Case Study: A Multi-Agent Home Automation System in Haskell

Tanja de Jong
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
t.dejong-4@student.utwente.nl

ABSTRACT

This proposal describes a case study of a multi-agent automation system in Haskell. As the name implies, a multi-agent system is a system that exists of multiple agents, that can individually solve problems. Because automation systems consist of multiple appliances and devices that are connected through a network and communicate with each other, these types of systems are very complex to control [5]. For this reason, a multi-agent system seems to be a good approach to defining a home automation system, which is the case that will be defined for this research. This home automation system should automatically determine when residents return home, in order to prepare the devices in the house without explicit human input. With a system like this in place, residents do not have to adjust their homes to their wishes after their return home, but can go ahead with their daily business.
One goal of this research will be to investigate how well functional languages can be used for the definition of multi-agent systems, because of their mathematical nature. Haskell is the functional language that will be used for this system. This research will hopefully result in an answer to the question of how suitable functional languages are for the definition of multi-agent automation systems.

Keywords

Multi-Agent System, Agent, Logic, Functional Language, Haskell, Home Automation System

1. INTRODUCTION

1.1 Multi-Agent System

As stated before, a multi-agent system exists of multiple intelligent agents that can individually solve problems. Multi-agent systems can be used for solving problems that can not -or not as easily- be solved by a single agent or a monolithic system. These types of systems can have several advantages, for example:

- robustness;
- scalability;
- adaptability.

Because home automation systems consist of multiple appliances and devices that are connected through a network and communicate with each other, these types of systems are very complex to control in a centralized way [5]. For this reason, a multi-agent system might be a good approach for defining a home automation system, which is the case that will be used for this research.

1.2 Haskell

Functional languages are of a mathematical nature, and this might have advantages for defining multi-agent systems [6]. Therefore, it seems to make sense to define a home automation multi-agent system in a functional language. Because Haskell is an advanced and widely used functional language, the system to be defined for this research will be defined in Haskell.

1.3 Home Automation

Home automation is a popular subject nowadays, because of numerous advantages of automation that can add convenience and safety to people's lives. Therefore, a new generation of such systems with even more automation is called for.

Home automation is ”a way of integrating technology and services in order to gain a higher quality of living”, according to Stichting Smart Homes, Nationaal Kenniscentrum Domotica & Slim Wonen. This is currently a very popular area of research as the number of new home installations are expected to increase enormously, according to Telecom research firm Berg Insight [3] and market research and intelligence firm ABI Research [1]. Nowadays, home automation systems can range from simple remote controls to complex networks with many different component that can each have their own form of intelligence and automation. The most intelligent systems that are currently in general use need user input through a pc, smart phone or tablet to start their automation processes. Automation results in numerous advantages for people, for example:

- added convenience/fun/peace of mind: people have to perform less boring tasks themselves
- timesaving: automation does work that people would have to do otherwise
- moneysaving on utilities: devices can be turned off automatically when nobody's home
- better for environment: devices can be turned off automatically when nobody's home
- increased home safety: lights can be turned on automatically when somebody arrives home

1.4 Problem Statement

Multi-agent systems can be widely used for various applications, among which home automation, and can have many advantages in comparison to single-agent or monolithic systems. Because logic can be used very well to reason about these multi-agent systems [7] and the mathematical nature of functional languages, it seems that a functional language would be a good choice for the definition of a multi-agent system. Proposed here is to investigate how suitable a functional language is for the definition of a multi-agent automation system, specifically, a home automation system in Haskell.

2. RESEARCH QUESTIONS

How suitable is Haskell for the definition of a multi-agent automation system, for example by looking at properties such as ease of programming, length of code, flexibility, readability, etcetera?

- For which specific properties will this suitability be investigated?
- How does such a definition in Haskell compare to one or more similar definitions in other programming languages, for these properties?

How well does the mathematical nature of functional languages comply with logical reasoning of multi-agent systems?

3. BACKGROUND

3.1 Multi-agent systems

Multi-agent systems are systems composed of multiple interacting computing elements, known as *agents* [8]. Some of the applications for multi-agent systems are workflow and business process management, distributed sensing, information retrieval and management, electronic commerce, human-computer interfaces, virtual environments, social simulation, and many more [8]. In such a system, each agent has its own set of knowledge about the environment that is gathered using sensors and communication with other agents, and its own desires that it attempts to fulfill through reasoning using actuators and communication with the other agents, as showed in figure 1. For this proposal, a multi-agent system consisting of rational agents, that reason according to the Belief-Desire-Intention model, is investigated.

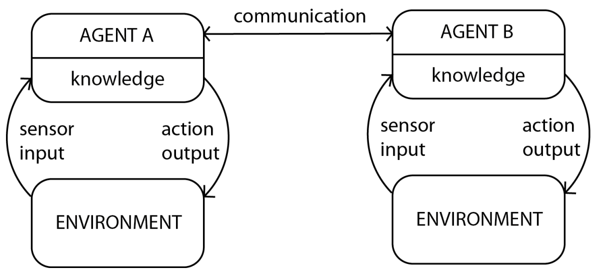

Figure 1. A multi-agent system

3.1.1 Rational Agents

According to Wooldridge "an *agent* is a computer system that is *situated* in some *environment*, and that is capable of *autonomous* action in this environment in order to meet its design objectives." [8]. Thus, it is an entity that acts upon the environment it inhabits. An agent is said to be rational if it chooses to perform actions that are in its own best interests, given the beliefs it has about the world. Therefore, rational agents are expected to have the following properties, according to Wooldridge and Jennings [7]:

- autonomy;
- proactiveness;
- reactivity; and
- social ability.

3.1.2 The Belief-Desire-Intention Model

The Belief-Desire-Intention (BDI) model sees agents' actions as rational in the sense that they are constituted of beliefs, desires, and intentions. The beliefs of an agend correspond to the agent's *knowledge* about its environment, but may be incomplete or incorrect. An agents *desires* then represent what an agent would want to accomplish and its *intentions* represent the desires that it has committed to achieving [7]. This model was originally developed by Michael Bratman [4].

3.1.3 Mathematical logic

Mathematical logic is one of the most important tools available in computer science and artificial intelligence today and can be used in the engineering of rational agents. By fixing on a structured, well-defined artificial language it is possible to investigate the question of what can be expressed in a rigorous, mathematical way. This way, any ambiguity can be removed and by adopting a logic-based approach, one makes available all the results and techniques of what is arguably the oldest, richest, most fundamental, and best-established branch of mathematics. One of the most desirable features of a software specification language is that it should not dictate how a specification should be satisfied by a definition. A logical specification language has exactly the necessary properties. It does not dictate how a certain agent *a* should go about an intention, it is simply expected that *a* behaves as a rational agent given such an intention. Therefore, it seems straightforward to define a logical language for the multi-agent system described in this proposal.

3.2 Functional languages

Functional languages are of a mathematical nature and have algebraic data types, which result in the possibility for pattern matching. Because of this, agents can reason and communicate using so-called *Propositions*, which are expressions that offer the opportunity to pattern match. For example, for the home automation system there may be a data type:

```
data Proposition = PassesBy ComputerAgent HumanAgent
|  Send Message ComputerAgent ComputerAgent
|  And Proposition Proposition
|  Not Proposition
```

The proposition `PassesBy ComputerAgent HumanAgent` can be used to let another agent know that a certain person, like a resident, has passed by. When another agent receives a message with this particular proposition, it can reason about it and decide if it needs to execute certain actions.

In this scenario, every agent basically consists of a function with its own state that can execute actions and update its state accordingly, whenever a proposition from another agent is received. In the home automation system, these actions may include sending a message to another agent, turning on the radio, etcetera.

4. RESEARCH METHOD

In order to answer the research questions, first of all, research will have to be done to find out whether comparable systems in other languages exist and what the advantages and disadvantages of these languages are. In this case, comparable solutions denote definitions of various multi-agent automation systems that do not necessarily have the exact same functionality as the language that will be defined. After this, it needs to be determined on what properties the systems will be compared and the system can be defined in Haskell. When this is decided, a comparison can be made based on these properties between the system that is defined in Haskell and at least one of the systems that were found in other research. Basically, the research method will consist of these steps:

- performing research into existing comparable definitions in other languages;
- determining on which properties the systems will be compared;
- defining a multi-agent home automation system in Haskell as described in this proposal;
- compare the newly defined system to the existing systems on the selected properties.

5. RELATED WORK

There is an extensive amount of research on home automation. However, most of this research does not focus directly on the topic of this proposal. The degree of automation most common in research needs user input in the form of using a smart phone, computer, or tablet to control the home automation system instead of having the intelligence to automate this. Besides this, there is a lot of focus on energy efficiency, for example by turning of devices when nobody is home [2]. Although this could of course be accomplished by a higher degree of automation, the focus of the research mentioned in this proposal is on turning on devices when habitants return home, not on turning them off when they leave the house. [5]

6. RESEARCH SCHEDULE

Date	Description
13-04-2014	Research on other systems completed
04-05-2014	Properties that will be compared decided
11-05-2014	Multi-agent automation system in Haskell defined
25-05-2014	Comparison of systems completed
31-05-2014	Draft paper finished
13-06-2014	Paper finished

7. REFERENCES

[1] ABI Research. Home automation, security, and monitoring. 2012.

[2] A. S. al sumaiti, M. H. Ahmed, and M. M. A. Salama. Smart home activities: A literature review. 2014.

[3] Berg Insight. Smart homes and home automation. *M2M Research Series*, 2013.

[4] M. E. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press: Cambridge, MA, 1987.

[5] G. Morganti, A. Perdon, G. Conte, and D. Scaradozzi. Multi-agent system theory for modelling a home automation system. 2009.

[6] A. Solimando and R. Traverso. Designing and implementing a framework forbdi-style communicating agents in haskell. 2012.

[7] M. Wooldridge. *Reasoning about Rational Agents*. The MIT Press: Cambridge, MA, 2000.

[8] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley and Sons, Ltd, 2002.