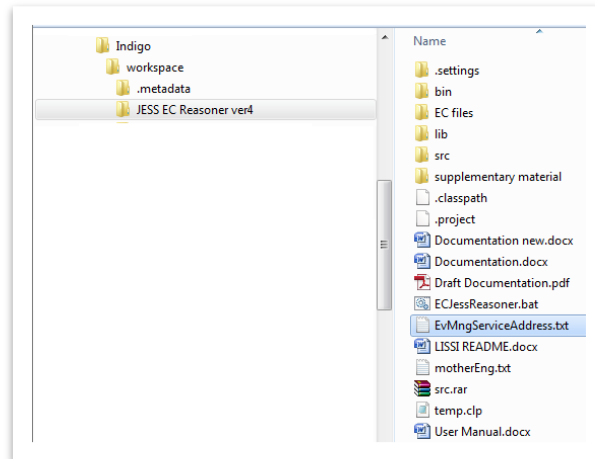**CEREBRO AmI Infrastructure**

Event Calculus Rule-based Reasoner

User's Manual

Theodore Patkos

# 1. Setting up the components before execution

- Connection with the Event Manager: Find the EvMngServiceAddress.txt file inside the JESS EC Reasoner folder and make sure the Port/IP address of the Event Manager is correct.

- Run Eclipse and load the Reasoner. Some packages that are loaded for the connection with the LiSSi infrastructure use paths that are only valid when executing the program with Eclipse.

- Specify the Event Calculus file: Inside the "ReasonerMain" class specify the path of the main Event Calculus file to load. E.g.,

```
commModule.InitializeECDomain ./EC files/Lissi2013/deliverable/
                                       LISSIvanKasteren.ec", false);
```

Important: All supplementary files that are loaded from this file, need to exist inside a folder having the same name as this file. For instance, in the previous example, they should be in a folder named "LISSIvanKasteren". In addition, if there are BN graph files (.xml files), they should be in a folder "BN graphs" inside this folder.
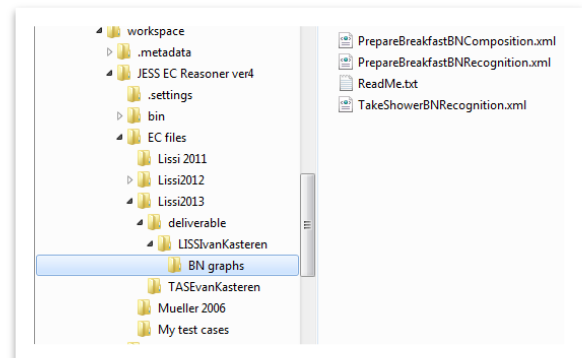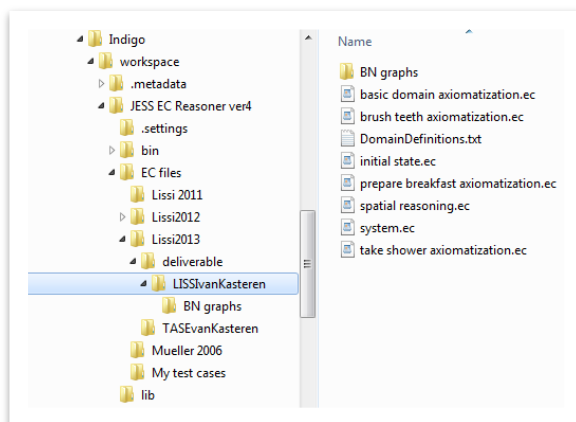
- Choose the Probabilistic Estimation method: Inside the ReasonerMain class locate the following command:

```
MonitorService commModule = new MonitorService(ui, "XXX");
```

and change parameter XXX to be either

1. WS

   Implements a weighted sum function. Requires no extra files, as all weights are encoded in the Event Calculus axiomatizations

2. BN

   Implements estimation based on belief networks. The appropriate .xml files need to be in the "BN graphs" folder.

3. DS

   Not implemented yet! It will include the Dempster Shaffer component.

- Make sure the "DomainDefinitions.txt" file is up-to-date and consistent with the EC axiomatization. The reasoner will read this file during initialization, in order to ask about the current state of all sensors/devices that exist in the environment.
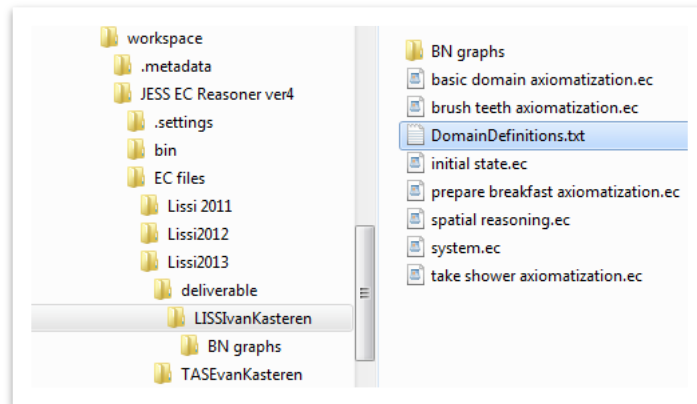
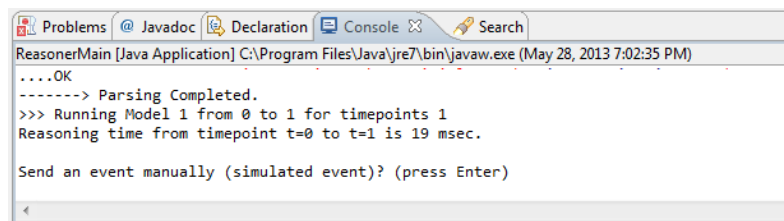You are now ready to start the Reasoner.

2. Familiarization with the UIs

When you first run the Reasoner from inside Eclipse, it will first try to connect with the Event Manager, then it will read and parse all related files (.xml and .ec files), it will open the reasoner's UI and then it will send queries to all sensors/devices to ask about their current state. This last step will take ~5 seconds.

All relevant information regarding these steps is printed on the Eclipse Console. If everything is correct, no errors and only three warnings will appear on the Console. If not, check the connections with the Event Manager service and the paths of the required EC files.
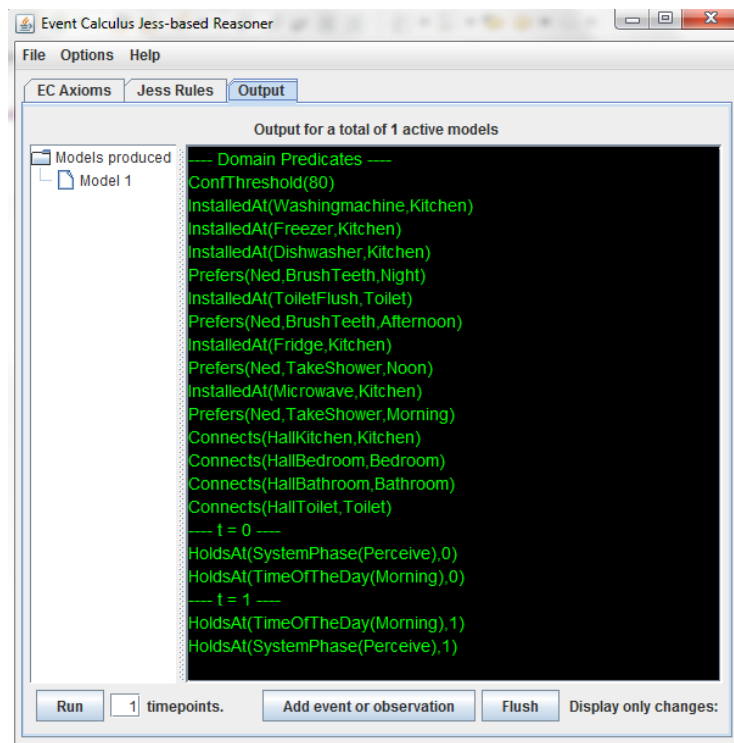
A message "Send an event manually (simulated event)? (press Enter)" will appear on the Console.

The user can receive important information both from the Eclipse Console and the Reasoner's UI:

Adding Events Manually.

This is a useful functionality for debugging the axiomatizations. In order to test the different use cases, the user can enter events (fake or simulated events) without interacting with the devices of the infrastructure.

To do so, press enter in the Eclipse Console. A pop-up window will appear.
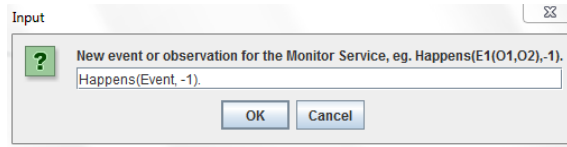
Enter the event you want to simulate, e.g., `Happens(DoorOpens(Ned,HallBedroom, 220),-1)`. Note that "-1" is needed to denote that the event will happen now. Any other timepoint will mean a specific time during the execution. Also, 200 is the actual time of occurrence; study the EC axiomatizations to get an understanding of how these event types are constructed.

After pressing the "OK" button, the Reasoner will react as if this event actually occurred in the environment. Notice that for each event three different reasoning steps are executed; in general, different tasks are considered in each step.

The event we manually added.

Other events triggered by the reasoner.

Reading and understanding the outputs.

The Reasoner's UI informs us about the current state of the world, as well as the events that have happened (this output can be sent as String to interested components). For instance, we can see which recognized or possible activities are or have been happening, their start and end times, the confidence values, the justification for these confidence values etc (see the related journal paper for information about these aspects). A sample output is as follows:

```
HoldsAt(LastLocation(Ned,Kitchen,300),5)
HoldsAt(CannotAccessRoom(Ned,Bathroom),5)
HoldsAt(PossActivity(Ned,PrepareBreakfast,PB1:Morning,1),5)

HoldsAt(PossActivity(Ned,PrepareBreakfast,DS,88),6)
HoldsAt(PossActivityStartTime(Ned,PrepareBreakfast,300),6)
HoldsAt(RecActivity(Ned,PrepareBreakfast,5),6)
HoldsAt(RecActivityStartTime(Ned,PrepareBreakfast,5,300),6)
```

In addition, the Eclipse console can provide other interesting information. For instance, if we use BNs as our probability estimation method, we can read related probabilities, such as:

```
Probabilistic Estimation component found that the current evidence is: [tm, !ts]
Pr(PB|[tm, !ts])=0.88
Probabilistic Estimation component sends:
Happens(UpdateActivityConf(Ned,PrepareBreakfast,300,DS,88),-1).
Probabilistic Estimation component found that the current evidence is: [!of, !cg, !gt]
The probabilities of the actions that have not been performed yet are as follows:
Pr(cg1|[!of, !cg, !gt])=8.55256
Pr(cg2|[!of, !cg, !gt])=6.12166
Pr(cf1|[!of, !cg, !gt])=6.19269
Pr(of1|[!of, !cg, !gt])=65.00234999999999
Pr(og1|[!of, !cg, !gt])=84.0049
Pr(og2|[!of, !cg, !gt])=56.987140000000004
----- End of list of pending actions -----
```

The first two lines state that the Prepare Breakfast (PB) activity has been recognized with 88% confidence, given the current evidence information (see the related .xml files for explanation about the IDs). The rest of the output is about the percentage of the actions that the user is expected to perform next. For example, Ned will most likely open the groceries cupboard (og1) or open the fridge (of1).

3. Execution Examples

Next, we provide two narratives to test the files that are given with the Reasoner, along with explanations on key points, to understand the behavior of the system.

    a. Narrative 1 – How Possible and Recognized Activities are initiated and terminated

Apply the following narrative using the **Weighted Sum (WS)** methodology.

```
Happens(DoorOpens(Ned,HallBedroom, 0),-1).
Happens(DoorOpens(Ned,HallBathroom, 100), -1).
Happens(TriggerAlert(NoActivity,240,340),-1).
Happens(TriggerAlert(NoActivity,480,580),-1).
Happens(DoorOpens(Ned,HallToilet, 636), -1).
```

Some remarks about the outputs are below (only some of the conclusions are shown):

- <u>Happens(DoorOpens(Ned,HallBedroom, 0),-1).</u>

No activity is triggered. See that the Reasoner infers that Ned cannot access the other rooms; this is because their doors are closed.

- <u>Happens(DoorOpens(Ned,HallBathroom, 100), -1).</u>

```
---- t = 5 ----
HoldsAt(ActiveAlert(NoActivity,480,100),5)
HoldsAt(ActiveAlert(NoActivity,900,100),5)
HoldsAt(ActiveAlert(NoActivity,240,100),5)
HoldsAt(PossActivityStartTime(Ned,TakeShower,100),5)
HoldsAt(PossActivityStartTime(Ned,BrushTeeth,100),5)
HoldsAt(PossActivity(Ned,TakeShower,TS8:NoShowerYet,3),5)
HoldsAt(PossActivity(Ned,BrushTeeth,BT3:Morning,3),5)
HoldsAt(PossActivity(Ned,TakeShower,TS2:Morning,2),5)
> Happens(UpdateActivityConf(Ned,BrushTeeth,100,DS,50),5)
> Happens(UpdateActivityConf(Ned,TakeShower,100,DS,63),5)
```

The user approaches the Bathroom; two activities become possible, TakeShower (TS) and BrushTeeth (BT). The EC axiomatization provides two justifications for the TS, i.e., TS8 and TS2, and one for the BS, i.e., BT3, with their corresponding confidence weights, i.e., 3, 2, 3 respectively. Moreover, some alerts have been initiated that are relevant to these activities.

Based on the weights, the WS function returns the probabilities for these activities to be the actual ones. It returns 63% for TS and 50% for BT.

```
---- t = 6 ----
HoldsAt(PossActivity(Ned,BrushTeeth,DS,50),6)
HoldsAt(PossActivity(Ned,TakeShower,DS,63),6)
> Happens(ChangeSystemPhaseTo(Act),6)
```

The activity values appear as conclusions. None activity is recognized though, since the confidence is below the threshold of 80%.

- Happens(TriggerAlert(NoActivity,240,340),-1).

Nothing happens for the next 4 minutes (240 sec); therefore, an alert is triggered. This has as a result for the BT activity to end, because the user cannot be reasonably brushing his teeth for more than 4 minutes.

```
---- t = 7 ----
> Happens(ChangeSystemPhaseTo(Perceive),7)
> Happens(TriggerAlert(NoActivity,240,340),7)
> Happens(EndPossActivity(Ned,BrushTeeth,340,BT7:InactiveTooLong,3),7)
```

Yet, the system again calculates the new confidence value for BT based on the new observation (an activity could be recognized even at the time it ends, see next action).

```
---- t = 8 ----
> Happens(UpdateActivityConf(Ned,BrushTeeth,340,DS,50),8)
```

Finally, the possible activity is retracted and only TS is possible to be happening in the bathroom.

```
---- t = 9 ----
> Happens(RetractPossActivity(Ned,BrushTeeth),9)
---- t = 10 ----
HoldsAt(PossActivity(Ned,TakeShower,DS,63),10)
HoldsAt(PossActivityStartTime(Ned,TakeShower,100),10)
HoldsAt(ActiveAlert(NoActivity,900,100),10)
HoldsAt(ActiveAlert(NoActivity,480,100),10)
```

- Happens(DoorOpens(Ned,HallToilet, 636), -1).

Now the user goes to another room (he has not closed the Bathroom door, but his interaction with the Toilet door allows the Reasoner to understand that the user is no longer in the Bathroom). The Reasoner can understand that Ned is no longer taking shower, but it needs to determine the confidence value for this activity, even though it has ended!

```
---- t = 13 ----
> Happens(DoorOpens(Ned,HallToilet,636),13)
> Happens(ChangesRoom(Ned,Toilet,636),13)
> Happens(EndPossActivity(Ned,TakeShower,636,TS6:LeftOnTime,1),13)
> Happens(ResetAlert(NoActivity,all,1),13)
```

Because the user stayed in the Bathroom the amount of time it usually takes him to have a shower (as well as because it is morning and according to his profile he takes showers in the morning), we are now

certain that the possible activity actually occurred. Therefore, it needs to change from possible to recognized, even though it has just ended (a recognized activity may trigger other actions).

```
---- t = 14 ----
> Happens(UpdateActivityConf(Ned,TakeShower,636,DS,100),14)
> Happens(BeginRecActivity(Ned,TakeShower,100),14)
---- t = 15 ----
HoldsAt(RecActivity(Ned,TakeShower,14),15)
HoldsAt(RecActivityStartTime(Ned,TakeShower,14,100),15)
HoldsAt(RecActivityEndTime(Ned,TakeShower,14,636),15)
> Happens(RetractPossActivity(Ned,TakeShower),15)
```

Notice that even though the possible activities are retracted, the recognized activities remain, in order to keep the history of what the user has performed. This can change in the "system.ec" file (just uncomment the last axioms). Inside the RecActivity predicates, the "14" argument is the unique ID that correlates the predicates of this activity: if at some other time point TakeShower is again recognized, a different ID will be assigned to correlate the new triplet of RecActivity predicates.

b. **Narrative 2** – Use of BNs and request for devices to perform actions

Apply the following narrative using the Belief Networks (BN) methodology.

```
Happens(DoorOpens(Ned,HallKitchen, 100), -1).
Happens(StartsInteractingWith(Ned,Fridge,203),-1).
Happens(CupboardOpens(Ned,Groceries,301),-1).
Happens(DoorOpens(Ned,HallBathroom, 400), -1).
```

Some remarks about the outputs are below (only some of the conclusions are shown):

- `Happens(DoorOpens(Ned,HallKitchen, 100), -1).`

Ned enters the Kitchen and PrepareBreakfast becomes a possible activity, as he has not taken breakfast in the last 14400 seconds (see "prepare breakfast axomatization.ec"). Then, the probabilistic component evaluates the conditional probabilities of the Belief Networks and concludes that the confidence value is 88% (due to the time of the day and the fact that he has not taken breakfast yet). This can be checked at the eclipse console, which shows the following message:

```
Probabilistic Estimation component found that the current evidence is: [tm, !ts]
Pr(PB|[tm, !ts])=0.88
Probabilistic Estimation component sends:
Happens(UpdateActivityConf(Ned,PrepareBreakfast,100,DS,88),-1).
```

Notice that in the "PrepareBreakfastBNRecognition.xml" file "tm" is the id for "TimeOfTheDay (Morning)" and "ts" is the id for taking shower in the last 900 seconds.

Moreover, the probabilistic component also checks which are the most probable actions that Ned is expected to perform. This information is extracted from the "PrepareBreakfastBNComposition.xml" file. The probabilities, as they appear in the console are:

```
Probabilistic Estimation component found that the current evidence is: [!of, !cg, !gt]
The probabilities of the actions that have not been performed yet are as follows:
Pr(cg1|[!of, !cg, !gt])=8.55256
Pr(cg2|[!of, !cg, !gt])=6.12166
Pr(cf1|[!of, !cg, !gt])=6.19269
Pr(of1|[!of, !cg, !gt])=65.00234999999999
Pr(og1|[!of, !cg, !gt])=84.0049
Pr(og2|[!of, !cg, !gt])=56.987140000000004
----- End of list of pending actions -----
```

As evidence now are the relevant fluents, specifically that the neither the fridge nor the groceries cupboard are open, and that Ned hasn't been in the toilet in the last 300 seconds. As a result, and according to his profile, the most probable action to perform is opening the groceries cupboard, followed by the opening of the fridge (notice that the percentage of closing these devices is very low, as the system knows that are already closed).

At the end, the Reasoner's UI displays that PrepareBreakfast is a recognized activity and that opening the groceries cupboard is a probable action. This is because the estimation is above the 80% threshold (this value can change from the Monitoring.class). Additionally, at the "Act" reasoning cycle it asks for a message to be presented on screen SCR1. By this we show how we can use inferrencing to perform actions in the environment. All predicates starting with "Do_" denote actions that the reasoner asks the infrastructure to be executed; the Event Manager receives such messages and forwards them to the appropriate component.

```
---- t = 4 ----
HoldsAt(RecActivityStartTime(Ned,PrepareBreakfast,2,100),4)
HoldsAt(RecActivity(Ned,PrepareBreakfast,2),4)
HoldsAt(PossActivity(Ned,PrepareBreakfast,DS,88),4)
> Happens(Do_DisplayMsg(BigBrother,og1,SCR1),4)
```

-Happens(StartsInteractingWith(Ned,Fridge,203),-1).

It turns out that Ned decided to open the fridge first, rather than opening the groceries cupboard. Reasoning proceeds in a similar way as before. Now, two actions are probable to happen next, closing the fridge or opening the groceries store.

```
Probabilistic Estimation component found that the current evidence is: [of, of1, !cg, !gt]
The probabilities of the actions that have not been performed yet are as follows:
Pr(cg1|[of, of1, !cg, !gt])=8.85285
Pr(cg2|[of, of1, !cg, !gt])=6.89596
Pr(cf1|[of, of1, !cg, !gt])=95.08648
```

```
Pr(og1|[of, of1, !cg, !gt])=87.92638000000001
Pr(og2|[of, of1, !cg, !gt])=66.07402
----- End of list of pending actions -----
```

Notice that now both the fact that the fridge is open and that the "OpenFridge" action has been performed are considered as evidences, in order to calculate the percentagies of the remaining actions.

- Happens(DoorOpens(Ned,HallBathroom, 400), -1).

Finally, the user decides to go to the bathroom. New possible activities are considered, i.e., TakingShower and BrushingTeeth, and the PrepareBreakfast activity ends. Notice, though, that even though this latter activity is not active any more, the Reasoner sends messages for the most probable actions that the user was expected to make: this works as a reminder for the user in case he forgot to perform some task (in this case he forgot to close the fridge!).

```
---- t = 10 ----
HoldsAt(PossActivityStartTime(Ned,TakeShower,400),10)
HoldsAt(PossActivityStartTime(Ned,BrushTeeth,400),10)
HoldsAt(RecActivityStartTime(Ned,PrepareBreakfast,2,100),10)
HoldsAt(RecActivity(Ned,PrepareBreakfast,2),10)
HoldsAt(InteractsWith(Fridge),10)
HoldsAt(RecActivityEndTime(Ned,PrepareBreakfast,2,400),10)
HoldsAt(PossActivity(Ned,BrushTeeth,DS,0),10)
HoldsAt(PossActivity(Ned,TakeShower,DS,57),10)
> Happens(Do_DisplayMsg(BigBrother,cf1,SCR1),10)
> Happens(Do_DisplayMsg(BigBrother,og1,SCR1),10)
```