WS1: Using PDDL in JACK

This workshop will introduce you with how to use PDDL and modeling files JavaFF code in the JACK environment.

In the first part you will use two pddl files: i) a domain file ii) a problem file. Collect these files from instructors.

You will also use the JavaFF library as a jar file and use it as an external library. Collect the JavaFF jar file from instructors

Make a new JACK project named AgentPlanning in a new folder of your name/group.

Copy the two pddl files in the project folder as well.

Firstly, You should be able to understand the domain and the planning problem in the given example. For this open the pddl files in a text editors. Read through the domain and problem files. Write small text based description of what the domain is? And what the planning problem is about? Show the description you have wrote to the instructors.

Now you should be able to run the given problem. For this Add a new design view named planningAgentDesign. In this view, add a new agent named “PlanningAgent”, a new event “StartPlanning” and a new plan named “DoPlanning”. Make sure that the agent, event and plan is within the package “planning”. The agent should post and handle the event. The Agent should be using the plan to handle the event. The design view should look like the one in figure below.

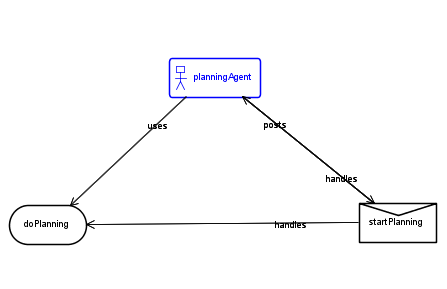


Figure1: design view for the planningAgent

How to program this you did it in exercise 2. Ask instructors for any help.

Next we need to use the JavaFF code. For this edit body of the plan and make sure it looks like this:

package planning;

import javaff.data.PDDLPrinter;

import javaff.data.UngroundProblem;

import javaff.data.GroundProblem;

import javaff.data.JavaFF\_Plan;

import javaff.data.TotalOrderPlan;

import javaff.data.TimeStampedPlan;

import javaff.parser.PDDL21parser;

import javaff.planning.State;

import javaff.planning.TemporalMetricState;

import javaff.planning.RelaxedTemporalMetricPlanningGraph;

import javaff.planning.HelpfulFilter;

import javaff.planning.NullFilter;

import javaff.scheduling.Scheduler;

import javaff.scheduling.JavaFFScheduler;

import javaff.search.Search;

import javaff.search.BestFirstSearch;

import javaff.search.EnforcedHillClimbingSearch;

import java.io.PrintStream;

import java.io.PrintWriter;

import java.io.File;

import java.io.FileOutputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.math.BigDecimal;

import java.util.Random;

public plan DoPlanning extends Plan {

#handles event StartPlanning ev;

public static BigDecimal EPSILON = new BigDecimal(0.01);

public static BigDecimal MAX\_DURATION = new BigDecimal("100000"); //maximum duration in a duration constraint

public static boolean VALIDATE = false;

public static Random generator = null;

public static PrintStream planOutput = System.out;

public static PrintStream parsingOutput = System.out;

public static PrintStream infoOutput = System.out;

public static PrintStream errorOutput = System.err;

static boolean relevant(StartPlanning ev)

{

return true;

}

context()

{

true;

}

#reasoning method

body(){

EPSILON = EPSILON.setScale(2,BigDecimal.ROUND\_HALF\_EVEN);

MAX\_DURATION = MAX\_DURATION.setScale(2,BigDecimal.ROUND\_HALF\_EVEN);

generator = new Random();

File domainFile = new File(ev.domainPath);

File problemFile = new File(ev.problemPath);

File solutionFile = null;

JavaFF\_Plan doPlan = doPlan(domainFile,problemFile);

if (solutionFile != null && doPlan != null) writePlanToFile(doPlan, solutionFile);

}

public static JavaFF\_Plan doPlan(File dFile, File pFile)

{

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

// Parse and Ground the Problem

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

long startTime = System.currentTimeMillis();

UngroundProblem unground = PDDL21parser.parseFiles(dFile, pFile);

if (unground == null)

{

System.out.println("Parsing error - see console for details");

return null;

}

//PDDLPrinter.printDomainFile(unground, System.out);

//PDDLPrinter.printProblemFile(unground, System.out);

GroundProblem ground = unground.ground();

long afterGrounding = System.currentTimeMillis();

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

// Search for a plan

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

// Get the initial state

TemporalMetricState initialState = ground.getTemporalMetricInitialState();

State goalstaten = performFFSearch(initialState);

long afterPlanning = System.currentTimeMillis();

TotalOrderPlan top = null;

if (goalstaten != null) top = (TotalOrderPlan) goalstaten.getSolution();

if (top != null) top.print(planOutput);

/\*javaff.planning.PlanningGraph pg = initialState.getRPG();

Plan plan = pg.getPlan(initialState);

plan.print(planOutput);

return null;\*/

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

// Schedule a plan

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

//TimeStampedPlan tsp = null;

//if (goalState != null)

//{

//infoOutput.println("Scheduling");

//Scheduler scheduler = new JavaFFScheduler(ground);

//tsp = scheduler.schedule(top);

//}

//long afterScheduling = System.currentTimeMillis();

//if (tsp != null) tsp.print(planOutput);

double groundingTime = (afterGrounding - startTime)/1000.00;

double planningTime = (afterPlanning - afterGrounding)/1000.00;

//double schedulingTime = (afterScheduling - afterPlanning)/1000.00;

infoOutput.println("Instantiation Time =\t\t"+groundingTime+"sec");

infoOutput.println("Planning Time =\t"+planningTime+"sec");

//infoOutput.println("Scheduling Time =\t"+schedulingTime+"sec");

return top;

}

private static void writePlanToFile(JavaFF\_Plan pplan, File fileOut)

{

try

{

FileOutputStream outputStream = new FileOutputStream(fileOut);

PrintWriter printWriter = new PrintWriter(outputStream);

pplan.print(printWriter);

printWriter.close();

}

catch (FileNotFoundException e)

{

errorOutput.println(e);

e.printStackTrace();

}

catch (IOException e)

{

errorOutput.println(e);

e.printStackTrace();

}

}

public static State performFFSearch(TemporalMetricState initialState) {

// Implementation of standard FF-style search

infoOutput.println("Performing search as in FF - first considering EHC with only helpful actions");

// Now, initialise an EHC searcher

EnforcedHillClimbingSearch EHCS = new EnforcedHillClimbingSearch(initialState);

EHCS.setFilter(HelpfulFilter.getInstance()); // and use the helpful actions neighbourhood

// Try and find a plan using EHC

State goalState = EHCS.search();

if (goalState == null) // if we can't find one

{

infoOutput.println("EHC failed, using best-first search, with all actions");

// create a Best-First Searcher

BestFirstSearch BFS = new BestFirstSearch(initialState);

// ... change to using the 'all actions' neighbourhood (a null filter, as it removes nothing)

BFS.setFilter(NullFilter.getInstance());

// and use that

goalState = BFS.search();

}

return goalState; // return the plan

}

}

The agents code should look like this:

package planning;

/\*\*

\*

\*/

public agent PlanningAgent extends Agent {

#posts event StartPlanning ev;

#handles event StartPlanning;

#uses plan DoPlanning;

public PlanningAgent(String name)

{

super(name);

}

public void performPlanning(String domainPath, String problemPath){

postEvent(ev.startPlanning(domainPath, problemPath));

}

}

The event code should look like this:

package planning;

/\*\*

\*

\*/

public event StartPlanning extends Event {

public String domainPath="";

public String problemPath="";

#posted as

startPlanning(String d, String p){

domainPath=d;

problemPath=p;

}

}

The main file should look like. Make sure that the class name has the same name as the main file:

import planning.PlanningAgent;

public class TestPlanning

{

public static void main (String args[]) {

PlanningAgent planner=new PlanningAgent("planner");

planner.performPlanning(args[0],args[1]);

}

}

Compile the program and run the program with these parameters (specified in the “Extra Args:” field):

/path/of/domain/file/domain.pddl /path/of/problem/file/pfile01

Make sure that the domain.ppdl and pfile01 file are in the same folder as your JACK-program.

Show the result to instructors.

Next we are supposed to implement the planning problem you modeled yesterday as exercise.

Show your completed modeling work to instructors. They will guide you further.