Cognitive Rovios – Connection to the SNePS System

By Peter Hoellig

May 6th, 2009

Abstract

A main task of creating a cognitive robot is establishing the ability to control a robot based on cognitive decisions made by the robot itself or by some attached cognitive system. This paper is a report on establishing a connection between a Rovio robot and the SNePS reasoning system using the GLAIR architecture. The Rovio platform will be discussed in detail, as will the portions of SNePS and GLAIR that are relevant to their connection. While cognitive action is always the primary concern of cognitive robotics, consideration of what primitive actions and combinations of those actions can or should be available to the reasoning system at the knowledge layer of the GLAR architecture will also be discussed.

1. Introduction

Of the many tasks involved in the creation of a cognitive robot, one of the most important is ensuring that the robot is capable of cognitive actions. Most commercially available robots are not capable of performing actions of their own will. Instead they are typically capable of a small set of predefined actions that will be executed when input is given from a user. Furthermore, they do not reason about their circumstances except where it is explicitly forced into its predefined actions. Because of this, any attempts of creating a cognitive robot from a commercially available robot must make use of some interface between the robots predefined actions and some system capable of performing the cognitive processing while leaving the execution to the robot itself.

In some robots, code can be downloaded to the robots memory and executed. This allows the robot to formulate and act cognitively as a autonomous system by simply downloading the cognitive system to the robots memory and allowing it to execute on the robot. However, many robots do not support this feature and thus must outsource the cognitive function to some other entity (typically a laptop computer with some kind of connection to the robot) who takes control of the robot. Such is the case with the Rovio (WowWee 2008) robot (henceforth referred to as DAFT (Delivering Acronym Focused Technology) that is used in this paper. Since it is not capable of cognitive action on its own, we have created an interface to handle communication with DAFT using the Java programming language. This Java API for DAFT is then connected to the SNePS (Shapiro 2000) reasoning system (which is LISP based) and serves as the SAL layer of the GLAIR (Shapiro 2003) architecture for this project. This connection is what allows DAFT to act cognitively as it allows for its sensory input to be processed into useful information about the current state of the world which can then be used in reasoning.

2. DAFT – Capabilities and Limitations

DAFT can perform a wide variety of simple actions and is also capable of reporting on many different aspects of its current status. As it stands however, DAFT is incapable of acting in response to his current status except in a few routines in which it is explicitly told to do so. While DAFT does allow for his basic actions to be combined into less simple actions, it does not benefit from this strongly as it does not allow for any controls to be placed in these combinations. What DAFT lacks in intelligence however, it more than makes up for in his capabilities.

DAFT is equipped with an assortment of features with which it can gather data about the world. DAFT possesses a microphone and speaker with which it can hear and speak to the world. Unfortunately, DAFT does not possess any abilities related to his innate ability to hear and speak except that it can pass sounds to and from a connected user. DAFT is also equipped with a camera that enables him to see. However, DAFT possesses no image processing capabilities and cannot tell the difference between an open area and the wall it is repeatedly crashing into. DAFT can determine if there is an object in front of him that would prevent him from moving forward by using an IR sensor that it has. Unfortunately, this sensor is somewhat temperamental and will often activate even when nothing is in front of DAFT or if it on certain types of floors. DAFT is a sturdy fellow and keeps his wheels on the ground quite well, it can move in eight directions and turn at various speeds based on input from a user.

DAFT is accessible to users who are on the same network as it and are capable of authenticating to the web server that it runs. DAFT will acquire a network connection upon configuration and can be accessed by any users on the same network through an interface provided by his web server. Commands issued to this interface are converted to the appropriate CGI-scripts which make up DAFT's basic actions. These scripts are accessible directly without the interface to anyone who knows their URLs and functions, which are defined in the Rovio API (citation to be included). It is these scripts that we will use to control DAFT based on the information that it can gather from the world, but first we must extend his senses a little.

3. The Java API

We have created a Java interface for DAFT to replace his standard user interface with one that can be automated and one that can extend his senses somewhat. First we must establish a connection to DAFT, this is done by connecting to DAFT's web server and issuing requests to executing DAFT's CGI scripts so that it will take pictures, move, return sensor data, etc. All of DAFT's useful scripts are encapsulated into Java methods in a class called RovioPrimitive. This class will be referenced by another Java class called RovioActions which will contain combinations of DAFT's actions and extensions of DAFT's senses and serve as the PML layer of the GLAIR architecture which we shall discuss in section 4. DAFT's sensory reports are also given methods that return only the needed data instead of all of the information that the script returns in order to prevent propagating unneeded information up to the cognitive area.

DAFT's IR sensor is expanded on by creating methods in which it is used to avoid obstacles while moving instead of merely popping up when an obstacle is in the way. DAFT's camera has been extended such that at this level DAFT is capable of discerning the color of whatever is in the center of his field of vision, as well as the ability to differentiate between twelve different colors and to search out certain colors on request. Others have extended this further by adding object discernment to DAFT's vision capabilities (Settembre 2009). Regardless of our particular implementation, this interface allows for whatever entity is connecting to DAFT to understand what DAFT sees using appropriate vision processing techniques even though DAFT himself does not understand. We have not extended DAFT's sense of hearing and ability to speak for this project. However, others have shown that it is possible with the SNePS system to implement some natural language recognition and understanding in DAFT (Kafender 2009).

3b) The Java API – Technical Details

The Java API makes use of DAFT's built in CGI scripts for control. When we want DAFT to perform an action we create an HTTP connection to DAFT's web server and execute the script causing DAFT to perform the requested action. This is done by creating a URL object from the java.net package using the web address of DAFT with the script location concatenated onto the end. Once done, the script will execute and we read any returned information into a buffer. A comprehensive list of all of DAFT's scripts is available from WowWee (WowWee 2008). We have chosen not to implement some of DAFT's scripts as Java methods as they are not all of use for our work. However, most of the scripts have been turned into Java methods by formatting the HTTP request string to match the location of the script and then making the request. We have created the following methods for use in our API and will comment on the ones that we have used in our implementation. For descriptions of all the methods implemented in the API please see Appendix A.

```
public void ManualDrive(int direction, int speed)
public void GetReport()
public void GetMCUReport()
public BufferedImage GetBufferedImage()
public Image GetImage()
public void ChangeResolution(int value)
public boolean IRBarrier()
public boolean IRBarrierMCU()
```

The ManualDrive method takes in an integer for the desired direction and another integer for the desired speed. This method accounts for all of the movements that DAFT can perform with eighteen directional options and ten speed options. To see the possible values of the parameters, see Appendix A for our Java code or the Rovio API from WowWee.

The GetReport() and GetMCUReport() methods call scripts which cause DAFT to return a large amount of sensory information in text form. This text is parsed and processed by various methods to aid in DAFTs understanding of its surroundings.

The GetBufferedImage() and GetImage() methods are the core of DAFT's vision system. Their calling causes DAFT to take a still picture and send it to the Java program for processing. All of DAFT's vision capabilities come from pictures that DAFT takes with these methods. The ChangeResolution method alters the resolution of the pictures that DAFT takes. Currently all of DAFT's vision processing methods are designed to use 640 x 480 images so this method is used to make sure that the pictures that DAFT is sending back are the right size.

The IRBarrier() and IRBarrierMCU() methods are used for DAFT's obstacle avoidance processes. They work by calling GetReport() and GetReportMCU() (respectively) and processing the report to discover the value returned by DAFT's IR sensor which informs the caller as to whether or not there is something in DAFT's way. It should be noted that the IR sensor is somewhat finicky in that it often returns that there are objects in front of it when there are none and so DAFT's movement capabilities are somewhat negatively impacted. For more information on all of the accessible Java methods please refer to the source code for the API located in appendix A.

4. GLAIR And Why We Chose It

Up to this point we have made passing reference to the GLAIR architecture without actually defining it. "GLAIR (grounded layered architecture with integrated reasoning) is a three-level architecture for cognitive robots and intelligent autonomous agents." (Shapiro 2003) GLAIR consists of three distinct layers that make up the cognitive system. First is the sensori-actuator level (SAL) which as implied by the name is where the system's senses and acts in the world. The next level up is the perceptuo-motor level (PML), this level is the middle ground of the system.

The PML layer handles primarily perception and motor control; it allows a certain amount of abstraction away from the SAL layer. The PML layer is where the senses are processed and turned into perception. For instance, the SAL layer may take a picture and then the PML layer will process that image to determine that the image is of a blue ball. It is used to process the sensory information, sorting out what should be passed up and what should be ignored. It also allows for refinement of motor functions from simple actions (move, turn) into more complex actions (move to the keys and pick them up). Further up the chain and the highest level in the system is the knowledge level (KL). The KL is responsible for understanding the input from the PML layer and reasoning about it. Although the KL is capable of acting based on input from some user it can also act on its own with goals and intentions rather than waiting for input from the world or some user to tell it what to do.

In our case, the GLAIR architecture is a perfect match for our setup due to its inherent separation of the sensory, perception, and reasoning layers. This separation is incredibly useful in our case because of our approach of connecting DAFT to Java which handles vision processing, refines the movement capabilities, and includes several useful behaviors for controlling DAFT. Furthermore, our intention is to then connect this Java code to the SNePS system for the purposes of reasoning, natural language understanding, and autonomous control. GLAIR is a perfect fit for our design with DAFT as the SAL, our Java code as the PML, and SNePS as the KL.

5. The PML

While we have already discussed the functional capabilities of our Java code in section 3, there is some discussion to be had as to what kinds of actions should be located in this level given its middle ground status. The PML is in a peculiar position because its interactions are entirely with other parts of the cognitive system; it does not interact directly with the world or with other entities. The PML layer is much like a generic API for use of the system it receives input from. It must be designed specifically for the system it interacts with, but because of this any KL must formulate its requests for action with respect to whatever PML implementation is being used.

While a PML layer by definition must be designed to work with and enhance one type of system, reasoning systems (which often make up the KL) are typically designed to interact with only itself and a user. Because of this, establishing the link between the PML and the KL tends to be awkward. Setting up the PML as a user of the KL to make use of its natural interface is generally out of the question because it places control into the PML layer rather than the KL layer. The approach which we are taking is creating an interface for the PML inside of the SNePS system. This will allow us to interact with the PML directly from the KL without stripping the PML of its generic design.

5b) The PML – Technical Details

The PML layer is made up of Java code which extends DAFT's capabilities as well as provides an interface for use by the KL. The sections of the code which are made available to the KL in our implementation will be discussed here. For more information on all of the methods that make up the PML and implementation level details of the methods we shall discuss here, please refer to the source code of the PML layer in appendix B. We make use of the following methods for our interface with the KL.

```
public   void Wander()
public void LeftWall()
public void LookAt(String color)
public void goBackward()
public void goRight()
public void goLeft()
public void goForward()
public void GoToFocussedObject()
public void getFocussedObject()
public void FollowFocussedObject()
```

The Wander() method is the first completely autonomous method that was written for DAFT. This method simply moves around the room while checking to make sure that it isn't about to run into any obstacles and it if is then it turns in a random direction until such a time as there is no longer an object in front of it and proceeds moving forward.

The LeftWall() method does as its name implies. DAFT will find a wall, place it on its left hand side and then follow along the wall. In the event that the wall ends, DAFT continues to follow walls to its left regardless of whether or not they are the original wall. Also, if there is a large enough space where the wall that DAFT is following ends before another wall begins (for example a doorway) then DAFT will turn into that space following around the wall to the other side of it.

The LookAt() method is one of DAFTs vision functions that is implemented at the PML layer. This function takes in a string representation of a color and if it understands what color it has been told to look at DAFT will then move around the room until it has found something of that color and placed it in the center of its field of vision. The colors that DAFT understands are red, orange, yellow, lime, green, emerald, cyan, light blue, blue, orchid, purple, and pink. There is an alternate version of this method which takes in a java.awt.color instead of a string representation of a color. This provides some additional understanding as there are some colors in java.awt that are not above, although the color recognition algorithm does not support white and black.

The GoToFocussedObject() method will cause DAFT to move next to an object of the same color as the object that it most recently looked at, went to, or found. This is typically called after LookAt as the way to move next to an object of a certain color.

The getFocussedObject() method simply returns the last color that DAFT looked at, found, or went to.

The FollowFocussedObject() method causes DAFT to move next to an object that is the same color that it most recently looked at, went to, or found. Subsequently, DAFT will continue to move next to that object as the object moves away from it.

The Go…() methods cause DAFT to move in the direction specified relative to it. For example, the GoForward() method will cause DAFT to move straight ahead for about one second. These methods do not alter the direction that DAFT is facing, so GoLeft() and GoRight() is more of a sidestep than a change in direction.

6. SNePS And The KL

The SNePS system has been used for many cognitive projects in the past and is a natural fit for our intentions. SNePS brings many features to the table that allow for cognitive tasks such as natural language processing, a relevance logic based reasoning system, episodic memory, sense of time and much more. We have borrowed much from previous SNePS projects, primarily the SNePS FEVAHR (foveal extravehicular activity helper-retriever) robot.

The FEVAHR robot implementation that we borrowed from was a software robot implementation that used the GLAIR architecture. This FEVAHR existed in a Java based virtual world. Java code made up the SAL, LISP the PML, and SNePS the KL. It was able to find objects in the virtual world based on a few colors and shapes, carry on conversations with the end user as well as other entities that existed in the world. SNePS provided its conversational ability as well as its memory which made it able to helpfully interact with the members of the world. We are borrowing heavily from this previous example at the KL because we desire a similar implementation, but in a physical robot. Essentially, the SNePS code should not have to be greatly modified to act as DAFT's control other than modifying it to interact with our PML which we discussed briefly in section 5. For further information on this interface, please see appendix C which contains our modification of the original FEVAHR code.

Once this connection has been properly established, the whole of SNePS's cognitive capabilities will be conjoined with DAFT's impressive array of basic actions that have been augmented by our PML layer. This setup allows for the SNePS system to move and act in the real world in a practical way and allows for the formulation of real world reasoning tasks that can test the viability of the system as a whole. These tests can also show the places in the system that need to be improved upon. In this way the system can begin evolving into a larger and more robust cognitive system that can handle many different types of tasks.

References

Kandefer, Michael W. (2009), “Cassie Can Speak: A .NET Interface to the SNePS Fevahr”.

Settembre, Scott (2009). “Controlling Rovio: Using RovioWrap and .NET to Control a Robot”.

Stuart C. Shapiro and Haythem O. Ismail (2003), “Anchoring in a Grounded Layered Architecture with Integrated Reasoning”, Robotics and Autonomous Systems 43, 2-3, 97-108.

Shapiro, Stuart C. (2000), "SNePS: A Logic for Natural Language Understanding and Commonsense Reasoning" [Postscript], in Lucja M. Iwanska, & Stuart C. Shapiro (eds.), Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language (Menlo Park, CA/Cambridge, MA: AAAI Press/MIT Press): 175-195.

WowWee Group Limited (2008), API Specification for Rovio Version 1.2, http://www.robocommunity.com/downloads/file/1425/Rovio_API_Specifications_v1.2.pdf

Appendix A – Java API for Primitive Actions Code

```
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;

import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;
import javax.imageio.*;
import javax.swing.JPanel;

import org.apache.commons.httpclient.HttpClient;

import org.apache.commons.httpclient.methods.GetMethod;


import sun.awt.image.URLImageSource;

public class RovioPrimitive extends JPanel {
        private HttpClient _client;
        private GetMethod _method;
        private String _currenturl;
        private String _server;
        private String[] _httpresult;
        private int _okay;
//      private EX17C _app;
        private Image _img;
        private BufferedImage _bufimg;
        public RovioPrimitive(String server)
        {
                _server = server;
                _client = new HttpClient();
                _method = new GetMethod("");
                _okay = 0;
//              _app = new EX17C();
                _httpresult = new String[50];
        }

        /*
         * private void CallMethod(String url) { // Create a method instance.
         * _method = new GetMethod(url);
         *
         *  // Provide custom retry handler is necessary
         * _method.getParams().setParameter(HttpMethodParams.RETRY_HANDLER,
new
         * DefaultHttpMethodRetryHandler(3, false));
         *
         * try { // Execute the method. int statusCode =
         * _client.executeMethod(_method);
         *
         * if (statusCode != HttpStatus.SC_OK) { System.err.println("Method failed: " +
         * _method.getStatusLine()); }
         *  // Read the response body. byte[] responseBody =
         * _method.getResponseBody();
         *  // Deal with the response. // Use caution: ensure correct character
         * encoding and is not binary data System.out.println(new
         * String(responseBody));
         *  } catch (HttpException ex) { System.err.println("Fatal protocol
         * violation: " + ex.getMessage()); ex.printStackTrace(); } catch
         * (IOException e) { System.err.println("Fatal transport error: " +
         * e.getMessage()); e.printStackTrace(); } finally { // Release the
         * connection. _method.releaseConnection(); } }
         */
        public void CallMethod2(String url)
        {
                int i =1;
                URL site;
                try
                {
                        site = new URL(url);

                        BufferedReader in = new BufferedReader(new
InputStreamReader(site.openStream()));

                        String inputLine;

                        while ((inputLine = in.readLine()) != null)
                        {
                                //System.out.println(inputLine);
                                _httpresult[i] = inputLine;
                                i++;
                        }

                        in.close();

                }
                catch (MalformedURLException e)
```

```
		{
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		catch (IOException e)
		{
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}

	public void ManualDrive(int direction, int speed)
	{
		/*
		 * d_value = 0 (Stop) 1 (Forward) 2 (Backward) 3 (Straight left) 4
		 * (Straight right) 5 (Rotate left by speed) 6 (Rotate right by speed) 7
		 * (Diagonal forward left) 8 (Diagonal forward right) 9 (Diagonal
		 * backward left) 10 (Diagonal backward right) 11 (Head up) 12 (Head
		 * down) 13 (Head middle) 14 (Reserved) 15 (Reserved) 16 (Reserved) 17
		 * (Rotate left by 20 degree angle increments) 18 (Rotate right by 20
		 * degree angle increments) s_value = 1 (fastest) – 10 (slowest)
		 */
		_currenturl = _server + "/rev.cgi?Cmd=nav&action=18&drive=" + direction
				+ "&speed=" + speed;
		this.CallMethod2(_currenturl);
	}
	public void GetReport()
	{
		_currenturl = _server + "/rev.cgi?Cmd=nav&action=1";
		this.CallMethod2(_currenturl);
	}
	public void StartRecording()
	{
		// Start recording a path.
		_currenturl = _server + "/rev.cgi?Cmd=nav&action=2";
		this.CallMethod2(_currenturl);
	}

	public void AbortRecording()
	{
		// Terminates recording of a path without storing it to flash memory.
		_currenturl = _server + "/rev.cgi?Cmd=nav&action=3";
		this.CallMethod2(_currenturl);
	}

	public void StopRecording(String pathname)
	{
		/*
		 * Stops the recoding of a path and stores it in flash memory;
		 * javascript will give default name if user does not provide one.
		 */
		_currenturl = _server + "/rev.cgi?Cmd=nav&action=4&name=" +
pathname;
		this.CallMethod2(_currenturl);
	}

	public void DeleteRecording(String pathname)
	{
		/*
		 * Stops the recoding of a path and stores it in flash memory;
		 * javascript will give default name if user does not provide one.
		 */
		_currenturl = _server + "/rev.cgi?Cmd=nav&action=5&name=" +
pathname;
		this.CallMethod2(_currenturl);
	}

	public void GetPathList()
	{
		// Returns a list of paths stored in the robot.
		_currenturl = _server + "/rev.cgi?Cmd=nav&action=6";
		this.CallMethod2(_currenturl);
	}

	public void PlayPathForward(String pathname)
	{
		// Replays a stored path from closest point to the end;
		// If the NorthStar signal is lost, it stops.
		_currenturl = _server + "/rev.cgi?Cmd=nav&action=7";
		this.CallMethod2(_currenturl);
	}

	public void PlayPathBackward(String pathname)
	{
		// Replays a stored path from closest point to the end;
		// If the NorthStar signal is lost, it stops.
		_currenturl = _server + "/rev.cgi?Cmd=nav&action=8";
		this.CallMethod2(_currenturl);
	}

	public void StopPlaying()
	{
        // Stop playing a path
        _currenturl = _server + "/rev.cgi?Cmd=nav&action=9";
        this.CallMethod2(_currenturl);
}

public void PausePlaying()
{
        // Pause playing a path
        _currenturl = _server + "/rev.cgi?Cmd=nav&action=10";
        this.CallMethod2(_currenturl);
}

public void RenamePath(String oldpath, String newpath)
{
        // Rename the old path
        _currenturl = _server + "/rev.cgi?Cmd=nav&action=11&name=" + oldpath
                        + "&newname=" + newpath;
        this.CallMethod2(_currenturl);
}

public void GoHome()
{
        // Drive to home location in front of charging station.
        _currenturl = _server + "/rev.cgi?Cmd=nav&action=12";
        this.CallMethod2(_currenturl);
}

public void GoHomeAndDock()
{
        // Drive to home location in front of charging station and dock
        _currenturl = _server + "/rev.cgi?Cmd=nav&action=13";
        this.CallMethod2(_currenturl);
}

public void UpdateHomePosition()
{
        // Define current position as home location in front of charging
        // station.
        _currenturl = _server + "/rev.cgi?Cmd=nav&action=14";
        this.CallMethod2(_currenturl);
}

public void SetTuningParameters()
{
        // Change homing, docking and driving parameters – speed for driving
        // commands
        _currenturl = _server + "/rev.cgi?Cmd=nav&action=15";
        this.CallMethod2(_currenturl);
}

public void GetTuningParameters()
{
        // Change homing, docking and driving parameters – speed for driving
        // commands
        _currenturl = _server + "/rev.cgi?Cmd=nav&action=16";
        this.CallMethod2(_currenturl);
}

public void ResetNavStateMachine()
{
        // Stops whatever it was doing and resets to idle state.
        _currenturl = _server + "/rev.cgi?Cmd=nav&action=17";
        this.CallMethod2(_currenturl);
}

public void GetMCUReport()
{
        // Returns MCU report including wheel encoders and IR obstacle
        // avoidance.
        _currenturl = _server + "/rev.cgi?Cmd=nav&action=20";
        this.CallMethod2(_currenturl);
}

public void ClearAllPaths()
{
        // Deletes all paths in FLASH memory
        _currenturl = _server + "/rev.cgi?Cmd=nav&action=21";
        this.CallMethod2(_currenturl);
}

public void GetStatus()
{
        // Reports navigation state
        _currenturl = _server + "/rev.cgi?Cmd=nav&action=22";
        this.CallMethod2(_currenturl);
}

public void SaveParameter(long index, long value)
{
        // Stores parameter in robot's flash
        /*
         * index = 0 – 19 value = 32bit signed integer
```

```
	 */
	_currenturl = _server + "/rev.cgi?Cmd=nav&action=23&index=" + index
			+ "&value=" + value;
	this.CallMethod2(_currenturl);
}

public void ReadParameter(long index)
{
	// Stores parameter in robot's flash
	/*
	 * index = 0 – 19
	 */
	_currenturl = _server + "/rev.cgi?Cmd=nav&action=value&index=" + index;
	this.CallMethod2(_currenturl);
}

public void GetLibNSVersion()
{
	// Returns string version of libNS and NS sensor
	_currenturl = _server + "/rev.cgi?Cmd=nav&action=25";
	this.CallMethod2(_currenturl);
}

public void EmailImage(String email)
{
	// Emails current image or if in path recording mode sets an action
	_currenturl = _server + "/rev.cgi?Cmd=nav&action=26&email=" + email;
	this.CallMethod2(_currenturl);
}

public void ResetHomeLocation()
{
	// Clears home location in flash
	_currenturl = _server + "/rev.cgi?Cmd=nav&action=27";
	this.CallMethod2(_currenturl);
}

// Get Data has issues, fix it later.
public BufferedImage GetBufferedImage()
{

	try
	{
		URL url = new URL(_server+"Jpeg/CamImg[value].jpg");
		_bufimg = ImageIO.read(url);
	}
```

```
		catch (IOException e)
		{
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		return _bufimg;
	}
	public Image GetImage()
	{
		URL url;
		try
		{
			url = new URL(_server+"Jpeg/CamImg[value].jpg");
			URLImageSource imgsrc = (URLImageSource) url.getContent();
			_img = Toolkit.getDefaultToolkit().createImage(imgsrc);
		}
		catch (MalformedURLException e)
		{
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		catch (IOException e)
		{
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		return _img;


	}


	public void ChangeResolution(int value)
	{
		_currenturl = _server + "/ChangeResolution.cgi?ResType=" + value;
		this.CallMethod2(_currenturl);
		// 0: lowest, 3: highest
	}

	public void ChangeCompressionRatio(int value)
	{
		_currenturl = _server + "/ChangeCompressRatio.cgi?Ratio=" + value;
		this.CallMethod2(_currenturl);
		// 0: low quality, 3:high quality
	}
```

```
public void ChangeFramerate(int value)
{
        _currenturl = _server + "/ChangeFramerate.cgi?Framerate=" + value;
        this.CallMethod2(_currenturl);
        // 2-32
}

public void ChangeBrightness(int value)
{
        // 0:darkest, 6:brightest
        _currenturl = _server + "/ChangeBrightness.cgi?Brightness=" + value;
        this.CallMethod2(_currenturl);
}

public void ChangeSpeakerVolume(int value)
{
        // 0 - 31
        _currenturl = _server + "/ChangeSpeakerVolume.cgi?SpeakerVolume="
                        + value;
        this.CallMethod2(_currenturl);
}

public void ChangeMicVolume(int value)
{
        // 0 - 31
        _currenturl = _server + "/ChangeMicVolume.cgi?MicVolume=" + value;
        this.CallMethod2(_currenturl);
}

public void SetCamera(int value)
{
        // 0 - Auto, 50 - 50Hz, 60 - 60Hz
        _currenturl = _server + "/SetCamera.cgi?Frequency=" + value;
        this.CallMethod2(_currenturl);
}

public void GetCamera(int value)
{

        _currenturl = _server
                        + "/GetCamera.cgi?[&JsVar=variable[&OnJs=function]]";
        this.CallMethod2(_currenturl);
}

public void Reboot()
{
			_currenturl = _server + "/Reboot.cgi";
			this.CallMethod2(_currenturl);
		}

		public boolean IRBarrier()
		{
			this.GetReport();

			//int x = _httpresult.charAt(_httpresult.length() - 2) - 48;
			//System.out.println("" + x);
			//System.out.print(_httpresult[4]);
			if ((_httpresult[4].charAt(_httpresult[4].length() - 1) - 48) >= 6)
			{
				return true;
			}
			else
			{
				return false;
			}
		}
		public boolean IRBarrierMCU()
		{
			this.GetMCUReport();

			char x = _httpresult[2].charAt(_httpresult[2].length() - 2);
		//	System.out.println("" + x);
			//System.out.println(_httpresult[2]);
			if (x == '6' || x == '7' || x == 'E' || x == 'F')
			{
				return true;
			}
			else
			{
				return false;
			}
		}
	}
```

Appendix B – Java Code for Non Primitive Actions – PML Layer

```
import java.awt.image.BufferedImage;
import java.awt.Color;
```

```
public class RovioActions {
	private   BufferedImage _img;
	private   int _row,_col;
	private   RovioPrimitive _rovio;
	private boolean _stop;
	private String _lookingAt;
	private boolean _follow;
	/**
	 * @param args
	 */

	public RovioActions()
	{
		_row = 1000;
		_col = 1000;

		//_mentalmap = new RovioMap(_row,_col);
		_follow = false;
		_lookingAt = "BLUE";
		_rovio = new RovioPrimitive("http://128.205.115.39/");

	}
	public String getFocussedObject()
	{
	return _lookingAt;
	}
	public void LeftWall()
	{
		_stop = false;
		while (_rovio.IRBarrier() == false)
		{
			_rovio.ManualDrive(1, 1);
		}
		while (_stop == false)
		{
			while (_rovio.IRBarrier() == true)
			{
				try
				{
					Thread.sleep(200);
				}
				catch (InterruptedException e)
				{
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
```

```
                        _rovio.ManualDrive(18, 1);
                }
                _rovio.ManualDrive(1, 1);
                _rovio.ManualDrive(1, 1);
                _rovio.ManualDrive(1, 1);
                _rovio.ManualDrive(1, 1);
                _rovio.ManualDrive(1, 1);
                _rovio.ManualDrive(1, 1);
                _rovio.ManualDrive(1, 1);
                _rovio.ManualDrive(1, 1);
                _rovio.ManualDrive(1, 1);
                _rovio.ManualDrive(1, 1);
                _rovio.ManualDrive(1, 1);
                while (_rovio.IRBarrier() == false)
                {
                        try
                        {
                                Thread.sleep(200);
                        }
                        catch (InterruptedException e)
                        {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                _rovio.ManualDrive(17, 1);
                }
        }
}

public void CaptureDocuments(java.awt.Color c)
{
        // set current location as (0,0), wander while keeping track of
        // current location relative to old location (polar coords).
        // Once you find the desired object, take a picture and then leave.
}
public void Recon()
{
        // follow left wall and periodically turn right and take pictures. Once
        // back to first wall contact stop.
}
public void LookAt(java.awt.Color c)
{
        _stop = false;
        int xloc,yloc;
        xloc = 0;
        _img = _rovio.GetBufferedImage();
```

```
		while ((xloc == 0 || (xloc > _img.getWidth() - 80 && xloc < _img.getWidth()
+ 80)) && _stop == false)
		{
			try
			{
				Thread.sleep(200);
			}
			catch (InterruptedException e)
			{
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			_img = _rovio.GetBufferedImage();
			xloc = ImageProcess(Hue(c));
			//System.out.println("xloc is:"+xloc);

			if (xloc == 0)
				{
				//System.out.println("Nothing in sight, going to look around");
				_rovio.ManualDrive(5, 1);
				Wander();
				}
			// determine location for left/right/straight and move in that direction.
			else if (xloc < (_img.getWidth()/2 - 80)  )
				{
				// left
				_rovio.ManualDrive(5, 10);
				_rovio.ManualDrive(5, 10);
				_rovio.ManualDrive(5, 10);
				//System.out.print("Target LEFT!!!!!!\n");
				}
			else if (xloc > (_img.getWidth()/2 + 80))
				{
				//right
				_rovio.ManualDrive(6, 10);
				_rovio.ManualDrive(6, 10);
				_rovio.ManualDrive(6, 10);
				//System.out.print("Target RIGHT!!!!!!\n");
				}
			else
				{
				_rovio.ManualDrive(1, 1);
				_rovio.ManualDrive(1, 1);
				_rovio.ManualDrive(1, 1);
				_rovio.ManualDrive(1, 1);
```

```
                //System.out.print("Target in sight\n");

                }
            xloc = 0;
            }
            //System.out.println("done");
    }
    public void LookAt(String color)
    {
        _stop = false;
        int xloc,yloc;
        xloc = 0;
        _img = _rovio.GetBufferedImage();

        while ((xloc == 0 || xloc < 240 || xloc > 400) && _stop == false)
        {
            try
            {
                Thread.sleep(200);
            }
            catch (InterruptedException e)
            {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            _img = _rovio.GetBufferedImage();
            xloc = 0;
            xloc = ImageProcess(Hue(color));
            //System.out.println("xloc is:"+xloc);

            if (xloc == 0)
                {
                //System.out.println("Nothing in sight, going to look around");
                _rovio.ManualDrive(5, 1);
                Wander();
                }
            // determine location for left/right/straight and move in that direction.
            else if (xloc < (_img.getWidth()/2 - 80)  )
                {
                // left
                _rovio.ManualDrive(5, 10);
                _rovio.ManualDrive(5, 10);
                _rovio.ManualDrive(5, 10);
                //System.out.print("Target LEFT!!!!!!\n");
                }
            else if (xloc > (_img.getWidth()/2 + 80))
```

```
                        {
                        //right
                        _rovio.ManualDrive(6, 10);
                        _rovio.ManualDrive(6, 10);
                        _rovio.ManualDrive(6, 10);
                        //System.out.print("Target RIGHT!!!!!!\n");
                        }
                else
                        {
                        _rovio.ManualDrive(1, 1);
                        _rovio.ManualDrive(1, 1);
                        _rovio.ManualDrive(1, 1);
                        _rovio.ManualDrive(1, 1);
                        //System.out.print("Target in sight\n");

                        }

                }
        _lookingAt = color;
                //System.out.println("done");
}
public void FindColor(java.awt.Color c)
{
        _stop = false;
        int xloc,yloc;
        _img = _rovio.GetBufferedImage();

        while (CenterImageProcess(Hue(c)) == 0 && _stop == false)
        {
                try
                {
                        Thread.sleep(200);
                }
                catch (InterruptedException e)
                {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                _img = _rovio.GetBufferedImage();
                xloc = ImageProcess(Hue(c));
                //System.out.println("xloc is:"+xloc);

                if (xloc == 0)
                        {
                        //System.out.println("Nothing in sight, going to look around");
                        _rovio.ManualDrive(5, 1);
```

```
						Wander();
						}
				// determine location for left/right/straight and move in that direction.
				else if (xloc < (_img.getWidth()/2 - 100)  )
						{
						// left
						_rovio.ManualDrive(5, 10);
						_rovio.ManualDrive(5, 10);
						_rovio.ManualDrive(5, 10);
						//System.out.print("Target LEFT!!!!!!\n");
						}
				else if (xloc > (_img.getWidth()/2 + 100))
						{
						//right
						_rovio.ManualDrive(6, 10);
						_rovio.ManualDrive(6, 10);
						_rovio.ManualDrive(6, 10);
						//System.out.print("Target RIGHT!!!!!!\n");
						}
				else
						{
						_rovio.ManualDrive(1, 1);
						_rovio.ManualDrive(1, 1);
						_rovio.ManualDrive(1, 1);
						_rovio.ManualDrive(1, 1);
						//System.out.print("Target in sight\n");

						}
				xloc = 0;
				}
				//System.out.println("done");
		}

	public void FindColorS(String s)
	{
		_stop = false;
		int xloc,yloc;
		_img = _rovio.GetBufferedImage();
		if (KnowColor(s) == -1)
		{
				return;
		}

		while (CenterImageProcess(Hue(s)) == 0 && _stop == false)
		{
				try
```

```
{
        Thread.sleep(200);
}
catch (InterruptedException e)
{
        // TODO Auto-generated catch block
        e.printStackTrace();
}
_img = _rovio.GetBufferedImage();
xloc = ImageProcess(Hue(s));
//System.out.println("xloc is:"+xloc);

if (xloc == 0)
        {
        //System.out.println("Nothing in sight, going to look around");
        _rovio.ManualDrive(5, 1);
        Wander();
        }
// determine location for left/right/straight and move in that direction.
else if (xloc < (_img.getWidth()/2 - 100)  )
        {
        // left
        _rovio.ManualDrive(5, 10);
        _rovio.ManualDrive(5, 10);
        _rovio.ManualDrive(5, 10);
        //System.out.print("Target LEFT!!!!!!\n");
        }
else if (xloc > (_img.getWidth()/2 + 100))
        {
        //right
        _rovio.ManualDrive(6, 10);
        _rovio.ManualDrive(6, 10);
        _rovio.ManualDrive(6, 10);
        //System.out.print("Target RIGHT!!!!!!\n");
        }
else
        {
        _rovio.ManualDrive(1, 1);
        _rovio.ManualDrive(1, 1);
        _rovio.ManualDrive(1, 1);
        _rovio.ManualDrive(1, 1);
        //System.out.print("Target in sight\n");

        }
xloc = 0;
}
```

```
                _lookingAt = s;
                //System.out.println("done");
        }
public void FollowFocussedObject()
        {
        _stop = false;
        _follow = true;
        while (_follow == true)
                {
                this.GoToFocussedObject();
                }
        }
public void GoToFocussedObject()
{
        _stop = false;
        int xloc,yloc;
        _img = _rovio.GetBufferedImage();
        if (KnowColor(_lookingAt) == -1)
        {
                return;
        }

        while (CenterImageProcess(Hue(_lookingAt)) == 0 && _stop == false)
        {
                try
                {
                        Thread.sleep(200);
                }
                catch (InterruptedException e)
                {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                _img = _rovio.GetBufferedImage();
                xloc = ImageProcess(Hue(_lookingAt));
                //System.out.println("xloc is:"+xloc);

                if (xloc == 0)
                        {
                        //System.out.println("Nothing in sight, going to look around");
                        _rovio.ManualDrive(5, 1);
                        Wander();
                        }
                // determine location for left/right/straight and move in that direction.
                else if (xloc < (_img.getWidth()/2 - 100)  )
                        {
```

```
					// left
					_rovio.ManualDrive(5, 10);
					_rovio.ManualDrive(5, 10);
					_rovio.ManualDrive(5, 10);
					//System.out.print("Target LEFT!!!!!!\n");
					}
				else if (xloc > (_img.getWidth()/2 + 100))
					{
					//right
					_rovio.ManualDrive(6, 10);
					_rovio.ManualDrive(6, 10);
					_rovio.ManualDrive(6, 10);
					//System.out.print("Target RIGHT!!!!!!\n");
					}
				else
					{
					_rovio.ManualDrive(1, 1);
					_rovio.ManualDrive(1, 1);
					_rovio.ManualDrive(1, 1);
					_rovio.ManualDrive(1, 1);
					//System.out.print("Target in sight\n");

					}
				xloc = 0;
				}
				//System.out.println("done");
		}
/*
public	void FindColorMAP(java.awt.Color c, char mapchar)
{
		int xloc,yloc;
		_img = _rovio.GetBufferedImage();
		_stop = false;
		while (CenterImageProcess(Hue(c)) == 0 && _stop == false)
		{
				try
				{
					Thread.sleep(200);
				}
				catch (InterruptedException e)
				{
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
				_img = _rovio.GetBufferedImage();
				xloc = ImageProcess(Hue(c));
```

```
                    //System.out.println("xloc is:"+xloc);


                    if (xloc == 0)
                            {
                            //System.out.println("Nothing in sight, going to look around");
                            WanderMAPOnce();
                            }
                    // determine location for left/right/straight and move in that direction.
                    else if (xloc <= 128)
                            {
                            // left
                            _rovio.ManualDrive(17, 1);
                            _mentalmap.UpdateLoc("left");
                            //System.out.print("Target LEFT!!!!!!\n");
                            }
                    else if (xloc >= 512)
                            {
                            //right
                            _rovio.ManualDrive(18, 1);
                            _mentalmap.UpdateLoc("right");
                            //System.out.print("Target RIGHT!!!!!!\n");
                            }
                    else
                            {
                            _rovio.ManualDrive(1, 1);
                            _mentalmap.UpdateLoc("forward");
                            //System.out.print("Target in sight\n");

                            }
                    xloc = 0;
                    }
                    //System.out.println("found it");
                    _mentalmap.AddObject(_mentalmap.getXloc(),
_mentalmap.getYloc(), mapchar);
            //      _mentalmap.PrintMap();
            }

        */
        public   void Wander()
        {
                    if (_rovio.IRBarrier() == true)
                    {
                            double x = Math.random();
                            while (_rovio.IRBarrier() == true)
                            {
```

```
                                        if (x >= 0.75)
                                        {
                                                _rovio.ManualDrive(5, 1);
                                                _rovio.ManualDrive(5, 1);
                                                _rovio.ManualDrive(5, 1);
                                        }
                                        else if (x >= 0.50)
                                        {
                                                _rovio.ManualDrive(6, 1);
                                                _rovio.ManualDrive(6, 1);
                                                _rovio.ManualDrive(6, 1);
                                        }
                                        else if (x >= 0.25)
                                        {
                                                _rovio.ManualDrive(17, 1);
                                        }
                                        else
                                        {
                                                _rovio.ManualDrive(18, 1);
                                        }
                                        }
                                }
                                else
                                {
                                        _rovio.ManualDrive(1, 1);
                                        _rovio.ManualDrive(1, 1);
                                        _rovio.ManualDrive(1, 1);
                                }
        }
        public   float max(float x, float y, float z)
        {
        float MAX = Math.max(x, y);
        MAX = Math.max(MAX, z);
        return MAX;
        }
        public   float min(float x, float y, float z)
        {
        float MIN = Math.min(x, y);
        MIN = Math.min(MIN, z);
        return MIN;
        }


public   float Hue(java.awt.Color c)
{
        float MAX,MIN,red,green,blue,hue;

        red = c.getRed();
        green = c.getGreen();
        blue = c.getBlue();

        red = red/255;
        green = green/255;
        blue = blue/255;
        MAX = max(red,green,blue);
        MIN = min(red,green,blue);

        if (MAX == red)
        {
                hue = ((green-blue)/(MAX-MIN))*60;
        }
        else if (MAX == green)
        {
                hue = (2+(blue-red)/(MAX-MIN))*60;
        }
        else
        {
                hue = (4+(red-green)/(MAX-MIN))*60;
        }

        if (hue < 0)
        {
                hue+= 360;
        }
        return hue;
}
public int KnowColor(String color)
{
int x = (int) Hue(color);
return x;
}
public float Hue(String color)
{
        float hue = -1;
        color = color.toUpperCase();
        if (color.equals("RED"))
        {
                hue = 0;
        }
        else if (color.equals("ORANGE"))
        {
```

```
			hue = 30;
		}
		else if (color.equals("YELLOW"))
		{
			hue = 60;
		}
		else if (color.equals("LIME"))
		{
			hue = 90;
		}
		else if (color.equals("GREEN"))
		{
			hue = 120;
		}
		else if (color.equals("EMERALD"))
		{
			hue = 150;
		}
		else if (color.equals("CYAN"))
		{
			hue = 180;
		}
		else if (color.equals("LIGHT BLUE"))
		{
			hue = 210;
		}
		else if (color.equals("BLUE"))
		{
			hue = 240;
		}
		else if (color.equals("ORCHID"))
		{
			hue = 270;
		}
		else if (color.equals("PURPLE"))
		{
			hue = 300;
		}
		else if (color.equals("PINK"))
		{
			hue = 330;
		}
		return hue;
	}

	public   int CenterImageProcess(float desiredhue)
```

```
{
        int RGB, rightmost, leftmost;
        float sat,red,green,blue,val,hue,MAX,MIN;
        leftmost = 0;
        rightmost = 0;
        boolean first = true;
        for (int x=(_img.getWidth()/2 - 80); x <= (_img.getWidth()/2 + 80); x+=2)
        {
                for(int y=400; y <= 475; y+=5)
                {
                RGB = _img.getRGB(x,y);
                ////System.out.print("Image RGB is:"+RGB+"\n");
                red = (RGB & 0x00ff0000) >> 16;
                green = (RGB & 0x0000ff00) >> 8;
                blue = RGB & 0x000000ff;
                ////System.out.println("r="+r+" g="+g+" b="+b);


                red = red/255;
                green = green/255;
                blue = blue/255;
                MAX = max(red,green,blue);
                MIN = min(red,green,blue);

                if (MAX == red)
                {
                        hue = ((green-blue)/(MAX-MIN))*60;
                }
                else if (MAX == green)
                {
                        hue = (2+(blue-red)/(MAX-MIN))*60;
                }
                else
                {
                        hue = (4+(red-green)/(MAX-MIN))*60;
                }

                sat = (MAX-MIN)/MAX;
                val = MAX;
                if (hue < 0)
                {
                        hue+= 360;
                }
        //      //System.out.println("X:"+x+"   Y:"+y);
                ////System.out.println("R="+red*255+"  G="+green*255+"   B="+blue*255);
                ////System.out.println("Hue is:"+hue+"   Sat is:"+sat+"   Val is:"+val+"
MAX:"+MAX+"   MIN:"+MIN+"  R="+red+"  G="+green+"  B="+blue);


                if (desiredhue >= hue-15 && desiredhue <= hue+15 &&  sat >= .6 && val
>= .4)
                        {
                        ////System.out.print("hit!\n");
                        ////System.out.println("X:"+x+"   Y:"+y);
                        ////System.out.println("Hue is:"+hue+"   Sat is:"+sat+"   Val is:"+val+"
MAX:"+MAX+"   MIN:"+MIN+"  R="+red+"  G="+green+"  B="+blue);

                        rightmost = x;
                        if (first == true)
                        {
                        leftmost = x;
                        first = false;
                        }
                        }
                }
        }
        ////System.out.print("rightmost is: "+rightmost+"  leftmost is: "+leftmost+"\n");
        if (rightmost != 0 && leftmost != 0)
        {
                //System.out.println("xloc is "+(rightmost+leftmost)/2);
        return (rightmost+leftmost)/2;
        }
        else
        {
                return 0;
        }
}
public   String FacingColor()
{
        _img = _rovio.GetBufferedImage();
        float red,green,blue, hue, sat, val, MAX, MIN;
        int RGB;
        red = 0;
        green = 0;
        blue = 0;
        for (int x = 310; x <= 330; x++)
        {
                for (int y = 230; y <= 250; y++)
                {
                        RGB = _img.getRGB(x,y);
                red += (RGB & 0x00ff0000) >> 16;
                green += (RGB & 0x0000ff00) >> 8;
                blue += RGB & 0x000000ff;
            }
        }
        red = red/400;
        green = green/400;
        blue = blue/400;

        red = red/255;
        green = green/255;
        blue = blue/255;

        MAX = max(red,green,blue);
        MIN = min(red,green,blue);

        if (MAX == red)
        {
            hue = ((green-blue)/(MAX-MIN))*60;
        }
        else if (MAX == green)
        {
            hue = (2+(blue-red)/(MAX-MIN))*60;
        }
        else
        {
            hue = (4+(red-green)/(MAX-MIN))*60;
        }

        sat = (MAX-MIN)/MAX;
        val = MAX;
        if (hue < 0)
        {
            hue+= 360;
        }
        return Pure(hue);
    }
    private String Pure(float hue)
    {
        if (hue < 15)
        {
            return "RED";
        }
        else if (hue < 45)
        {
            return "ORANGE";
```

```
}
else if (hue < 75)
{
        return "YELLOW";
}
else if (hue < 105)
{
        return "LIME";
}
else if (hue < 135)
{
        return "GREEN";
}
else if (hue < 165)
{
        return "EMERALD";
}
else if (hue < 195)
{
        return "CYAN";
}
else if (hue < 225)
{
        return "LIGHT BLUE";
}
else if (hue < 255)
{
        return "BLUE";
}
else if (hue < 285)
{
        return "ORCHID";
}
else if (hue < 315)
{
        return "PURPLE";
}
else if (hue < 345)
{
        return "PINK";
}
else if (hue <= 360)
{
        return "RED";
}
else
```

```
        {
                return Pure(hue - 360);
        }

}
public   int ImageProcess(float desiredhue)
{
        int RGB, rightmost, leftmost;
        float sat,red,green,blue,val,hue,MAX,MIN;
        leftmost = 0;
        rightmost = 0;
        boolean first = true;
        for (int x=0; x < _img.getWidth(); x+=5)
        {
                for(int y=0; y < _img.getHeight(); y+=5)
                {
                RGB = _img.getRGB(x,y);
                ////System.out.print("Image RGB is:"+RGB+"\n");
                red = (RGB & 0x00ff0000) >> 16;
                green = (RGB & 0x0000ff00) >> 8;
                blue = RGB & 0x000000ff;
                ////System.out.println("r="+r+" g="+g+" b="+b);


                red = red/255;
                green = green/255;
                blue = blue/255;
                MAX = max(red,green,blue);
                MIN = min(red,green,blue);

                if (MAX == red)
                {
                        hue = ((green-blue)/(MAX-MIN))*60;
                }
                else if (MAX == green)
                {
                        hue = (2+(blue-red)/(MAX-MIN))*60;
                }
                else
                {
                        hue = (4+(red-green)/(MAX-MIN))*60;
                }

                sat = (MAX-MIN)/MAX;
                val = MAX;
                if (hue < 0)
```

```
				{
					hue+= 360;
				}
				/*if (x > 300 && y > 175 && x < 400 && y < 250)
				{
				//System.out.println("X:"+x+"   Y:"+y);
				//System.out.println("R="+red*255+"  G="+green*255+"   B="+blue*255);
				//System.out.println("Hue is:"+hue+"   Sat is:"+sat+"   Val is:"+val+"
MAX:"+MAX+"   MIN:"+MIN+"  R="+red+"  G="+green+"  B="+blue);
				}*/

				if (desiredhue >= hue-15 && desiredhue <= hue+15 &&  sat >= .6 && val
>= .4)
					{
					//System.out.print("hit!\n");
					//System.out.println("X:"+x+"   Y:"+y);
					//System.out.println("Hue is:"+hue+"   Sat is:"+sat+"   Val is:"+val+"
MAX:"+MAX+"   MIN:"+MIN+"  R="+red+"  G="+green+"  B="+blue);

					rightmost = x;
					if (first == true)
					{
					leftmost = x;
					first = false;
					}
					}
				}
		}
		////System.out.print("rightmost is: "+rightmost+"  leftmost is: "+leftmost+"\n");
		if (rightmost != 0 && leftmost != 0)
		{
				//System.out.println("xloc is "+(rightmost+leftmost)/2);
		return (rightmost+leftmost)/2;
		}
		else
		{
				return 0;
		}
}
public void Stop()
{
		_stop = true;
		_follow = false;
}
public boolean getFollow()
{
```

```
return _follow;
}
public void goDirection(int direction)
{
        if (direction == 0)
        {
        _rovio.ManualDrive(1, 1);
        _rovio.ManualDrive(1, 1);
        _rovio.ManualDrive(1, 1);
        _rovio.ManualDrive(1, 1);
        }
        else if (direction == 1)
        {
                _rovio.ManualDrive(2, 1);
                _rovio.ManualDrive(2, 1);
                _rovio.ManualDrive(2, 1);
                _rovio.ManualDrive(2, 1);
        }
        else if (direction == 2)
        {
                _rovio.ManualDrive(3, 1);
                _rovio.ManualDrive(3, 1);
                _rovio.ManualDrive(3, 1);
                _rovio.ManualDrive(3, 1);
        }
        else if (direction == 3)
        {
                _rovio.ManualDrive(4, 1);
                _rovio.ManualDrive(4, 1);
                _rovio.ManualDrive(4, 1);
                _rovio.ManualDrive(4, 1);
        }
}
public void goBackward()
{
        _rovio.ManualDrive(2, 1);
        _rovio.ManualDrive(2, 1);
        _rovio.ManualDrive(2, 1);
        _rovio.ManualDrive(2, 1);
}
public void goRight()
{
        _rovio.ManualDrive(4, 1);
        _rovio.ManualDrive(4, 1);
        _rovio.ManualDrive(4, 1);
        _rovio.ManualDrive(4, 1);
```

```
}
public void goLeft()
{
        _rovio.ManualDrive(3, 1);
        _rovio.ManualDrive(3, 1);
        _rovio.ManualDrive(3, 1);
        _rovio.ManualDrive(3, 1);
}


}
```

Appendix C – The KL/PML interface and KL layer

Pmlb.cl

```
;;; Connection to Java Simulated World
;;; Stuart C. Shapiro
;;; October 11, 2002
;;; Revised to the "class model", October 21, 2002
;;;
;;; Referred to as PMLw in various papers.
;;;

(defpackage :Pmlb
  (:shadow common-lisp:find)
  (:export #:descriptionOfFocussedObject #:destroyWorld
          #:find #:FocussedObjectLooksLike
          #:followFocussedObject
          #:goToFocussedObject
          #:satisfies-description #:description-of-object
          #:goTo #:goDir #:stop
          #:attendToPoint
          #:talkto #:lookat
          #:world #:Cassie
          #:blue #:green #:yellow #:cyan #:red #:purple #:pink #:orange #:lime
          #:up #:down #:left #:right
          #:human #:robot))

(in-package :Pmlb)
(require :jlinker)
(use-package :javatools.jlinker)
(load "/projects/robot/Fevahr/Java/jl-config.cl")
(jlinker-init)

(defvar world)
(defvar Cassie)

(defconstant up 0)
(defconstant down 1)
(defconstant left 2)
(defconstant right 3)
```

```
(defconstant blue 0)
(defconstant green 1)
(defconstant yellow 2)
(defconstant cyan 3)
(defconstant red 4)
(defconstant human 0)
(defconstant robot 1)


(def-java-class RovioActions () () () ())
(def-java-constructor newRovioActions (RovioActions))
(def-java-method Wander(RovioActions))
(def-java-method LeftWall(RovioActions))
(def-java-method LookAt(RovioActions "string"))
(def-java-method FindColorS(RovioActions "string"))
(def-java-method Hue(RovioActions "string"))
(def-java-method goDirection(RovioActions "int"))
(def-java-method Stop(RovioActions))
(def-java-method GoToFocussedObject(RovioActions))
(def-java-method getFocussedObject(RovioActions))
(def-java-method FollowFocussedObject(RovioActions))


;(def-java-class Fevahr (Agent) () () ())
;(def-java-method getFocussedObject (Fevahr))
;(def-java-method (findObj "find") (Fevahr "int" "int"))
;(def-java-method (findCat "findByShape") (Fevahr "int"))
;(def-java-method findByColor (Fevahr "int"))
;(def-java-method (fGoTo "goTo") (Fevahr "int" "int"))
;(def-java-method (fGoDir "goDir") (Fevahr "int"))
;(def-java-method (fgoToFocussedObject "goToFocussedObject") (Fevahr))
;(def-java-method follow (Fevahr))
;(def-java-method (fStop "stop") (Fevahr))
;(def-java-method lookForPoint (Fevahr))

(defun createWorld ()
  (setf Cassie (newRovioActions)
       ))

(defun destroyWorld ()
  (jlinker-end))

;(defun goTo (x y)
;  "Move Cassie to position (x, y) smoothly."
;  (fgoTo Cassie x y))


(defun goDir (d)
  "Move Cassie 1/3 of the way across the room in the given direction."
  (assert (and (<= 0 d) (<= d 3)) (d)
    "The direction ~A is not between 0 and 3." d)
  (goDirection Cassie d))

(defun goToFocussedObject ()
  "Moves Cassie to be next to the object she is looking at."
  (GoToFocussedObject Cassie))
```

```
(defun descriptionOfFocussedObject ()
      (let ((fo (getFocussedObject Cassie) )) )
)
;  "Returns a list of the color and shape of the object
;      Cassie is looking at."
;  (let ((fo (getFocussedObject Cassie)))
;    (list (getColor fo) (getShape fo))))

(defun description-of-object (object)
  "Returns a list of the color and shape of the world object."
  (list (KnowColor object) ))


(defun find (description)
  "Causes Cassie to be looking at an object
      that satisfies the description,
      which may be incomplete."
  (unless (if (first description)
            (if (second description)
              (LookAt Cassie (first description)
                     (second description))
            (LookAt Cassie (first description)))
          (if (second description)
            (LookAt Cassie (second description))
            (warn
             "Can't find an object with an empty description.")))
    (warn "Can't find an object looking like ~A." description)))


(defun FocussedObjectLooksLike (description)
  "Returns t if the object Cassie is looking at
     has the given description,
     which may be incomplete."
  ;; If Cassie is not looking at anything, returns nil.
  (when (getFocussedObject Cassie)
    (let ((objdescr (descriptionOfFocussedObject)))
      (and (or (null (first description))
             (= (first description) (first objdescr)))
         (or (null (second description))
             (= (second description) (second objdescr)))))))

(defun satisfies-description (object description)
  "Returns t if the object has the given description,
   which may be incomplete."
  (and (or (null (first description))
         (= (first description) (getColor object)))
       (or (null (second description))
         (= (second description) (getShape object)))))



(defun followFocussedObject ()
  "Causes Cassie to start following the object she is looking at."
  (FollowFocussedObject Cassie))

(defun stop ()
  "Causes Cassie to stop moving and to stop looking at anyone."
  (Stop Cassie))

(defun lookat (obj)
  "At least temporarily a noop."
  (LookAt Cassie obj))

(defun talkto (name)
  "At least temporarily a noop."
  (declare (ignore name)))

;;; Create and display the Fevahr World.

(createWorld)
```

jfevahr.snepslog

```
;;; Demo file for FEVAHR in the Java Simulated World
;;; Stuart C. Shapiro
;;; October 19, 1995
;;;
;;; Load requisite files
;;;
^ (cl:load "/home/unmdue/phoellig/CogRob/pmlb.cl")
demo "/home/unmdue/phoellig/CogRob/fevahr.snepslog"
;;;
;;; Establish alignments
;;;
^^
(setf *alignments*
  (list (cons (sneps:choose.ns #2!((find (object- ! propername) Bill)))
              (list Pmlb:blue Pmlb:human))
        (cons (sneps:choose.ns #2!((find (object- ! propername) Stu)))
              (list Pmlb:yellow Pmlb:human))
        (cons (sneps:choose.ns #2!((find (object- ! propername) Cassie)))
              (list Pmlb:cyan Pmlb:robot))
        (cons (sneps:choose.ns #2!((find lex green)))
              (list Pmlb:green nil))
        (cons (sneps:choose.ns #2!((find lex red)))
              (list Pmlb:red nil))
        (cons (sneps:choose.ns #2!((find lex person)))
              (list nil Pmlb:human))
        (cons (sneps:choose.ns #2!((find lex robot)))
              (list nil Pmlb:robot))
        (cons (sneps:choose.ns #2!((build lex up)))
              Pmlb:up)
        (cons (sneps:choose.ns #2!((build lex down)))
              Pmlb:down)
        (cons (sneps:choose.ns #2!((build lex left)))
              Pmlb:left)
        (cons (sneps:choose.ns #2!((build lex right)))
              Pmlb:right)))
^^
;;;
;;; Turn tracing on/off
```

```
;;;
untrace inference acting
lisp
```

Pmla.cl

```
;;; Primitive Action Functions for FEVAHR
;;; Stuart C. Shapiro
;;; February 24, 1995
;;;
;;; Referred to as PMLa in various papers.

(in-package snepslog)

(cl:load "/home/unmdue/phoellig/CogRob/pmla2.cl")

(defvar *STMn* nil
  "A global variable for the node
     representing the entity Cassie is looking at.")

;;; Make sure *STMn* in accessible in the snepsul package.
(import '*STMn* (find-package :snepsul))

;;; Make *STMn* empty when reloading.
(setf *STMn* nil)

(defun pml-version (node)
  (declare (special *alignments*))
  (cdr (assoc node *alignments*)))

(defsetf pml-version (node) (pmlobj)
  `(progn (setf *alignments* (acons ,node ,pmlobj *alignments*))
        ,pmlobj))

(defun pml-description-of (nodes)
  (let* ((color nil)
       (shape nil))
    (sneps:do.ns
     (nde
      ;; Find all nodes that might be aligned with descriptions of NODES.
      ;; This includes NODES themselves, equivalent nodes,
      ;; properties of any of these, and classes of any of these
      #!((find (compose
            ;; them or their properties or their classmods
            (or ()
                classmod-
                (compose property- ! object))
            ;; them or classes they nodes are in
            (compose
             (kstar (or classhead-
                      (compose superclass- ! subclass)))
             (or ()
                 (compose class- ! member)))
```

```
               ;; nodes at the same level
               (kstar (compose equiv- ! equiv)))
                ~nodes))
        (if (or color shape) (list color shape)
        nil))
       (let ((color-n-shape (pml-version nde)))
         (setf color (or color (first color-n-shape))
             shape (or shape (second color-n-shape)))
         (when (and color shape) (return (list color shape)))))))

(sneps:defsnepscom nlip ((state &optional (*trace-level* -1)) sneps:build)
  "Reads an NL string, parses it starting at GATN STATE, and returns
   the popped node(s)."
  (declare
   (special
    *trace-level* sneps:outunit parser:*atn-arcs-hashtable*
englex:*lexicon*))
  (unless (sneps:initialized-p englex:*lexicon*)
    (warn "No lexicon is loaded. You should load a lexicon via~
           ~%`(lexin \"<lexicon-filename>\")' before using `nlip'."))
  (unless (sneps:initialized-p parser:*atn-arcs-hashtable*)
    (sneps:sneps-error
     (format
      nil "No ATN grammar is loaded. You must load a grammar via~
           ~%`(atnin \"<atn-grammar-filename>\")' before using `nlip'.")
     'describe 'nlip))
  (let* ((*all-parses* nil)
       (*terminating-punctuation-flag* nil)
       (expression (parser::atn-read-sentence))
       result)
    (declare (special *all-parses*))
    (if (and (upper-case-p (elt (first expression) 0))
           (null (englex::lexic (first expression)))
           (englex:lookup
            (string-downcase (first expression) :end 1)))
      (setf expression
        (cons (string-downcase (first expression) :end 1)
            (rest expression))))
    (flistify (parser:flatten
             (parser::internal-parse expression state)))))

(defun choose-random (list)
  (nth (random (length list)) list))

(defun recognize (description)
  "Returns the SNePS node representing the entity
      that looks like the description."
  (let ((individuals (sneps:+ (sneps:find (object- ! propername)
                              snepsul::Stu)
                    (sneps:+ (sneps:find (object- ! propername)
                                 snepsul::Bill)
                      (sneps:find (object- ! propername)
                           snepsul::Cassie))))
      (groups (sneps:+ (sneps:find classmod (sneps:find lex snepsul::green)
                    classhead (sneps:find lex snepsul::robot))
                 (sneps:find classmod (sneps:find lex snepsul::red)
                    classhead (sneps:find lex snepsul::robot)))))
    (sneps:do.ns (individual individuals)
             (when (equal description (pml-description-of
                                  individual))
               (return-from recognize individual)))
    (sneps:do.ns (group groups)
             (when (equal description
                          (pml-description-of group))
               (or (sneps:* 'npx) (sneps:$ 'npx))
               (return (choose-random
                       #!((find member-
                             (deduce member *npx class ~group)))))))))

(defun recognize-as (object node)
  "Returns T if the object looks like
  the entity represented by the node."
  (Pmlb:satisfies-description object (pml-description-of node)))

(define-primaction find-deicticfun (object)
  "Finds an object of the given category
     represented by the object node."
 ; (Pmlb:lookat (first (pml-description-of object) ))
 (lookat (recognize (pml-description-of object))))
```

Pmla2.cl

```
;;; PML Functions for FEVAHR to remain uncompiled
;;; Stuart C. Shapiro
;;; Started November 18, 1995
;;;
;;; Continuation of PMLa

(in-package snepslog)

;;; Utilities for the primitive actions

(defun lookat (entity)
(Pmlb:lookat (first (pml-description-of entity)))
)

;(defun lookat (entity)
;  "Believes that *I is looking at the entity and not some other;
;   If already believes that *I is looking at that entity does nothing."
;  (declare (special *STMn*))
;  (cond (#!((= (findassert
;            time (find contains *NOW)
;            event
;            (find
;             agent *I
;             act (find action (build lex "look")
;                    (|at| (kstar (compose equiv- ! equiv))) ~entity)))
;             assertion))
;         #!((surface *assertion)))
;      (t (end-durative-acts #!((find lex "follow")))
;         (Pmlb:lookat (first (pml-description-of entity)))
;         (setf *STMn* entity)
;         (assert-durative-act
;          #2!((build action (build lex "look") |at| ~entity)))
;         (continue-durative-acts #!((find lex "talk"))))))
;
(defun end-durative-acts (actions)
  "Finds all current beliefs of the form `I am doing <action>',
   where <action> is one of the action nodes on the list ACTIONS,
   and believes that their times are before now."
  (let ((beliefs
       #3!((findassert
             time (find supint-
                    (findassert subint
                              (+ *NOW
                               (find before-
                                      (findassert after *NOW)))))
             event (find agent *I (act action) ~actions)))))
    (when beliefs
      #!((add before (+ *NOW (find time- ~beliefs)) after #NOW)))))

(defun assert-durative-act (act)
  "Asserts that 'I am doing <act>', where act is a durative act node."
  ;; Change to:
  "<act> is a durative act node.
   If *I am already doing <act>, don't do anything;
   Else, end any current doing of the action of <act>,
         and assert that *I am now doing <act>."
  (unless
      #!((= (findassert time (find contains *NOW)
                  event (find agent *I act ~act))
          assertion))
      (end-durative-acts #!((find action- ~act)))
      #!((add before *NOW after #event-time)
       (add supint *event-time subint #NOW)
       (= (assert time *event-time
            event (build agent *I act ~act))
          assertion)
       (add time *event-time
        event (build agent *I act ~act))))
  #!((surface *assertion)))

(defun continue-durative-acts (actions)
  "Finds all current beliefs of the form `I was doing <action>'
   that are not known already to be in the past,
   where <action> is one of the action nodes on the list ACTIONS,
   and believes that their times include now."
  (let ((beliefs
       #!((findassert
           time (- (find supint-
                     (findassert subint
                              (find before-
                                  (findassert after *NOW))))
                  (find before- (findassert after *NOW)))
           event (find agent *I (act action) ~actions)))))
    (when beliefs
      #!((add supint (find time- ~beliefs) subint *NOW)))))
```

```
(defun assert-punctual-act (act)
  "Asserts that 'I did <act>', where act is a punctual act node."
  #!((add before *NOW after #event-time)
     (add before *event-time after #NOW)
     (surface (assert time *event-time
            event (build agent *I act ~act)))
     (add time *event-time
        event (build agent *I act ~act))))


;;; Primitive Actions
(define-primaction talkfun (parser:to)
  "Makes the object of talk the value of *You,
   and believes that *I is talking to that agent."
  (Pmlb:talkto (format nil "~A"
                  #2!((sneps:choose.ns
                        (find (propername- object) ~parser:to)))))
  (assert-durative-act
   #2!((build action (build lex "talk") parser:to ~parser:to = You)))
  (continue-durative-acts #!((find lex ("follow" "look")))))

(define-primaction findfun (object)
      (lookat object)
)
;(define-primaction findfun (object)
;  "Finds a world object that fits the PML description of
;   the SNePS node object,
;   and looks at it;
;   But if *I am already looking at it, do nothing."
;  (declare (special *STMn*))
;  (let ((description (pml-description-of object)))
;    (cond ((Pmlb:FocussedObjectLooksLike description)
;        (unless (sneps:iseq.n object *STMn*)
;          #!((add equiv (~object ~*STMn*))))
;        #!((surface
;             (findassert
;            time (find contains *NOW)
;            event
;            (find agent *I
;                  act (find action (build lex "look")
;                        |at| ~*STMn*))))))
;          (t (cond
;            (description
;             (end-durative-acts #!((find lex "look")))
;             (Pmlb:find description)
;             (setf *STMn* (recognize
;                        (Pmlb:descriptionOfFocussedObject)))
;             (unless (eql object *STMn*)
;                      #!((add equiv (~object ~*STMn*))))
;             (assert-punctual-act
;            #2!((build action (build lex find) object ~*STMn*)))
;             (lookat *STMn*)
;             (continue-durative-acts #!((find lex talk))))
;            (t (error "I don't know what ~A look(s) like."
;                  object)))))))

(define-primaction followfun (object)
  (Pmlb:followFocussedObject)
  (assert-durative-act #2!((build action (build lex "follow")
                          object ~object)))
  (continue-durative-acts #!((find lex ("look" "talk")))))

(define-primaction gofun (parser:to)
  (cond (#!((findassert member ~parser:to class (find lex direction)))
       ;; Go up/down/left/right.
       (end-durative-acts #!((find lex "follow")))
       (Pmlb:goDir (pml-version (sneps:choose.ns parser:to)))
       (assert-punctual-act
        #2!((build action (build lex "go") parser:to ~parser:to)))
       (continue-durative-acts #!((find lex ("talk" "look")))))
      ;; Go to an object.
      (#!((= (findassert
            time (find contains *NOW)
            event (find arg1 *I
                        rel (build lex "near")
                        (arg2 (kstar (compose equiv- ! equiv)))
                        ~parser:to))
             assertion))
         ;; Already near the object
         #!((surface *assertion)))
      (t ;; Go to the object.
       (end-durative-acts #!((find lex "follow")))
       (Pmlb:goToFocussedObject)
       (assert-punctual-act
        #2!((build action (build lex "go") parser:to ~parser:to)))
       #!((surface (assert time *NOW
                   event (build arg1 *I rel (build lex "near")
                              arg2 ~parser:to)))
          (add time *NOW
             event (build arg1 *I rel (build lex "near") arg2
                        ~parser:to)))
       (continue-durative-acts #!((find lex ("talk" "look")))))))

(define-primaction stopfun ()
  "Cassie stops,
   but continues talking to whomever she was talking to before."
  (Pmlb:stop)
  (end-durative-acts #!((find lex ("follow" "look"))))
  (assert-punctual-act #2!((build action (build lex "stop"))))
  (continue-durative-acts #!((find lex "talk"))))
```

fevahr.snepslog

```
;;; Demo file for FEVAHR in the Simulated World
;;; Stuart C. Shapiro
;;; Started June 21, 1995
;;; SNePSLOG version started March 16, 2004
;;;
set-mode-3
;;; Import into snepslog symbols that are already in use in the snepsul
package.
```

```
^(import '(parser:lex snepsul::contains snepsul::before snepsul::after)
       (find-package :snepslog))
;;; DEFINE CASEFRAMES
;;;
;;; Functional Individuals
;;; ======================
define-frame act (nil action object)
define-frame compCat (nil classmod classhead)
define-frame doAt (nil action at)
define-frame doTo (nil action parser:to)
define-frame event (nil agent act)
define-frame lex (nil lex)
define-frame string (nil beg snepsul:end)
define-frame word (nil word)
;;; Propositions
;;; ============
define-frame Equiv (nil equiv)
define-frame EventGoal (nil event goal)
define-frame EventPrecondition (nil event precondition)
define-frame Has (nil possessor rel object)
define-frame Member (nil member class)
define-frame Name (nil object propername)
define-frame Occurs (nil event time)
define-frame Pred(nil pred word)
define-frame Property (nil object property)
define-frame Rel (nil rel arg1 arg2)
define-frame Results (nil event results)
define-frame Subclass (nil subclass superclass)
define-frame SubInt (nil subint supint)
;;;
;;; Define Paths
;;; ============
^^
;;; Import relation names into the snepsul package
;;;    for use in the grammar and in paths.
;;; Except for these, which would cause a name conflict: to lex end contains
before after
(defparameter *Vocabulary*
    '(snepslog::act snepslog::action snepslog::agent snepslog::arg1
      snepslog::arg2 snepslog::at snepslog::beg snepslog::class
      snepslog::classhead snepslog::classmod snepslog::equiv
      snepslog::event snepslog::goal snepslog::member snepslog::object
      snepslog::possessor snepslog::precondition snepslog::pred
      snepslog::propername snepslog::property snepslog::rel
      snepslog::results snepslog::subclass snepslog::subint
      snepslog::superclass snepslog::supint snepslog::time
      snepslog::word)
      )
(export (append *Vocabulary*
            (mapcar #'sneps::converse.r *Vocabulary*))
      :snepslog)
(import (append *Vocabulary*
            (mapcar #'sneps::converse.r *Vocabulary*))
      :snepsul)
^^
define-frame Before(nil before after)
define-path class (or class
     ;; If x is not a member of class c,
     ;;    then it is not a member of any subclass of c.
     ;; If x is a member of class c,
     ;;    then it is a member of any superclass of c.
     ;; However, this inheritance of superclasses is blocked
     ;;    by properties that are negative adjectives.
     ;; I.e., if you say
     ;;                  `former' is a negative adjective.
     ;; then a former teacher will not be considered to be a teacher.
     ;; However, it needs some negative adjective to be in the network.
         (domain-restrict
          ((compose arg- ! max) 0)
          (compose class
                   (kstar (or classhead-
                              (compose superclass- ! subclass)))))
         (compose ! class
                  (kstar (or (domain-restrict
                              ((not (compose classmod lex word-
                                             (and beg- end-)
                                             member- ! class
                                             (domain-restrict
                                              ((compose classmod lex)
                                               "negative")
                                              classhead)
                                             lex)) "adjective" ) classhead)
                             (compose subclass- ! superclass)))))
define-path member (compose member
                     ;; The member relation is transparent across
coextensional entities.
                     (kstar (compose equiv- ! equiv)))

define-path superclass (compose superclass
                        ;; The superclass relation goes up the class
hierarchy,
                        ;; and also from a complex category to the main
category.

                        (kstar (or classhead (compose subclass- !
                                                      superclass))))

define-path subclass (compose subclass
                       ;; The subclass relation goes down the class
hierarchy,
                       ;; and also from a main category to a complex
subcategory.

                       (kstar (or classhead- (compose superclass- !
                                                      subclass))))

define-path equiv (compose equiv
                    ;; equiv is transitive.
                    (kstar (compose equiv- ! equiv)))

define-path before (compose before
                     ;; before is transitive
                     ;; and if t1 is before t2,
                     ;;   then it's before any subinterval of t2.
```

```
                       (kstar (or (compose after- ! before)
                              (compose supint- ! subint))))

define-path after (compose after
                     ;; after is transitive
                     ;; and if t1 is after t2,
                     ;;   then it's after any subinterval of t2.
                     (kstar (or (compose before- ! after)
                              (compose supint- ! subint))))
define-path supint (or time        ; for reflexivity of a time that's a time
of some event.
                   ;; for transitivity
                   (compose supint (kstar (compose subint- ! supint))))

define-path subint (or time        ; for reflexivity of a time that's a time
of some event.
                   ;; for transitivity
                   (compose subint (kstar (compose supint- ! subint))))

define-frame Contains(nil contains)
define-path contains (kstar (compose supint- ! subint)
                      ;; A virtual arc from one time to another
                      ;; when the second is during the first.
                      ;; Any node also contains itself.
                      ;; For use by find and findassert
                      )

^^
;;;
;;; Load requisite files
;;; ====================
(cl:load "/projects/robot/Fevahr/newachieve")
(cl:load "/projects/robot/Fevahr/newact")
(cl:load "/home/unmdue/phoellig/CogRob/pmla")
(in-package :snepsul)
(atnin "/projects/robot/Fevahr/grammar.cl")
(lexin "/projects/robot/Fevahr/lexicon.cl")
(in-package :snepslog)
;;;
;;; Attach the primitive actions
;;; ============================
(attach-primaction
 believe believe disbelieve disbelieve withall withall withsome withsome
 snsequence snsequence sniterate sniterate do-all do-all
 snepsul::finddeictic find-deicticfun
 (= (sneps:build lex snepsul::talk) talk) talkfun
 (= (sneps:build lex snepsul::find) find) findfun
 (= (sneps:build lex snepsul::go) go) gofun
 (= (sneps:build lex snepsul::follow) follow) followfun
 (= (sneps:build lex snepsul::stop) stop) stopfun)
^^
;;;
;;; Establish *I
;;; ============
Member(cassie,lex(snepsul::FEVAHR)).
Name(cassie,snepsul::Cassie).
%(= snepslog::cassie I)
```

```
;;;
;;; Establish *NOW
;;; ==============
SubInt(b2,{b3,b4}).
%(= snepslog::b2 NOW)
;;;
;;; Establish *You*
;;; ===============
Occurs(event(cassie,doTo(lex(snepsul::talk),stu)),b3).
Occurs(event(cassie,doAt(lex(snepsul::look),stu)),b4).
%(= snepslog::stu You)
;;;
;;; Conceive of other known entities.
;;; =================================
Name(stu,snepsul::Stu).
Name(bill,snepsul::Bill).
Member({stu,bill},lex(snepsul::person)).
Member(robbie,compCat(lex(snepsul::green),lex(snepsul::robot))).
Member({b8,b9,b10},compCat(lex(snepsul::red),lex(snepsul::robot))).
Member({lex(snepsul::down),lex(snepsul::up),lex(snepsul::right),lex(snepsul::
left)},lex(snepsul::direction)).
;;;
;;; Class Hierarchy
;;; ===============
Subclass({lex(snepsul::astronaut),lex(snepsul::woman),lex(snepsul::man)},
        lex(snepsul::person)).
Subclass(lex(snepsul::FEVAHR),lex(snepsul::robot)).
Subclass({lex(snepsul::robot),lex(snepsul::person)},lex(snepsul::agent)).
Subclass(lex(snepsul::agent),lex(snepsul::thing)).
;;;
;;; Some extra individuals
;;; ======================
Member(b11,lex(snepsul::woman)).
Member(b12,lex(snepsul::man)).
Member(string(word(snepsul::fake),word(snepsul::fake)),
       compCat(lex(snepsul::negative),lex(snepsul::adjective))).
;;;
;;; Semantics of some words
;;; =======================
Member(string(word(snepsul::near),word(snepsul::near)),
       compCat(lex(snepsul::locative),lex(snepsul::relation))).
Member({lex(snepsul::eat),lex(snepsul::dial),lex(snepsul::follow)},transparen
t).
all(name,agt)(Has(agt,lex(snepsul::name),name) <=> Name(agt,name)).
all(rln,psr,obj)(Has(psr,rln,obj) => Member(obj,rln)).
;;;
;;; Domain plans and rules
;;; ======================
all(obj)(Member(obj,lex(snepsul::thing))
         =>
ActPlan(doAt(lex(snepsul::look),obj),act(lex(snepsul::find),obj))).

all(agt)({Member(agt,lex(snepsul::agent))}
         v=> {all(obj)({Member(obj,lex(snepsul::thing))}
                      v=>
{EventPrecondition(event(agt,act(lex(snepsul::follow),obj)),
Rel(lex(snepsul::near),agt,obj)),
                            EventGoal(event(agt,doTo(lex(snepsul::go),obj)),
                                      Rel(lex(snepsul::near),agt,obj)),
                            EventGoal(event(agt,act(lex(snepsul::find),obj)),

event(agt,doAt(lex(snepsul::look),obj))),

EventPrecondition(event(agt,doTo(lex(snepsul::go),obj)),

event(agt,doAt(lex(snepsul::look),obj))))}),
              ActPlan(act(lex(snepsul::help),agt),
                      do-all({act(lex(snepsul::follow),agt),
                            doTo(lex(snepsul::talk),agt)}))}).

;;; Transparency Rules
;;; ==================
all(a)(Member(a,transparent)
       => (all(time,obj2,agt,obj)({Equiv({obj2,obj}),
                                   Occurs(event(agt,act(a,obj)),time)}
                                  &=>

{Equiv({Occurs(event(agt,act(a,obj2)),time),

Occurs(event(agt,act(a,obj)),time)}),
                                   Occurs(event(agt,act(a,obj2)),time)}))).

all(time,obj2,agt,obj)({Occurs(event(agt,doAt(lex(snepsul::look),obj)),time),
                        Equiv({obj2,obj})}
                       &=>

{Equiv({Occurs(event(agt,doAt(lex(snepsul::look),obj2)),time),

Occurs(event(agt,doAt(lex(snepsul::look),obj)),time)}),

Occurs(event(agt,doAt(lex(snepsul::look),obj2)),time)}).

all(time,obj2,agt,obj)({Occurs(event(agt,doTo(lex(snepsul::go),obj)),time),
                        Equiv({obj2,obj})}
                       &=>

{Equiv({Occurs(event(agt,doTo(lex(snepsul::go),obj2)),time),

Occurs(event(agt,doTo(lex(snepsul::go),obj)),time)}),

Occurs(event(agt,doTo(lex(snepsul::go),obj2)),time)}).

all(agt2,time,agt,obj)({Equiv({agt2,agt}),
                        Occurs(event(agt,doTo(lex(snepsul::go),obj)),time)}
                       &=>

{Equiv({Occurs(event(agt2,doTo(lex(snepsul::go),obj)),time),

Occurs(event(agt,doTo(lex(snepsul::go),obj)),time)}),

Occurs(event(agt2,doTo(lex(snepsul::go),obj)),time)}).
```

```
all(agt2,time,obj2,agt,obj)({Equiv({agt2,agt}),

Occurs(event(agt,doTo(lex(snepsul::go),obj)),time),
                             Equiv({obj2,obj})}
                            &=>

{Equiv({Occurs(event(agt2,doTo(lex(snepsul::go),obj2)),
                                   time),

Occurs(event(agt,doTo(lex(snepsul::go),obj)),
                                 time)}),

Occurs(event(agt2,doTo(lex(snepsul::go),obj2)),time)}).

all(prop,class,obj)({Member(obj,compCat(prop,class))}
                    v=> {all(class2)(Subclass(class,class2)
                                    =>
                                    Subclass(compCat(prop,class),
                                   compCat(prop,class2))),
                         Subclass(compCat(prop,class),class)}).
```