MicroTESK - Task #7678

Generation of LLVM configuration files from nML specifications

11/04/2016 11:41 AM - Andrei Tatarnikov

Status:	New	**Start date:**	11/04/2016
Priority:	Low	**Due date:**	
Assignee:	Alexander Kamkin	**% Done:**	0%
Category:	LLVM	**Estimated time:**	0.00 hour
Target version:	2.5	**Spent time:**	0.00 hour
Detected in build:	svn	**Published in build:**	

Description

Subject. To solve this task, you need implement an extension in Java that will traverse nML IR and generated the required files.

Such extensions implement the TranslatorHandler interface and are registered in the constuctor of the NmlTranslator class (see the code fragment below).

```
public final class NmlTranslator extends Translator<Ir> {
  private static final Set<String> FILTER = Collections.singleton(".nml");
  public NmlTranslator() {
    super(FILTER);

    getSymbols().defineReserved(NmlSymbolKind.KEYWORD, ReservedKeywords.JAVA);
    getSymbols().defineReserved(NmlSymbolKind.KEYWORD, ReservedKeywords.RUBY);

    // Detects parent-child connections between primitives
    addHandler(new ReferenceDetector());
    // Adds the list of root operations to IR
    addHandler(new RootDetector());

    addHandler(new ArgumentModeDetector());
    addHandler(new BranchDetector());
    addHandler(new MemoryAccessDetector());
    addHandler(new Analyzer(this));
    addHandler(new PrimitiveSyntesizer(this));
    addHandler(new ExceptionDetector());

    // Generate Java code of the ISA model
    addHandler(new MetaDataGenerator(this));
    addHandler(new Generator(this));
  }
```

All these handlers are examples of how to implement logic traversing IR. Some of them perform analysis and some generate code. Code generation is done using the StringTemplate library (STG files + java classes).

To traverse the IR, the following classes and interfaces can be used: IrVisitor, IrVisitorDefault and IrWalker (or its variations IrWalkerFlow, IrWalkerShortcuts) defined in the ru.ispras.microtesk.translator.nml.ir package. You need to implement IrVisitor (use IrVisitorDefault as default implementation with empty methods) and pass it to the most suitable walker. *NOTE: implementation of IR walker and visitor is raw and a subject to improvements. Any questions/feedback are appreciated.*

Expressions require using a separate walker and visitor implemented in Fortress: ExprTreeVisitor, ExprTreeVisitorDefault and ExprTreeWalker (ru.ispras.fortress.expression). Examples of using them you can find both in MicroTESK and in Fortress.

History

#1 - 01/27/2020 04:40 PM - Alexander Kamkin

- Target version changed from 2.4 to 2.5

- Priority changed from Normal to Low

- Assignee changed from Alexander Ulyanov to Alexander Kamkin

- Category set to LLVM