PROMETHEUS AI PHASE 1

Sean Stappas
260639512

April 10, 2017

Abstract

Acknowledgments

Contents

Abbreviations

NN	Neural Network
KNN	Knowledge Node Network
ES	Expert System
META	Meta Reasoner
OOP	Object-Oriented Programming

1 Introduction

The goal of this project is to create an artificial intelligence system to control multiple robots. Applications for this type of system include robots in hazardous environments, such as in outer space (Mars, Moon, etc.), in nuclear plants after a nuclear disaster, and in military zones.

The structure of the system is inspired from the functionality of the human brain, and is composed of the following four layers (in order of increasing abstraction): the Neural Network (NN), the Knowledge Node Network (KNN), the Expert System (ES), and the Meta Reasoner (META), as can be seen in Figure 1. These will be described in Section 2.

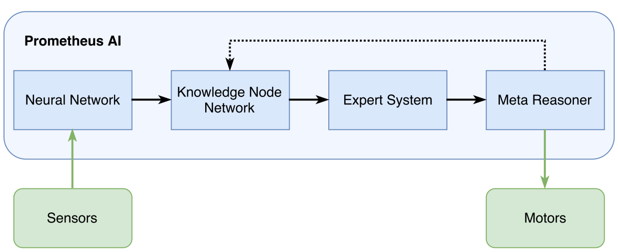

Figure 1: Prometheus AI model.

2 Background

2.1 Neural Network

The NN layer consists of a network of neurons with a similar structure to neurons in the human brain. In the context of this project, it is the interface between the robots' sensors and the rest of the AI system. Informational tags are generated by this layer based on observations that the robots make in their environment. These tags are passed

2.2 Knowledge Node Network

The KNN layer represents memory in the human brain. It takes in the tags provided by the NN and outputs tags based on its knowledge of the environment. These output tags can be simple facts, such as "I see a wall", or can be recommendations for future actions such "turn left". These tags are passed on to the next layer, the Expert System (ES).

The KNN has three main ways of thinking: forwards, backwards, and lambda. Forwards thinking is the simplest, and can be seen in Figure 2.

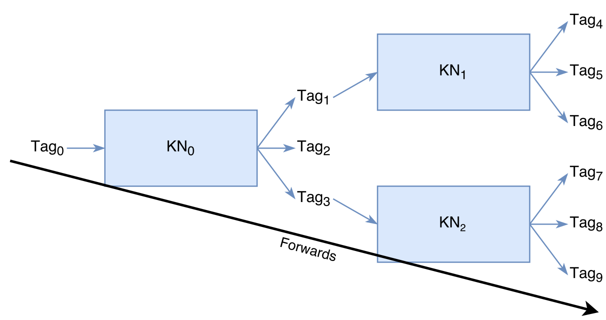

Figure 2: Thinking forwards in the KNN.

Thinking backwards starts at the output tags of Knowledge Nodes, and works backwards, as seen in Figure 3.

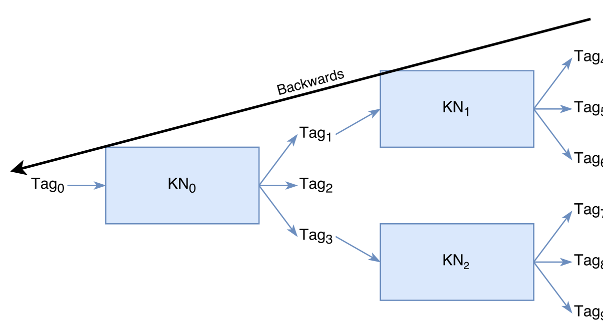

Figure 3: Thinking backwards in the KNN.

Lambda thinking uses a combination of forwards and backwards, and can be seen in Figure 4.

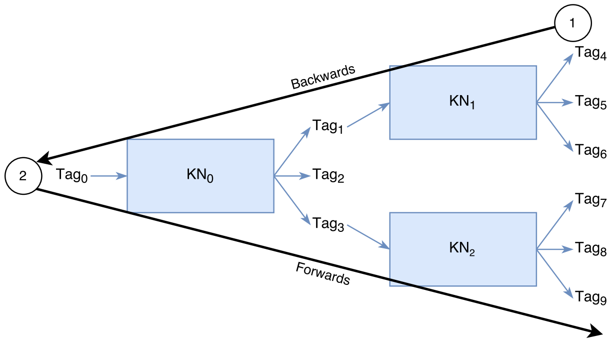

Figure 4: Lambda thinking in the KNN.

2.3 Expert System

The ES layer is a decision maker based on pure facts. It is not aware of its current reality, or any context. It simply has a list of known Facts, in the form of tags, and outputs tags based on known relationships, called Rules. These output tags are passed on to the final layer, the Meta Reasoner (META).

2.4 Meta Reasoner

The META layer represents high-level reasoning in human brains. It is aware of its environment and context, and makes decisions based on what it believes to be right. It is paranoid, and constantly checks whether the tags reported by the rest of the AI system make sense based on its expected view of the world. If it decides to make a decision, it sends a command to the actuators of the robots to decide how to move. If it is not happy with the recommendation(s) from the ES, it may initiate another think cycle in the KNN to generate new recommendations(s).

3 Problem

The assigned task was to construct two out of the four layers listed in Section 2: the Expert System (ES), and the Knowledge Node Network (KNN). The other two layers are to be completed by another Honours Thesis student.

As a deliverable for the end of this first semester, it was required to complete a prototype of these two layers, with basic versions of the functionality described in Section 2. This was to be done entirely in Java.

4 Design

4.1 Efficiency (Space & Time)

A very important consideration when designing the system is speed. Since the robots may have to react very quickly to stimulus in the environment, the reasoning in the AI must be as fast as possible. This is especially true in the hazardous environments for which this system could be useful for, as specified in Section 1. An example of a design choice that was made to improve speed is the use of Java Sets for most of the collections in the ES and KNN layers. The original specifications mentioned using ArrayLists, but, since there is no specific iteration order necessary for most operations in the ES and KNN, these collections were changed to Sets.

When designing every algorithm in the system, the first criterion in mind is speed. Indeed, for the `think()` methods of the ES and KNN, it was important to think about the right way to leverage all the available data structures to maximize speed.

4.2 Object Oriented Design

Another important choice is to leverage object-oriented design as much as possible. Object-oriented programming (OOP) allows extensive planning before even beginning to write code, which can identify any flaws in the initial design. It also allows the code to be very clean and reusable. Since Java is the programming language chosen for the project, OOP is also the natural way to proceed. To follow OOP, all the structures described in Section 2 will be designed around Java classes and the methods associated with those classes. OOP principles such as polymorphism and encapsulation will be followed closely.

For example, the "tags" described in Section 2 need to be as general as possible. A natural choice for this structure would be a Java String, which would be relatively simple to pass around the system. However, these tags represent various concepts; each tag can either be a Fact, a Recommendation, or a Rule. If implemented as Strings, the tags would have to be encoded on creation to represent each concept and decoded on use to retrieve the important information. This seems like a bad use of the OOP principles of Java. For this reason, the tags are instead implemented using a Tag Java class, with Recommendation, Fact, and Rule subclasses. This should also make manipulating the Tags faster, while incurring a slight memory overhead. To store these Tags in a database, they can be converted to JSON format. On read from the database, they can be easily decoded.

4.2.1 Abstraction

One important design aspect is that the system should be as abstract as possible, while still performing its desired task. For instance, the system should be general enough to perform under simulations, as well as in real-life environments. It should also ideally be able to perform in vastly different environments, with different tasks.

One example of making use of abstraction is the choice to create an interface for both the ES and KNN, since they share some functionality, like the ability to think.

4.2.2 Encapsulation

Each layer of the system has unique, localized functionality that does not need to be visible from the rest of the system. For instance, the intricacies of the `think()` method in the ES and KNN layers do not need to be known to the rest of the system.

4.2.3 Polymorphism

Many methods in the system can have different functionality depending on context. For instance, the `think()` method in the KNN and ES layers can take an optional number of think cycles to specify a threshold for quiescence.

4.2.4 Inheritance

4.3 Implementation

The details of the specific design choices for the Expert System and Knowledge Node Network will now be discussed.

4.3.1 Expert System

The Expert System layer is based around the `ExpertSystem` Java class.

4.3.2 Knowledge Node Network

The Knowledge Node Network layer is based around the `KnowledgeNodeNetwork` Java class.

4.4 Testing

4.4.1 Unit Tests

Unit tests were generated for the KNN and ES.

4.4.2 Integration Tests

Integration tests were made to test the combined functionality of the KNN and ES.

4.5 Documentation

Proper documentation of the source code is very important. This is to ensure that anyone wanting to work with the code which was designed has an easy time doing so. Also, for anyone wanting to understand how the system works, documentation is essential. This documentation was achieved through Javadoc and UML diagrams.

4.5.1 Javadoc

Extensive Javadoc was generated for the entire code base. This can be found here: `http://cs.mcgill.ca/~sstapp/prometheus/index.html`.

4.5.2 UML

UML diagrams of every package in the code base were created.

5 Plan for Next Semester

The plan for next semester is to finalize the two layers that were started this semester, and to test them on the robots available in Prof. Vybihal's lab.

5.1 Finalization

5.1.1 Knowledge Node Network

Many complex features of the KNN layer are still to be implemented, such as backwards and lambda thinking, confidence, aging...

5.1.2 Expert System

Some features of the ES layer are still to be implemented, such as more complex Fact checking.

5.2 Integration

One very important task left to be done is to integrate the two layers described in this report (ES and KNN) with the other layers developed separately (NN and META). Ideally, the layers should be able to work together, but there will surely be some conflicts at the interface of the layers. These will have to be resolved when the time comes.

5.3 Testing

5.3.1 Simulation

Once all the layers are functioning together, they can be tested. The first and easiest way to test would be in a simulated environment. One simulator that may be used is Simbad, which is a Java 3D robot simulator [2]. This can allow for some early debugging and fixes.

5.3.2 Physical

One the simulation testing is completed and working properly, the system can be tested in the lab. Prof. Vybihal's lab has multiple robots with ultrasonic sensors, and these will be the test subjects of this phase.

6 Impact on Society and the Environment

6.1 Use of Non-renewable Resources

As a purely software-oriented project, there are no physical materials needed to construct this system.

6.2 Environmental Benefits

The system could be used as a tool to control robots in dangerous environments such as nuclear plants after a radioactive disaster. Indeed, the system could coordinate robots to help contain the damage faster than humans could, thus limiting the risk on the environment.

6.3 Safety and Risk

It is critical that, once this system is completed, it is used in an ethical way, and for the right purposes.

One example of use that may cause ethical concern is in a military setting, where an AI system like the one described here could be used in a battlefield in place of soldiers.

In the very long term, there are concerns with the possibility that an AI system might achieve intelligence and awareness close to a human. If such an AI were to obtain "consciousness" in its own way, should that entity be entitled to its own rights, like humans do?

6.4 Benefits to Society

Going back to the example use case in a radioactive disaster, the system could be used to send robots in an area that would otherwise be very dangerous for humans. This would therefore help prevent the unnecessary loss of life in cleaning up these leaks.

7 Conclusion

This semester, prototypes of the Expert System (ES) and Knowledge Node Network (KNN) layers of the AI model were completed in Java. These prototypes were tested individually and together, with positive results.

The main goal for next semester is to finalize the entire system, implementing more complex features that were omitted for this prototype stage.

References

[1] J. Vybihal, "Full AI model," 2016.

[2] "Simbad 3d robot simulator," http://simbad.sourceforge.net/index.php, (Accessed on 03/27/2017).