Translating B to TLA\(^+\) for Validation with TLC

Dominik Hansen and Michael Leuschel

Institut für Informatik, Universität Düsseldorf
Universitätsstr. 1, 40225 Düsseldorf
{hansen,leuschel}@cs.uni-duesseldorf.de

Abstract. The state-based formal methods B and TLA\(^+\) share the common base of predicate logic, arithmetic and set theory. However, there are still considerable differences, such as the way to specify state transitions, the different approaches to typing, and the available tool support. In this paper, we present a translation from B to TLA\(^+\) to validate B specifications using the model checker TLC. We provide translation rules for almost all constructs of B, in particular for those which are not built-in in TLA\(^+\). The translation also includes many adaptations and optimizations to allow efficient checking by TLC. Moreover, we present a way to validate liveness properties for B specifications under fairness conditions. Our implemented translator, Tlc4B, automatically translates a B specification to TLA\(^+\), invokes the model checker TLC, and translates the results back to B. We use ProB to double check the counter examples produced by TLC and replay them in the ProB animator. We also present a series of case studies and benchmark tests comparing Tlc4B and ProB.

Keywords: B-Method, TLA\(^+\), Tool Support, Model Checking, Animation.

1 Introduction and Motivation

B \cite{B} and TLA\(^+\) \cite{TLA+} are both state-based formal methods rooted in predicate logic, combined with arithmetic, set theory and support for mathematical functions. However, as already pointed out in \cite{Hansen:2013}, there are considerable differences:

- B is strongly typed, while TLA\(^+\) is untyped. For the translation, it is obviously easier to translate from a typed to an untyped language than vice versa.
- The concepts of modularization are quite different.
- Functions in TLA\(^+\) are total, while B supports relations, partial functions, injections, bijections, etc.
- B is limited to invariance properties, while TLA\(^+\) also allows the specification of liveness properties.

* Part of this research has been sponsored by the EU funded FP7 project 287563 (ADVANCE) and the DFG funded project Gepavas.

© Springer-Verlag Berlin Heidelberg 2014
– The structure of a B machine or development is prescribed by the B-method, while in a TLA+ specification any formula can be considered as a system specification.

As far as tool support is concerned, TLA+ is supported by the explicit state model checker TLC [13] and more recently by the TLAPS prover [3]. TLC has been used to validate a variety of distributed algorithms (e.g. [4]) and protocols. B has extensive proof support, e.g., in the form of the commercial product AtelierB [2] and the animator, constraint solver and model checker ProB [9,10]. Both AtelierB and ProB are being used by companies, mainly in the railway sector for safety critical control software. In an earlier work [5] we have presented a translation from TLA+ to B, which enabled applying the ProB tool to TLA+ specifications. In this paper we present a translation from B to TLA+, this time with the main goal of applying the model checker TLC to B specifications. Indeed, TLC is a very efficient model checker for TLA+ with an efficient disk-based algorithm and support for fairness. ProB has an LTL model checker, but it does not support fairness (yet) and is entirely RAM-based. The model checking core of ProB is less tuned than TLA+. On the other hand, ProB incorporates a constraint solver and offers several features which are absent from TLC, in particular an interactive animator with various visualization options. One feature of our approach is to replay the counter-examples produced by TLC within ProB, to get access to those features but also to validate the correctness of our translation. In this paper, we also present a thorough empirical evaluation between TLC and ProB. The results show that for lower-level, more explicit formal models, TLC fares better, while for certain more high-level formal models ProB is superior to TLC because of its constraint solving capabilities. The addition of a lower-level model checker thus opens up many new application possibilities.

2 Translation

The complete translation process from B to TLA+ and back to B is illustrated in Fig. 1.