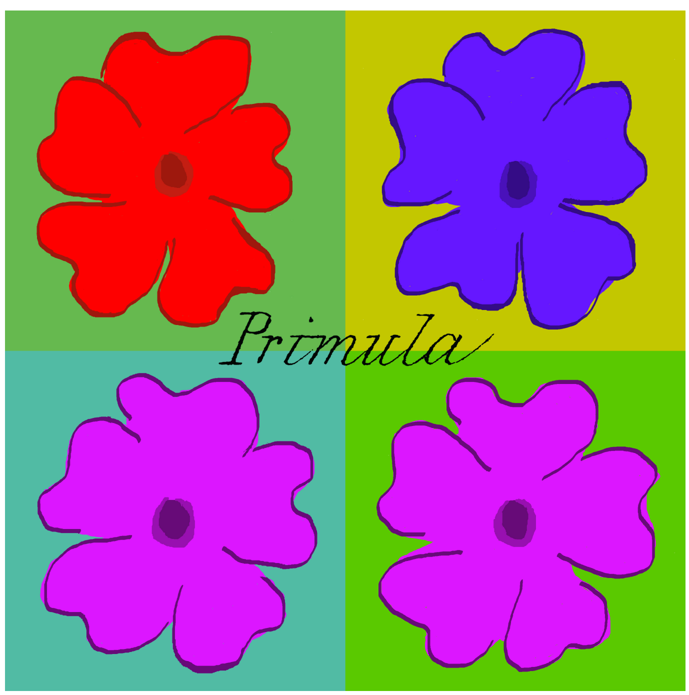
The *Primula* System: user's guide
Version 3.0
Example: `Generalized inference for GNNs by RBNs`

Raffaele Pojer, Manfred Jaeger
Institut for Datalogi, Aalborg Universitet, Selma Lagerløfs Vej 300, 9220 Aalborg Ø
`rafpoj@cs.aau.dk`, `jaeger@cs.aau.dk`

Primula homepage: www.cs.aau.dk/∼jaeger/Primula

September 19, 2023



Introduction

Graph Neural Networks (GNNs) are machine learning models that operate on graphs achieving state-of-the-art results in many domains. Even if it seems obvious, as all the deep-learning frameworks, GNNs are model that after being trained, the inference task is limited on the trained task. On the other hand we have Relational Bayesian Networks (RBNs), which are graphical models coming from the statistical relational learning family, that are much able to perform inference and learning in unseen data, due its probabilistic nature. Furthermore, the powerful language of RBNs allows to represents computationally equivalent GNNs models. Those GNNs models, represented as RBNs, use the best of the two worlds by using the low-level neural components of GNNs and the high-level reasoning of RBNs.

In this example, taken from the paper of Pojer et al. *Generalized Reasoning with Graph Neural Networks by Relational Bayesian Network Encodings*, we want to show the inferential capabilities of this methods by embedding a trained GNN model for node classification into an RBN. This RBNs will be able to perform the same inference task of the GNN model, but also it will be able to have the high-level inference power of graphical models. We have trained a GNN on synthetic node labels that represent a first-order logic formula, as in Barceló et al [1]. The logical-classifier we have adopt for this example is the following:

$$\alpha(x) := \exists^{[2,3]} y(\text{Blue}(y) \wedge \neg\text{edge}(x, y)). \tag{1}$$

Where a node x can be considered as α if there exists between 2 and 3 nodes y that are blue and there is no edge between x and y. We have trained a single layer GNN on graphs with 5-8 nodes and with random coloring labels for the nodes, and the α label as defined before. The trained GNN was then embedded in a RBN for the inference task.

We will dive this example in two parts, the first we will show how this method can enable us to invert the problem, and the second part is focused on inference the graph structure.

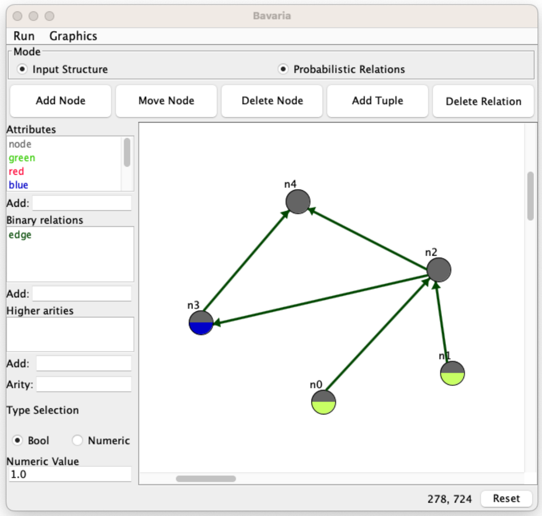
Blue attributes probabilities

The GNN model after being trained, is able to classify if a node x is α or not given the various configurations of node labels in the graph, especially the *blue* attribute. In this example we will see the problem in the opposite direction: given the α label, what is the probability of the *blue* label of being true? The example provided has a graph with partial observation of the *blue* attribute, and full observation on the α attribute. To do this, will use *Primula* specifically with MCMC.

Primula settings

Load the model file `rbn_acr.rbn` and the data file `alpha1-blue.rdef` from the folder.

Select in the *Primula* console Modules:Bavaria to open the graphical data editor. In *Bavaria* press the toggle Probabilistic Relations to view also the attributes of nodes, blue color is for the blue attribute, lime color is for the α_1 attribute, while no colors or grey means nodes without any assignments. You will see something like this:



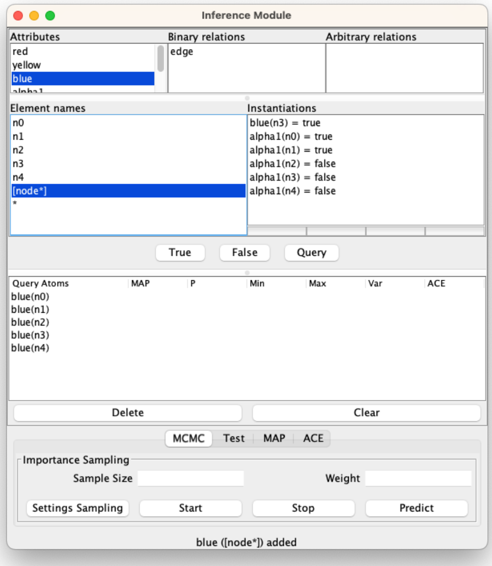
The *Bavaria* window displays the relational structure contained in `alpha-blue.rdef`.

NB: With RBNs and in Primula all the graphs are directed, as you can see also in this graphs edges have direction, but the RBN we have just imported do not take in consideration the direction of the edges.

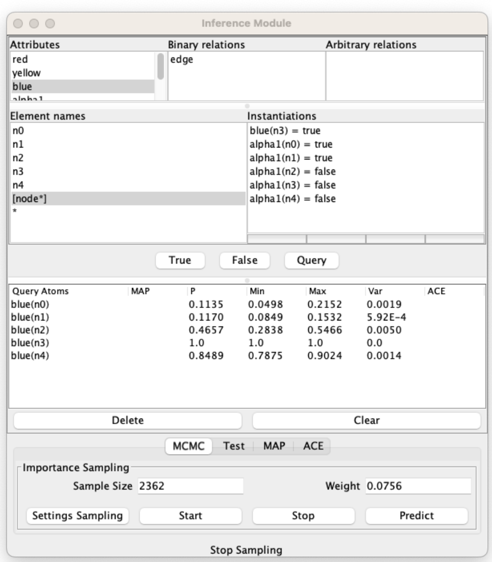
The model `rbn_acr.rbn` is a RBN which represents an ACR-GNN model. The model is a single-layer ACR-GNN, and it was trained on a synthetic dataset by taking inspiration from the paper of The Logical Expressiveness of Graph Neural Networks of Barceló et al (2019). For more details about the model and the dataset, see the paper *Generalized Reasoning with Graph Neural Networks by Relational Bayesian Network Encodings* from Pojer et al (2023). For a more easy reading, we report here the formula of the α_1 logical classifier we have adopted in this example:

$$\alpha(x) := \exists^{[2,3]} y(\text{Blue}(y) \wedge \neg\text{edge}(x, y)). \tag{2}$$

Compute the probability of the blue color for each node. Open the Modules:Inference Module to compute the probabilities of the blue color for each node in the graph. Select the 'Query' button to activate the query mode. Now select the blue attribute from the 'Attributes' list and click on the `[node*]` to select all the nodes.



Use MCMC to compute the probabilities Select the 'MCMC' at the bottom of the window and press the 'Start' button. After few seconds you will see the results in the table above, under the 'P' column. Press the 'Stop' button to stop the computation.



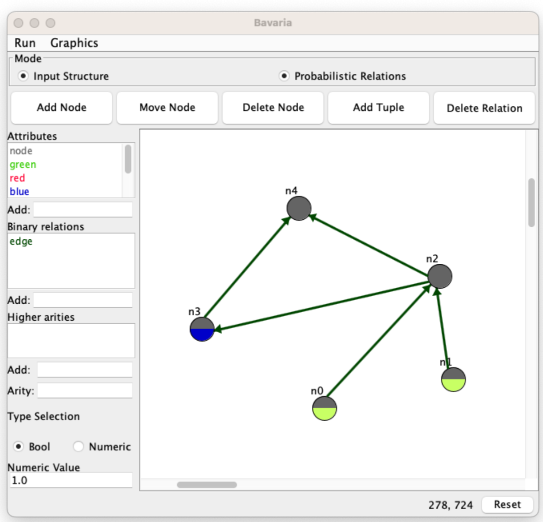
Inference of the graph structure

In this part of the example we are considering a new different setting of inference problem that with the single GNN model will be not possible to do directly. We full observe all the node attributes and where the α labels are true. The *edge* attribute will be set as *probabilistic*, with a prior probability of $p_{edge} = 0.5$. From this setting, we want to infer which is the graph structure that keeps the α labels true (thus the logical formula valid). In this example is it possible to see the capabilities of this method, and if one want to change the p_{edge} value in the RBN, it is possible also to see how this method deal with extreme probability values.

Primula settings

Load the model file `rbn_acr.rbn` and the data file `alpha1-edge.rdef` from the folder.

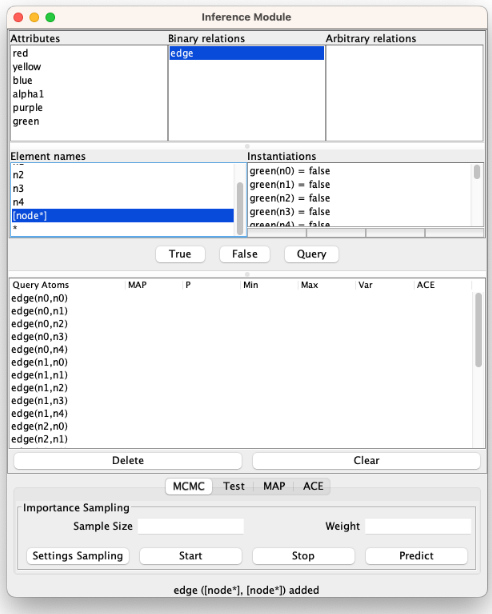
Select in the *Primula* console Modules:Bavaria to open the graphical data editor. In *Bavaria* press the toggle Probabilistic Relations to view also the attributes of nodes, blue color is for the blue attribute, lime color is for the α_1 attribute, while no colors or grey means nodes without any assignments. You will see something like this:



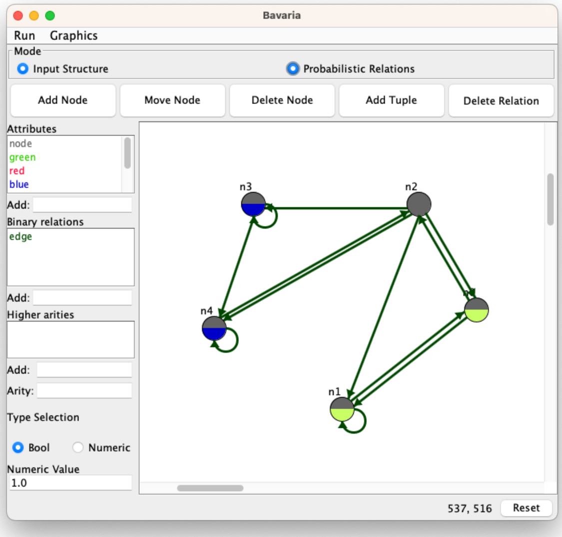
The *Bavaria* window displays the relational structure contained in `alpha-blue.rdef`.

NB: With RBNs and in Primula all the graphs are directed, as you can see also in this graphs edges have direction, but the RBN we have just imported do not take in consideration the direction of the edges.

Generate the graph structure using MAP inference. Open the Modules:Inference Module and press the 'Query' button to activate the query mode. Now select the edge attribute from the 'Binary Relations' list and double-click on the `[node*]` to query the edge attribute for all the nodes. You will see something like this:



Configure the MAP inference tool to compute the graph structure. Select the 'MAP' at the bottom of the window and press the 'Setting MAP' button. A new window will appear. Enter the number of restarts you want to perform in the first text box (e.g. -1 for infinite restarts, or 10 as in the paper examples). After that, click on the 'Start' button to start the MAP inference. After a while, the MAP inference will stop to the restarts number you have set, otherwise press the 'Stop' button to stop the MAP inference. Assign the results of the MAP inference to the graph using the 'Set MAP Vals' button. In *Bavaria* you will see the results with the new graph structure.



References

[1] Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. Logical expressiveness of graph neural networks. 2019.