FACULTY OF ENGINEERING TECHNOLOGY
DE NAYER (SINT-KATELIJNE-WAVER) CAMPUS

User-friendly House Automation using IDP

Thijs ALENS

Promotor(en): J.Vennekens, S. Vandevelde

Master's Thesis submitted to obtain the degree of Master of Science in Electronica-ICT afstudeerrichtingprogramme Engineering Technology

Academic year 2024 - 2025

Contents

Chapter 1

Beschrijving van deze masterproef in de vorm van een wetenschappelijk artikel

The thesis should also contain a short scientific article. If you write your thesis in Dutch, you must write the article in English, and vice versa. We advise you to employ the IEEE Manuscript Templates for Conference Proceedings (`https://www.overleaf.com/latex/templates/ieee-conference-template/grfzhhncsfqn`).

Chapter 2

Introduction

2.1 Positioning

Home automation is a way to automate tasks in a home. These can be simple tasks such as automatically turning on lights upon entering a room, to more advanced operations like regulating home climate. Recently, with the coming of Internet of Things (IoT), lots of devices became "smart" and are therefore capable of automation. This event has the potential to significantly streamline daily routines, reduce costs, and improve overall comfort. Typically, a home automation system runs on a server or Raspberry Pi in conjunction with Home Assistant. Home Assistant is an application that integrates various devices from different brands into a single functional app. It also provides a framework through which users can automate their homes.

IDP-Z3 is a reasoning engine that makes decisions based on a formal FO(·) description, which is an extension of FO incorporating types, partial functions, inductive definitions, and aggregates. A key advantage of IDP-Z3 and FO(·) is the separation of knowledge from its interpretation. This allows the user to avoid programming the decision-making model themselves, as this is handled by the reasoning engine. (? link leggen tss IDP en HA ?)

2.2 Problem statement

While Home Assistent is relatively user-friendly, it has limitations. More complex automations are not possible through the user interface (UI) and still require users to modify configuration files, which is far from ideal. The user also needs to manage the various rooms individually and coordinate them with each other. Thus, seamlessly integrating home automation into an existing household presents numerous challenges, especially for individuals with limited technical expertise.

One of the main challenges is the steep learning curve associated with setting up and maintaining home automation systems. If users wish to implement complex automations within their home, they must have a basic understanding of programming and be familiar with specific technical tools. For those without a technical background, this can be a daunting and difficult process. It would be unfortunate if the benefits of home automation were accessible only to those with programming skills or technical expertise.

However, the IDP language itself is not particularly user-friendly and can become quite complex, especially when dealing with intricate rules or scenarios. When users are confronted with a large vocabulary and theory, it can be difficult to maintain an overview and make necessary adjustments. It also has a lot of possibilities that are not usefull in the context of home automation

2.3 Objectives

This thesis seeks to address this issue by investigating how IDP, specifically IDP-Z3, can be utilized to make home automation more accessible. Instead of needing to create automations, triggers, and aligning them with each other, the user only needs to describe the desired behavior of the home in the IDP language. A graphical user interface (GUI) that provides structure, along with a more limited subset of the IDP language, could help make IDP a more viable option for home automation.

The objectives of this thesis are:

- Design a user-friendly UI that is most suited for IDP in combination with home automation.
- Deside on a subset of the IDP-language that has all the functionallity needed for home automation.

The primary research question addressed in this thesis is:
How do we design an IDP-Z3 framework that enables end-users to automate their homes in a user-friendly way?
This overarching inquiry is further explored through the following sub-questions:

- What is the optimal user-friendly interface for addressing this problem?
- Which subset of the IDP-language is needed to configure a home?

Chapter 3

Literature review

3.1 Home automation

3.1.1 What is home automation

Home automation is a very brought term that is used when talking about automating a home. It can be very simple like automaticially turning on a light when entering a room, to very complex problems like tuning the temperature of the house depending on various variables. It also refers to the security of a home: when should the security camera automaticially record, when should the doors automaticially lock, what should happen if the alarm goes of... Automaticially doing things, like automaticially making coffee at the start of the day, is also considered home automation. It can also be used to control the energy usage throughout the day to reduce costs.

Home automation consists of 3 main parts:

- Smart home devices
- Some kind of smart hub/server
- An application

3.1.1.1 Smart home devices

A smart home device (or smart devices) can be either of two things: a sensor or an actuator. A sensor can detect an event while an actuator can do something based on a trigger. A simple example can be a motion detector (sensor) that automaticially turns on a light (actuator) when there is movement. Depending on the device that is used, their may be some more advanced (smart) features in them. For example, the light could come with a sensor that mesures the brightness of a room, which automaticially adjusts the brightness of the light based on how much natural light is comming in. This is rarely used, because most of the time the user would like to configure this themself. The smart devices are, in other words, the actual hardware behind a smart home which, in most of the cases, do not have any smart home logic in them.

3.1.1.2 Smart hub/server

The smart hub is a central device that forms a connection between the complex hardware of the devices and the user. It's function is to recieve data from the sensors and sent commands to actuators, a central hub for the devices so to speak. This hub can be provided by a manufacturer of smart devices specifficially for their devices, or it can be a generic one that tries to be as complient with as many devices as possible.

3.1.1.3 Application

The application is a way for the user to configure their home. Most of the time this is done by creating a GUI (graphical user interface) in which the user can create automations, see the status of devices, see what automations are running, easily communicate with the devices...

3.1.2 Home assistent

Home assistent is an open source, all in one application for home automation. It runs a local server where smart devices can be connected to and configured, because it is local, it ensures great security of the users data. It has a large (and growing) comunity and therefore lots of compatiblity for different devices and brands. So it doesn't take 20 apps anymore to control a home enviroment. Home assistent also provides multiple UIs (dashboards) where users can see the state of their home. The user can pick a dashboard that was made by an other user, design one themself or build/expand on an existing dashboard. They can interact with the devices in their dashboard. The user can also create automations using a UI (not the same as the dashboard), these are actions that are excecuted if a trigger for set automation happens. To fully understand how these automations work, a further understanding of how home assistent works is required.

3.1.2.1 Entities

these are the lowest level possible. They represent single sensors/actuators like a temperature sensor, a lightswitch, a light...

3.1.2.2 Devices

These are a group of entities. It could be that a device has 1 entity (ex. a lightswitch), but it also could have multiple entities (ex. a motion sensor that is also capable of capturing the temperature).

3.1.2.3 Areas

These are groups of devices that could correspond to rooms in a house. The living room could have devices like a light-switch, a motion sensor (that detects if someone is in the room), a set of speakers... All of these devices could be grouped together in one area.

3.1.2.4 Scenes

Scenes are used to set an enviroment to a specific state. This enviroment doesn't need to be an area, it could manage smaler or bigger. If the user would want to watch a movie, a scene could be set up. It would turn on the tv, dim the lights, turn off the music that was playing througout the house...

3.1.2.5 Automations

Automations are a used to automate things. These automations consist of three things.

- Triggers

  These are used for the activation of the automation (ex. a motion-sensor detects motion)

- Conditions

  These are extra conditions that need to be met before excecuting the automation (ex. someone needs to be home)

- Actions

  These are things that happen after a trigger and the conditions are met (ex. turn on the light in the room)

3.1.2.6 Scripts

Scripts are automations without a trigger. This means they can't be run without being activated by an automation or another trigger. This is mostly used to do the same tactions from different automations. If you set the lights in a single room up in scripts (turn all the lights in the room on). You could create a script that runs all the scripts to turn on the lights when the user enters the house. This way all the lights (that are defined in different rooms), could be activated all at once.

kleine samenvatting met vb

3.1.3 The UI for automations

leg uit hoe automations gedaan worden in HA.

3.1.4 Available options

3.2 IDP-Z3

IDP-Z3 is a reasoning engine that can perform a variety of reasoning tasks on knowledge bases in the FO(·) language (IDP-Z3: a reasoning engine for FO(·)). The idea is to provide knowledge (in the form of FO(·)) that is interpreted by the IDP-Z3 system to produce an output, which can be lots of things thanks to the knowledge based paradigm.

3.2.1 The knowledge based paradigm

Reasoning engines enable the Knowledge Base paradigm (Denecker and Vennekens 2008), in which systems store declarative domain knowledge, and use it to solve a variety of problems. The knowledge based paradigm states that the knowledge base of a sertain domain should be seperated from its inference tasks (interpretations). This implies that the knowledge could be re-used for multiple use cases in the same domain. For example: The knowledge states that a sertain number needs to be greater then 7. Interpretations could be:

- 5 is not greater then 7 (it checks if a given value complies with the knowledge)
- 8, 9, 10, 11, 12... (generates models that comply with the knowledge)
- ...

In this example the knowledge is used for 2 interpretations (more are possible see below). When this would have been described in an imperative programming language, to seperate programs would have been nessecairy for this problem. This is where the power of a knowledge based system lies.

It is explained below with simple examples, it can of course handle much more complex rules.

- It can produce models based on the knowledge that is provided (model expansion)

  It is defined that a person must be between 0 and 100 years old. IDP-Z3 can provide models where persons are 1, 4, 40, 99... years old.

- It can understand links between types (propagation)

  It is defined that a window can not be open if the heating is on. IDP-Z3 can work out that if the heating is off, the window can be open.

- It can explain why a sertain model can exist (explanation)

  EX?

- It can optimise situations (optimisation)

  It is defined that a sertain number must be greater then 2 and must be devisable by 7. IDP-Z3 can work out that the minimal value of that number is 7.

- It can check if some knowledge is redundend (relevance)

  It is defined that a sertain number must be positive, greater then 5 and greater then -3. IDP-Z3 can work out that the last rule is redundend.

3.2.2 FO(·)

FO(·) (aka FO-dot) is the Knowledge Representation language used by the IDP-Z3 reasoning engine (IDP-Z3: a reasoning engine for FO(·)). It is an extention of first-order logic (FOL), which makes use of operators the following constructs â§, â¨, Â¬, *â*, *â*, *â*.

(tabelletje dat symbooltjes uitlegt)

The FO(·) language consists of, at least, a vocabulary and a theory. The vocabulary is used to describe what symbols will be used in the theory. The theory consists of rules that apply to the symbols. The structure, which is optional, describes a single, specific situation. A quick and easy example to make this clear: Let's say you wanted to represent when a person can speak and read. The rules would be the following:

- the person should be at least 1 year old to be able to speak
- the person should be at least 6 years old to be able to read

To represent this in FO(·) there should be a type Person which represents the people that we would like to check, a function to check the age of a person and two predicates to check whether a person can speak or read. This should come in the vocabulary V. The rules should be implemented in the Theory T which makes use of the previously defined vocabulary V. In this example the rules are implemented as definitions, a new and more clear way to represent rules. They can be interpreted as implications. These defenitions can be read as following:

- For every person, it holds that they can read if and only if they can speak and are at least 6 years old.
- For every person, it holds that they can speak if and only if they are at least 1 year old. Outside of the definitions, an additional rule is described to ensure that a person cannot have a negative age.
- For every person, it holds that they must be at least 0 years old.

If these rules were to be applied specifically to three individuals, for example, Tibo, Luca and Daan, this particular situation can be described within the structure. By not specifying the ages of the individuals, this can be interpreted by the reasoning engine (in this case, IDP-Z3).

The output of the reasoning engine gives possible models. These models all comply with the theory that was provided. It also shows that the reasoning engine has put in some values for the ages of the persons, altough this was never specified in the FO(·)-description. This is the power of splitting the knowledge (FO(·)-description) and the interpretation (done by IDP-Z3). One knowledge base (KB) can be used for multiple purpusses. It can be used to generate models (like in the example), it can be used to check whether a model is valid, it can be used to teach a user the toughtprocess behind a sertain dissicion... " The KB is âonlyâ a formal representation of declarative properties of the domain. This imposes a strong requirement on the KB language (FO(·)): its expressions should be interpretable as (informal) propositions about the domain, and this interpretation, its informal semantics, should be as clear, precise and objective as possible" (Building a Knowledge Base System for an integration of Logic Programming and Classical Logic). This is in strong contrast to declarative programming frameworks. When using such a framework, all the different operations (generation, validation, explaining...) must be programmed seperatly. On top of that, if a change was to occur, all the programs have to be rewritten, unlike with a split KB and interpretation, where only the KB must be adjusted.

constraints dat verder gaan dan implicaties
logische of die geinterpreteerd wordt door idp-z3
conflicting rules
validation (simulatie)