package anatlyzer.evaluation.mutators.deletion;

import java.util.List;

import java.util.Random;

import org.eclipse.emf.ecore.EObject;

import org.eclipse.emf.ecore.EStructuralFeature;

import org.eclipse.emf.ecore.util.EDataTypeEList;

import org.eclipse.m2m.atl.core.emf.EMFModel;

import anatlyzer.atlext.ATL.LocatedElement;

import anatlyzer.atlext.ATL.Module;

import anatlyzer.atlext.ATL.OutPatternElement;

import anatlyzer.atlext.OCL.OperationCallExp;

import anatlyzer.evaluation.mutators.ATLModel;

import anatlyzer.evaluation.mutators.AbstractMutator;

public abstract class AbstractDeletionMutator extends AbstractMutator {

//private static final Container not = null;

//private static final Container and = null;

static int i=1;

static ATLModel wrapper ;

/\*\*

\* Generic deletion. It allows subtypes of both the container class and the class to delete.

\* @param atlModel

\* @param outputFolder

\* @param ContainerClass container class of the class of objects to delete (example OutPattern)

\* @param ToDeleteClass class of objects to delete (example Binding)

\* @param relation containment relation (example bindings)

\*/

protected <Container extends LocatedElement, ToDelete/\* extends LocatedElement\*/>

void genericDeletion(EMFModel atlModel, String outputFolder, Class<Container> ContainerClass, Class<ToDelete> ToDeleteClass, String relation) {

genericDeletion(atlModel, outputFolder, ContainerClass, ToDeleteClass, relation, false);

}

/\*\*

\* Generic deletion. It allows subtypes of the class to delete. The parameter 'exactContainerType'

\* allows configuring whether the type of the container must be exactly the one received, or if

\* the subtypes should be also considered.

\* @param atlModel

\* @param outputFolder

\* @param ContainerClass container class of the class of objects to delete (example OutPattern)

\* @param ToDeleteClass class of objects to delete (example Binding)

\* @param relation containment relation (example bindings)

\* @param exactContainerType false to consider also subtypes of the ContainerClass, true to discard subtypes of the ContainerClass

\*/

protected <Container extends LocatedElement, ToDelete/\* extends LocatedElement\*/>

void genericDeletion(EMFModel atlModel, String outputFolder, Class<Container> ContainerClass, Class<ToDelete> ToDeleteClass, String relation, boolean exactContainerType) {

// if (i==1){

this.save(atlModel, outputFolder);

System.out.println("111111");

// static final ATLModel wrapper = new ATLModel(atlModel.getResource());

if (i==1){

AbstractDeletionMutator.wrapper = new ATLModel(atlModel.getResource());

i=i+1;

}

System.out.println("2222");

System.out.println(atlModel.getResource());

List<Container> containers = (List<Container>)wrapper.allObjectsOf(ContainerClass);

// we will add a comment to the module, documenting the mutation

Module module = wrapper.getModule();

EDataTypeEList<String> comments = null;

System.out.println(comments);

if (module!=null) {

EStructuralFeature feature = wrapper.source(module).eClass().getEStructuralFeature("commentsBefore");

comments = (EDataTypeEList<String>)wrapper.source(module).eGet(feature);

}

//System.out.println("ccccccccccccc");

//System.out.println(comments);

// filter subtypes (only if parameter exactContainerType is true)

if (exactContainerType) filterSubtypes(containers, ContainerClass);

System.out.println("size");

boolean choose=false;

boolean choose2=false;

while(choose==false){

//Random randomGenerator = new Random();

int randomInt=(int) (Math.random() \* (containers.size()));

//int randomInt = randomGenerator.nextInt(containers.size()) + 1;

System.out.println("Random number generated is : " + randomInt);

// for (Container container : containers) {

System.out.println("container");

System.out.println(containers.size());

//EStructuralFeature feature = wrapper.source(container).eClass().getEStructuralFeature(relation);

EStructuralFeature feature = wrapper.source(containers.get(randomInt)).eClass().getEStructuralFeature(relation);

System.out.println(feature);

if (feature!=null) {

// CASE 1: monovalued feature .........................................................

if (feature.getUpperBound() == 1 && feature.getLowerBound() == 0) {

EObject link = (EObject) wrapper.source(containers.get(randomInt)).eGet(feature);

System.out.println("mono");

// mutation: remove object

if (link!=null) {

LocatedElement object = (LocatedElement)wrapper.target(link);

if (ToDeleteClass.isAssignableFrom(object.getClass())) {

wrapper.source(containers.get(randomInt)).eSet(feature, null);

// mutation: documentation

if (comments!=null) comments.add("\n-- MUTATION \"" + this.getDescription() + "\" " + toString(object) + " in " + toString(containers.get(randomInt)) + " (line " + object.getLocation() + " of original transformation)\n");

System.out.println("monovalue");

// save mutant

this.save(atlModel, outputFolder);

choose=true;

// restore: restore object and remove comment

//wrapper.source(containers.get(randomInt)).eSet(feature, link);

//if (comments!=null) comments.remove(comments.size()-1);

}

}

}

// CASE 2: multivalued feature ........................................................

else {

System.out.println("feature");

System.out.println(feature);

System.out.println("multi");

//List<EObject> link = (List<EObject>)wrapper.source(container).eGet(feature);

List<EObject> link = (List<EObject>)wrapper.source(containers.get(randomInt)).eGet(feature);

System.out.println(feature.getLowerBound());

System.out.println(link.size());

if (feature.getLowerBound() < link.size()) {

System.out.println("multivalue");

int size = link.size();

System.out.println(size);

Random randomGenerator2 = new Random();

//int randomInt2 = randomGenerator2.nextInt(link.size())+1;

int randomInt2=(int) (Math.random() \* (link.size()));

System.out.println("Random number generated is : " + randomInt2);

//for (int i=0; i<size; i++) {

// mutation: remove object

EObject eobject = link.get( randomInt2);

System.out.println("eobject");

System.out.println(toString(eobject));

LocatedElement object = (LocatedElement)wrapper.target(eobject);

System.out.println(object.getClass());

if (ToDeleteClass.isAssignableFrom(object.getClass())) {

link.remove( randomInt2);

// mutation: documentation

if (comments!=null) comments.add("\n-- MUTATION \"" + this.getDescription() + "\" " + toString(object) + " in " + toString(containers.get(randomInt)) + " (line " + object.getLocation() + " of original transformation)\n");

// System.out.println(feature.getLowerBound());

// System.out.println(link);

// save mutant

this.save(atlModel, outputFolder);

choose=true;

// restore: restore object and remove comment

//link.add( randomInt2, eobject);

//if (comments!=null) comments.remove(comments.size()-1);

this.save(atlModel, outputFolder);

}

//}

}

else{

choose=true;

}

}

}

//}

}

// i=i+1;

// }

//else{

// AbstractDeletionMutator.wrapper = new ATLModel(atlModel.getResource());

/\* List<Container> containers = (List<Container>)wrapper.allObjectsOf(OutPatternElement.class);

// System.out.println(containers.size());

boolean choose=false;

// System.out.println(containers.get(21));

// System.out.println(containers.get(21).eContainer());

// we will add a comment to the module, documenting the mutation

Module module = wrapper.getModule();

EDataTypeEList<String> comments = null;

System.out.println(comments);

if (module!=null) {

EStructuralFeature feature = wrapper.source(module).eClass().getEStructuralFeature("commentsBefore");

comments = (EDataTypeEList<String>)wrapper.source(module).eGet(feature);

}

int randomInt=(int) (Math.random() \* (containers.size()));

//while(choose==false){

// if(containers.get(randomInt).eContainer() )

choose=true;

//}

EStructuralFeature feature = wrapper.source(containers.get(randomInt)).eClass().getEStructuralFeature(relation);

List<EObject> link = (List<EObject>)wrapper.source(containers.get(randomInt)).eGet(feature);

int randomInt2=(int) (Math.random() \* (link.size()));

EObject eobject = link.get( randomInt2);

LocatedElement object = (LocatedElement)wrapper.target(eobject);

link.remove( randomInt2);

if (comments!=null) comments.add("\n-- MUTATION \"" + this.getDescription() + "\" " + toString(object) + " in " + toString(containers.get( randomInt2)) + " (line " + object.getLocation() + " of original transformation)\n");

this.save(atlModel, outputFolder);\*/

}

//}

}