, Grumberg and Clarke [6,24], in which the concrete and the abstract data domain are related via a homomorphic function and another based on Cousots' Abstract Interpretation theory (we use Abstract Interpretation with upper cases to refer to Cousots' work and abstract interpretation with lower cases to denote the general framework), see for example [7,8,20,21], in which data is related by Galois Connections. A lifting mechanism is also implemented which allows to automatically build Galois Connections from homomorphisms, see [28].

Standard abstraction frameworks are only based on the abstraction of states which make them unable to deal with infinitely branching systems with action labels. A unique feature of our tool is that it allows the abstraction of