ABTOOLS: Another B Tool

Jean-Louis Boulanger
University of Technology of Compiegne
HEUDIASYC UMR CNRS 6599
BP 20529
60205 COMPIEGNE cedex - France
jean-louis.boulanger@utc.fr

Abstract

ABTools[1] *([2]) provides an open environnement based on ANTLR and JAVA and provides somes facilities for design and test an extension for the B language.*

1. Introduction

The formal B **method** [1] is more and more used in France for the development of safety-critical software for railway systems. The B method , developped by Jean-Raymond Abrial, is a 'model-based' formal method like Z [7] or VDM [4] but, unlike others, B allows an incremental process development starting from abstract specification to (automatically generated) code. In this paper we are particularly concerned by the fact that this process is covered by *one and only one language* : the Abstract Machine Notation (AMN). In the following we will use the name 'B language'. We suppose in the following that the reader is rather familiar with the B language.

2. ANTLR: ANother Tool for Language Recognition

ANTLR , formerly PCCTS, is a parser and translator generator tool that lets one define language grammars in either ANTLR syntax (which is YACC and EBNF (Extended Backus-Naur Form) like) or a special AST (Abstract Syntax Tree) syntax. ANTLR is written entirely in JAVA and it is more than just a grammar definition language. Lexers, parsers and tree parsers are specified using the same syntax and the same code generator is used for all three. Parser generated must interact with the lexical analyser, report parsing errors, construct abstract syntax trees, and call user actions. You combines the lexical analyser and parser specifications in one file, optionally generates abstract syntax trees (ASTs) and tree-walking classes, and allows very fine-grained control of the parse throught syntactics predicates. ANTLR is distributed in source-code form and it's free. ANTLR ([6]) converts an extended form of context-free grammar into a set of functions which directly implement an efficient form of deterministic recursive-descent LL(k) parser. Context-free grammars may be augmented with predicates to allow arbitrary semantics and syntactic context to influence parsing; this allows a form of context-sensitive parsing. Selective backtracking is also available to handle non-LL(k) and even non-LALR(k) constructs.

3. ABTools : Another B Tool

3.1. What is a B-Tool?

There are currently two recent commercialized tools supporting the B method: the `AtelierB` of ClearSy[2] and the *B-Toolkit* of BCore[3]. The facilities provided by each of them are summarized in figure 1. A tool session can be described as follows: once the source of a machine (which can be either a top level machine, a refinement, or an implementation) has passed the syntactic and type checks, a collection of POs are generated in conformity with the B method. Then the POs (Proof Obligation) are processed by the automatic prover that works in conjunction with a library of mathematical rules. Some of the POs will not be discharged: the interactive prover allows the user to create tactics or to add new mathematical rules for their proof. Once each of the remaining POs has been successfully processed, the automatic prover must be invoked again on the whole set of POs in order to verify that the tactics and rules added by the user do not invalidate the previous proofs. Fi-

[1] www.chez.com/abtools

[2] http://www.atelierb.societe.com/index.html

[3] http://www.b-core.com

nally, if the target B abstract machine is an implementation, translation into a programming language can be launched

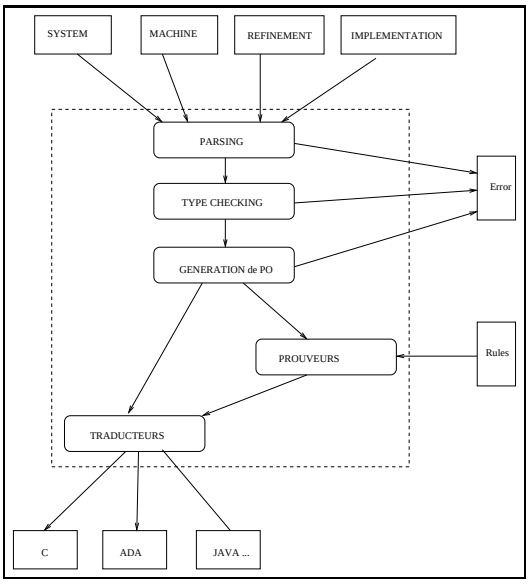

Figure 1. A B Tool

3.2. ABTools

Lexing and Parsing The syntactic analysis ensures that the source of an abstract machine satisfies the syntactic rules of the Abstract Machine Notation. Then, type checking is carried out. This stage is important because no attempt will be made to prove a predicate involving constructs that have not been checked for type consistency. Note that it is possible to determine the type of any expression because the set-theoretic model used in AMN is a simplification of the classical set theory. For the B "classic", we defines a ll(k) grammar where k is the lookahead set to 2. A lexer translates from a stream of characters to a stream of Tokens. Our lexer is a ll(k) grammar with a lookahead k set to 5. We implemented 3 extensions of the B language (BPRIME [3], BEvent [5] and BSystem) by grammar inheritance and overloading.

Treewalker ANTLR parser can automatically construct an abstract syntacx tree (AST), saving the user from having to explicitly call tree construction routines. An AST is a special kind of tree that can have an arbitrary number of subtrees (children) which are ASTs themselves. To create an AST, the user annotates the grammar to indicate what is a root node, what is a leaf node and what is to be excluded from the AST. When walking the tree one can manipulate the order in which nodes are visited with all expressiveness of the implementation language. The treeparser is being used to walk the tree and decompile the tree in ASCII, LaTeX, HTML or XML forms and in futur to generate C or JAVA code.

Typing The type checking allows to reveal faults like those related to missing or incorrect declarations, type inconsistency within an expression, inconsistency of the signature of an operation with the signature of its refined version, violation of the visibility rules induced by the composition clause part. Another treeparser is being used to walk the tree and type-check it.

Proof obligation In the B development, the proofs accompany the construction of software. Each time an abstract machine is defined or modified, there are POs related to its mathematical consistency; if the machine is a refinement or an implementation, there are also proofs of its correctness with respect to the previous steps of the development chain. Our B tools allow to generate automatically the POs for each abstract machine by walking the tree with a treeparser.

4. Conclusions

ANTLR proved to be a robust and effective toolkit. For this kind of translation task the tree rewriting facilities seemed a natural fit to the problem. ABTools is a B development environnement (parsing, typechecking, PO and Documentation Generation) that provides the possibily to prototype (rapidely) extension for the B language. ABTools is free, public-domain software.

References

[1] J.-R. Abrial. *The B Book - Assigning Programs to Meanings*. Cambridge University Press, Aug. 1996.

[2] B. Jean-Louis. Abtools, une suite d'outil pour la méthode b développé avec antlr. In *Journées «Outils pour et autour de la méthode B», 15 et 16 octobre 2001*.

[3] B. Jean-Louis, M. George, and T. Bruno. Revisiting b language syntax. Technical Report 99-07, CNAM Laboratoire CEDRIC, 1999.

[4] C. B. Jones. *Systematic Software Development Using VDM*. Prentice-Hall International, Englewood Cliffs, New Jersey, second edition, 1990. ISBN 0-13-880733-7.

[5] MATISSE. Event b reference manual. Technical report, Methodologies and Technologies for Industrial Strength Systems Engineering, 2001.

[6] T. J. Parr. *Obtaining practical variants of LL(K) for K>1 by splitting the atomic K-Tuple*. PhD thesis, Purdue University, 1993.

[7] J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall International Series in Computer Science, 2nd edition, 1992.