Optimality in Abstractions of Model Checking

Rance Cleaveland1*, Purush Iyer1** and Daniel Yankelevich2

1 Dept of Computer Science, North Carolina State University, Raleigh, NC 27695-8206, USA.
2 University of Buenos Aires, Argentina.

Abstract. This paper investigates the use of abstract-interpretation-inspired techniques for improving the performance of procedures for determining when systems satisfy formulas in branching-time temporal logic. A framework for abstracting system descriptions is developed, and a particular method for generating abstract systems from given abstractions on system states is defined and shown to be both safe and optimal, in the sense that concrete systems satisfy all the temporal formulas enjoyed by their abstracted counterparts. One may then use a model checker on an abstracted (and hence smaller) system in order to infer properties of a concrete system.

1 Introduction

One popular approach to the analysis of systems of concurrent processes involves the use of a temporal logic to formulate properties that the system being studied should enjoy [13]. Such logics enable users to specify the behavior a system should exhibit as it evolves over time. The utility of this approach to specification and verification has been illustrated in numerous case studies, and in the case of finite-state systems model-checking algorithms have been developed for determining automatically when systems satisfy formulas in certain logics [5, 14]. These algorithms have been implemented, and the resulting tools used to analyze a variety of different systems. However, the practical applicability of model checking remains limited by the state-explosion phenomenon: the number of states a concurrent system exhibits may be exponential in the number of processes, and as a model checker must in the worst case examine every system state, the automatic analysis of systems of many processes remains in many cases beyond the capabilities of existing tools. Coping with state explosion remains one of the central issues confronting researchers in automatic verification of finite-state concurrent programs.

In this paper we develop a framework for alleviating state explosion arising in the checking of systems against specifications given in the very expressive branching-time temporal logic CTL* [9]. The framework is based on the use of

* Research supported by NSF grant CCR-9120995, ONR Young Investigator Award N00014-92-J-1582, NSF Young Investigator Award CCR-9257963, and NSF grant CCR-9402807.

** Research supported in part by NSF grant CCR-9404619.
state abstractions, which intuitively group states in a concrete system into equivalence classes that may be viewed as states in an abstract, and therefore smaller, system. In particular, using the methodology of abstract interpretation \[6, 10\] we identify when system abstractions induced by state abstractions are safe in the sense that the CTL* formulas satisfied by abstracted systems are guaranteed to hold for the associated concrete systems. We also establish a restricted optimality result by showing that a particular system abstraction is the most precise one possible among a class of safe abstractions.

The remainder of the paper develops along the following lines. In the next section we present Kripke structures, which we use to model systems, and define the semantics of CTL*. The section following then introduces democratic Kripke structures, which will be used as abstractions of Kripke structures, and recasts the semantics of CTL* with respect to them. Section 4 defines an abstraction ordering on the states of a democratic Kripke structure and explores properties of such an ordering. Section 5 then defines a class of democratic Kripke structures that are "safe" approximations for a given Kripke structure, while the section following establishes that a particular method for constructing such approximations is, in a certain sense, optimal. The final section contains our conclusions and directions for future research.

**Related Work.** Several recent papers also address the use of state abstractions in model checking \[1, 3, 4, 8, 11\]. The approaches generally fall into two categories. The first (and the one pursued by this paper) treats abstractions as functions mapping system states to abstract states and is followed by \[1, 3\]. Both papers present techniques for "lifting" such abstractions to "safe" system abstractions for fragments of branching time logics and present several compelling examples illustrating the savings that such approaches may yield in model checking. However, in \[3\] the abstraction mechanism presented does not work for all of CTL*, and connections with abstract interpretation are not pursued. While the authors of \[1\] present their results in the context of abstract interpretation, they restrict their abstract models to be bisimilar to the concrete model when considering their full branching-time logic. However, authors of both papers do not characterize when other constructions might be safe or address the question of optimality.

The second approach requires that the collection of abstract states form a complete lattice and that the abstraction function mapping (sets of) concrete states to abstract states be a component in a Galois insertion/connection. This framework is pursued by \[4, 8, 11\]; each presents a mechanism for lifting a particular abstraction on states into a safe one on Kripke structures in which formulas in all of CTL* are preserved. However, the connection with abstract interpretation is generally left unexplored, as no optimality results are presented.

In light of these comments, the contributions of this paper may be summarized as follows:

- We extend the intuitive "functions as abstractions" approach of \[1, 3\] to handle all CTL* specifications.