**P4 Compiler - Semantic Checks Features & Technical Design Document - Draft**

**INDEX**

Low Level Design - Semantic Checks Pass ………….……………………………………. 2

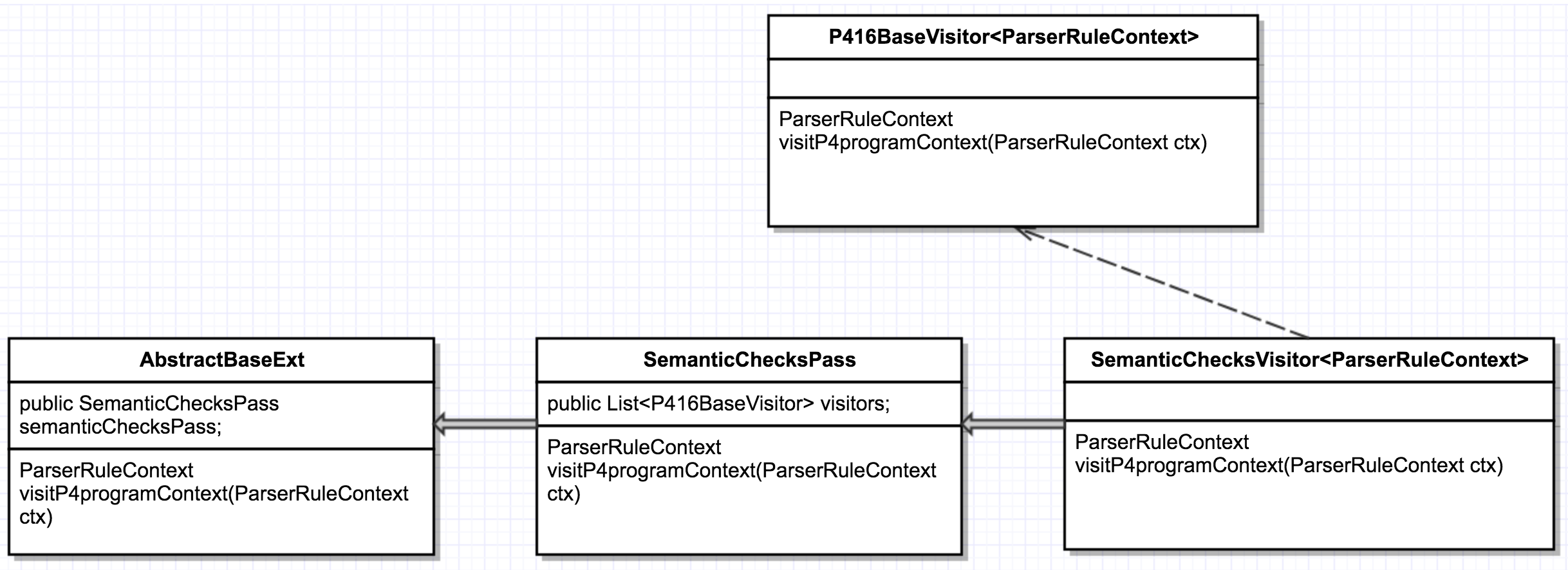
Class Diagram

Features of P4 Semantic Checker……….…………………………………………….......... 3

Semantic Checker Design Overview ……………………………………………………….... 3

**APPENDIX**

**Low Level Design - Semantic Checks Pass - Class Diagram**



**Features of P4 Semantic Checker**

1. Symbol Checker: Verifies whether Symbols or Variables are declared before their usage.
2. Type Checker: Verifies Type compatibility of operands in an expression and assignment statements.
3. Implementing checks for other P4 language semantics based on current p4c prototype compiler implementation.
4. Implementing checks specific to target hardware we are supporting.

**Semantic Checker Design Overview**

1. Semantic Checker is implemented based on Visitor Design Pattern.
2. The design intention is to perform the semantic checks without the need to modify the existing ANTLR generated code or to keep modifications to the minimum. The existing code is treated as an API with respect to the Semantic checker and is not modified unless there is a design need to do so.

For example, the Type interface is added with getTypeSize () and isTypeCompatible () methods which Semantic checker can make use of.

1. Multiple Visitors are implemented based on specific group (type) of semantic checks that are being done.
2. SemanticChecksPass is used as a singleton instance in AbstractBaseExt. The SemanticChecksPass class encapsulates all the visitors in it.
3. Each of the Visitor extends P416BaseVisitor<ParserRuleContext> which extends the ANTLR framework provided AbstractParseTreeVisitor<T>.