Федеральное государственное бюджетное образовательное учреждение   
высшего образования

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | «Московский государственный технический университет  им. Н.Э. Баумана (национальный исследовательский университет)»(МГТУ им. Н.Э. Баумана) |

ФАКУЛЬТЕТ – Информатика и управления

КАФЕДРА – Информационные системы и телекоммуникации

РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе на тему

Веб-сервис хранения данных набора Eclipse Workspaces

Студент группы ИУ3-73 (подпись) 16.02.2017 Д.А. Козлов

Руководитель курсовой работы (подпись) 16.02.2017 А.М. Иванов

Москва, 2017

package ru.agentlab.tweety.arg.tests;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collection;

import org.junit.Assert;

import org.junit.Test;

import net.sf.tweety.arg.deductive.CompilationReasoner;

import net.sf.tweety.arg.deductive.DeductiveKnowledgeBase;

import net.sf.tweety.arg.deductive.accumulator.SimpleAccumulator;

import net.sf.tweety.arg.deductive.categorizer.ClassicalCategorizer;

import net.sf.tweety.arg.deductive.semantics.DeductiveArgumentNode;

import net.sf.tweety.arg.deductive.semantics.attacks.Defeat;

import net.sf.tweety.commons.ParserException;

import net.sf.tweety.logics.pl.parser.PlParser;

import net.sf.tweety.logics.pl.syntax.PropositionalFormula;

public class DeductiveTest {

private static final String RAINY = "rainy";

private static final String CLOUDY = "cloudy";

private static final String TODAY = "today";

private static final String ASHLEY = "ashley";

private static final String UMBRELLA = "umbrella";

private static final String TRAIN = "train";

private static final String SUNNY = "sunny";

private static final String WET = "wet";

private static final String AND = "&&";

private static final String OR = "||";

private static final String NOT = "!";

@Test

public void propositionTest() throws ParserException, IOException

{

DeductiveKnowledgeBase knowledgeBase = new DeductiveKnowledgeBase();

PlParser parser = new PlParser();

knowledgeBase.add((PropositionalFormula) parser.parseFormula(RAINY + AND + CLOUDY));

knowledgeBase.add((PropositionalFormula) parser.parseFormula(TODAY + AND + RAINY));

CompilationReasoner reasoner = new CompilationReasoner(knowledgeBase, new ClassicalCategorizer(), new SimpleAccumulator());

Assert.assertTrue(reasoner.query((PropositionalFormula) parser.parseFormula(TODAY + AND + CLOUDY)).getAnswerBoolean());

}

@Test

public void deductionTest() throws ParserException, IOException

{

PlParser parser = new PlParser();

Collection<PropositionalFormula> firstPersonSupport = new ArrayList<>();

//today ashley took umbrella

firstPersonSupport.add((PropositionalFormula) parser.parseFormula(TODAY));

firstPersonSupport.add((PropositionalFormula) parser.parseFormula(ASHLEY));

firstPersonSupport.add((PropositionalFormula) parser.parseFormula(ASHLEY + AND + UMBRELLA));

DeductiveArgumentNode firstPerson = new DeductiveArgumentNode(firstPersonSupport, (PropositionalFormula) parser.parseFormula(TODAY + AND + RAINY));

Collection<PropositionalFormula> secondPersonSupport = new ArrayList<>();

//if sunny then not rainy

secondPersonSupport.add((PropositionalFormula) parser.parseFormula(SUNNY + OR + NOT + RAINY));

//Ashley used train

secondPersonSupport.add((PropositionalFormula) parser.parseFormula(ASHLEY + AND + TRAIN));

DeductiveArgumentNode secondPerson = new DeductiveArgumentNode(secondPersonSupport, (PropositionalFormula) parser.parseFormula(TODAY + AND + NOT + RAINY));

Assert.assertFalse(Defeat.getInstance().isAttackedBy(firstPerson, secondPerson));

Assert.assertFalse(Defeat.getInstance().isAttackedBy(secondPerson, firstPerson));

//umbrella was wet

firstPersonSupport.add((PropositionalFormula) parser.parseFormula(UMBRELLA + AND + WET));

Assert.assertFalse(Defeat.getInstance().isAttackedBy(firstPerson, secondPerson));

Assert.assertFalse(Defeat.getInstance().isAttackedBy(secondPerson, firstPerson));

//if rainy then wet

firstPersonSupport.add((PropositionalFormula) parser.parseFormula(NOT + RAINY + OR + WET));

//today ashley goes by train

secondPersonSupport.add((PropositionalFormula) parser.parseFormula(ASHLEY + TRAIN));

secondPersonSupport.add((PropositionalFormula) parser.parseFormula(NOT + TODAY + OR + TRAIN));

Assert.assertFalse(Defeat.getInstance().isAttackedBy(firstPerson, secondPerson));

Assert.assertFalse(Defeat.getInstance().isAttackedBy(secondPerson, firstPerson));

//if umbrella was wet then it was rainy

firstPersonSupport.add((PropositionalFormula) parser.parseFormula(NOT + "(" + UMBRELLA + AND + WET + ")" + OR + RAINY));

Assert.assertTrue(Defeat.getInstance().isAttackedBy(firstPerson, secondPerson));

Assert.assertFalse(Defeat.getInstance().isAttackedBy(secondPerson, firstPerson));

}

}