*23311085 Qichong Huang*

1.Use Protege-5.5.0, please find the ***HW3Q1.owl*** file in the folder

a) Refer to ***HW3Q1.owl*** in the folder. First, the class ***Projects*** has 4 subclasses, they are: ***commits***, ***failures***, ***faults***, and ***versions***. The class ***commits*** has 3 subclasses: ***file\_modifications***, where a series of file modifications are put here with codes, ***author***, which indicates the author of this commit, and ***time-stamp*** which saves the time when this commit is madshould e. These 3 subclasses are disjoint with others. The class ***failures*** represents different type of failures which has 3 subclasses: ***user\_reported\_issues***, ***system\_crashes***, and ***failed\_tests***. These 3 subclasses are disjoint with others. The class ***failures*** is the subclass of ***user\_reported\_issues*** or ***system\_crashes*** or ***failed\_tests***. The class ***faults*** represents all the types of faults which has 4 subclasses: ***the\_size\_of\_the\_project***, ***the\_length\_of\_the\_file***, ***the\_author\_of\_a\_commit***, and ***other***. These 4 subclasses are disjoint with others. The class ***faults*** is the subclass of the union of these 4 subclasses. The class ***version*** has no subclass, it corresponds to a version of commits.

There are 4 object properties: ***corresponds\_to***, which is functional and has domain commits and ranges versions, ***manifests***, which has domain failures and ranges faults, ***lead\_to,*** which has domain faults and ranges failures, and ***has\_faults***, which has domain file\_modifications and ranges faults. The properties ***manifests*** and ***lead\_to*** are inverse to each other.

There is one data property: ***has\_fault*** which is a Boolean value and has domain of ***file\_modifications***.

All classes also have limitation of the type of data it should store, for example, ***time-stamp*** should have data of numbers in certain format.

b) User can use **SPARQL** for the sorts of queries that can be asked and a format for the answers produced by the system.

c) When enter new data into the system, user will assert data in corresponding class as an individual. After the assertion, user should start reasoner for auto-classify and error detection etc.

When execute queries, user should first type their queries in the **Snap SPARQL Query** subtab in **DL query** tab using **SPARQL.** Then click ***Execute*** button for execution and get results.

d) User can use **SPARQL** in the **Snap SPARQL Query** subtab in **DL query** tab. Theoretically, user can query about the state of the project at any time by querying the ***version*** class of this project; track reported failures and their likely cause by querying the ***failures*** class and querying the property ***manifests***; determine the likelihood of faults existing in a given file by querying the property ***has\_fault*** of some individuals in class ***file\_modifications***.

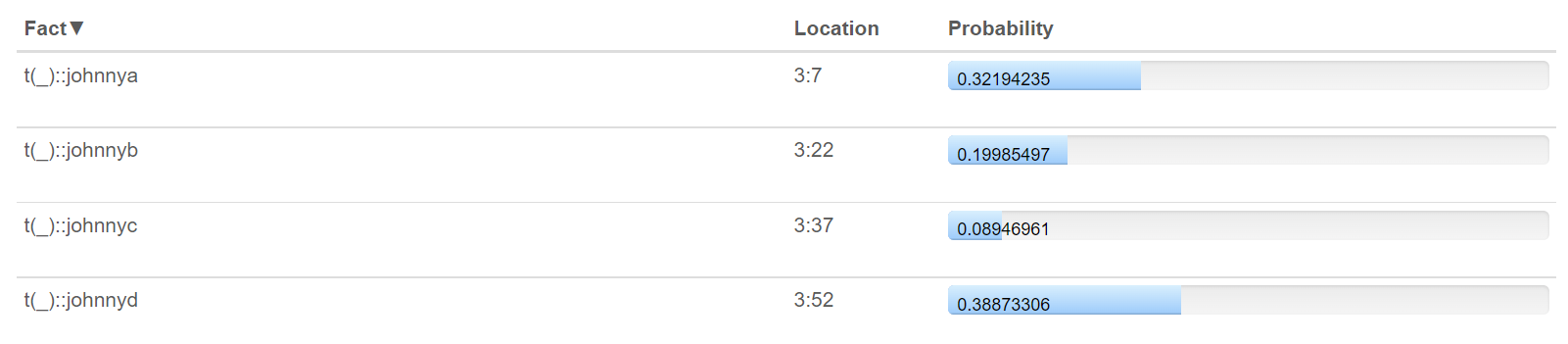
There are also other query functionalities user can explore, like query about the author or time stamp of a commits.

2.

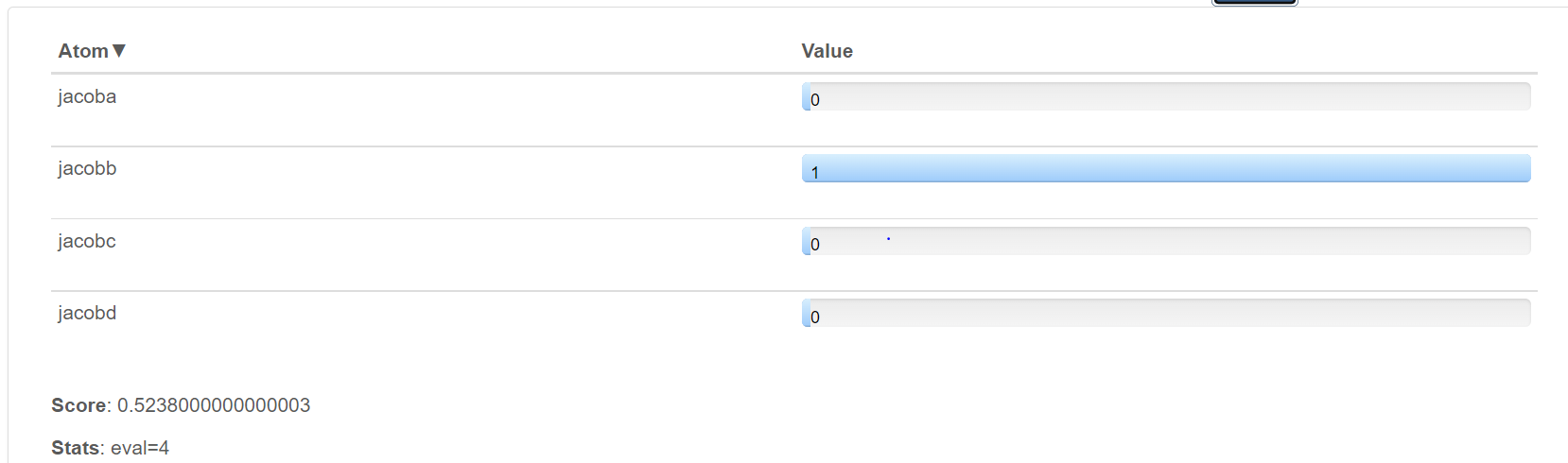
At the beginning, all probability distributions of each player’s four actions are set to 0.25. The probability of Jacob winning is 0.25. Refer to the ***HW3Q2.pl*** file in the folder.

Then using sampling to collect samples. Collected samples refer to the ***HW3Q2sample.pl*** file in the folder.

Next, change Johnny’s probability for each action to t(\_) for learning. Refer to the ***HW3Q2learning.pl*** file in the folder.

Results: 

Since Jacob has learnt Johnny’s strategy, change Johnny’s probability for each action to the corresponding one in results above (round to 4 digits), and Jacob’s probability for each action to ?. Add utilities for each situation. Refer to the ***HW3Q2learned.pl*** file in the folder.

Results: 

In this scenario, Jacob should only choose action 2(action b).

Theroetically, under this case, they should reach an equilibrium, where Jacob choose only action 2(action b) and Johnny choose only action 3(action c).

3.

**1) What do you think are the main opportunities and challenges for knowledge representation and reasoning in contemporary artificial intelligence? Do you think the material in this unit will be useful in your future endeavours?**

I think the main opportunities for knowledge representation and reasoning in contemporary artificial intelligence are creating AI that can accept literal expressions fias input and conduct logical responds in various environments and creating AI that has excellent logic reasoning ability to help human discover new knowledge from what we have now and possesses the ability to transform the result into a high-level abstract expression for human to comprehend.

The main challenges in my thoughts are the limit of hardware to support calculation in logic reasoning in massive system and ambiguities in expressions we used making some knowledge representation and reasoning tasks very hard.

Yes, I think the material in this unit will be useful in my future endeavours. Even I may not study in this specific field in future, but those materials provide a chance for me to get a glimpse at KRR as we are in a world where AI is everywhere and KRR taking a big part in it.

**2) Which section of the unit (logic, logic programming, ontologies, uncertainty etc.) did you find most interesting and engaging, and why?**

I find the logic programming part most interesting and engaging, since this is the part where we put all those theoretical knowledge and logic-related mathematical practises we went through (not only in CITS3005) together, trying to create something that can be potentially directly implement in real-life. That section is challenging, the differences between logic programming and normal programming put me in a position where I need to quickly adapt changes and learning the how logic reasoning applications work is quite interesting.

**3) Is there any material you would like to see included in the unit, or would you like to see more detail on any specific content?**

I think the contents in this unit are full now, with 4 main sections. However, I would love to see more philosophy part of this unit.

**4) How would you explain this unit to another student who was considering taking the unit next year?**

I would say this unit is the mixture of theoretical and programming practises which begins at logic knowledge we learned from CITS2200, then gradually shifts to more logic programming side and reasoning part. This unit provides students a glimpse at KRR and taking this unit will be a fun but not lack of challenges journey.

**5) Would you recommend this unit to other students?**

Yes, I will.