Seven at one stroke: LTL model checking for High-level Specifications in B, Z, CSP, and more

Michael Leuschel and Daniel Plagge

Softwaretechnik und Programmiersprachen
Institut für Informatik, Heinrich-Heine-Universität Düsseldorf
Universitätsstr. 1, 40225 Düsseldorf, Germany
{leuschel,plagge}@cs.uni-duesseldorf.de

Abstract. The size of formal models is steadily increasing and there is a demand from industrial users to be able to use expressive temporal query languages for validating and exploring high-level formal specifications. We present an extension of LTL, which is well adapted for validating B, Z and CSP specifications. We present a generic, flexible LTL model checker, implemented inside the PROB tool, that can be applied to a multitude of formalisms such as B, Z, CSP, B‖CSP, as well as Object Petri nets, compensating CSP, and dSL. Our algorithm can deal with deadlocking states, partially explored state spaces, past operators, and can be combined with existing symmetry reduction techniques of PROB. We establish correctness of our algorithm in general, as well as combined with symmetry reduction. Finally, we present various applications and empirical results of our tool, showing that it can be applied successfully in practice.
Keywords: Validation and Verification, Notations and Languages, LTL, model checking, B-method, CSP, Z, Integrated Methods, symmetry reduction.

1 Introduction and Motivation

The B-Method and Z are used in railway systems (Dollé et al., 2003), the automotive sector (Pouzancre, 2003), as well as avionics (Hall, 1996). The size of the formal models is steadily increasing and there is a big demand from industrial users to be able to animate and validate high-level specifications (Essamé and Dollé, 2007), in order to ensure that the correct system is built. The PROB tool set can be used to animate B (Leuschel and Butler, 2003) as well as Z specifications (Plagge and Leuschel, 2007). It can also be used to detect invariant violations, deadlocks and check refinement. However, there is also an industrial demand for expressive temporal query and validation languages[1], in order to validate temporal properties of the system (not easily expressed in B or Z), as well as to navigate in the state space, and ask questions about the future and past of the current state.

In this paper we present a methodology and implementation to satisfy this industrial need by

[1] Private communication from Kimmo Varpaaniemi, Space Systems Finland.