HANDS-ON Building a Studentmodel using MEBN/PrOWL2

Jeroen Donkers, Maastricht University
Jeroen.donkers@maastrichtuniversity.nl

Preinstalled software needed for this session:

- Java JRE (recent version)
- Protégé ontology editor 5.0 or higher (http://protege.stanford.edu)
- UnBBayes (http://sourceforge.net/projects/unbbayes)
    - GUI and plugins unbbayes.prs.mebn, unbbayes.gui.mebn.ontology.protege
    - Versions: Gui: 4.21.15, plugins 1.13.11 and 1.1.4, *or*
      Gui: 4.21.18, plugins 1.14.13 and 1.2.5.
- Some Java IDE (e.g. Eclipse) (JDK needed)

We are going to create a very simple student model (with two MFrags).

Part 1: setting up your ontology

1a. create empty ontology file

- Start Protégé.
- Rename the ontology (field Ontology IRI) to: "studentmodel".
- Save the model in OWL/XML format under the filename "studentmodel.owl"

1b. create Classes

- Switch to tab "Classes"
- Create classes Time, Task, Skill (all three, subclasses of Thing)

1c. create Instances (individuals)

- Switch to tab "individuals"
- Create individuals for Skills: "writing", "desinging", "programming".
- Create individuals for Tasks: "task_1", "task_2"
- Save ontology

1d. create object properties (relations)

- Switch to tab "object properties"
- Add property "needsSkill" (under topObjectProperty)
- Select "Task" for the domain and "Skill" for the range of this object property.
- Save ontology

1e. create data properties

- Switch to tab "data properties"
- Add property "finishedTask"
- Add as domain: Task and Time
- Select as range the built-in data type "xsd:Boolean"
- Add properties "practicedSkill" and "hasSkill"
- Add as domain: Skill and Time to both
- Select as range the built-in data type "xsd:Boolean" for both
- Save ontology

1f. create relation between instances

- Switch to tab "individuals"
- Select "task_1"
- Add object property assertion: needsSkill designing
- Add object property assertion: needsSkill writing
- Select "task_2"
- Add object property assertion: needsSkill programming

- Add object property assertion: needsSkill writing
- Save ontology
- Close Protégé

Part 2: creating the MEBN

2a. create an MEBN file pair (.ubf + .owl)

- Open UnBBayes
- Open file studentmodel.owl form part 1
  - Select file type "MEBN with PR-OWL 2.0"
- Save the file under name "studentmodel.ubf"
  - Select file type "MEBN with PR-OWL 2.0" (**DO THIS ALL THE TIME!!!**)
- Close UnBBayes
- You will find that two linked files have been created:
  - Studentmodel.owl
  - Studentmodel.ubf
- Open studentmodel.owl in Protégé and observe the additions (imported ontology "pr-owl2.owl") DO NOT SAVE!

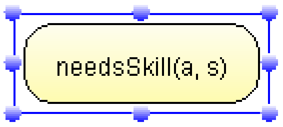
2b. add time steps

- Open UnBBayes
- Open file studentmodel.**ubf** form part 2a (**DO NOT SELECT THE OWL FILE)**
  - Select file type "MEBN with PR-OWL 2.0"
- In the MTheory tab, select MTheory entities pane (button with the orange circle):
- Select Class "Time" and check the box "is Ordenable"
- Select the Entity instances pane (diamond and orange circle):
- Select "Time [Ord]"
- Add time instances "T0", "T1", "T2", "T3" (enter name and use "+" button).
- Save the file (Select file type "MEBN with PR-OWL 2.0")

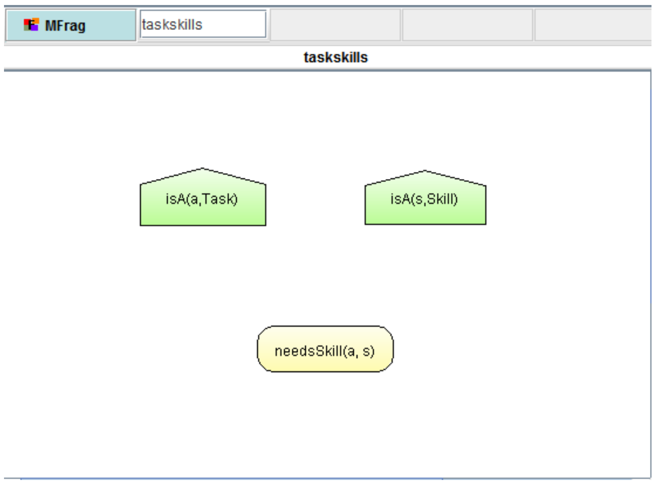
2c. create the MFrag "taskskills"

- Insert a new MFrag (use third vertical button):
- Change the name of the MFrag in "taskskills" (name field at top of right pane, press Enter)
- Insert a task variable
  - Press "insert Ordinary variable" button
  - Click in the right pain to place the variable node
  - Select "Task" from the dropdown box
  - Enter "a" as the name for the variable
  - Enlarge the "IsA" node symbol to make the text is readable:

- Add a variable "s" of type Skill
- Add the relation node "needsSkill"
  - Select the "Property2Node" tab on the top
  - Select the middle button ("Show OWL properties")
  - Locate property "needsSkill" and drag it to the right pane
  - Enlarge the node to make the text readable.
  - Select the "Mtheory" tab on de top
  - Unselect and reselect the "needsSkill" node (the left pane will chan
  - Select the button for parameters (parantheses)
  - Double click on "a (Task)" and double click on "s (Skill)" to add the parametes
  - Click on the "T+" button and the "+" button to set Boolean values for this node (true, false, absurd).



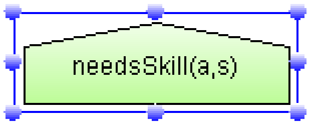
- Save your file



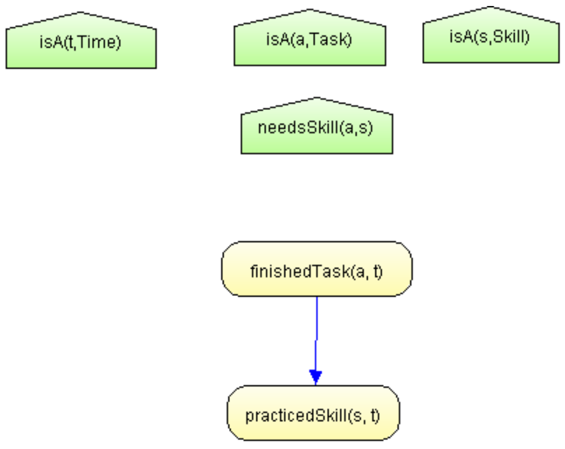
The result of 2c: MFrag taskskills

2d. create the MFrag "practice"

- Insert a new MFrag "pratice"
- Insert variable t of type Time
- Add a variable "a" of type Task and a variable "s" of type Skill
- To restrict the relation between task and skill:
  - Add a context node to the Mfrag (use button "C"):
  - Double click "formula" in the left upper pane
  - Double click node "needsSkill" in the left lower pane (tree)
  - Double click on "needsSkill" that just appeared in the upper pa
  - Select "a" for the Task_label and "s" for the Skill_label
  - The node should now look like:



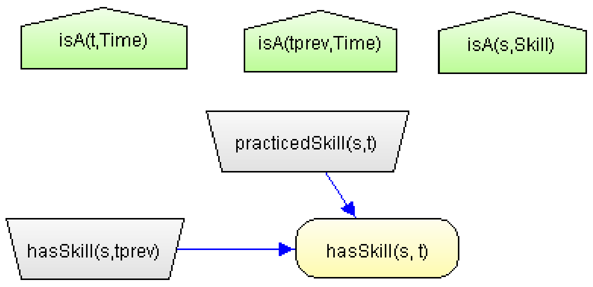
- From the Property2Node tab drag property "finishedTask" to the MFrag.
  - Set parameters to "a" and "t"
  - Add Boolean values to this node
- From the Property2Node tab drag property "practicedSkill" to the MFrag.
  - Set parameters to "s" and "t"
  - Add Boolean values to this node
- Add an arrow from finishedTask to practicedSkill:
- Save your file



Result of 2d: MFrag practice

2e. create MFrag "skilldevelopment"

- Insert a new MFrag "skilldevelopment"
- Insert variables t and tprev of type Time
- Add a variable "s" of type Skill
- From the Property2Node tab drag property "hasSkill" to the MFrag.
  - Set parameters to "s" and "t"
  - Add Boolean values to this node
- Add an input node using the "I" button
  - Select node "hasSkill" from the left tree
  - Select "s" for the Skill_label
  - Select "tprev" for the Time_label
- Add an arrow from inputnode "hasSkill(s,tprev)" to "hasSkill(s,t)"
- Add a second input node
  - Select node "practicedSkill from the left tree
  - Select "s" and "t" for the labels
- Add an arrow from inputnode "practicedSkill(s,t)" to "hasSkill(s,t)"
- Save your file



The result of 2e: MFrag skilldevelopment

2f create local distributions

- Open MFrag Practice, click on node "practicedSkill"
- Open the probability table editor:
- Enter the following formula (using the buttons):

```
 if any t have ( finishedTask = true)  [
   true = 0.8,
   false = 0.2,
   absurd = 0
 ] else [
   true = 0.2,
   false = 0.8,
   absurd = 0
 ]
```

- Click on "Save" and then on "Compile"
- In node "finishedTask" add the formula

```
[
   true = 0.01,
   false = 0.99,
   absurd = 0
]
```

- Remember to press "Save" and then "Compile"

- Open MFrag skilldevelopment, click on “hasSkill”
- Edit the probability table and fill in the following formula:

```
if any t have ( practicedSkill=true ) [
if any tprev have (hasSkill=true) [
   true = 0.9,
   false = 0.1,
   absurd = 0
] else [
   true = 0.7,
   false = 0.3,
   absurd = 0
]
] else [
if any tprev have (hasSkill=true) [
   true = 0.6,
   false = 0.4,
   absurd = 0
] else [
   true = 0.1,
   false = 0.9,
   absurd = 0
]]
```

- Remember to press “Save” and then “Compile”
- Save your file

Part 3: querying the network

Now we are ready to query the MEBN network.

First add some context knowledge:

- Open the findings editing pane
- Select needsskill and use the pencil button to add
    - Task 1 needs skill writing
    - Task 2 needs designing

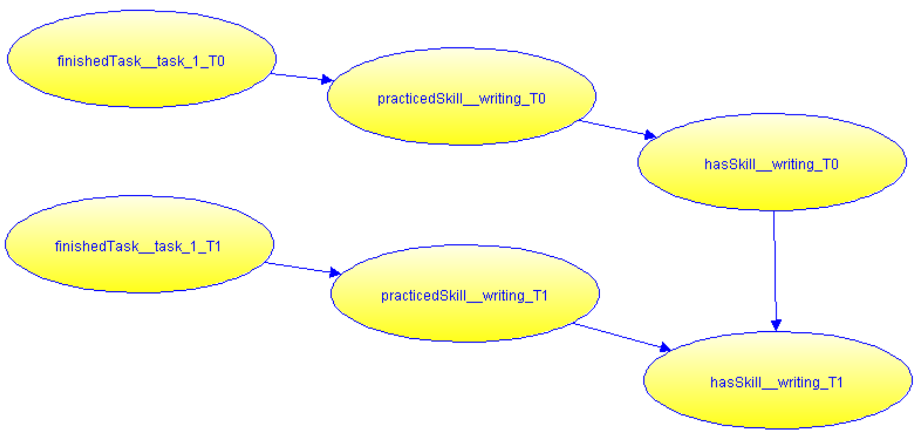
Then add an observation:

- Select finishedTask and use the pencil to add:
    - Task 1 was finished on T0

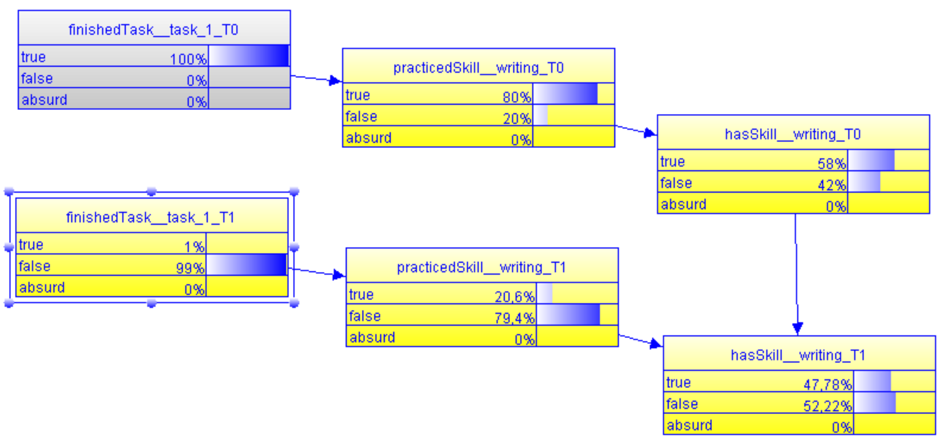
Now we can query:

- Press the query button:
- Select "hasSkill" from the list
- Select skill "writing" and time "T1"
- Press Execute

The following query-graph (SSBN) appears (after rearranging the nodes a bit):



On the lft you can see the resulting properties. By right-clicking on the nodes, you can show the belief bars in the nodes:



finishedTask__task_1_T0	
true	100%
false	0%
absurd	0%

practicedSkill__writing_T0	
true	80%
false	20%
absurd	0%

hasSkill__writing_T0	
true	58%
false	42%
absurd	0%

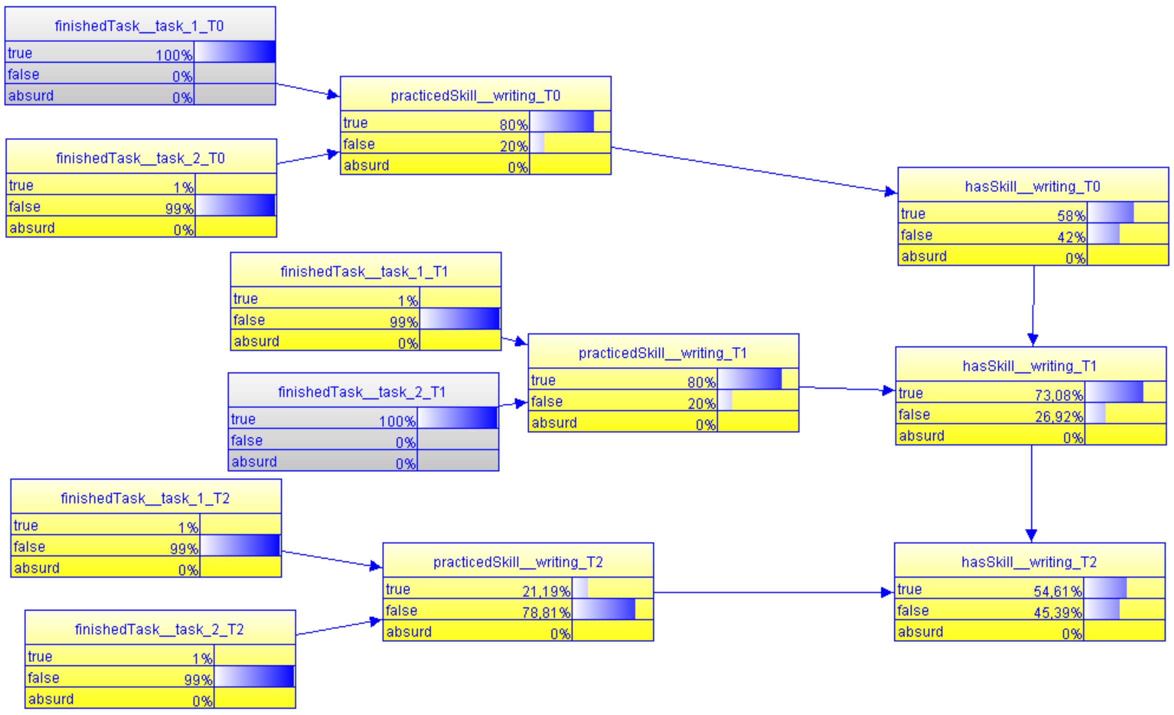
finishedTask__task_1_T1	
true	1%
false	99%
absurd	0%

practicedSkill__writing_T1	
true	20,6%
false	79,4%
absurd	0%

hasSkill__writing_T1	
true	47,78%
false	52,22%
absurd	0%

You can see that the skill decreases a bit, because it was practiced only at T0.
Use the edit network button to return to the edit mode:

Now change the context such that Task 2 also needs writing skill. Add the observation that task 2 was finished at T1. Query the writing skill at time T2.



Additional tasks

- Try adding some more context knowledge, observations and queries.
- What happens if you change the probability tables?
- Discuss the shortcomings and possible improvements of this simple model.
- Open and study the Watchme student model in UnBBayes

Part 4: querying the network in Java

Import the eclipse project into Eclipse and open the "main.java" file:

```
public static void main(String[] args) {

File ubf = new File("C:\\\\gala_workshop\\\\models\\\\studentmodel.ubf");
try {
    UnBBayesWrapper ub = new UnBBayesWrapper(ubf);
    ProbabilisticNetwork pr=null;

    // add context knowledge
    ub.addBooleanEntityFinding("needsSkill", new String[] {"task_1", "writing"},true);
    ub.addBooleanEntityFinding("needsSkill", new String[] {"task_2", "writing"},true);

    // add observations
    ub.addBooleanEntityFinding("finishedTask", new String[] {"task_1", "T0"},true);
    ub.addBooleanEntityFinding("finishedTask", new String[] {"task_2", "T1"},true);

    // make query
    List<QueryItem> q = Arrays.asList(new QueryItem("hasSkill",
                                Arrays.asList("writing", "T2")));

    // run query
    pr = ub.query(q);

    // collect output probabilities
    System.out.println("OUTPUT:");

    ProbabilisticNode pn = (ProbabilisticNode) pr.getNode("hasSkill__writing_T0");
    System.out.println("p(has writing skill at T0) = "+pn.getMarginalAt(0));

    pn = (ProbabilisticNode) pr.getNode("hasSkill__writing_T1");
    System.out.println("p(has writing skill at T1) = "+pn.getMarginalAt(0));

    pn = (ProbabilisticNode) pr.getNode("hasSkill__writing_T2");
    System.out.println("p(has writing skill at T2) = "+pn.getMarginalAt(0));


} catch (Exception e) {
     System.out.println(e.toString());
     e.printStackTrace();
}}
```

Study the above code, run it (change the path to the ubf file as needed).

Try to create more queries to the network.