

Jml Winter School 2008 Exercise 2

Adding full support for annotated loops
prepared by Perry R. James and Patrice Chalin

Objective: Based on lessons learned in Exercise 1, complete the implementation of the Reference Manual's possibly-annotated-loop non-terminal.

1. Unit tests:
 - a. Before starting each of the following steps, write a unit test that fails.
 - b. After completing each step, also add tests that are needed to give better coverage of the code you added.
2. Add support for loop variant, using just the decreases keyword.
 - a. Make the necessary changes to the grammar, scanner and parser.
 - b. Instead of adding more productions to the while statement, factor the annotations into a non-terminal
 - c. Call the new AST node that holds loop variants and invariants
Jml LoopAnnotations.
 - d. Add a new class, Jml LoopVariant, and implement its constructor, resolve, analyze, and code generation methods.
 - e. Make any necessary changes to WhileLoop and Jml WhileLoop.
3. Add support for sequences of loop variants and invariants.
 - a. Look at the productions for SpecCaseSeq and Parser.consumeSpecCaseSeq() for ideas. Note the grammar is LALR, not LL as it was for JML2.
 - b. Question: How do the fields and methods of Jml WhileLoop need to change?
4. Add support for keyword synonyms, including _redundantly forms.
5. Add support for a label between the annotations and the while keyword.
 - a. Question: can an annotation be just a label? Why or why not?
 - b. Look at the implementation of LabeledStatement.
 - c. Question: can a solution be implemented that only changes the grammar and parser?
 - d. Hint: Once a Jml WhileStatement is on the AST stack, wrap it with a LabeledStatement.
6. Add support for annotated for and do-while loops.