Science of Computer Programming 94 (2014) 217–237

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Supervisory control theory with Alloy ☆

Benoît Fraikin, Marc Frappier, Richard St-Denis *

Département d'informatique, Université de Sherbrooke, Sherbrooke (Québec), J1K 2R1, Canada

ARTICLE INFO

Article history:
Received 15 February 2013
Received in revised form 29 April 2014
Accepted 30 April 2014
Available online 22 May 2014

Keywords:
Alloy
Supervisory control theory
SAT solver
Kodkod

ABSTRACT

Scientific literature reveals that the symbolic representation techniques underlying some formal methods are useful in verifying properties or synthesizing parts of large discrete event systems. They involve, however, encoding complex schemata and fine-tuning heuristic parameters in order to translate specific problems into efficient BDD- or SAT-based representations. This approach may be too costly when the main goal is to explore a theory, use simulation to understand its underlying concepts and computation procedures, and conduct experiments by applying them to problems in different fields as the theory evolves over time. To achieve this goal, this paper investigates the use of Alloy in modeling and prototyping varying fragments of the supervisory control theory, including the verification of nontrivial properties such as controllability, normality and observational equivalence. It also shows how to apply abstract models for synthesizing optimal supervisors and reports on an experiment suggesting that Alloy can be used to synthesize supervisors for concrete systems using hierarchical decomposition.

1. Introduction

Novice researchers face numerous challenges in learning a new theory for the first time, particularly if their mental representations of knowledge acquired in their previous learning do not correspond to those required to grasp new difficult concepts. Developing an abstract model of a theory in a declarative manner with Alloy [1] provides an interactive simulation platform that can be used to explore various instances of the model. As people gradually become familiar with concepts and acquire a deep intuition of the theory, they can explore the abstract model with their own instances, even adding new relationships between concepts, in order to verify properties, generate solutions to practical problems and, ultimately, extend the theory. Prototyping and applying a theory in this way constitutes a source of motivation and creativity in identifying and solving new problems. This paradigm shift contrasts with the usual approach for tackling the verification of models expressed in a new theory, which consists in codification of verification or synthesis procedures in a conventional programming language or modification of open-source tools. In fact, these actions require manipulation of complex data structures and substantial effort to understand more than ten thousand lines of code compared with hundreds written in Alloy, as this paper shows.

☆ The research described in this paper was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).
* Corresponding author.
E-mail addresses: Benoit.Fraikin@USherbrooke.ca (B. Fraikin), Marc.Frappier@USherbrooke.ca (M. Frappier), Richard.St-Denis@USherbrooke.ca (R. St-Denis).

http://dx.doi.org/10.1016/j.scico.2014.04.016
0167-6423/

The aforementioned approach has been investigated with a control theory for discrete event systems (DES) called the *supervisory control theory* (SCT). Modeling sufficiently large fragments of this theory with the new kind of computation paradigm provided by Alloy has not been explored yet and represents a great challenge, considering the current limitations of SAT solvers. Overcoming this challenge constitutes an interesting avenue of research, since current DES tools do not provide support for experimenting with definitions, proving properties and generating counterexamples, tasks that are quite common in teaching SCT to students. Despite an overlap, for instance with state exploration techniques in formal verification, SCT is still relatively unknown to the computer science community, even if its relevance to computer science problems appears more and more obvious. Computer programs are intrinsically discrete objects that transform inputs into outputs using a set of discrete rules. Control is everywhere in these rules, generally scattered throughout the code. Furthermore, there is an increasing body of work on applying SCT to problems in computer science and software engineering, particularly in the areas of concurrent programming [2], software deadlock avoidance [3] and component-based software engineering [4]. In the same spirit of interdisciplinarity, several researchers from eight American universities launched the ExCAPE project[1] with the intention of transforming the way programmers develop software by advancing the theory and practice of software synthesis as advocated, for instance by SCT. There are also examples of cross-fertilization between planning in artificial intelligence and SCT [5–7]. Overall, the growing importance of this field is evidenced by the control and coordination of large networks consisting of arbitrary numbers of finite-state processes, such as multi-agent systems.

Based on a meaningful subset of SCT that illustrates many of its facets with Alloy, this paper explores issues in using this tool for solving control problems, details solutions for them and suggests potential modifications to Alloy. It is organized as follows. Section 2 briefly introduces SCT to relate control problems to the SAT-solving approach, as suggested in recent work. Section 3 presents typical control problems formulated within the framework of SCT with the aid of an example. It is useful for readers unfamiliar with this theory. Section 4 shows how to take advantage of Kodkod to implement an iterative specification refinement process to synthesize optimal supervisors when Alloy is ineffective in generating them in a single step. Section 5 pays special attention to the modeling of nontrivial SCT properties, such as controllability, normality and observational equivalence. Section 6 positions the Alloy specifications presented in this paper within a wider project and reports statistics of some experiments. Section 7 contains a discussion about Alloy's advantages and limitations in comparison with other tools that can be potentially used not only to solve control problems, but also to formulate the underlying theory. Finally, Section 8 ends with remarks about future work on the use of Alloy as a tool for solving real problems in the context of SCT.

2. A brief overview of the Supervisory Control Theory

SCT deals with the behavior of discrete event dynamic systems under the control of supervisors that aim to restrict their actions to satisfy control specifications. Since its proposal by Ramadge and Wonham [8], this theory has evolved over recent decades to give rise to various variants, each of them being specific to a system architecture and a class of control problems. Two dual formulations have been considered in developing these variants: *language-based formulation* and *state-based formulation*. In the former, a control specification is given in terms of a formal language over the system's event alphabet, called the *legal language*, and the control is exercised from the observation of event traces. In the latter, a control specification is expressed by the means of a predicate defined on the system's state space, which defines a set of *good states*, and the supervisor makes decisions based on the current state of the system. In both cases, a supervisor enables controllable events so that the system's controlled behavior remain in the legal language or good-state space according to the total or partial observation of event traces or states, respectively. Even if the language-based approach has been used in the development of many variants of the theory, the state-based approach seems to be more convenient and intuitive in some situations (e.g., fuzzy control [9], implementation of supervisors in programmable logic controllers [10]).

The development of a variant requires representing the constituent units of the system architecture with mathematical objects (e.g., formal languages over an event alphabet, state transition systems, predicates on a state space, mask functions), which are used to define properties (e.g., controllability, observability, nonblockingness) to be satisfied by the control specification. Then, based on these properties, necessary and sufficient conditions are formulated for the existence of supervisors. When these conditions are not fulfilled by the original control specification, the infimal or supremal element of a family of languages or predicates satisfying these conditions (if such an element exists) is considered to solve the problem. Then, synthesis algorithms allow for the automatic derivation of supervisors from mathematical models of DES, often expressed using finite automata for practical reasons [11]. Finally, translation procedures specific to code generation for software controllers from supervisors can be exploited to obtain executable solutions that are correct by construction. After many years of effort, several system architectures (e.g., limited lookahead, distributed, horizontal and vertical, conditional) and control patterns (e.g., on-line, decentralized, hierarchical, multi-decision) along these lines have been explored with success (e.g., [12–15]). Fig. 1, used several times in the sequel, illustrates some of them and shows that all of these architectures are organized as a closed-loop system.

Compared to BDD-based techniques, which have gained widespread use in the SCT community [16], little effort seems to have been invested in model checking with SAT solvers to verify properties formulated in the context of SCT or generate

[1] https://excape.cis.upenn.edu.