ProB 2.0 Tutorial*

Jens Bendisposto, Joy Clark, Ivaylo Dobrikov, Philipp Körner, Sebastian Krings, Lukas Ladenberger, Michael Leuschel and Daniel Plagge

Heinrich-Heine Universität Düsseldorf

1 Introduction

We believe that any proof centric formal method must be complemented by animation and visualization to be useful because there are always properties of a model that cannot be covered by proof. In particular a proof can guarantee that a model is internally consistent but it cannot guarantee that this model is actually what its designer had in mind. Animation and visualization can help to review models. ProB is a mature toolset[1] that supports a whole range of different formalisms such as classical B, Event-B, CSP and Z.

2 Java Interface

Previously, we have developed a ProB plug-in to add animation and model checking to the Rodin platform. This plug-in was designed to be extensible using so called commands. A command encapsulates a call to Prolog and the result of that call. The architecture was successfully used to build third-party tools on top of ProB. For instance, the UML-B Statechart animation plug-in uses ProB.

However, while the design was very flexible there were some abstractions missing. For instance, the API had no notion of a state space or a trace. Each tool that required one of these concepts had to reinvent them. Also, building tools on top of ProB was a task for a plug-in developer and it required a good bit of ceremony, i.e., one needs to create an eclipse plug-in. We wanted to introduce a more lightweight approach allowing an end user to customize the tool itself. In ProB 2.0 this can be done using Groovy as an embedded scripting language. Our major design principles for the new version of ProB were

- **Embrace Groovy.** We try to make (almost) everything accessible for the scripting language. This is very important because it enables customization and extension of the tool from within the tool. Actually we even try to go further by designing the tool as if it were a library for Groovy. This made a couple of extensions to ProB extremely easy. For instance, adding CSP support to the plug-in only took a few hours.

* Parts of this research has been sponsored by the DFG funded research project GEPAVAS and the EU funded research FP7 project 287563: ADVANCE).

[1] The first version was published 10 years ago

– **Add reoccurring abstractions.** We have identified three main abstractions: model, state space and history. The model represents the static properties of a specification, i.e., its abstract syntax. For Event-B this could have been done using the Rodin abstractions but we wanted to use Event-B and classical B uniformly.
– **Prefer immutability.** This is more a implementation principle than a design principle but we think it is essential to make everything look immutable. If we add a new step to a history, we do not change the history but we generate a new history. Because of the consistent use of immutable values, we can use structural sharing to avoid expensive copying. The state space is actually mutable, but in a safe way: only new information can be added, old information always stays the same.

3 The tutorial

The tutorial is for people who want to use the new API programmatically. We will quickly demonstrate the current state of development and the new features of ProB 2.0 including B-Motion Studio and Worksheets, but the main focus of the tutorial is on using the scripting language and on leveraging the new abstractions. We will develop a few example scripts together during the tutorial (for example, this could be scripts to export the state space to some other format such as dot, generating test-cases, or calling ProB's constraint-solver).

4 Additional Information

Additional material for the tutorial will be provided on our website
`http://stups.hhu.de/ProB/index.php5/Tutorial13`