**Приложение Д**

**Исходный код**

**<Launcher.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import org.antlr.runtime.ANTLRFileStream;

import org.antlr.runtime.RecognitionException;

import proto.core.Compiler;

import proto.core.TinyCompiler;

import proto.generator.Generator;

import proto.generator.PromelaGenerator;

public class Launcher {

public static final Map<String, Generator> GENERATORS = new HashMap<String, Generator>() {{

put("promela", new PromelaGenerator());

}};

public void launch(Map<String, String> args) throws IOException, RecognitionException {

Generator generator = GENERATORS.get(args.get("target"));

Compiler compiler = new TinyCompiler(generator);

compiler.compile(new ANTLRFileStream(args.get("source")));

}

public static void usage() {

String usage = "Welcome to Proto Compiler 0.0.1! [http://github.com/hisohito/proto-code]\n\n" +"USAGE:\n" +

"\t java -cp proto-0.0.1.jar proto.Launcher [--target=<TARGET>] [--output=<OUTPUT\_FILE>] <SOURCE>\n\n" +

"\t <SOURCE> is an proto source file\n" +

"\t <TARGET> is one of supported targets (promela by default)\n" +

"\t <OUTPUT\_FILE> is an output file name (out.pml by default)\n";

System.out.println(usage);

}

public static void main(String[] args) throws IOException, RecognitionException {

Map<String, String> yeld = new HashMap<String, String>();

for(String arg: args) {

if (arg.matches("--\\w+=\\w+")) {

arg = arg.substring(2);

String[] parts = arg.split("=");

yeld.put(parts[0], parts[1]);

} else {

yeld.put("source", arg);

}

}

if (yeld.get("source") == null) {

usage();

return;

}

if (yeld.get("output") == null) yeld.put("output", "out.pml");

if (yeld.get("target") == null) yeld.put("target", "promela");

new Launcher().launch(yeld);

}

}

**<CallNode.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto.antlr;

import org.antlr.runtime.CommonToken;

import org.antlr.runtime.Token;

import org.antlr.runtime.tree.CommonTree;

public class CallNode extends CommonTree {

String classname;

public CallNode(int ttype) {

token = new CommonToken(ttype, "");

}

public CallNode(Token t) {

token = t;

}

public CallNode(int ttype, Token t, String cname) {

token = t;

this.classname = cname;

}

public String toString() {

String name = (token.getText().indexOf(".") != -1) ? token.getText()

.substring(token.getText().indexOf(".") + 1) : token.getText();

return this.classname + "::" + (token != null ? name : "");

}

}

**<ClassNode.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto.antlr;

import org.antlr.runtime.CommonToken;

import org.antlr.runtime.Token;

import org.antlr.runtime.tree.CommonTree;

public class ClassNode extends CommonTree {

public ClassNode(int ttype) {

token = new CommonToken(ttype, "");

}

public ClassNode(Token t) {

token = t;

}

public String toString() {

return (token != null ? token.getText() : "");

}

}

**<MethodNode.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto.antlr;

import org.antlr.runtime.CommonToken;

import org.antlr.runtime.Token;

import org.antlr.runtime.tree.CommonTree;

public class MethodNode extends CommonTree {

String classname;

public MethodNode(int ttype) {

token = new CommonToken(ttype, "");

}

public MethodNode(Token t) {

token = t;

}

public MethodNode(int ttype, Token t, String cname) {

token = t;

this.classname = cname;

}

public String toString() {

return this.classname + "::" + (token != null ? token.getText() : "");

}

}

**<PrototypeNode.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto.antlr;

import org.antlr.runtime.CommonToken;

import org.antlr.runtime.Token;

import org.antlr.runtime.tree.CommonTree;

public class PrototypeNode extends CommonTree {

String classname;

public PrototypeNode(int ttype) {

token = new CommonToken(ttype, "");

}

public PrototypeNode(Token t) {

token = t;

}

public PrototypeNode(int ttype, Token t, String cname) {

token = t;

this.classname = cname;

}

public String toString() {

String name = token.getText();

return this.classname + "::" + (token != null ? name : "");

}

}

**<SpecNode.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto.antlr;

import org.antlr.runtime.CommonToken;

import org.antlr.runtime.Token;

import org.antlr.runtime.tree.CommonTree;

public class SpecNode extends CommonTree {

public SpecNode(int ttype) {

token = new CommonToken(ttype, "");

}

public SpecNode(Token t) {

token = t;

}

public String toString() {

String name = token.getText();

return (token != null ? name : "");

}

}

**<StateNode.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto.antlr;

import org.antlr.runtime.CommonToken;

import org.antlr.runtime.Token;

import org.antlr.runtime.tree.CommonTree;

public class StateNode extends CommonTree {

String classname;

public StateNode(int ttype) {

token = new CommonToken(ttype, "");

}

public StateNode(Token t) {

token = t;

}

public StateNode(int ttype, Token t, String cname) {

token = t;

this.classname = cname;

}

public String toString() {

return classname + "::" + (token != null ? token.getText() : "");

}

}

**<Compiler.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto.core;

import org.antlr.runtime.ANTLRFileStream;

import org.antlr.runtime.RecognitionException;

public interface Compiler {

public String compile(ANTLRFileStream fs) throws RecognitionException;

}

**<AbstractCompiler.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto.core;

import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

import org.antlr.runtime.tree.CommonTree;

import proto.antlr.CallNode;

import proto.antlr.PrototypeNode;

import proto.antlr.StateNode;

import proto.generator.Generator;

import proto.ir.Automata;

public abstract class AbstractCompiler implements Compiler {

protected CommonTree mainTree;

protected Generator generator;

public AbstractCompiler(Generator generator) {

this.generator = generator;

}

private void findStates(CommonTree tree, Automata automata) {

if (tree != null) {

for (int i = 0; i < tree.getChildCount(); i++) {

CommonTree current = (CommonTree) tree.getChild(i);

if (current instanceof StateNode) {

automata.addState(current.toString());

}

findStates((CommonTree) tree.getChild(i), automata);

}

}

}

private CommonTree getSubTree(CommonTree tree, String name) {

if (tree != null) {

for (int i = 0; i < tree.getChildCount(); i++) {

String current = tree.getChild(i).toString();

if (current.equals(name)) {

return (CommonTree) tree.getChild(i);

}

CommonTree t = getSubTree((CommonTree) tree.getChild(i), name);

if (t != null)

return t;

}

return null;

}

return null;

}

private void formPaths(CommonTree tree, Automata automata) {

if (tree != null) {

for (int i = 0; i < tree.getChildCount(); i++) {

CommonTree currentNode = (CommonTree) tree.getChild(i);

String current = currentNode.toString();

if (currentNode instanceof CallNode) {

if (!(currentNode.getParent() instanceof StateNode))

continue;

if (automata.hasState(current)) {

// this is call if StateNode -> ad path to automata

CommonTree parent = (CommonTree) currentNode.getParent();

String from = parent.toString();

if (automata.hasState(from)) {

automata.addPath(from, current);

}

} else {

// if it is MethodCall - check if there are StateNodes in his depth

Collection<String> resolve = resolveCall(tree, current, automata);

if (resolve != null) {

// check all CallNode's in method

for (String r : resolve) {

CommonTree fromNode =

(CommonTree) currentNode.getParent();

while (!(fromNode instanceof StateNode)) {

fromNode = (CommonTree)

fromNode.getParent();

}

String from = fromNode.toString();

if (automata.hasState(from)) {

automata.addPath(from, r);

}

}

}

}

} else if (currentNode instanceof PrototypeNode){

formSpec(currentNode, automata);

break;

}

formPaths((CommonTree) tree.getChild(i), automata);

}

}

}

private String formSpec (CommonTree tree, Automata automata){

if (tree != null) {

String result = "";

for (int i = 0; i < tree.getChildCount(); i++) {

CommonTree current = (CommonTree) tree.getChild(i);

result = current.toString()+" ";

for (int j = 0; j < current.getChildCount(); j++) {

result += current.getChild(j).toString() +" ";

if (current.getChild(j).getChildCount() != 0){

result = formSpec((CommonTree) current.getChild(j),automata)+" ";

}

}

automata.addSpec(result);

}

}

return "";

}

private Collection<String> resolveCall(CommonTree tree, String id,

Automata automata) {

List<String> list = new ArrayList<String>();

// find exactly this MethodNode in tree

CommonTree methodNode = this.getSubTree(this.mainTree, id);

// if there is no childs - :`-(

if (methodNode == null || methodNode.getChildCount() == 0) {

return null;

} else {

// for all node childs:

// 1. if it is CallNode to StateNode - add to list

// 2. if it is CallNode to MethodNode - add to list recursive work result

// return list

for (Object child : methodNode.getChildren()) {

String callName = child.toString();

if (automata.hasState(callName)) {

list.add(callName);

} else {

List<String> addList = (List<String>) resolveCall(tree, callName, automata);

list.addAll(addList);

}

}

}

return list;

}

protected Automata compile0(CommonTree tree) {

mainTree = tree;

Automata automata = new Automata();

findStates(tree, automata);

formPaths(tree, automata);

automata.print();

return automata;

}

}

**<TinyCompiler.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto.core;

import org.antlr.runtime.ANTLRFileStream;

import org.antlr.runtime.RecognitionException;

import org.antlr.runtime.TokenRewriteStream;

import org.antlr.runtime.tree.CommonTree;

import proto.antlr.ProtoLexer;

import proto.antlr.ProtoParser;

import proto.generator.Generator;

import proto.ir.Automata;

public class TinyCompiler extends AbstractCompiler implements Compiler {

public TinyCompiler(Generator generator) {

super(generator);

}

public static void printTree(CommonTree t, int indent) {

if (t != null) {

StringBuffer sb = new StringBuffer(indent);

if (t.getParent() == null) {

System.out.println(sb.toString() + t.getText());

}

for (int i = 0; i < indent; i++)

sb = sb.append(" ");

for (int i = 0; i < t.getChildCount(); i++) {

System.out.println(sb.toString() + t.getChild(i));

printTree((CommonTree) t.getChild(i), indent + 1);

}

}

}

public String compile(ANTLRFileStream fs) throws RecognitionException {

ProtoParser parser = new ProtoParser(new TokenRewriteStream(new ProtoLexer(fs)));

CommonTree tree = (CommonTree) parser.program().getTree();

printTree(tree, 0);

Automata ir = compile0(tree);

return generator.generate(ir);

}

}

**<Generator.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto.generator;

import proto.ir.Automata;

public interface Generator {

public String generate(Automata automata);

}

**<PromelaGenerator.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto.generator;

import java.io.DataInputStream;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Collection;

import java.util.Collections;

import java.util.HashSet;

import java.util.Set;

import proto.ir.Automata;

public class PromelaGenerator implements Generator {

private Collection<String> generateSpecs(Automata automata) {

Set<String> allSpec = new HashSet<String>();

Collection<String> specs = automata.specs();

for (String spec : specs) {

// replace words

spec = spec.replaceAll("finally", "<>")

.replaceAll("globally", "[]")

.replaceAll("until", "U")

.replaceAll("release", "R");

spec = spec.replaceAll("not", "!")

.replaceAll("and", "&&")

.replaceAll("or", "||")

.replaceAll("if", "->");

String tmp[] = spec.split(" ");

StringBuilder oneSpec = new StringBuilder();

for (String string : tmp) {

if (!string.equals("R") && !string.equals("->")

&& !string.equals("||") && !string.equals("&&")

&& !string.equals("U") && !string.equals("!")

&& !string.equals("{") && !string.equals("}")

&& !string.equals("(") && !string.equals(")")

&& !string.equalsIgnoreCase("[]")

&& !string.equalsIgnoreCase("<>")) {

if (automata.hasState(string)) {

// insert state 'stateA1' expression

oneSpec.append("stateA1 == "+ string.toUpperCase().replace(':', '\_'));

} else {

System.err.println("ERROR: Wrong specification description!");

break;

}

} else {

oneSpec.append(string);

}

}

// send string to spin

oneSpec.insert(0, "!(");

oneSpec.append(")");

Runtime rt = Runtime.getRuntime();

Process proc;

try {

System.out.println("spin.exe -f \"" + oneSpec.toString()+"\"");

proc = rt.exec("spin.exe -f \"" + oneSpec.toString()+"\"");

int exitVal = proc.waitFor();

StringBuilder out = new StringBuilder();

DataInputStream din = new DataInputStream(proc.getInputStream());

String temp = "";

try {

while ((temp = din.readLine()) != null) {

out.append(temp);

}

allSpec.add(out.toString());

} catch (IOException e) {

System.exit(0);

}

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

return Collections.unmodifiableSet(allSpec);

}

public String generate(Automata automata) {

StringBuilder output = new StringBuilder();

output.append("int stateA1;\nint lastEvent;\n");

int index = 0;

String entryPoint = "";

// states info defines

for (String state : automata.states()) {

state = state.replace(':', '\_');

output.append("#define " + state.toUpperCase() + " " + index + "\n");

index++;

if (state.indexOf("main") != -1) {

entryPoint = state;

}

}

// main function begins here

output.append("\ninline A1() {\n");

// body start

output.append("\tstateA1 = "+ entryPoint.replace(':', '\_').toUpperCase() + ";");

output.append("\n\tdo\n");

for (String state : automata.states()) {

output.append("\t\t:: ( stateA1 == "+ state.replace(':', '\_').toUpperCase() + " ) ->\n");

if (automata.paths(state) != null) {

// printf for log

output.append("\t\t\tprintf(\"" + state + "\");\n");

output.append("\t\t\tif\n");

for (String path : automata.paths(state)) {

output.append("\t\t\t\t::stateA1 = "+ path.replace(':', '\_').toUpperCase() + ";\n");

}

output.append("\t\t\tfi;\n");

} else {

output.append("\t\t\tprintf(\"" + state + "\");\n");

if (state.indexOf("finalize") != -1){

output.append("\t\t\tbreak;\n");

}

}

}

// body end

output.append("\tod;\n}\n");

output.append("proctype Model() {\n");

output.append("\tA1();\n}\n");

output.append("init {\n\trun Model();\n}\n");

// spec time

Collection<String> ltlSpec = generateSpecs(automata);

// for each generated specification - create file

String defaultFilename = "C://Users//Jill//diploma//proto-code//target//protoOutput";

int number = 0;

for (String string : ltlSpec) {

File promela = new File(defaultFilename+(number++)+".pml");

try {

promela.createNewFile();

FileWriter fw = new FileWriter(promela);

fw.write(output.toString()+string);

fw.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

if (ltlSpec.isEmpty()){

File promela = new File(defaultFilename+".pml");

try {

promela.createNewFile();

FileWriter fw = new FileWriter(promela);

fw.write(output.toString());

fw.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

return "Success created "+number+" files";

}

}

**<Automata.java>**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Proto Project

//

// Copyright (c) 2012 Julia Shatilina <julia.shatilina@gmail.com>

//

// Licensed under the Apache License, Version 2.0 (the "License");

// You may not use this file except in compliance with the License.

// You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package proto.ir;

import java.util.Collection;

import java.util.Collections;

import java.util.HashMap;

import java.util.HashSet;

import java.util.Map;

import java.util.Set;

public final class Automata {

private Set<String> states;

private Map<String, Set<String>> paths;

private Set<String> LTLSpec;

public Automata() {

states = new HashSet<String>();

paths = new HashMap<String, Set<String>>();

LTLSpec = new HashSet<String>();

}

public void addState(String id) {

states.add(id);

}

public boolean hasState(String id) {

return states.contains(id);

}

public Collection<String> states() {

return Collections.unmodifiableSet(states);

}

public void addPath(String fromId, String toId) {

Set<String> path = paths.get(fromId);

if (path == null) {

path = new HashSet<String>();

path.add(toId);

paths.put(fromId, path);

} else {

path.add(toId);

}

}

public Collection<String> paths(String state) {

return paths.get(state);

}

public boolean hasPath(String fromId, String toId) {

return paths.get(fromId).contains(toId);

}

public void addSpec(String id) {

LTLSpec.add(id);

}

public Collection<String> specs() {

return Collections.unmodifiableSet(LTLSpec);

}

public void print() {

for (String state : this.states()) {

System.out.println(state);

if (this.paths(state) != null) {

for (String path : this.paths(state)) {

System.out.println("\t" + path);

}

}

}

}

}