Prometheus AI

Sean Stappas — Supervised by Prof. Vybihal

McGill

Introduction

Prometheus is an artificial intelligence system designed to control and coordinate multiple robots. All the thinking and decision-making is done by the system, which collects data from all the robots' sensors.

The architecture of Prometheus is loosely inspired from the structure of the human brain, and is composed of the following four layers: the **Neural Network (NN)**, the **Knowledge Node Network (KNN)**, the **Expert System (ES)**, and the **Meta Reasoner (META)**.

The **Neural Network** layer consists of a network of neurons with a structure similar to neurons in the human brain. It is the interface between the robots' sensors and the rest of the AI system.

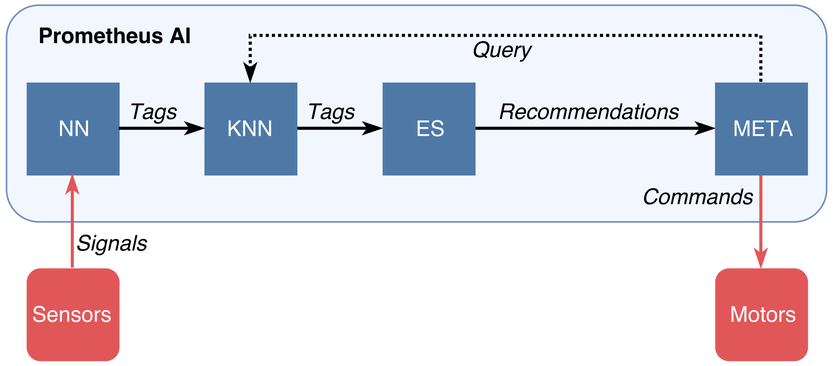
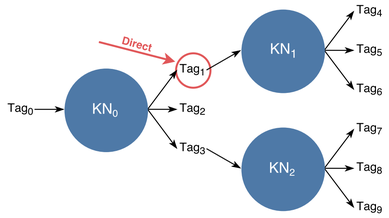
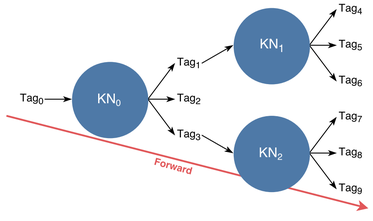


Figure 1: Prometheus AI model with labeled input and output.

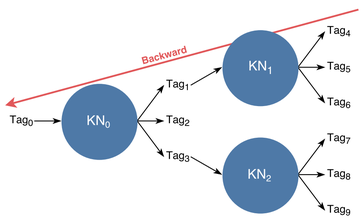
The **Knowledge Node Network** layer is the analog to memory in the human brain. It takes in the tags provided by the NN and outputs tags based on its knowledge of the environment. It consists of interconnected **Knowledge Nodes** (KNs), which are abstract structures representing memories and their connections to other memories. The KNN has four ways of searching to fire KNs and activate tags: **direct**, **forward**, **backward** and **lambda**.



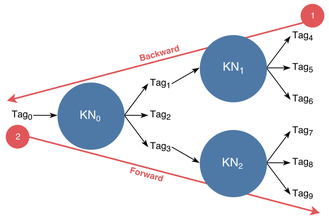
(a) Direct searching.



(b) Forward searching/thinking.



(c) Backward searching/thinking.



(d) Lambda searching/thinking.

Figure 2: Methods of searching and thinking in the KNN.

The **Expert System** layer is a basic logic reasoner. It is not aware of its current reality or any context. It takes in the tags provided by the KNN and interprets them as either **facts**, **recommendations** or **rules**.

Fact	Meaning
$P(x)$	x is true or active.
$P(x = 1)$	x is equal to 1.
$P(x\ != 1)$	x is not equal to 1.
$P(x > 1)$	x is greater than 1.
$P(x < 1)$	x is less than 1.
$P(\&x)$	x can take any integer value.

(a) Examples of simple facts in the ES.

$$Fact := P(A_1, \ldots, A_n)$$

(b) The form of a fact in the ES. P is a predicate, and A_i is an argument.

$$Rule := Fact_1 \ldots Fact_m \rightarrow Tag_1 \ldots Tag_n$$

(c) The form of a rule in the ES, with m input facts and n output tags.

Figure 3: Facts and Rules in the ES.

The **Meta Reasoner** represents high-level reasoning in the human brain. It is aware of its environment and context, and makes decisions based on what it believes to be right.

Problem

There are two main assigned tasks over the past two semesters:

1. Design and implement the **Expert System** and **Knowledge Node Network** in Java.
   - Create an initial design.
   - Build a code skeleton.
   - Implement integration and unit tests.
2. Supervise undergraduate students working on Prometheus.
   - Provide resources.
   - Review code.

Design & Implementation

The **tags** are implemented using a `Tag` Java class, with `Predicate` and `Rule` subclasses. The `Predicate` class is a superclass of the `Fact` and `Recommendation`. Searching in the **Knowledge Node Network** is implemented by `directSearch()`, `forwardSearch()`, `backwardSearch()` and `lambdaSearch()`.

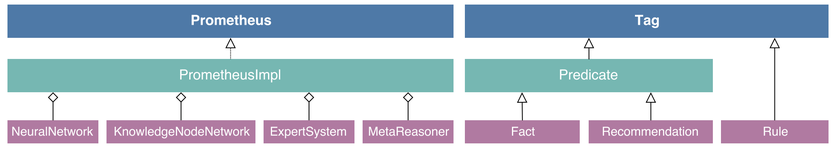


Figure 4: UML diagrams of the Prometheus and Tag packages.

The most important method in the **Expert System** is `think()`, which executes thinking cycles and output recommendations.

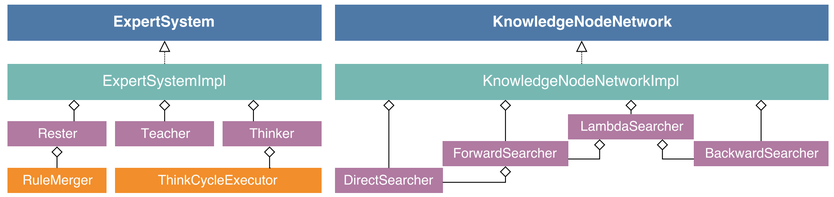


Figure 5: UML diagrams of Expert System and Knowledge Node Network packages.

The Java **graph visualization** library GraphStream was used to visualize the KNN and the process of searching or thinking through it.

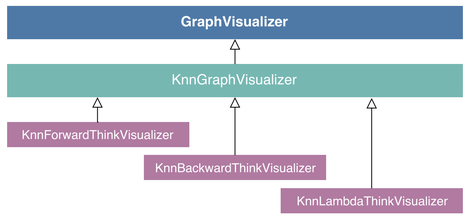


Figure 6: UML diagrams of the graphing package.

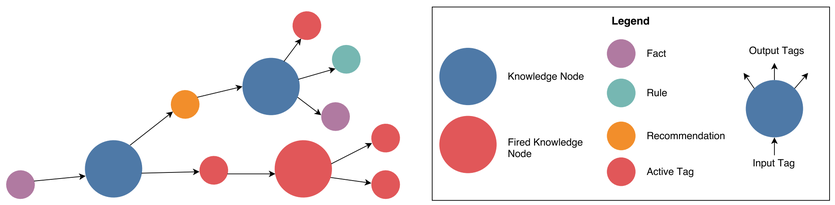


Figure 7: Legend for the visualization of the KNN.

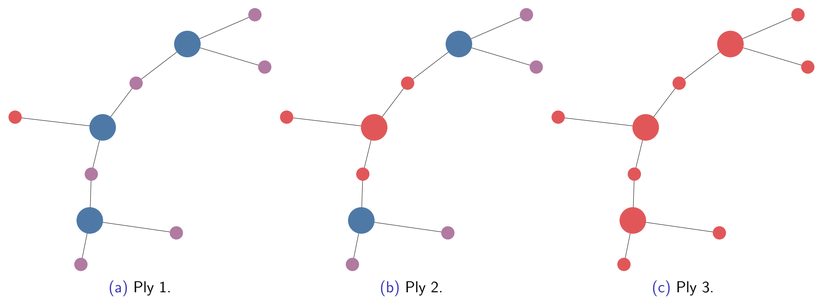
Results & Tests

Unit tests were created using **Mockito**, **TestNG** and **TestNG**. These tests test the functionality of isolated methods. **Integration tests** were also created with **TestNG**, testing end-to-end behavior.

State	Ready Rules	Active Rules	Active Facts	Active Recommendations
Initial	$(A)(B) \rightarrow (D)$ $(D)(B) \rightarrow (E)$ $(D)(E) \rightarrow (F)$ $(G)(A) \rightarrow (H)$ $(E)(F) \rightarrow (\#Z)$		$(A), (B)$	$(\#X), (\#Y)$
⋮	⋮	⋮	⋮	⋮
Final	$(G)(A) \rightarrow (H)$	$(A)(B) \rightarrow (D)$ $(D)(B) \rightarrow (E)$ $(D)(E) \rightarrow (F)$ $(E)(F) \rightarrow (\#Z)$	$(A), (B),$ $(D), (E)$ (F)	$(\#X), (\#Y), (\#Z)$

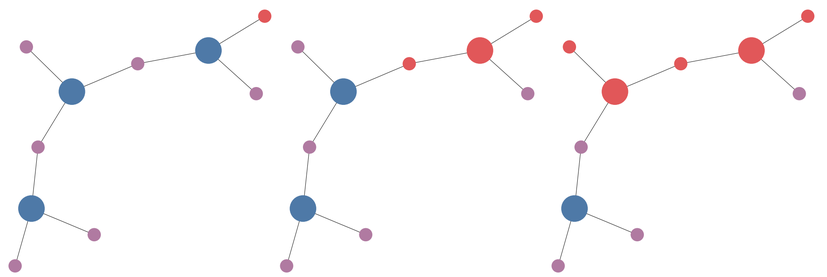
Figure 8: ES test setup representing simple rules and facts that must be brought to quiescence.

- Graph visualization tests.
  - Iterations of the KNN searching algorithms are presented visually.
  - Small nodes are Tags, big nodes are Knowledge Nodes (KNs).
  - Red nodes represent active Tags or fired KNs.
  - For non-active Tags, Facts are purple, Rules are blue and Recommendations are orange.



(a) Ply 1. (b) Ply 2. (c) Ply 3.

Figure 9: Forward thinking visualization in the KNN.



(a) Ply 1. (b) Ply 2. (c) Ply 3.

Figure 10: Backward thinking visualization in the KNN.

Conclusion

- Skills learned:
  - Planning and implementing large software project.
  - Time management.
  - People management.
- Possible future work:
  - Implement the missing NN and META layers.
  - Explore further features in the KNN, such as learning and attention.