FPL Object Models and Algorithms

Derek Pappas

CTO

Fastpath Logic, Inc.

Programming conventions

- Coding convention
- Use types (See TypeDefs.h)
- Use STL
- Use Boost smart pointers
- Use implicit casting of inherited objects
- Do not cast objects

Overview

- Language
  - Syntax
  - Semantics
  - Scoping
  - Grammar

Overview (cont.)

- Lexer (csl.lexer.g)
- Parser (csl.parser.g)
- Abstract syntax tree (AST)
- Scope tree (SymTree.cpp)
- Tree Walker (csl.treewalker.g)
- Function Argument Checker
- CSL object model (CSLOM)

Overview (cont.)

- CSLOM checker
- CSLOM command
- CSLOM auto router
- Adapter
- Chip design object model

Overview (cont.)

- Visitors
  - Code generators
  - GUI controller
- GUI
  - QT

Boost

- Boost pointers are templates
- They wrap around the pointer and the pointer is accessed with the get() method
- Reference counters track the number of pointers to the object pointed to by the pointer
- Delete is called on the object when ref counter is zero
- shared_ptr<>
- weak_pointer<>

Cyclic depdencies

- Objects which point to one another have a cyclic dependency
- Example:
  - A parent object points to a child object
  - the child object points to the parent object

When does delete happen?

- How does the “implicit” garbage collector know when to delete the objects since they point to one another?

Break cyclic dependencies

- The delete methods does not count the weak pointer to the parent and when all other pointers to the parent object are deleted the parent object’s reference count is zero even thought the child object’s weak pointer still exists
- The parent object can contain a Boost shared_pointer<childPtr*>
- The child object can contain a Boost weak_ptr<parentPtr*>

Object Deletion

- The parent object is deleted
- After the parent object is deleted the child is deleted when the child object has no other pointers to it

creating OM objects

- constructors are private and are called by the OM class’s build() function
- The build() function checks that the arguments passed to the build function are correct.
- The build function calls the CDOMBase functions which populate the CDOMBase member variables such as CDOMBase* parent and m_children. As well as adding scoping and typing information to the CDOMBase parent.

class OMBase

- Contains a list of CDOMBase pointers (m_children) and a parent pointer which is used to construct the OM tree
- In addition the class also contains information about scoping
- All classes in the CDOM are derived from the CDOMBase class

OM Tree

- The OM tree is a representation of the AST which contains additional semantic information such as scoping and is checked for warnings and errors.
- The OM tree is constructed using the setParent(CDOMBase*) which called by the classes build() function

Visitor

- The visitor patterns uses a call back mechanism and inheritance to traverse the OM tree
- The Visitor class is inherited and the Visitor beforeTraversal, inTraversal, and afterTraversal methods are overridden.
- The traversal methods are passed a this pointer by the calling OM object’s acceptVisitor
- The traversal methods contain the logic which access the OM object’s state through the OM object’s methods.

Call back mechanism

- OM acceptVistor()
- Visitor visit()
- Visitor visit() methods traverse the OM by calling the OM acceptVisitor methods
- Each OM class has an acceptVisitor() method which calls back the next lower level of the Visitor visit() methods

Visitor call back

- The visitor visit function
  visit(CDOM<type>* pT)
  will call the corresponding leaf level visit function.
- For example
  visit(CDOMExpr*)
  will not call visit(CDOMExpr* pT) which does not exist but instead will call the CDOMExpr (e.g. CDOMExprOp, CDOMExprLink) visit function.