TOKYO METROPOLITAN UNIVERSITY

首都大学東京

Human-centric Semantic Reasoning and Optimization for Smart Home

Thesis submitted for
acquiring the degree

of

Doctoral of Engineering

by

Noel Tay Nuo Wi

Under the guidance of
Professor Naoyuki Kubota

Department of Human Mechatronics System
Graduate School of System Design
Tokyo Metropolitan University

Tokyo, Japan 2017

Abstract

Smart home consists of various kinds of Internet of Things (IoT) devices connected to the private house that cooperatively provide inhabitants (users) with proactive services related to comfort, security and safety. Examples of services include 1) manipulation of lighting and temperature based on time and context, 2) reminder service of user's schedules by using the nearest output device, and 3) device organization to realize surveillance system. However, current smart homes are developed mostly from the viewpoint of technical capabilities, where users have to decide how the connected devices are going to serve them. They may have to setup the devices based on the available functionalities and specifications of the devices, and also have to alter their living styles according to the role of each device. Besides, most devices can only provide simple services independently. Thus, cooperation among the devices is important.

On the other hand, human-centric approach, which centered on humans' need to enhance their living experience, is an important technological paradigm where services are provided anywhere and anytime based on situation. Smart home abiding this approach should cooperatively maximize fulfillment of quality of life (QOL) for individual users subject to personal constraints. In this respect, the devices are bound to enable communication of information, and their operations are coordinated to deliver services cooperatively via a sequence of device actions called a plan.

Due to personalization and automation, a number of problems have to be solved. First, a means of automatic binding between loosely coupled devices depending on services delivered have to be devised, as manual setup is impractical. Secondly, coordination of devices needs to generate complex plans, without requiring manual specification of sub-plans. Besides, issue of over-constrained goals during service provisions that arises from flawed or contradicting specification from multiple users should be considered. Apart from that, low training data in general environment setting for individual identification should be addressed.

The aim of this research is to establish an integrated system for the human-centric smart home (HcSH) that provides personalized service through loosely coupled devices automat-

ically. This research modularizes the overall system into three modules, which are human identification (HIM), automated planner (APM), and semantic reasoner (SRM). HIM helps select the appropriate QOL, SRM binds the devices by associating them with planning components, which are then used by APM to generate plans for device coordination to maximize QOL fulfillment.

Chapter 1 gives the introduction and design motivation. Chapter 2 presents the related works and literature reviews, as well as justifications relevant to this thesis.

Chapter 3 deals with HIM, which is realized via face identification. For face identification, problems faced are heavy computational load and insufficient learning data. The solution is to use transfer learning to handle data issue while being able to build generalized face model. For face model refinement, active learning is implemented. Experimental results show the method is competitive in terms of accuracy and computational cost compared to current state of the art.

Chapter 4 presents APM, where planning via solving Constraint Satisfaction Problem (CSP) is laid out. CSP in planning is declarative without requiring prior specification of sub-plans, and can handle variables of larger domains. Due to the high possibility of having over-constrained QOL as in practical cases, CSP planner cannot fulfill all of them. An example is a contradicting TV channel request from 2 persons. Optimization through weighted CSP is therefore used to maximize QOL fulfillment. Experiments on weighted CSP shows that the method is capable of performing optimization while generating complex plans.

Chapter 5 is on SRM, where knowledge representation is constructed by Web Ontology Language (OWL) description logic. It models knowledge on home and building layout and device functionalities. OWL is used because it is decidable and that it is endorsed by World Wide Web Consortium (W3C). We deal with case studies based on further inference on building state as an important example to discuss the applicability of the proposed method, and demonstrate the use of building ontology. This is followed by automated device binding and the method to generate basic planning components of rules in automated planning. Finally, an extension to robot complex planning is provided to demonstrate how it can be easily extended.

Chapter 6 demonstrates the applicability of the HcSH, which integrates all three modules through its implementation in a prototype smart home with 5 rooms, which houses 2 persons. Various tests are performed to show the generated plans are near optimal without redundancy. The system is also shown to be scalable given increasing amount of devices. Case studies show that the system can perform well even under short time threshold.

Finally, chapter 7 summarizes the thesis. Future vision of the work is also laid out, which is to implement it as a community-centric system.

Contents

List of Figures

List of Tables

List of Abbreviations

ASP Answer set programming

BMA Bacterial memetic algorithm

CSP Constraint satisfaction problem

FOL First order logic

IoT Internet of things

ISS Informationally structured space

LBP Local binary pattern

MDP Markov decision process

OWL Web ontology language

PDDL Planning domain definition language

QOL Quality of life

RDF Resource description framework

SOA Service oriented architecture

SWRL Semantic web rule language

W3C World wide web consortium

WCSP Weighted Constraint satisfaction problem

WSDL Web service definition language

XML Extensible markup language

Chapter 1

Introduction

"Smart home" is defined as a communication network with sensors and actuators that combines building automation components with other information sharing components in the private home, where building automation is related to the control and communication networks in buildings [39]. The smart home observes the inhabitants and provides proactive services that can deliver comfort, security and safety, energy saving and sustainability. It integrates the devices and information pertaining the home to provide services, which are activities to fulfill desires of recipients. In this thesis, the notion of "service" is used to abstract away the specifics of software and hardware implementation through the engagement of standardizations and well-defined functional descriptions.

1.1 Current State of Smart Home

The Internet of Things (IoT) is the next step in the evolution of wireless networks, Big data, and connected devices. Sensors shrink in size and start to transition from our smartphones to other everyday objects that can broadcast themselves in the web [45]. At the beginning, the goal of IoT was to attach RFID tags on objects such that a virtual representation can be made that relates to the physical entity. The novelty of IoT is to connect objects or things to Internet, hence connecting the physical world to the virtual world.

Market analysis predicts the IoT will double in size to nearly 50 billion things by 2020, comprising a $1.7 trillion market [1]. Their application extends from power plant, factories, and jet engines performance monitoring to vital signs collection from bracelets and watches for medical and health purposes. In each of these cases, the IoT is both saving lives and transforming industries and societies.

[1] See http://www.idc.com/getdoc.jsp?containerId=prUS25658015

One of the greatest extensions is the realization of smart home. Smart homes typically evoke visions of robot maid or refrigerators ordering milk from Amazon or home that tends to your entertainment needs, but they also offer possibilities for energy and cost savings, greater home efficiency through automation, as well as improved home security. Smart homes use communication and networking technology to integrate the various common devices and appliances found in almost all homes, plus building environment systems more common in factories and offices, so that an entire home can be controlled centrally and/or remotely as a single machine. Smart homes have the potential to provide for consumers' growing expectations of convenience, sustainable living, safety, and security. Besides, it can also be an entertainment hub, communication center, and an extension as smart assistant that helps us in our daily living [66]. From the Gartner Hyper Cycle for emerging technologies shown in Figure 1.1, Connected homes and IoT Platforms are on the rise in expectation. Consumers are expecting an integration of IoT devices through a suitable platform in order to realize a smart and connected home. It is reported that 77% of people believe smart homes will be as common in 2025 as smartphones [2].

Major technology companies waste no time in diving into the world of smart home and smart devices to provide a self-sustaining ecosystem of intelligent control and sensing. Apple has introduced the HomeKit, a framework for communicating with and controlling connected accessories in the user's home [3]. Google Nest develops various smart products like the Nest Cam and smart thermostat that can also act as a hub for various other devices [4]. Samsung SmartThings enables monitoring and control of the home securely [5]. Amazon delivers Echo, a smart hub with smart agent control that controls everything from home entertainment to light control. It is also compatible to other smart devices from WeMo, Philips Hue, Samsung SmartThings, Wink, Insteon, Nest, and Ecobee [6]. Huawei gets involve by deploying its cloud-based platform for Telefónica to develop an end to end smart home solution [7].

Besides devices and physical hubs, certain useful general applications and services for smart home are created that can equally be useful, such as Yonomi, which prides itself on being one of the few home automation services that doesn't require the purchase of a central "hub" or physical syncing device [8]. IFTTT (Short for "If This Then That) is a unique,

[2]See https://newsroom.intel.com/news-releases/intel-securitys-international-internet-of-things-smart-home-survey/

[3]See https://developer.apple.com/homekit/

[4]See https://nest.com/

[5]See https://www.smartthings.com/

[6]See http://www.amazon.com/Amazon-SK705DI-Echo/dp/B00X4WHP5E

[7]See http://www.huawei.com/en/news/2016/2/Smart-Home-category-in-Latin-America

[8]See http://yonomi.co/

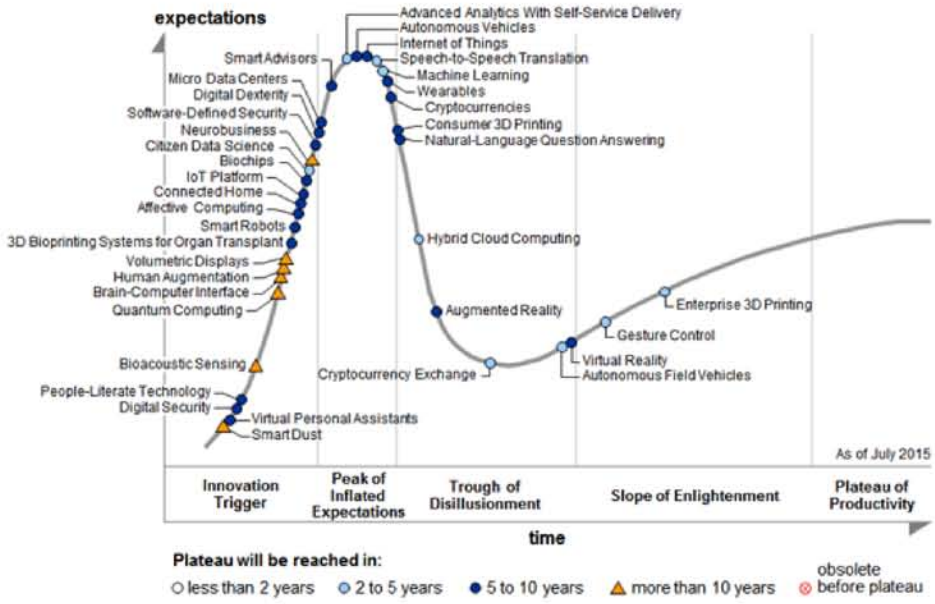


Figure 1.1: *Gartner Hype Cycle for emerging technologies August 2015 (adopted from http://www.gartner.com/newsroom/id/3114217)*

customizable service that lets users automate tasks within their homes via allowing users to create "if, then" concoction of rules based on their compatible devices [9].

In the field of research, capabilities of the smart home are continually being extended. Extensive research and development have been done on various applications and aspects of the smart home, such as general building automation [57, 111], home intelligence [17, 16], automation and planning [70, 71], weather controlled home [122], home for the elderly [1, 99], monitoring [87, 68, 28], communication technologies [94], energy efficiency [152, 42, 32], interoperation [72] and simulation tools [100, 47].

1.2 Human-centric Approach

As Mark Weiser, the pioneer in ubiquitous computing, puts it: technologies should fit the human environment, instead of humans having to adapt to technologies [140]. [82] has argued that smart homes are developed mostly from the viewpoint of technical capabilities, which is considered ego-centric. Gadgets are running well ahead of consumer desires, and thus, they haven't yet found reasons to buy [10]. If support or connection is lost or any outages are encountered, most gadgets cannot stopped gracefully as exhibited by the Nest smart

[9] See https://ifttt.com/categories/connect-your-home

[10] See http://www.wsj.com/articles/smart-home-gadgets-still-a-hard-sell-1451970061

(a) *Ego-centric* **(b)** *Human-centric*

Figure 1.2: *Ego-centric VS Human-centric*

Thermostat [11]. High profile cases like Quirky sell of bid for Wink home automation hub from Flextronics [12] and the discontinuation of Resolv's smart home hub from Nest [13] exhibit such problems. Consumers are left hanging and uncertain of the gadgets they purchased. Besides, most devices can only provide simple services independently. Thus, cooperation among the devices is important.

Leitner proposes to shift the smart home paradigm from technological capabilities to that which centered around human's need to enhance their living experience, which is called human-centric [82]. Figure 1.2 illustrates the difference between ego-centric and human-centric approach in the adoption of technology. But it is by no means stated that technical capabilities are unimportant. In fact, they are extremely crucial in realizing the smart home with their technological capabilities. The emphasis is that smart home should function in a way that tries to satisfy personal goals based on devices connected to it, instead of humans having to decide how the devices are going to serve them. Besides, most devices can only provide simple services independently. Thus, cooperation among the devices is important. That said, every person has a set of goals or constraints that they hope the smart home can fulfill. The smart home's role is perform binding and control of devices to fulfill such goals cooperatively.

The smart home devices can be used individually, or can be coordinated through complex plans[14] to achieve goals. For example, garden lights, security camera, video streaming

[11]See http://www.cbc.ca/news/technology/nest-smart-home-problems-1.3410143
[12]See http://fortune.com/2015/09/22/quirky-files-bankruptcy/
[13]See http://www.wired.com/2016/04/nests-hub-shutdown-proves-youre-crazy-buy-internet-things/
[14]A plan is a sequence of actions to be performed by the devices

service and television (where all these individual devices can be used individually) can be coordinated given an alarm is triggered such that video from the camera be sent to the TV that is turned on, with the lights turned on as well, to form a monitoring system. Therefore, smart home should possess the capability to perform late binding[15] of devices due to them being loosely coupled [16] without too much manual effort from the user. It must also support intelligence for device configurations, making choices and generating complex plans. "Intelligence" refers to the capability to make inference (or reasoning) of logical consequences from a set of facts, where, in the context of this thesis, are axioms pertaining the home layout, device functions and classes, and ontologies that provide the necessary schema for inference.

Given that most goals are over-constrained, smart home should try to maximize the fulfillment of personal goals that are often conflicting. For example, it should try to fulfill as much as possible the requests made by both the mother and daughter who wants to watch TV but both having different channel preferences.

Abundant researches had been done (refer to literature review in Chapter 2) on device binding, service composition, automated planning and knowledge representation for the smart home, but from current literatures, there is no known works on device binding and implementation of the smart home that maximizes fulfillment of personal goals.

Given that, the concept of design is "Personalized service provision of loosely coupled smart home devices with optimization and semantic reasoning capability". Therefore, human-centric semantic reasoning and optimization for smart home is proposed. Semantic reasoning deals with device binding, while optimization deals with the maximization of personal goals fulfillment. Human-centric computing is a new technology paradigm where computing resources are provided to humans anywhere and anytime based on their situation [67]. The work in this thesis devises a system to deliver human-centric computing with the available devices and service providers around for the smart home. With this system, human-centricity applies to the binding of devices and implementation to maximize the fulfillment of personal goals of every user within the vicinity of smart control. As this system is meant to be augmented into a home to realize a smart home, in this thesis, the terms human-centric system and human-centric smart home will be used interchangeably.

In the context of human-centricity, quality of life (QOL) is a set of personalized objective goals that needs to be fulfilled. Every person has their own QOLs, which are constraints that should be fulfilled for them, such as that they prefer dim lights at night when they just wake up, the duration between meal times must not exceed certain amount of time, sleep-

[15] association between different devices during run-time

[16] devices that have little to no knowledge of the definition of others

ing duration should not exceed certain amount of time, medication schedule etc. These constraints are objective and pragmatic such that they can be explained via logic or mathematical formulation. This work does not deal with subjective goals like trying to improve a person's mood (unless there is an objective indicator and pragmatic executions to handle such problems, which, as of now, is not possible). For unidentified people or people without any records, a default QOL can be assigned to them. Therefore, two main parts of work are required, which are:
1) Real-time open set face identification that is invariant to transformation for determining the identity in order to select their QOL for the smart home
2) Service provision by the smart home with reasoning capabilities that revolves around the human it is responsible for to maximize the QOLs

The first part deals with identifying and selecting appropriate QOLs and data to be run by, the second part, which tries to fulfill the QOLs. The second part requires automation to bind and coordinate devices, as well as intelligence to provide necessary information crucial for automation.

Devices abiding to the computing paradigm of human-centricity do not need information of what other devices are doing or their capabilities, except their own. The devices are connected to the central controller, which reasons and controls them according to the functionalities and semantics of the devices (which means the devices need to have enough information about themselves in terms of their functions and meaning to feed to the central controller). It is up to the central controller to bind the devices together depending on situation. In this respect, the devices are bound to enable communication of information, and their operations are coordinated to deliver services cooperatively via a sequence of device actions called a plan.

The aim of this research is to establish an integrated system for the human-centric smart home (HcSH) that provides personalized service through loosely coupled devices automatically. It provides a platform to implement human-centricity through modularizing the overall system into three modules, which are human identification (HIM), automated planner (APM), and semantic reasoner (SRM). Figure 1.3 shows the discrete methodologies used to achieve human-centric smart home. HIM helps select the appropriate QOL, SRM binds the devices by associating them with planning components, which are then used by APM to generate plans for device coordination to maximize QOL fulfillment.

HIM identifies (and possibly tracks) people who are within the vicinity of control such that their personalized data can be extracted, like their personal information and QOL.

With the devices connected to the smart home and the house layout, SRM will associate device nodes and wrapped up their atomic services. These are performed to generate

planning operators for APM, as well as generating rules for decision making during planning. Apart from inherent ontology, SRM obtains information from structural information (information regarding home layout and adjacency), service information (functional information and classes of the connected devices), and state information (variable instantiations from devices, humans and home). Personal information and QOLs (selected by HIM) are also used during the generation of planning operators.

APM performs automated planning that coordinates the devices to maximize all QOLs. Planning is done using planning operators generated from SRM. Therefore, state information is fed into this module as initial condition. The output of APM is the manipulation of state information. Given this model, QOLs are a set of conditions for the state information. APM tries to manipulate these state information to maximize the QOL fulfillment by coordinating the available devices that are connected to it.

The definition of "Human-centricity" used in this thesis is the emphasis around human's need to enhance their living experience, where computing resources are provided to them anywhere and anytime depending on situations. Various smart homes that fit this definition are built, but problems faced are the need for manual configuration and procedural programming to obtain the services that are specific. That said, this thesis refers to these as smart homes providing human-centric services. There is a subtle difference between this aspect of smart home and this thesis's work. The thesis's emphasis is on building automation of the smart home that can be automatically configured, depending on the devices connected to the smart home. Contrary to the former, this thesis is on smart home that provides human-centric automation (through reasoning and optimization), where the definition of human-centricity points to the fact that the automation will maximize personal goals of human (thus, enhance their living experiences) given the devices available around the home (thus, services provided anywhere and anytime).

To avoid confusion between this work and context-aware computing, smart home can be divided into two parts, 1) sensing and classification part, and 2) knowledge representation and automation. Context-aware computing refers to a system that can sense the environment and adapt. Sensing and classification of new information is the key. Therefore, it lies in the first part. On the other hand, this work is on the second part, where it uses the sensed and classified information from part 1 to perform control and coordination. Information from part 1 provides the objective indicators as mentioned previously.

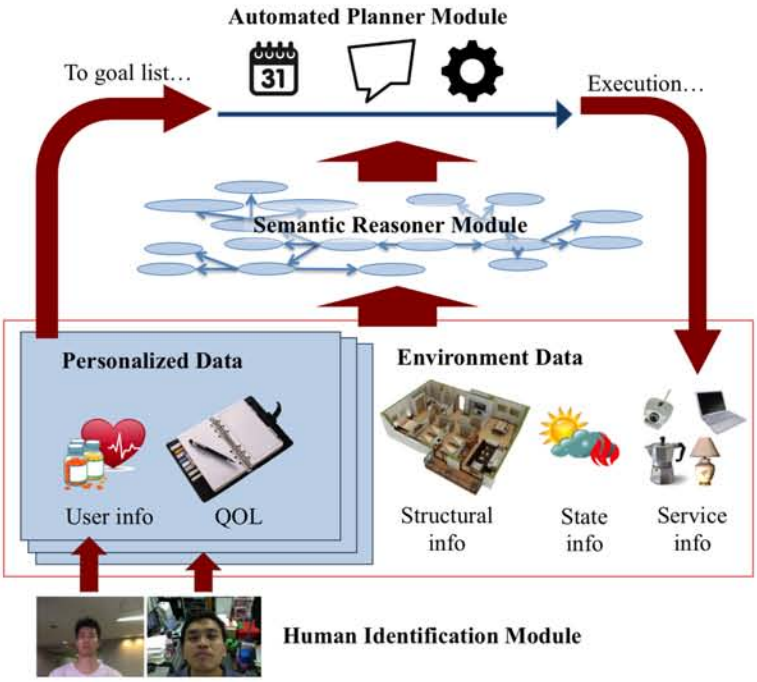


Figure 1.3: *Gist of methodologies for human-centric smart home*

1.3 Problems and Objectives

The proposed system delivers personalized service provision through automation and intelligence, which reduces manual effort from human. Due to personalization and automation, a number of problems have to be solved.

To reduce manual effort is to reduce the need to configure the devices. The devices need to be loosely-coupled, and are only bound during run-time when necessary.

Most devices can only provide simple services each. To extend their usefulness, multiple devices have to work together and also sequentially. Such tasks require more complicated plans, which assemble a sequence of activities to coordinate the devices.

A prominent challenge is due to over-constrained specification. QOLs, which are a set of personalized constraints, can often be over-constrained or contradictory due to imperfect specification by the user, or due to multiple people having contradictory constraints.

Finally, identification that is impervious to variances in order to extract QOLs need to be open set, and that it can work under low training data, is required. This is because homes are unique, and the intention of this system is to be able to be implemented over a wide range of homes.

Given the challenges, the objective is to devise a declarative approach (contrary to procedural approach that requires much manual effort overhead) capable of generating com-

plex plans after automatic device binding, where the generated plans will provide the means to coordinate the devices. Optimization will be used during planning to deal with issue of over-constraint. Open set face identification is developed, which utilizes transfer learning to handle problem of low training data. Methods used are supported by literature reviews as explained in Chapter 2.

More detailed objectives are below:

Open-set face identification:

- Development of real time open set face identification with transfer learning
- Design of prototype data compression method for efficient storage and processing

Service provision with reasoning capabilities:

- Establishment of a declarative approach in implementing service oriented architecture (SOA) to achieve late binding of devices
- Implementation of automated planning and optimization via constraint satisfaction
- Establishment of ontology for knowledge representation that supports knowledge sharing and device binding
- Implementation of personalized service provision through QOL maximization

Semantic reasoning is needed for device binding, such that their services can be coordinated to maximize (or optimize) the fulfillment of personalized goals in the smart home, and thus, the title "Human-centric semantic reasoning and optimization for smart home".

1.4 Thesis Scope and Organization

This work is on service provision that tries to fulfill personalized QOLs. This is realized through the devices that are connected to the system. Therefore, this work emphasizes on the planning to coordinate devices, instead of acting, which deals with the low level implementation. Distinction between planning and acting can be found in [53]. Although not directly on acting, this work follows certain standardization in device specification that allows abstraction, and a link between the planning and acting part. That said, middlewares and communication protocols will not be emphasized.

This work is also intended to be general. The intention is for it to be used by developers in realizing a human-centric system. Therefore, specific service type will be used only for

case studies to verify the proposed system. Hence, test is done on the optimality and time consumption of the system, which must be scalable.

Besides that, the proposed system requires constraints that depend on objective indicators, and plausible pragmatic executions to fulfill them. Therefore, states involving subjectivity like sentiments should be encoded in a way such that the encoding objectively reflects the intended meaning relative to task (which is decided by the designer or prespecified standards). Encodings can be obtained from direct measurements (if there is any) or indirect inferences (which may involve machine learning on multiple sensor inputs or the construction of reasoning structures like the probabilistic graphical models). In short, a means of measurement is needed to objectify factors that we need to use as indicators.

As stated, the theme is the personalization of service provision of loosely coupled smart home devices. This work requires face identification in order to select the QOL of the identified person to optimize. Therefore, real-time open set face identification that is able to run on low powered processors is built such that they can be accommodated for various smart devices as shown in Chapter 3. Transfer learning and incremental learning (implemented under spatio-temporal association) to assist face identification are also developed to handle insufficient face prototype. At the same time, the algorithm requires a special reformulation of problem that can help in storage and processing.

For the second part of objectives, which is service provision with reasoning capabilities, constraint processing in automated planning is devised to support loose binding of devices to fulfill goals, which is in accordance to SOA architecture as presented in Chapter 4. Extension to optimization is explained as well. Knowledge representation and ontology construction for semantic inference of the smart home and variable binding are explained in Chapter 5. All these fulfill the objectives under Service provision with reasoning capabilities. This work is intended to be general. Thus, no particular information structure is assumed. It acts as a framework to support various applications via different design of services and ontologies. But for experiment purposes, this thesis adapts the system to Informationally Structured Space (ISS) (explained in Section 2.5), which provides the information structure for the smart home.

Figure 1.4 shows the organization of the thesis. The description is as follows:

Chapter 2 provides the related works for this thesis relevant to the research done that supports the objectives given, which involves face recognition, building automation system and knowledge representation for the smart home. It also gives a brief description of ISS, which provides the design motivation behind the system as well as the architecture and modules that make up the whole functioning structure, and how ISS is going to relate to this work to be interfaced with personalized service provision.

HIM is explained in Chapter 3. This chapter starts off by presenting the general framework to give a rough idea of the whole recognition system. It then details the image preprocessing required to obtain appropriate image representation. Detailed mathematical formulation of the problem is presented in Section 3.3. Analysis and modifications for transfer learning is detailed in Section 3.4, followed by algorithmic implementation of recognition and data mergence in Section 3.5. For real-time incremental learning, Section 3.6 explains the spatio-temporal association method to achieve that. Finally experiments to verify the validity of the face identification system for open set problem and real-time problem, as well as computational evaluation, are shown in Section 3.7.3.1.

Chapter 4 explains APM that uses constraint satisfaction approach in realizing automated planning for the smart home, which coordinates devices and can support complex sequence of plans. Planner module that supports dynamic re-planning is shown in Section 4.1. Domain description and planning as constraint satisfaction problem is given in Section 4.2 and 4.3 respectively. Enhancements for planning is then shown in Section 4.4. This is followed by case studies of planning using hard constraints in Section 6.2.3. Due to hard constraints not being able to perform optimization, in Section 4.6.1, constraints are equipped with weights to support plan optimization. Experiments are also shown to validate the method.

Chapter 5 explains the SRM, which gives the knowledge representation method for smart home intelligence and variable binding for automated planning. Section 5.1 provides description of building ontology construction as well as case studies that utilizes it for further inference on the state of the building. This is followed by Section 5.2, which details the method to generate rules for automated planning from semantic associations of connected devices. Finally, an extension to robot planning with ontology support is provided in Section 5.3.

Chapter 6 describes the implementation of human-centric system in a prototype smart home to demonstrate the applicability of the approach.

Finally, Chapter 7 summarizes the thesis, where future work is also laid out.

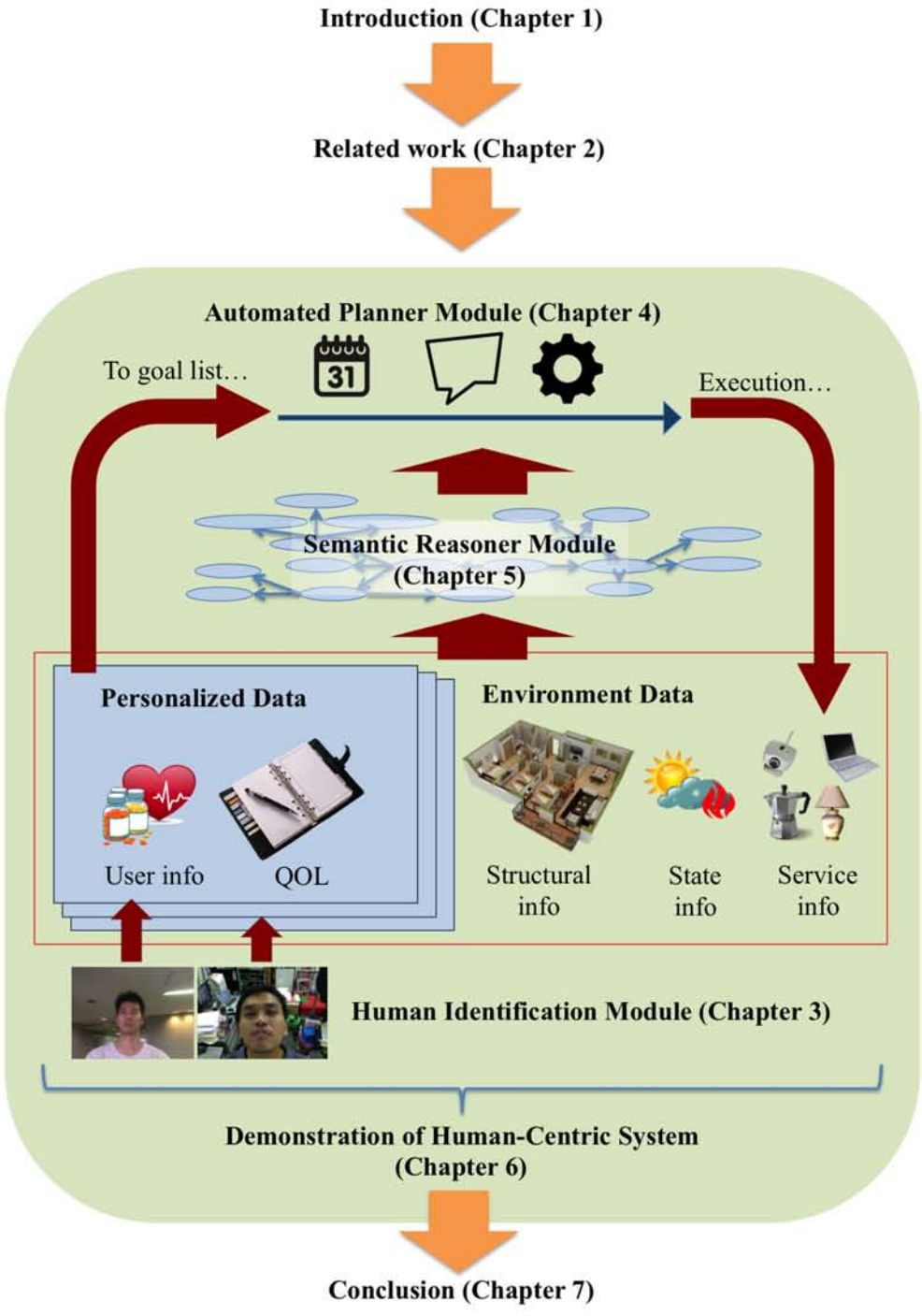


Figure 1.4: *Thesis organization*

Chapter 2

Related Works

2.1 Building Automation System

Due to high overhead cost to establish a smart home that caters for individual use, building automation system (BAS), which is a much cheaper choice, is preferred [15]. Besides, current smart homes still remains mostly in experimental stage. Therefore, it is more practical to extend the current domotic system. With various vendors providing devices with their own functional goals, to maintain interoperability of devices connected to the BAS, various middleware support are also available, such as OSGI, KNX, UPnP, X10, OpenWebNet and DPWS. Various home automation systems have been developed over time as subsequently described.

Home automation for inferring environment state is developed in [17]. It uses DogOnt ontology [16] to determine environment states like whether a certain room is smoke free and mosquito free depending on which rooms are adjacent and whether certain doors or windows are open or not. Although it does not include service composition, since it only deals with sensory information, nevertheless, it provides insight in generating rules as goals for service composition to work on.

In [111], enhancements are made to the KNX standard in order to support knowledge-based and context-aware functionalities in home automation. The work uses knowledge representation and automated reasoning to create a self-adapting framework for managing user profiles and device services. Rule matching need not be exact, where it allows potential or intersection matches, which is the case in most practical situations. Interestingly, the system also supports partial or disjoint matches, where requests and supplied resources have conflicting features and when no other matches are possible. Service composition is performed through constant running of Concept Abduction algorithm [38] and selecting

services to cover missing features. But such service composition will not be applicable if service sequence is crucial. Situation is made worse under uncertainty. A more flexible system is developed in [86], where the case study is to maximize human comfort and energy efficiency.

Building automation adopting SOA and device profile for web service (DPWS) is proposed in [57]. Context information is obtained for processing before being used to guide service composition, complemented by policy rules and composition plans. The system also includes reasoning capability through building ontology. To handle dynamic changes during service composition, it uses service composition plans to describe users' requirements by Composition Plan Description Language (CPDL). Similar to [111], service composition does not support construction of complex sequence of services, such as the case of robot query on user's preferred activity before preparing the environment for such activity.

Kaldeli et. al. [70, 71] developed a home automation service composition based on SOA that is capable of complex composition of a sequence of services using constraint processing. Unlike previous methods, the system does not require manual construction of subplans. It handles context awareness and is able to deal with uncertain situation by dynamic re-planning. By using constraint processing, it can also handle variables with large domain efficiently. But the work does not consider intelligence or make use of ontology and knowledge base for reasoning. Besides, although sub-plans need not be manually made, this comes at a cost where the logical rules have to be painstakingly specified as propositional logic.

To enable complex reasoning, semantic information pertaining the domotic environment can be organized by knowledge represented through a suitable ontology schema. [71] and [57] argued that the main focus of previous platforms of pervasive applications is on device interoperation and spontaneous networking, which do not consider complex and intelligent functionalities involving higher levels of information. Dynamic composition of complex sequence of services under uncertainty is necessary. Currently, to enable higher functionalities, tedious programming is required from the user side. Besides, certain applications require manipulation and reading of values, such as counting. [71] uses artificial intelligence method, namely constraint processing, to handle such service composition task where variables of a wider domain is required in the planning process. It offers a declarative approach for planning where services are loosely bound. The disadvantage is that there is no intelligence and reasoning involved. It performs service composition through solving Constraint Satisfaction Problem (CSP), and implements them by appropriate devices. Its implementation requires logical rules that need to be manually defined.

2.2 Web Technologies for Home Automation

Recent progress in information processing and communication technology has made possible the creation of smart home, smart environments and office buildings that is able to increase comfort, security and efficiency. Ambient intelligence played a big part, which has the aim of surrounding building inhabitants with intelligence and unobstrusive sensors and actuators to acquire knowledge, adapt and dispense favorable services [114]. Mobile devices are utilized and objects (including non-electronic objects) are assigned with RFID tags that carry their information essential for the construction of knowledge structure and inference.

This is further enhanced by the progress of web technologies and the introduction of the semantic web, where various tools for knowledge representation, semantic annotations and querying are developed and standardized by the World Wide Web Consortium (W3C). This progress also brings forward the Internet of Things (IoT), where everyday objects are equipped with embedded data and communications capabilities. With this, IoT realizes the concept of pervasive and ubiquitous computing that is crucial for ambient intelligence.

Web service aims to change the web from a database of static documents to an electronic platform for conducting business. The widespread adoption of web services is due to its simplicity and data interoperability provided by various technology like Extensible markup language (XML), Simple object access protocol (SOAP) and Web services description language (WSDL). The issue is that data in the web is represented as just mere data for the sake of presentation. Associations between them for semantic interoperability and reusability are difficult. Semantic web is introduced by Tim Berners-Lee, which is an extension of the current web where web information are endowed with semantics such that they can be operated and processed by machines [10, 115]. Its initiative addresses the problem of XML's lack of semantics by developing a set of XML based languages, such as Resource description framework (RDF) and Web ontology language (OWL).

RDF is used to model information found on the web in a structured manner to manage distributed data [138]. Unlike relational database, it models data as graphs [4]. All the semantic web standards are built on this foundation of distributed data. RDF does this in the form of triples, which consists of three parts: a subject, a predicate and an object. Using this form, a model of almost any information can be represented by combining triples.

Substantial amount on research has been done that utilized Semantic Web Technologies to build intelligent applications in various domains [64, 90, 108]. They have been applied to domotic applications [15, 111], agent communication [103] and emergency management [116] that exploits the knowledge structure built up via tools such as the ontology markup language (DAML, OWL), rule mark up language (SWRL), querying tool SPARQL and RDF

schema, all of which are endorsed by W3C.

Description logics (DLs) are a family of knowledge representation languages used for ontological modeling [75]. DLs are equipped with formal semantics that allows the exchange of ontologies without ambiguity as to their meaning. They also enable inference to be made given the axioms stated in the ontology. The advantage of using description logic is that they are always guaranteed to terminate during inference, unlike using pure First Order Logic (FOL). There are different kinds of DLs. We choose OWL Web Ontology Language to model the building ontology due to its compatibility with the semantic web endorsed by W3C. Besides, it subsumes the capabilities of RDF schema and RDFS-Plus.

2.3 Automated Planning for Home Automation

Researches related to web services and semantic web provide concepts and technologies that can support service discovery, the design and implementation of services, as well as providing machine understandable semantics for reasoning. Service composition is initially applied to create composite services for various complex business requirements like travel booking applications. Overtime, the idea is adopted by ubiquitous and pervasive computing due to its ability to discover services and enable loose binding of services. In order to support service discovery and composition, web services are endowed with semantic markups that describe their functionalities, properties and conditions (pre and post conditions of the service). With the semantic markup, it is much more convenient to discover, plan, reason and monitor services, instead of relying on detailed descriptions of the functions. Advancements in the web of things and web service technologies can shed light on service composition for home automation. To enable complex reasoning, semantic information pertaining the domotic environment can be organized by knowledge represented through a suitable ontology schema.

Web services are technologies that can support the design and implementation of distributed system. The prominence of the web of things and advancement in web services shed light into service composition for planning in home automation, as they concern the interoperation of different objects. There are a lot of similarities between both of them, that is, high heterogeneity of objects, dynamic environment due to objects connection and disconnection from the system or due to contextual changes, and the need for integration and coordination of different objects to deliver complex services. Service oriented architecture (SOA) [104] complemented by automatic service composition provides the crucial approach in delivering what is required of home automation. SOA consists of independent but interoperable services that are loosely coupled, where each service only exposes its

functionality to others, keeping the details of implementation hidden. Besides, user can use the composition capabilities of SOA without having to deal with details and technicalities [84]. As argued by [71], previous platforms of pervasive applications focus only on device interoperation and spontaneous networking, without considering complex and intelligent functionalities involving higher levels of information. Thus, tools and design concepts of web services can be applied for intelligent home automation. Currently, W3C standard of service description through Web service description language (WSDL) and message format between services through Simple object access protocol (SOAP) is regarded as the most popular implementation of SOA.

A large amount of work has been done on dealing with web services from service discovery to policy checking. Here, we will concentrate on service composition. More detailed review of web service composition can be obtained from [118]. Initially, service composition is applied to create composite services for various complex business requirements. Overtime, the idea is adopted by ubiquitous and pervasive computing. In order to support service discovery and composition, web services come with semantic markups that describe their functionalities, properties and conditions (pre and post conditions of the service). With the semantic markup, it is much more convenient to discover, plan, reason and monitor services, instead of relying on detailed descriptions of the functions. It endows them with capabilities of being a planning operator. Several well-known workflow languages are developed for semantic descriptions, such as WSDL-S, OWL-S and WSMO.

With the advancements made in web services and AI planning, these works are extended to various real-life applications. Service composition has been applied to work flow planning for disaster management. To handle hydrogeological disaster, in [11] a tool was developed to generate abstract plan and translate it into an executable process language via WS-BPEL that takes environmental changes into account. A system to handle fire emergencies with context awareness is developed by [41]. Virtual objects, which are derived from physical objects ranging from security guards to specific sensors to location entities, are managed through semantic ontology model. With the concept of web of things, semantic ontology model is established to enable information reusability, extensibility and interoperability. Decisions and plans during a fire incident are obtained from the assistance of log repository of previous similar circumstances. In [57], SOA is applied to building automation, where context information is collected and processed in order to compose plans for device/service coordination. Web connected sensors for home surveillance is implemented in [62]. Web service composition also gains prominence in the field of engineering and industrial automation [41, 106] and smart grid application [49] as well as energy saving application [51].

In [74], OWLS-Xplan uses the semantic descriptions of atomic web services defined in OWL-S for planning purposes. The XPlan planning module will then generate the composite services. In the work, an XML dialect of planning domain definition language (PDDL) is developed, thus, the system is PDDL compliant. Although the system obtains semantic descriptions in OWL-S, it is not utilized and semantic awareness is not achieved. In this case, the planning module is required to perform exact matching for service inputs and outputs.

In [58, 59] a framework was developed that converts web service composition tasks into planning problems expressed in PDDL. The framework will then convert the devised plan into an OWL-S composite process description. The framework translates atomic OWL-S processes to planning operators, from which it derives the set of actions to achieve goals. When no exact composite services can be found, semantic information is utilized to obtain composite services that best approximate the goal.

To deal with complex tasks and to reduce planning complexity, hierarchical task network (HTN) [50] is introduced. It uses method definitions in its planning domain description, which specifies how the complex tasks can be broken down into more manageable tasks [121, 6]. The planning problem can then be specified as a list of tasks to perform. The planner will then solve the problem by applying the breaking down of tasks to every task in the task list. This process continues until the tasks are reduced to their atomic planning operator constituents that corresponds to a solution plan. The main disadvantage of this approach lies in the fact that the planning process requires that certain decomposition rules be specified due to its hierarchical nature. This means that it needs to be encoded in advance by an expert.

Other popular planning solutions for service composition are based on the Golog language. Golog is a logical programming language and has been further developed in [93] to support customized constraints and non-determinism in sequential executions. Through PDDL, it has been used in order to support service composition. Modeling through Situation calculus, the encoding and translation process for this approach is relatively complex.

Answer set programming (ASP) is another popular approach used for planning [146, 149]. It is a declarative language that is suitable for knowledge representation and non-monotonic reasoning. [146] integrated ASP with cost learning to improve the performance of the planner for robot planning, while [149] combines with MDP to endow it with the capability to handle uncertainty. At the moment, for complex sequence of services, these methods require heavy computational load the long planning time.

The methods discussed thus far require exact match ups between inputs, outputs and variables, or that it assumes certain ontologies to handle heterogeneities, or requires specifications of user anticipation and procedural templates. Domain and goal modeling through

Constraint satisfaction problem (CSP) is developed to create a language that allows users to express goals without having to know about the details and interdependencies between services [71, 70, 69]. Its domain representation is of similar concept with the Multi-valued Planning Task (MPT) encoding [61]. Besides, another advantage is that it is able to handle variables with large domain efficiently, which is quite prevalent in the field of building automation such as temperature value and user location. Although the CSP planner might be slower than some state of the art methods [109, 6, 63], it can support complex goals and can handle variables with large domain efficiently.

2.4 Face Recognition

The presence of domestic robots is quite common nowadays, which ranges from household robots to edutainment robots. However, one type of robot that is also rising to prominence is the robot partner that provides security, assistance and companionship to human occupants. They provide temporary companionship and also a focus point for other services like tele-presence, news announcement and entertainment.

To ensure favorable interaction with robot partner's human occupants, face recognition is one of the fundamental requirements as interaction mode and style differ between individuals. Apart from just identification or verification, it also needs to discern whether a face is considered familiar or not, which is termed as open universe scenario recognition. The HIM will be on open universe real-time face recognition. Given unconstrained and uncooperative condition during recognition, it also needs to be able to run on low powered processors, which is the case for most robots' onboard processors. Designing a good robot partner means seriously taking the tradeoff between accuracy and computational load into account.

Three issues we are addressing are different modality, recognition under uncontrolled condition and low computational load.

One can think of different modality as capturing images under different hardware like Kinect, webcam and security cameras, which imposes different scale and resolution. Recognition under different modality is limited to comparing normal photo images, and does not involve heterogeneous recognition [73]. Robot partners need to recognize faces under unconstrained and uncooperative situation [34]. It includes recognizing faces under different poses, expression and illumination. An efficient and fast method needs to be devised for the task without imposing constraints to its human occupants as this will hamper robot-human interaction.

As low powered processors normally run robot partners, face recognition process can-

not be too heavy. For this work, we will use the iPhone 5s as a case study to function as a robot, which is from our previous work that is called the iPhonoid [23]. The motivation behind the iPhonoid is that robot partners should be able to accompany their owners, helping and collecting information during the course of everyday life, while being able to dispense services depending on the devices that are attached to it. Smartphone is a perfect choice as implementation medium for such robots due to their wide adoption by general public and interfacing capabilities with various devices from standardization. Face recognition system devised needs to be able to fit into the robot, among other modules such as motion control, speech recognition and generation, interface and gesture recognition (The other modules will not be covered). Although we use iPhone 5s for robot execution, it is by no means the best and we believe this work (face recognition with transfer learning) can traverse to other low-powered processors as well.

Given the three issues, a system built based on the reformulated joint probabilistic face matching method to support transfer learning is devised [30, 129]. Besides, online learning based on spatio-temporal association is used to support initial low number of samples.

As robot partners are expected to function in uncontrolled situation, their recognition system must be invariant to changes due to pose, illumination, and expression difference. Despite the requirement for such significant level of invariance, it needs to be discriminative enough to distinguish between individuals. Moreover, as robot partners are expected to come across people whom they have never met before, they need to be able to tell whether a person is familiar or not, and not just the nearest expected identity. This is termed as an open universe identification problem [83, 8, 102]. Recent progress in recognition for open universe scenario has been rapid both in verification [65, 30, 9, 148] and identification [102]. Given the scope of robot partners, where population size is not too massive, identification can be performed through the scores of independent verification [133, 55]. HIM will concentrate on face recognition with 2D image as input. For 3D face recognition, interested readers can refer to works by [135, 136, 25, 24].

Current face recognition algorithms can reach near human level accuracy, or even seemed to surpass it [125, 89, 151]. This includes 3D morphable faces [14], sparse representation method [144, 102] and deep networks [125].

Multi-layered networks and deep convolution networks have received significant attention recently in the field of machine learning and pattern recognition. They consist of multiple layers of networks that are greedily learned through each layer. Data abstraction increases with level of layers. In terms of face recognition, the higher level layers retain the identities of a person, which is invariant to distortions from illumination and pose. Deep architecture is currently applied in the state-of-the-art system in pair-wise comparison test

for the challenging (Labeled Faces in the Wild) LFW database [65, 113], which achieves near human level face recognition performance. Deep learning method achieves success through obtaining the appropriate representation that is abstract enough to be invariant to intra-class variations, but retains its discriminative power between inter classes. They can also be easily incorporated into other algorithms like the joint probabilistic verification [30] by performing the role of representation extraction [123]. But a drawback is that they are extremely computationally intensive and require huge amount of training data to construct.

On the other side of the continuum for face recognition approach are the model-based methods. A notable example is the morphable face method, which can estimate illumination, pose, and texture parameters of the face through the use of 3D face model [14]. Given the 3D face model of a person (estimated from a 2D image given a trained 3D face prior), various variations such as lightning, pose, expression and even weight difference can be simulated to compare with input face image for recognition purposes. It also achieves state of the art performance for Labeled Faces in the Wild test [150]. However, the computational load is very high, and is reported to take several minutes per face. The use of 2D models, complemented by prior information to compensate for the missing dimension [29, 54], though faster, faces issues like self-occlusion. But recently, a fast variant of the 3D morphable face model has been developed [147]. It relies on transforming the probing filter instead of building a model of the face.

Although recognition system trained using large amount of data can produce good results [124], this is not practical for robot partners. Individual templates need to be learnt incrementally overtime, yet at the same time, they do not face over-fitting issue from the lack of samples. Besides, continual learning and recognition are to be performed in real-time. Therefore, computational load and template size should be considered during design to ensure practicality, especially when on board computers have a more limited capability. Support vector machine (SVM) based methods [85, 141] have been popular in identification. They can also be used to create signature sequence [9, 79] that can be used for verification. However, SVM method requires a long time to train for large-scale one-versus-many mode of recognition, and that ample number of samples per identity is required to ensure reasonable accuracy. Another popular method for identification is the sparse representation classification (SRC) methods [144, 139], which have been shown to perform well, even on the realistic open-universe set. The disadvantage is that the ℓ_1 minimization during test phase is complex under high dimensional dictionaries. Recent methods like Linearly Approximated Sparse Representation-based Classification (LASRC) [102] have managed to improve the convex optimization by combining ℓ_1 and ℓ_2 minimization such that it can handle

pair-matching tasks in real-time without affecting accuracy. Apart from issues regarding complexity, SRC needs ample samples to support the reconstruction.

A method that is increasingly gaining momentum is the probabilistic face method. Given a number of hypotheses regarding the input image and its connection with the prototype references, identification is achieved by selecting the hypothesis with highest likelihood [83]. Based on the foundation work of [96], which classifies based on the differences of inputs, [30] further improved the method as a joint probabilistic problem. The method tries to reduce the correlation between inter-class faces and increase for intra-class faces. Although simple in implementation, it is able to achieve high performance in LFW face verification test. Expectation-Maximization (EM) algorithm for the joint probabilistic method has been devised such that training is fast given the right implementation, as will be shown in Chapter 3. Result can be improved with the help of Fisher vector [120] or high dimensional representation [31], and can even be extended to deep learning [123]. Another major advantage comes from its transparency, which allows cross-domain transfer learning to be easily implemented as in the work of [27]. Therefore, joint probabilistic formulation for face recognition is used in this work to exploit its simplicity both in training and testing and also its transparency such that transfer learning can be easily carried out to counter the lack of sample images of the intended identities. Simplicity is crucial to reduce computational load, while transfer learning is required to prevent issues of sample scarcity. The work can also be extended to using Gaussian Process for extracting the face prior, which provides a richer description [88].

These properties are crucial for robot partner with limited computational power like the iPhonoid [23], one of the robot partners we developed that is powered by the iPhone to operate in real time. It cannot sustain processes involving high dimensional representation without sacrificing the fluidity of human-robot interaction. Representation under low dimension is not linearly separable due to inherent non-linearity of facial distortion from pose and illumination changes given different environment the robot is in. The problem can be alleviated using training samples from the same domain selectively obtained from a collection. Thus, transfer learning comes into play again to selectively merge the intended target domain to the current source domain. Different set of samples can be used as target domain depending on the situation, such as whether the robot is indoor or outdoor or which room it is in [43]. Besides, the samples should be compressed in such a way that the face recognition system does not require too much storage space, yet easy to manipulate given new input data. As mobile usage is expected to increase that requires constant processing based on models of Big Data, cloud network and processing will gain prominence [56]. Such samples will be crucial for online cloud based data sharing and processing among

the robots as well, which can increase the effectiveness of recognition.

2.5 Informationally Structured Space

2.5.1 Introduction

There is always a tradeoff between accuracy and computational load when prediction and inference are involved. One alternative over the tradeoff is to exploit surrounding information that it can acquire through external sensors and data loggers, complemented by storage mediums and processors via Informationally Structured Space (ISS), which provides an environment for information gathering, storage, processing and control [76]. Apart from providing contextual information, the knowledge can also be used as reinforcements for machine learning. Together with these peripheral devices and systems, they formed an ecosystem of information flow which is the fundamentals of ISS. Surrounding information can then be supplied to the robot and other devices implicitly [77], so that it can perform the appropriate task, instead of solely relying on its onboard sensors. Readers can refer to the following works [76] for more information related to ISS. ISS, complemented by robots, found many useful applications in daily life. Various applications had been developed and applied to various areas, such as in rehabilitation [131], business [119], search and rescue [78], companionship [23], household [101, 127], transportation [126], activity monitoring [21] and navigation and mapping [142]. Its sensors range from sound recorders to motion sensors to cameras in order to establish a more complete information foundation for data interpretation. It should be emphasized that this work does not deal explicitly with ISS, but dwell on the implication it brings to designing the smart home.

2.5.2 Design Motivation

The motivation of design is to support the ability for plug and play, ability to be implemented on multiple levels, ability to perform loose binding of devices, ability to automate services to achieve goals, abstraction and specialization of work.

Ability to plug and play means being able to communicate with a device the instance it is plugged into the system, where the system will also know the device's functionality. ISS will configure itself to accommodate for what the device has to offer when it is plugged in, or the function it loses when the device is disconnected. This implies that the ISS and the device have to speak a common language.

The ability to be implemented on multiple levels means that ISS should be able to run on

a single device or govern an assortment of devices of a building to a whole neighborhood. The motivation is that no matter the medium and its size, ISS should support the modules connected to it while remain the same in terms of its workings.

As it is uncertain which devices will be connected/disconnected from the system, inter-operations between them cannot be pre-programmed. Thus, ISS needs to be able to perform loose binding, which means the devices are bounded to achieve a task only during run-time.

Given the devices are bounded, the ISS also needs to be capable of generating plans and execute them by orchestrating the relevant devices. As the role of devices are closely related to the goals, therefore, this ability is closely related to the late binding of devices as the goal can act as a means for the devices to synchronize with each other to fulfill it.

With the complexity of the task that is required of ISS, support for abstraction is important to obscure and partition the level of complexity on which a layer interacts.

Abstraction works vertically by suppressing the more complex details below the current level, whereas specialization-of-works works parallely. It means that different devices or modules need not know what the others are capable of or even their presence in the ISS. All they need to provide is their own functionalities and the semantics that defined them.

ISS should be designed and function in a way that it tries to satisfy goals by calling on the objects connected to it, where human is treated the same as another object connected to the ISS. Goals are introduced either through human command (turning the TV on), event trigger (goals in an event of falling), schedule trigger (environmental checking) or introduced during implementation by other activities (sub-goals of a particular activity). In the context of human-centricity, quality of life (QOL) is a set of objective rules that the ISS needs to satisfy to ensure the wellbeing of its human inhabitants.

2.5.3 Architecture

Figure 2.1 shows the basic architecture of the ISS. ISS framework consists of 4 blocks, which are explained as follows:

Planning and implementation block – This block plans a sequence of service activities to be implemented to satisfy certain goals. Service activities come from pervasive block, where different devices attached to the block introduces their own services.

Information block – This block processes and interprets signals and data such that useful information can be extracted. It consists of specific processors, state tables, look up tables and storage unit.

Pervasive block – This block is an interface between the ISS and the external world. It introduces to the internal world (other blocks) the variables and services offered by the ex-

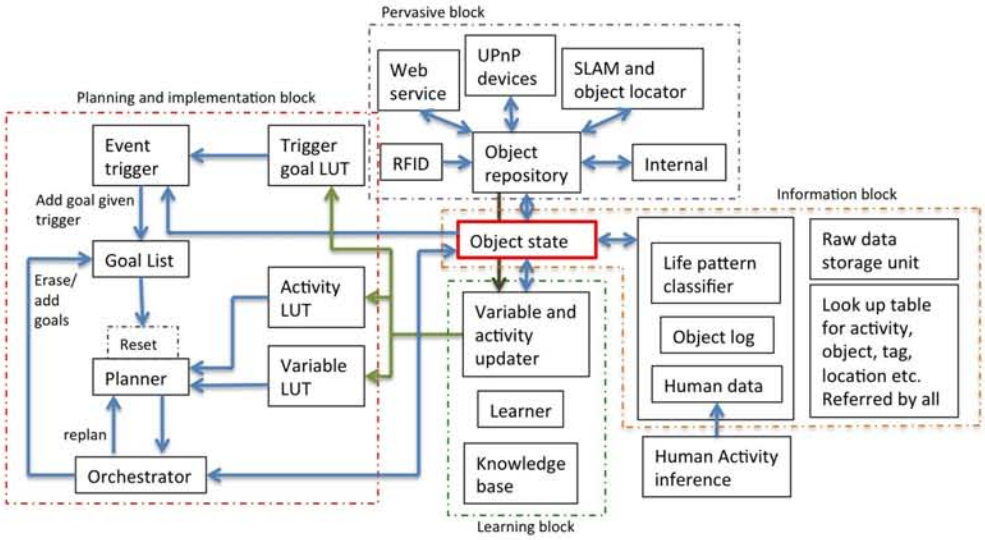


Figure 2.1: *ISS framework*

ternal world in an abstract manner and also perform service discovery and ranking. Details of services like protocols and connection procedure are not passed on to the ISS beyond the pervasive block. Objects and their services are loosely coupled; therefore, this block is independent of the architecture used to expose objects as web services.

Learning block – This block learns the possible undertakings of activities given a certain event via semantic ontology acquired through time (For example, via log repository as applied in [116]). This block is the subject of future research.

Desciption of the modules in Figure 2.1 are explained as below.

- Event trigger: This module continually probes the object state module, and will be triggered given a pre-specified condition of the object state module is met. The pre-specified conditions are coded in the Trigger goal LUT module. Upon triggering, a goal will be added to the goal list module. For design purpose, trigger should come from human command and events such as raining and emergency situation. Given a trigger, it will place a corresponding goal in the goal list module (ex: ask about leisure, goal during raining, lack of sleep, not taking meals, falling etc.). Conditions are to be set to prevent continual triggering (in an event of unsatisfied goals given a persistent triggering condition met).

- Trigger goal Look-up-table (LUT): Rules of QOL and emergencies, where the user wants a certain reaction from the ISS, should be coded into this module with respect to the state variables in object state module. Its formalization is to be coherent with

that in the variable/activity updater module.

- Goal list: It contains the goals, which the planner module will use to plan the sequence of activities to satisfy the goals.
- Activity LUT: This module contains all the available service activities that currently exist in the ISS, which are provided by external devices via the pervasive block.
- Planner: This module uses the goal list, activity LUT and Variable LUT modules to plan a sequence of activities such that they can satisfy the goals. It uses constraint processing and employs an optimistic approach to planning. Goals can come from within the planner through activities via including it to the goal list module. Two types of planner are devised to deal with sequencing problem, which is, task activation method and through goals.
- Orchestrator: Upon receiving the plans from the planner module, orchestrator module will implement the plan by checking the conditions before implementation and distributing tasks to the respective devices according to the plan. It will request the planner module to re-plan if contradiction is met.
- Object repository: It keeps and updates the records of all devices and objects that exist within the grasp of ISS. It includes the atomic services the devices can offer, the names, properties and tags of devices and objects (location, able to be sit, clothesline etc. such that variable/activity updater can use it to determine the appropriate set of suiting for it). It also keeps track of devices being plugged in or detached from the ISS, and will inform variable/activity updater module. Every device connected to this module need to provide two things, that is, 1) what it can provide (given input) and 2) what it can receive (given input). For example, for weather web service, it can provide weather information (given location as input), and for TV service, it receives a TV switching state variable to turn on or off the TV. These are termed as atomic services. How the devices perform the atomic services is hidden from the module. For "human" object, some information can be provided by its own atomic services like asking for preferred activity and TV channel. More complex services (that requires additional objects to provide their service) like obtaining their location and emotion can be performed after being suit up by variable/activity updater module. This is subject of future work. Object repository module receives these atomic services and relay them to variable/activity updater module in order to suit the atomic services up (with pre-conditions and multiple output effects) for more descriptive and complex services.

- Object state: This module is the central mediator of the blocks. It keeps the current value of the variables, record the states, and also can be changed to get reaction from actuators. The variable/activity updater module updates its list of variables (not the value of variables, which are being manipulated by activities).
- Variable/Activity updater: Activity pre-conditions and proposition of goals depends on current available list of devices and variables. It is up to this module to update these propositions and pre-conditions in terms of names, without affecting the actual proposition. It suits up the atomic services provided by the object repository module for more complex service activities according to the tags and properties associated with the object. This is performed via variable binding through the help of semantic annotations of devices as explained in Section 5.2. It also retracts and includes instances of activity, goal and variable into the LUTs for the planner depending on the available devices connected to the object repository module, which it gets its input from.
- Learner: A learning module that learns association between goals and trigger event, propositions and activity.
- Knowledge base: A large collection of semantic associations that is used for reasoning and learning. Its representation will be based on ontology stored in the Look up table module.
- Life pattern classifier: Classifiers and inference machine to make sense of the raw data, and to provide useful information.
- Object log: Consists of logs that are needed for learning and inference. It stores information with necessary taggings and timestamps such that they can be rearranged for further use in the future.
- Human data: Consists of schedules, state, preferences and general information of people.
- Raw data storage unit: A place to store raw data that is needed for further processing, such as image, video, sound, and stream of data from sensors.
- Look up table: It contains the association between ID and a declaration (like location, object type and behavior), so that all modules in the ISS will be talking the same language. This module is crucial for specifying the ontology for semantic inference as well.

- RFID: Module that handles objects identified by their RFID. Currently it is intended to be used on consumable products in the house (ex: toothpaste, butter, oil).
- Web service: Module that handles web services. Currently it includes news and weather information web service
- UPnP devices: Module that handles hardware devices like robots, sensor module, URG locator, TV, home lighting, air-conditioning. These devices need to support plug and play capability.
- SLAM/Object locator: For objects that can't be directly plugged to the ISS (chairs, tables), they will be located and recognized. Their recognized properties are being handled by this module.
- Internal: This module handles objects that are required for the functioning of the ISS.

2.5.4 Relationship with the Current Work

This work deals with personalized service provision endowed with intelligent capabilities. It does not deal with the technicalities of middleware, where it assumes that devices are connected with their information handled by ISS. It also does not handle classifications to obtain objective indicators. It assumes these information are already obtained.

Therefore, the main blocks dealt in this work is the planning/implementation block, which handles the planning and execution part explained in Chapter 4. This is where planning is performed, and its execution instructions are then sent to the pervasive block. State information is obtained from Object state module, which store the variables and their changes described in subsequent chapters. For Learning block, Variable and activity updater module is used for variable binding, which is explained in detail in Section 5.2. This module obtains the device semantics from knowledge base module, which is passed from the pervasive block. Inference is made from the knowledge base based on the ontology specified in the Look up Table of Information block.

When face identification is performed, user information is obtained from Human data module of the Information block. This information includes the QOL list explained in Section2.5.2. It is assumed that ISS can keep track of the identified person (and thus, his/her QOL will be in the Goal List module of Planning/implementation block of the ISS of the environment the person is in), until he/she left. For unidentified people, default QOL is assigned.

Chapter 3

Joint Probabilistic Open Set Face Identification with Transfer Learning

For human-centric approach to work, it needs to deliver services that fulfill personal goals, or QOL. This means identities of the users are important, since personal QOLs are extracted given identities are known. This chapter deals with Human Identification Module (HIM), which employs open set face identification in order to extract QOLs. It needs to be open set as the system needs to determine whether it knows the person or not (and if not, default QOL will be assigned instead).

One may choose between a peripheral identifier (like RFID) and biometrics for identification. Since peripheral identifiers will not be widely used as personal identifiers in the near future, biometric is used. In this case, face is chosen since it can be handled without cooperation from the humans.

The requirements for the face recognition system of robot partners are that they need to function well in uncontrolled and uncooperative situation, all of which are performed by low powered processors. Although informationally structured space can provide powerful processing capability, here, it is used only as storage. The reason is that for this work, the intention is to perform test on the feasibility of running the whole system using low powered processor.

3.1 General Framework

A framework as shown in Figure 3.1 is developed for real-time face recognition. The framework involves open set face recognition, transfer learning and real-time adaptive learning. For clarification, the term "template" refers to the comparison measurement in

the joint probabilistic comparison module, which is trained using samples from the baseline domain termed as source domain. Domain specific target domain (which normally has a significantly lower number of samples, but which addresses the current situation more directly) can be merged with the source domain under transfer learning for improvements. As identification is achieved through selecting the class that has its reference image nearest to the input, these reference images are termed as "prototype".

As illustrated, input images are preprocessed before the descriptor vector is passed to the recognition system. The preprocessing modules are illustrated in Figure 3.2, and will be described in more detail in Section 3.2.

Open universe (or open set) face recognition needs to be able to reject unknown faces and not just giving the nearest identity the input is associated to, contrary to close universe (or close set) recognition, which tries to find the closest fit to the list of identities in its database. Therefore, methods that are able to handle open universe scenario are much more realistic. For this, joint probabilistic method [30] is used because of its transparency in data handling and simplicity.

Due to the possibility of prototype sample scarcity and different environmental conditions, transfer learning is required to apply knowledge from selected domains or classes in order to facilitate recognition. In transfer learning, knowledge from target domain (information for the intended problem) is merged with source domain (information from a different but related source to the intended problem) to construct the recognition template.

Real time adaptation can also help mold a suitable baseline for face recognition in a new environment. Identification system start off with limited samples to construct templates for identification. Over time, adaptation towards the environment is attained through constant accumulation of samples for learning and constructing better face prior.

To provide a gist of the framework in Figure 3.1, during identification, the preprocessed input will be passed to the joint probabilistic comparison module, where it is compared with prototype data of different individuals stored in the database. The output is the identity and the familiarity of the input image. The template for comparison is trained via data from source domain and target domain from the database. Detailed explanation will be given in Section 3.3. Besides comparing with prototype data, the input data is also extracted and organized through spatio-temporal association such that useful data can be extracted to help refine the template of the joint probabilistic comparison module. Details of the spatio-temporal association method are given in Section 3.6.

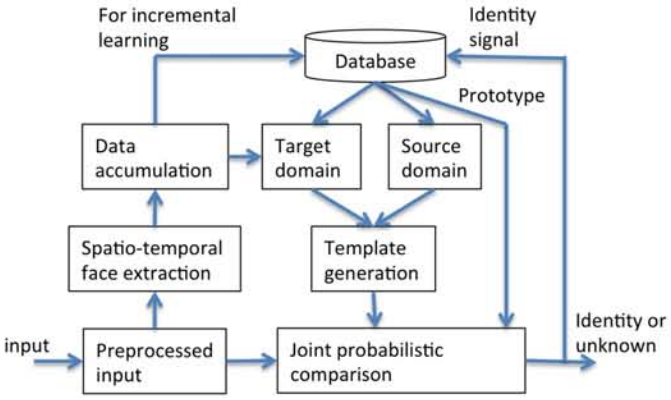


Figure 3.1: *Real-time face recognition framework*

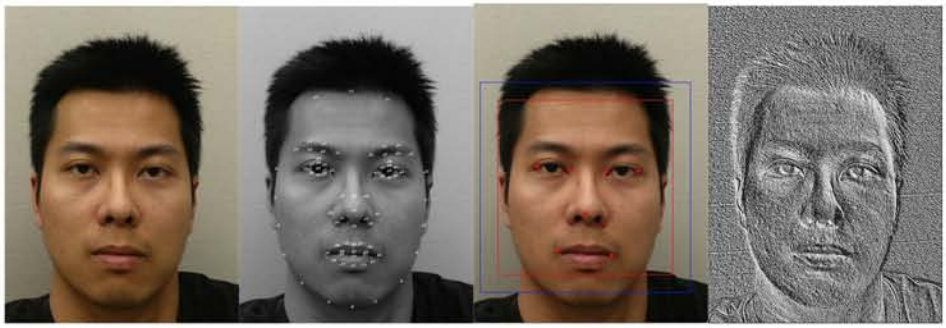
Figure 3.2: *Input image preprocessing sequence to extract appropriate descriptor for joint probabilistic face comparison*

3.2 Image Preprocessing

Before identification can be performed, input images are preprocessed to obtain suitable descriptors that are sufficiently informative. The modules are shown in Figure 3.2. The preprocessing is required to reduce variances without significantly affecting the information content, while also improve discriminability between faces of different individuals.

Given an input image, face detection is first performed through Viola-Jones Haar Cascade method [137]. Facial landmarks are located using Flandmark [134]. Although this approach only locate a very small amount of landmark compared to Active Shape Models like the STASM [95], this approach is faster and more suitable for low powered processors. The face will then be re-aligned according to the eye pairs as detected by the landmark detector, thus, eliminating variances of rotation and scale. For better alignment, in the future, the cascade regression method [26], which can locate more than 80 landmark points in milliseconds, will be used to improve alignment.

The aligned face is loosely cropped approximately as demonstrated in the work by [110]. The face is also partitioned into one (for experiment in Section 3.7.4.2) or four patches (for experiment in Section 3.7.4.3 and subsequent experiments after that) of varying location and sizes. This is to create different partition for local binary pattern (LBP) to work on to generate overcomplete representation. It has been shown that overcomplete high dimensional representation can almost always improve accuracy after dimension reduction [31].



(a) *Original Photo* **(b)** *Feature Localization via STASM* **(c)** *Feature Localization via Flandmark* **(d)** *LBP Representation*

Figure 3.3: *Face localization and preprocessing*

LBP descriptors are then extracted and histogramed [2]. Uniform LBP of $LBP^{u}_{2,8}$ is used, after which the LBP image patch is divided into regions. The histograms for all regions and patches are concatenated to form the final overcomplete descriptor. The descriptor with a dimension of around 20k is reduced to 400 using principal component analysis. Principal components extracted are fixed from the start. The preprocessing steps are illustrated in Figure 3.4.

3.3 Joint Probabilistic Method

Here we will provide a summary of the approach, before elaborating further the reformulated approach that can easily support transfer learning. For more details, readers can refer to [30].

Lets denote that x_1 and x_2 are column vectors of 2 preprocessed face representations to be compared, $P(x_1, x_2|H_I)$ is the intra-personal model (H_I is the hypothesis that both inputs are of the same identity) and $P(x_1, x_2|H_E)$ is the extra-personal model (H_E is the hypothesis that both inputs are not from the same identity), m is the number of prototype images per subject or per class, d is the dimension of input, n is the identity or class index with N classes (but this definition is limited to one configuration as will be discussed in Section 3.4), Ω is the configuration index and Γ is the individual identity index

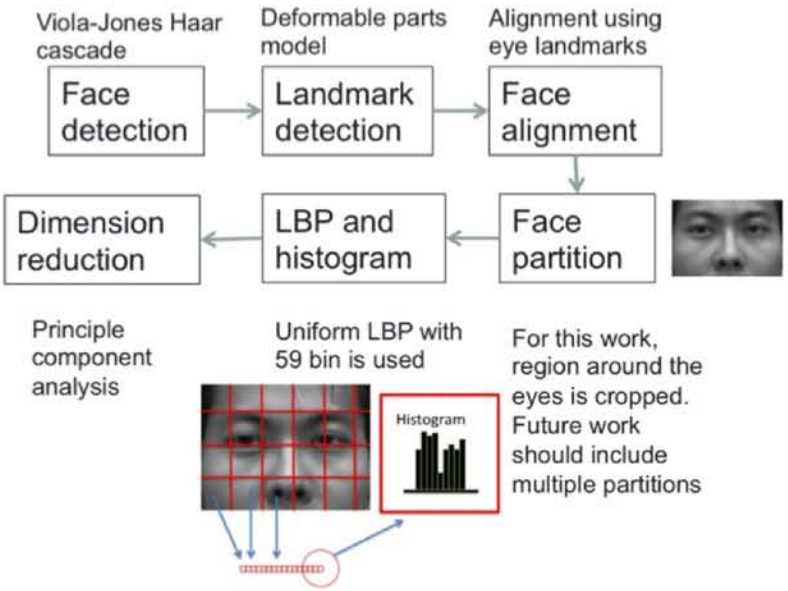


Figure 3.4: *Face preprocessing steps*

3.3.1 Pair-wise Similarity Inference

Given the following equation:

$$r(x_1, x_2) = \log(P(x_1, x_2|H_I)/P(x_1, x_2|H_E)) \tag{3.1}$$

$r(x_1, x_2)$ of (3.1) gives a scalar value that measures the similarity between the inputs, where the higher it is, the more similar x_1 and x_2 is. Let us assume that the face representation can be constructed by a summation of μ (the identity latent vector that encodes the identity) and ε (intra-class variation factor that encodes the variation given an identity) that are normally distributed, where both have covariances denoted as S_μ and S_ε respectively, and that both are obtained using m number of images per subject (where there is a total of N subjects). μ and ε are assumed to be independent, thus, the covariance of vector $x = S_\mu + S_\varepsilon$.

Given the conditional probability:

$$P(x|a) = \frac{1}{\sqrt{2\pi^d|\Sigma|}} e^{-\frac{1}{2}(x-\mu_a)^T\Sigma_a^{-1}(x-\mu_a)} \tag{3.2}$$

Under H_I model, the prior condition is that the random variable μ comes from the same mean (due to same identity) for x_1 and x_2, thus, same covariance. Another prior condition is that ε is independent for two different intra-class variations. Therefore, the covariance Σ_I for cocatenated column vector $[x_1; x_2]$ is:

$$\Sigma_I = \begin{pmatrix} S_\mu + S_\varepsilon & S_\mu \\ S_\mu & S_\mu + S_\varepsilon \end{pmatrix} \tag{3.3}$$

Under H_E model, the prior condition is that the random variable μ are generated from distribution of different means. Identities are independent of each other. Therefore, the covariance Σ_E for vector $[x_1; x_2]$ is:

$$\Sigma_E = \begin{pmatrix} S_\mu + S_\varepsilon & 0 \\ 0 & S_\mu + S_\varepsilon \end{pmatrix} \tag{3.4}$$

By substituting Eq. 3.4 and 3.3 into Eq. 3.1, given the conditional distribution shown in Eq. 3.2, and ignoring the constant values, the following is obtained for $r(x_1, x_2)$:

$$r(x_1, x_2) = x_1^T A x_1 + x_2^T A x_2 - 2x_1^T G x_2 \tag{3.5}$$

where

$$A = (S_\mu + S_\varepsilon)^{-1} - (F + G) \tag{3.6}$$

and

$$\begin{pmatrix} S_\mu + S_\varepsilon & S_\mu \\ S_\mu & S_\mu + S_\varepsilon \end{pmatrix}^{-1} = \begin{pmatrix} F + G & G \\ G & F + G \end{pmatrix} \tag{3.7}$$

3.3.2 Covariance Learning

In order for Eq. 3.5 to perform similarity inference, covariance S_μ and S_ε need to be found. This sub-section will show how both covariances are obtained.

Expectation-Maximization (EM) algorithm is used to learn S_μ and S_ε that is required for the model.

Expectation part proceeds as follows:

Given $\bar{h} = [\mu; \varepsilon_1; \varepsilon_2 ...; \varepsilon_3] \in \Re^{(m+1)d \times 1}$ and $\bar{x} = [x_1; x_2 ... x_m] \in \Re^{md \times 1}$, both with 0 means, lets denote:

$$\bar{p} = \begin{pmatrix} I & I & 0 & \cdots & 0 \\ I & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & 0 & 0 & \cdots I & \end{pmatrix} \tag{3.8}$$

where I = identity matrix $\in \Re^{md\times(m+1)d}$

We have:

$$\bar{p}\bar{h} = \bar{x} \tag{3.9}$$

Covariance for $\bar{x}$ is:

$$\Sigma_x = \begin{pmatrix} S_\mu + S_\varepsilon & S_\mu \cdots & S_\mu & \\ S_\mu & S_\mu + S_\varepsilon & \cdots & S_\mu \\ \vdots & \vdots & \ddots & \vdots \\ S_\mu & \cdots & \cdots & S_\mu + S_\varepsilon \end{pmatrix} \tag{3.10}$$

Covariance for $\bar{h}$ is:

$$\Sigma_h = diag(S_\mu, S_\varepsilon \cdots S_\varepsilon) \in \Re^{(m+1)d\times(m+1)d} \tag{3.11}$$

Therefore, the covariance of $[\bar{h}; \bar{x}]$ is:

$$cov([\bar{h}; \bar{x}]) = \begin{pmatrix} \Sigma_h & cov(\bar{h}, \bar{x}) \\ cov(\bar{x}, \bar{h}) & \Sigma_x \end{pmatrix} \tag{3.12}$$

where $cov(\bar{h}, \bar{x}) = \Sigma_h \bar{p}^T$

From Eq. 3.12, through Schur complements, the expected $\bar{h}$ given input $\bar{x}$ (or expected μ with its intra-variants ε given a set of inputs xs) is:

$$\bar{h}|\bar{x} = \Sigma_h \bar{p}^T \Sigma_x^{-1} x \tag{3.13}$$

Eq. 3.13 can be broken down to solve for the latent vectors μ and ε separately due to independence property. Therefore, with the relation between Eq. 3.7 and Eq. 3.13, given a single identity or class index n with m images, its μ and ε can be found as:

$$\mu^n = \sum_{i=1}^{m^n} S_\mu (F + mG) x_i^n \tag{3.14}$$

$$\varepsilon_j^n = x_j^n + \sum_{i=1}^{m^n} S_\varepsilon G x_i^n \tag{3.15}$$

where $F = S_\varepsilon^{-1}$ and $G = -(mS_\mu + S_\varepsilon)^{-1} S_\mu F$

At the current stage, it is assumed that $m^{n=1} = m^{n=2} ... = m^{n=N}$. This condition is limited to one configuration, thus, it is more appropriate to represent it as m^Ω. Configurations

will be explained in detail in Section 3.4.

For maximization part of the EM algorithm, covariance for μ and ε are calculated to obtain new S_μ and S_ε.

3.3.3 Reformulation of Joint Probabilistic Learning

Lets denote $x_i^n \in \Re^{d\times 1}$ is a column vector representing image i of identity or class n, $H = S_\mu(F + mG)$, and $K = S_\varepsilon G$. Both H and K are functions of m. Therefore, they are configuration Ω dependent.

Given the following equations:

$$\bar{X}^n = \sum_{i=1}^{m} x_i^n \tag{3.16}$$

$$\hat{X} = \sum_{n=1}^{N} \bar{X}^n (\bar{X}^n)^T \tag{3.17}$$

$$\bar{E} = \sum_{n=1}^{N}\sum_{i=1}^{m} \frac{x_i^n}{Nm} \tag{3.18}$$

$$X = \sum_{n=1}^{N}\sum_{i=1}^{m} x_i^n (x_i^n)^T \tag{3.19}$$

Through substitution, Eq. 3.14 can be re-written as $\mu = H\bar{X}^n$. S_μ can be also be calculated as $S_\mu = (N-1)^{-1}(\sum_{n=1}^{N}(H\bar{X}^n - E(H\bar{X}^n))(H\bar{X}^n - E(H\bar{X}^n))^T)$. Let's also denote :

$$\hat{X}_o = \hat{X} - Nm^2\bar{E}\bar{E}^T \tag{3.20}$$

$$X_o = X - Nm\bar{E}\bar{E}^T \tag{3.21}$$

Due to condition that $m^{n=1} = m^{n=2} ... = m^{n=N}$ as stated previously, $E(\bar{X}^n) = mE(x_i^n) = m\bar{E}$, using equations from Eq. 3.16 to 3.19, we can reduce the equation to:

$$S_\mu = H(\hat{X} - Nm^2\bar{E}\bar{E}^T)H^T/(N-1) \tag{3.22}$$

If the mean of the input samples are inherently 0, then the numerator of the equation is reduced to $H\hat{X}H^T$. Given Eq. 3.20 and 3.21, Eq. 3.22 can be represented as:

$$S_\mu = (H\hat{X}_o H^T)(N-1)^{-1} \tag{3.23}$$

Eq. 3.15 can be re-written as:

$$\varepsilon = x_j^n + K\bar{X}^n \tag{3.24}$$

Therefore, to calculate S_ε, we have:

$$S_\varepsilon = (x_j^n + K\bar{X}^n - \frac{1}{N}\sum_{j=1}^{N}(x_j^n + K\bar{X}^n))(x_j^n + K\bar{X}^n - \frac{1}{N}\sum_{j=1}^{N}(x_j^n + K\bar{X}^n))^T(Nm-1)^{-1} \tag{3.25}$$

Given Eq. 3.20 and 3.21, Eq. 3.25 can be represented as:

$$S_\varepsilon = (X_o + \hat{X}_o K^T + K\hat{X}_o^T + mK\hat{X}_o K^T)(Nm-1)^{-1} \tag{3.26}$$

It can be observed from (3.23) and (3.26) that the training database of a domain can be represented more compactly by five components, which are, symmetric matrices $\hat{X}$ and X, vector $\bar{E}$, and scalar N and m, given combined inter-classes, which we termed as template Ω for that particular domain. Therefore, given a template Ω, S_μ and S_ε are obtained by iterating through (3.23) and (3.26). More details on the implementation are given in Section 3.5.

3.4 Fixed Sample Number per Subject Constraint Relaxation

3.4.1 Rigidity due to Fixed Sample Number per Subject

As discussed for the joint probabilistic method, it is assumed that the number of images per subjects, m, is the same given template Ω. This is hardly the case and a troublesome constraint to oblige in practical sense. It is much more convenient if domains can easily be constructed from individuals or situations without such imposition, and that it can easily be assembled according to the task at hand. Thus, an alternative is to standardize m to increase robustness.

For explanation purpose, it is assumed that there is only one subject ($N = 1$) with m^Ω sample images. Specific for this sub-section, let us consider this as sample images to build template Ω. To standardize m sample images, (3.16) and (3.19) can be re-written as:

$$\bar{X} = \frac{m}{m^\Omega}\sum_{i=1}^{m^\Omega} x_i \tag{3.27}$$

$$X = \frac{m}{m^{\Omega}} \sum_{i=1}^{m^{\Omega}} x_i (x_i)^T \tag{3.28}$$

This can be seen as artificially increasing or decreasing the number of sample images to a constant m (where calculation of G and A will be based on). Experimental result shows that by artificially increasing the number of sample images, the accuracy will increase while variance drops [129], thus, giving a stable performance as shown in experiments in Section 3.7.2.

Every domain template therefore can be represented by $C_1^{\Omega} = m^{\Omega}$, $C_2^{\Omega} = \frac{1}{m^{\Omega}} \sum_{i=1}^{m^{\Omega}} x_i$ and $C_3^{\Omega} = \frac{1}{m^{\Omega}} \sum_{i=1}^{m^{\Omega}} x_i (x_i)^T$, which overall will be denoted as C^{Ω}. C_2^{Ω} and C_3^{Ω} are needed to obtain the covariance, whereas C_1^{Ω} is crucial for incremental learning when one needs to manipulate C_2^{Ω} and C_3^{Ω}. Intuitively, C^{Ω} belongs to an identity. With standardization in (3.27) and (3.28), C^{Ω} from various identities can be merged through (3.16)-(3.19).

The size of the template is $0.5d^2 + 1.5d + 1$, where d is the dimension of the image representation. Large number of training samples will not affect the computational load and speed. Apart from that, with C^{Ω}, template can be easily generated and merged for transfer learning from different domains, depending on the application.

3.4.2 Number per Subject Variation Induced Distortion

In this sub-section, analysis is done of the effects caused by using arbitrary number of images per subject, a, compared to the actual, m. It is assumed that the 0 mean condition is met. Lets denote:

$$\hat{E} = \frac{\hat{X}}{m^2} \tag{3.29}$$

If a is the new parameter for number of images per class (instead pf m), then Eq. 3.23, ignoring coefficients, can be re-written as:

$$a^2 H_a \hat{E} H_a^T = a^2 H_a (\frac{\hat{X}}{m^2}) H_a^T = \frac{a^2}{m^2} H_a \hat{X} H_a^T \tag{3.30}$$

where H_a means a is the parameter that replaces m. Now, we need to find out the difference as:

$$\Delta^2 = \frac{a^2}{m^2} H_a \hat{X} H_a^T - H_m \hat{X} H_m^T \tag{3.31}$$

Expanding Eq. 3.31, we obtain:

$$\Delta = [(a-m)I - (a^2 S_{\mu}(aS_{\mu} + S_{\varepsilon})^{-1} - m^2 S_{\mu}(mS_{\mu} + S_{\varepsilon})^{-1})] \frac{S_{\mu} S_{\varepsilon}^{-1}}{m} \tag{3.32}$$

It can be observed that distortion on S_μ is introduced by S_ε as parameter m varies. But the larger the m and a, the less effect S_ε has on S_μ. Large m, meaning more images per class, is crucial for overall reduction of the difference. More analysis needs to be performed to obtain features that will induce more optimal S_μ and S_ε.

For S_ε, Eq. 3.26, ignoring coefficients and X (since m doesn't have effect on X), can be re-written as:

$$\frac{a^2}{m^2}\hat{X}G_a^T + \frac{a^2}{m^2}G_a\hat{X}^T + \frac{a^3}{m^2}G_a\hat{X}G_a^T \tag{3.33}$$

The difference can be obtained as
First component (Similar to second component transposed) of Eq. 3.33:

$$\Delta = \frac{S_\varepsilon}{m^2}(m^2(mS_\mu + S_\varepsilon)^{-1} - a^2(aS_\mu + S_\varepsilon)^{-1})S_\mu S_\varepsilon^{-1} \tag{3.34}$$

Third component of Eq. 3.33:

$$\Delta = \frac{\sqrt{a}S_\varepsilon}{m}(m(mS_\mu + S_\varepsilon)^{-1} - a(aS_\mu + S_\varepsilon)^{-1})S_\mu S_\varepsilon^{-1} \tag{3.35}$$

As can be observed from Eq. 3.34 and 3.35, larger m can reduce distortion due to more sample images. At the same time, larger a can also reduce the effects of S_ε. Therefore, given sufficient samples (means sufficient m), and a very large a, the difference will approach 0.

3.4.3 Modifications for Transfer Learning

For transfer learning to occur, in M-step as shown by [27], target domain with scarce samples can be merged directly with the information rich source domain to obtain new covariance shown in (3.36) and (3.37) given that $\ddot{\mu}$ and $\ddot{\varepsilon}$ are from target domain samples:

$$T_\mu = wS_\mu + (1-w)n^{-1}\sum_i \ddot{\mu}_i\ddot{\mu}_i^T \tag{3.36}$$

$$T_\varepsilon = wS_\varepsilon + (1-w)k^{-1}\sum_j \ddot{\varepsilon}_j\ddot{\varepsilon}^T \tag{3.37}$$

$w \in (0,1)$ acts as a weight to determine how much the source domain and target domain will impose its effect on the final covariance, and k and n are number of samples for the factors. Source domain provides the appropriate base line for the target domain to work on. With transfer learning, this method achieves state-of-the-art accuracy. It has been shown that the transfer learning method gives a similar performance for a large-scale database (20 times larger) trained system, meaning that it is an extremely efficient method.

The joint probabilistic method as described is rigid due to the fact that it does not directly cater for arbitrary number of training samples per subject during template training. This is important because if the number of samples does not oblige to a single m, one may need to perform the E-step separately for the different m. Apart from that, the way the samples are trained is not efficient as every samples need to be stored and looped through during training. This problem is worsened if continual accumulation of samples occurs.

Fortunately, the joint probabilistic method is reformulated as described in Section 3.3.3 to enable more efficient training and storage, yet easy to store, categorize and merge between domains. E-step and M-step as described can be combined.

With the new representation for database (as configurations Ω) and the conditions to apply different images per class m, transfer learning can be implemented. Let's denote:

$$E^X = \frac{X}{(Nm)^2} \tag{3.38}$$

Given configurations ($\Omega = 1, 2...Q$), where they have the same m but different N, with Eq. 3.29 and 3.38, the combination of the two configurations (similar to traning the two sets of database) can be performed as:

$$X_{new} = \sum_{i=1}^{Q} (N^{\Omega=i} m)^2 E^{X^{\Omega=i}} \tag{3.39}$$

$$\hat{X}_{new} = \sum_{i=1}^{Q} m^2 \hat{E}^{\Omega=i} \tag{3.40}$$

$$N_{new} = \sum_{i=1}^{Q} N^{\Omega=i} \tag{3.41}$$

For the case where m is different for different Ω, there are two alternatives, which are 1) apply different H and K (since they are parameterized by m) respectively of the configurations, and, 2) H and K have the same m, but $\hat{X}$ and X are re-scaled to a standard m with Eq. 3.29 and 3.38.

For alternative 1, the method is similar to that of Eq. 3.13 to 3.15, but with additional process of constructing the different H and K matrices. For alternative 2, there will be distortions due to difference in $\hat{E}(\Delta \propto \hat{E}_1 - \hat{E}_2)$ of the same class given different m of each $\hat{X}$. To obtain the new configurations, the method is the same as Eq. 3.39 to 3.41, but with m according to its repective database. If m is large, according to the law of large numbers, the distortion will be negligible.

3.5 Domain Mergence

As shown in Figure 3.5, prototype images (images used to compare with the input) are stored according to identities. Across the identities, they can be separated by domains (For example, what type of medium the image is taken, the type of lighting, what room they are taken etc.). Transfer learning is applied, where one applies the information from the target domain (often times, scarce in sample image) of other identities with the source domain (built from large amount of non-specific sample images) to facilitate recognition. From the figure, target domain A molds the source domain B (which provides a baseline) to further suit it for recognition for a human subject currently captured under the condition of domain A. Algorithm 1 and 2 show the implementation of domain specific recognition. Ω is referring to one subject with C_1^{Ω} number of sample images, meaning domain refers to a specific identity. Whether the identity should be unique or not, and also its effects, are not studied in this work.

For domain merging shown in Figure 3.5, input consists of the templates for the domains needed to be merged, and the output is a new template that will be used for subsequent training for S_μ and S_ε (as shown in Algorithm 2. To increase the effects of certain domain, we multiply the data using w_f. By default, it is set to 1. This parameter is used when we want to multiply the data samples of the target domain. From experiments shown in Section 3.7.3.2, it is shown to be able to increase accuracy.

S_μ and S_ε are used to obtain A and G via Algorithm 1, which is termed the template that is to be used for face recognition. Both the matrix A and G are applied to (3.5) to compare prototype images with input image. The class where the prototype image achieves the highest value is the identity of the input. To achieve an open-set recognition, one may evaluate the maximum value of the winner, where it is considered unknown if the value is below certain threshold.

3.6 Spatio-temporal Association

Real time face learning while performing face recognition can be exploited to further improve the accuracy. Faces are collected at real time such that useful information can be continually obtained related to the environment to improve recognition. Real time extraction provides a source of target domain samples that is much similar to the current working condition. This can be very effective in a crowded place where there are a lot of variations from different identities, thus a great source of information, which can contribute to learning the facial baseline in that particular environment.

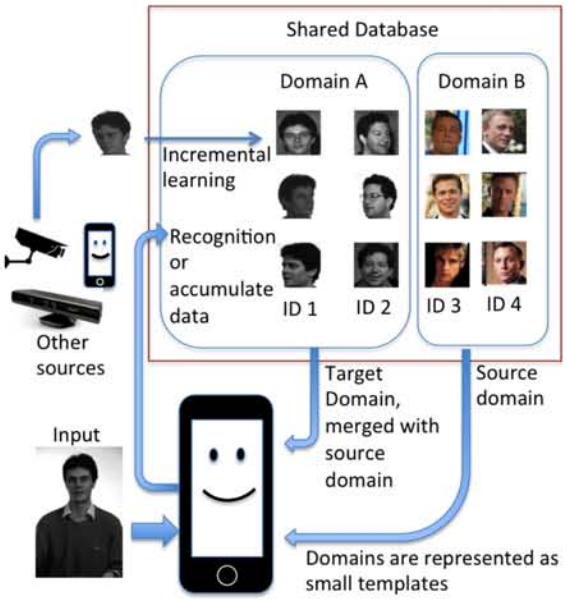


Figure 3.5: *Source domain and target domain information flow for face recognition. Domain B is the source domain that contains massive amount for sample image. Domain A is the target domain with low number of sample image. Input image is captured under the same condition as Domain A, therefore, template from Domain A will contribute to its recognition.*

In accordance to the method employed for transfer learning of the joint probabilistic face comparison, one needs to categorize faces according to identities before transfer learning can occur. In an uncooperative situation, the identification system needs to collect and categorize the faces based on identity automatically. It needs to know whether subsequent captured faces belong to the same person or not. To solve this, we apply a simple spatio-temporal association, where faces that occur close in space and time are considered the same face. Let us denote x_n and y_n as x and y (in pixels) of the center point of the bounding box after face detection at time t_n (in seconds), and x_len_n and y_len_n as the length of the width and height of the bounding box, where n is the sequence index of input image stream. The proximity function PF is then described as:

$$PF = 4\frac{|x_n - x_{n-1}| + |y_n - y_{n-1}|}{x\_len_n + y\_len_n} + |t_n - t_{n-1}| \tag{3.42}$$

The smaller the value of PF gets, the nearer the distance between two subsequent samples in space and time. The max PF is denoted as PF_{MAX}, where two subsequent samples are considered not to be in the same category if PF exceeds this max value.

Besides proximity measure, another procedure is required to prevent a constant inflow of similar samples. For example, it is not favorable to keep on collecting samples from a

Data: Domains selected $\Omega = 1, 2, 3...N$
Result: X_o and $\hat{X}_o$
Set the constant m;
Number of subjects $n = 0$;
//Looping through all domains;
for $f = 1$ *to* N **do**
  //Merging domains;
  $\hat{X} = \hat{X} + w_f m^2 C_2^{\Omega=f} (C_2^{\Omega=f})^T$;
  $X = X + w_f m C_3^{\Omega=f}$;
end
$\bar{E} = \sum_{j=1}^{n} w_j C_2^{\Omega=f} / \sum_{i=1}^{n} w_i$;
//Zero mean;
$\hat{X}_o = \hat{X} - Nm^2 \bar{E}\bar{E}^T$;
$X_o = X - Nm\bar{E}\bar{E}^T$

Algorithm 1: Algorithm for domain mergence

human subject who just remains stationary and idle. Not just incurring redundancy, but this will reduce dispersion (and therefore, reducing invariance) when the samples are merged to the source domain to calculate the covariance matrices. Accuracy will deteriorate as a result. Therefore, a mechanism to ensure no constant stream of stationary faces is collected needs to be devised. Thus, similarity measure is used to calculate based on (3.5) between the input and the latest sample in the category after proximity measure. The input will be rejected from being added to the category with minimum PF if the similarity measure is above certain preset threshold. The implication is that no two similar samples will occur side by side in the category.

In terms of architecture, 5 categories are allocated to temporarily store the input samples, where the identities for each category are the same. PF for each input is calculated for every category. The input will be added into the category given that the lowest PF value is below the max proximity and that the similarity measure is below certain threshold. This process is shown in the flow chart in Figure 3.6. If PF is larger than the max proximity value, this indicates a new person appears in the range of view. A new category will then be used to store the samples for this person. If all categories are used up, the new sample will be rejected.

Input samples are only compared with the latest sample in the category as described in (3.42). Every category needs to keep track of the information (face boundary length, center point, and timestamp) of its latest sample such that PF can be calculated. An exception is the case when PF is below max value, but similarity measure is above the threshold. In this case, the sample will not be added to the category, but the information of the latest sample

Data: X_o and $\hat{X}_o$
Result: A and G
Randomize S_μ and S_ε;
for $f = 1$ *to* T **do**
  //Calculate matrix based on current S_μ and S_ε;
  $G = -(mS_\mu + S_\varepsilon)^{-1} S_\mu S_\varepsilon^{-1}$;
  $H = S_\mu(S_\varepsilon^{-1} + mG)$;
  $K = S_\varepsilon G$;
  //Update S_μ and S_ε;
  $S_\mu = (H\hat{X}_o H^T)(N-1)^{-1}$;
  $S_\varepsilon = (X_o + \hat{X}_o K^T + K\hat{X}_o{}^T + mK\hat{X}_o K^T)(Nm-1)^{-1}$
end
$A = (S_\mu + S_\varepsilon)^{-1} - (S_\varepsilon^{-1} + G)$

Algorithm 2: Algorithm for efficient covariance calculation

is set according to the input sample.

When conditions are right, samples stored in the categories are collected and used as target domain to be merged with the source domain to construct face recognition template. The category will also be cleared to make way for samples of new identities. The process is illustrated in Figure 3.7. The process will loop through every category to check whether its latest timestamp has expired. For the system, an expiration of T_E second is set, which means if no samples are obtained within T_E seconds, it will collect and clean up the category for others. After this, a second check is performed to determine if minimum number of samples per category is reached or not. Samples will be accumulated in the target domain for the former, and for the latter, the samples are discarded.

Samples collected that are to be added into the target domain are performed in the data accumulation block shown in Figure 3.1. Accumulation can be performed easily through Algorithm 1.

3.7 Experiments

Experiments are performed to validate the proposed method. The scope is on face recognition. Detailed explanations of the experiments are provided in their respective sections.

3.7.1 Face Dataset

Four public face databases are used, which are, Pubfig [79], FERET [105], ExtendedYale [48], FEI database [130] and Labeled Faces in the Wild (LFW) [65] database. Sample images

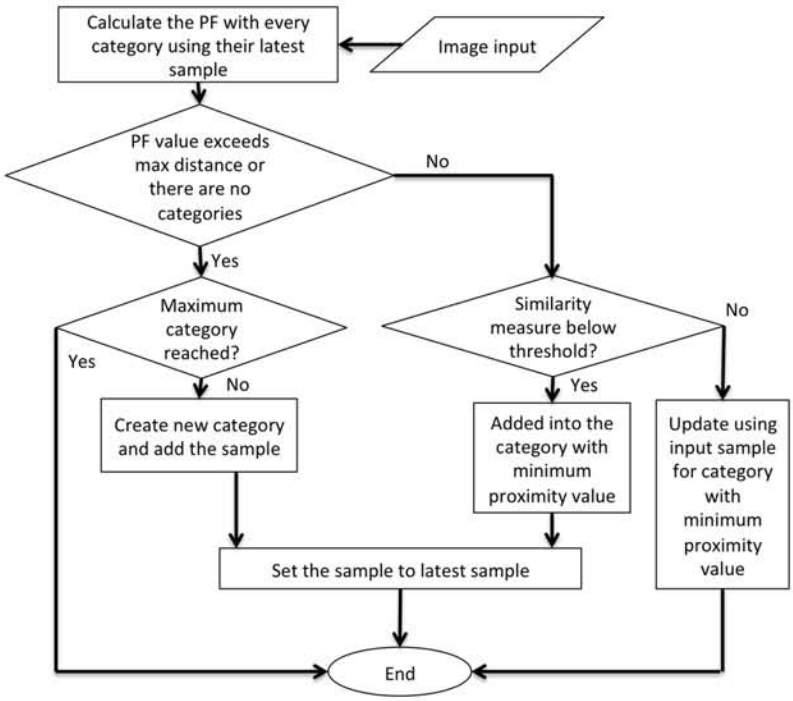


Figure 3.6: *Flow chart for spatio-temporal association*

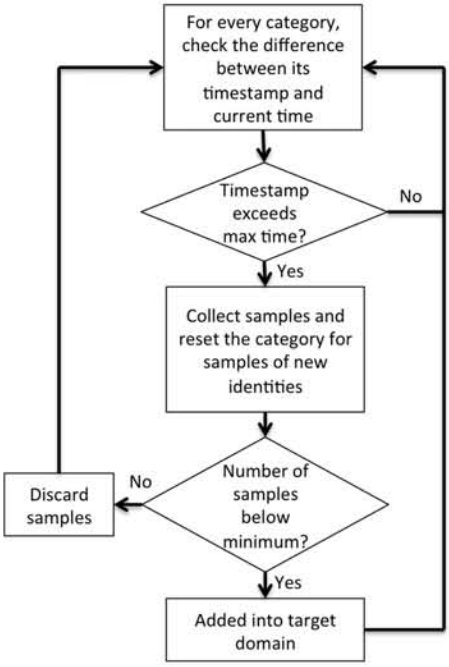


Figure 3.7: *Flow chart for control of category*

for Pubfig, FERET, ExtendedYale and FEI are shown in Figure 3.8 to 3.11 respectively.

Pubfig dataset is a large, real-world face dataset that consists of 58797 images of 200 people collected from the internet. These images are taken in completely uncontrolled situations with non-cooperative subjects. Thus, there is large variation in pose, lighting, expression, scene, camera, imaging conditions and parameters. It is this reason that Pubfig is chosen as there are numerous images for every subject that can be used for training and testing. Pubfig is appropriate to be used as source domain. LFW contains 13233 images of 5749 people, and is of similar spirit to Pubfig dataset. It is broader than Pubfig, but shallower, where significant amount of subjects have less than 10 prototype images. Due to this issue, LFW is not appropriate for training, but it is a good candidate to be used as distractors since LFW does not overlap with Pubfig.

FERET dataset is a standard dataset used for face recognition system, which includes 2413 still facial images representing 856 individuals. For this work, a subset of 200 individuals are used that images of them under different horizontal poses. Images captured in this dataset are controlled, but can be used as test benchmark for recognition under different pose and also a good target domain.

Extended Yale contains 16128 images of 28 human subjects under 9 poses and 64 illumination conditions. Similar to FERET, this dataset is a good benchmark to test on recognition under different poses and illumination.

FEI dataset contains 14 images for 200 individuals, which has a total of 2800 images. All images are colourful and taken against a white homogenous background in an upright frontal position with profile rotation of up to about 180 degrees. This dataset is mainly used as source domain.

3.7.2 Number of Images per Subject Standardization

3.7.2.1 Experimental Setup

Pubfig is used as source domain, FERET is chosen due to its large number of identity with standard settings, and ExtendedYale due to its large number of sample image per subject. Unless otherwise stated, 80 individuals from Pubfig with an average of 89 images per person is used as source domain, which, altogether, around 8000 faces. These 80 individuals will never again be used in tests. 200 principle components are also selected (instead of 2000 as in [30])from these individuals and be used for all tests, regardless of the database (Pubfig, FERET and ExtendedYale). Given an input image, face is detected using Viola-Jones Haar face detection method [137]. Alignment is then performed given eye-pair detection under deformable model [134]. Eye-pair is then segmented and used for recognition. But from the

Figure 3.8: *Pubfig sample images*

Figure 3.9: *FERET sample images*

Figure 3.10: *Extended Yale sample images*

Figure 3.11: *FEI sample images*

work of [18], it shows that other facial components also contain nearly the same magnitude of information as the eyes. The usage of other components will be included in future work, as current work is on the study of recognition improvement given efficient cross-domain mergence. LBP with 59 words [2] are then used on the segmented eye pair. After 10 by 10 partitioning, the final descriptor of every image is 5900 dimension vector, which is reduced to 200 via the PCA described previously. This vector will be used for the joint probability formulation and recognition. Coefficients are obtained with only 5 iterations of the modified EM algorithm described in Section 3.5.

3.7.2.2 Result and Discussion

A standard number of images per subject, m, is crucial in the robustness of the system. Given different number of m, we achieve a standard number by artificially increasing or decreasing m. Experiment is performed to determine its effects on the accuracy.

From Pubfig database, 50 images from 80 subjects are chosen to train, therefore, m=50. Result of various standard sample images m_S is shown in Table 3.1. Test is performed on 50 randomly chosen subjects (not included in training subjects), where each subject has 1 to 3 prototype images. Results for m_S that is near m =50 doesn't show any significant changes in performance. Performance drops as m_S decreases further. But, surprisingly, performance improves as m_S increases. Besides the improvement in recognition rate, the standard deviation decreases, which means, the system is getting more and more stable as number of sample images are increased artificially. The result remain the same as m_S is increased up to 10k. More importantly, the implication of this result is that the system does not need to oblige to m. Given different subjects with different number of sample images, the number

Table 3.1: *Standardization of image sample per subject given fixed amount (50 Samples) on Pubfig dataset*

m_S	10	20	40	m	60	100	200	400	1k
Mean(%)	59.9	65.1	66.0	66.3	65.9	66.0	66.8	67.2	67.5
SD	5.29	2.77	2.12	2.42	2.39	2.01	1.74	1.66	1.28

Table 3.2: *Standardization of image sample per subject given arbitrary amount on Pubfig dataset*

m_S	20	40	100	200	500	1k	10k
Mean(%)	68.3	75.0	76.2	76.5	77.2	77.2	78.0
SD	4.38	2.98	1.66	1.52	0.75	0.19	0.13

can be standardized to one m_S, and therefore, calculation of coefficients can be easier.

In order to test on arbitrary number of sample images per subject, experiment is the same as previous, but with a slight difference that m for each subject is chosen at random, ranging from 10 to 150. Table 3.2 shows the recognition result for various m_S. Results show that with arbitrary m for each subject, likewise, performance and stability improves as m_S increases. Besides, this property makes it easier for cross-domain mergence, where one does not need to take into account the sample image for each. This enables processing to be modularised for each domain, thus, improving efficiency. Apart from transfer learning, this also paved a way for incremental learning.

3.7.3 Cross-domain Mergence Effect

This section shows the influence of mergence between source and scarce target domain on performance.

3.7.3.1 Experimental Setup

Given the source domain, experimental studies are performed to study the effects of cross-domain mergence on the accuracy of recognition. As discussed, source domain comes from the selected subjects from Pubfig database with around 8000 images. Standardization is set to 5000, and will remain for the rest of the experiment. Identification test performed on 50 randomly chosen Pubfig subjects (excluding those in the source domain) gives an average rank 1 accuracy of 77.9% as shown in Table 3.3, with each subject having only 1 to 3 prototype image to compare with. As this is the result for the source domain alone, it will be used as comparison in the subsequent sub-section.

Table 3.3: *Pubfig identification test with source domain*

Rank	1	2	3
Accuracy(%)	77.9	85.48	92.43

Table 3.4: *Contribution of scarce target domain to identification on Pubfig dataset via sample multiplication*

Data Multi.	1	2	3	4	5	6	7	8
Accuracy(%)	78.2	81.4	82.3	83.1	80.6	79.6	77.9	75.3

3.7.3.2 Result and Discussion

To study the influence of mergence, there are two cases: First, multiplication of data sample without any increase of information. Second, direct increase the number of subjects (and thus, information).

Simulating the first case with small sample size of a different domain, we randomly select 10 subjects from the FERET database for every trial, where each subject has an average of 8 sample images, to be used as a separate domain to be merged. With the 10 subjects, we tested on different multiplication of the data to be merged with the source domain (means merging with the source domain n multiple times). This is to test the effects a small sample on the performance by manipulating the magnitude of exertion without actually increasing the information (more samples). Results in Table 3.4 show that a small sample can improve the accuracy, where by using its original sample size (multiplication 1), performance increases from 77.9% to 78.2%. But accuracy can be further improved by just multiplicating the sample further. Yet as more exertion is applied, the accuracy will start to drop, even below the accuracy without cross-domain mergence. This is due to the effects of lack of information bypassing the information contributed by the source domain. The amount of optimum exertion is subject of future research.

For the second case, experimental condition is the same as previous, except that more subjects are sampled from FERET database to be merged with the source domain. Table 3.5 shows the result. As expected, there is improvement in rank 1 accuracy compared to Table 3.4. Accuracy doesn't drop as subject number approach 80, compared to the first case.

Table 3.5: *Contribution of target domain to identification without sample multiplication*

No. of Subjects	5	10	15	20	30	40	60	80
Accuracy(%)	78.2	78.2	80.5	81.4	81.6	82.4	82.4	82.6

3.7.4 Cross-domain Mergence Test on Controlled Variations

This section shows the contribution of cross-domain mergence on invariant identification. Invariance on 3D pose and illumination variations are tested. Two kinds of tests are performed. The first test will evaluate the influence of the number of target domain on identification. The second test will evaluate on the use of more face patches and higher number of dimension in the representation on the performance.

3.7.4.1 Experimental Setup for Target Domain Size Evaluation

For FERET, images with letter codes "ba, bb, bcfi bk" are used, which consists of 200 subjects, where the pose ranges from -60 degree to 60 degree. Out of the 200, 150 subjects are randomly chosen for testing, where each subject only has one prototype image for recognition that is randomly chosen as well. The remaining 50 subjects will be used for training, where they act as target domain. Due to ExtendedYale database's large sample image per subject (more than 500), we performed experiment to study how the number of sample image per subject effect the accuracy. The database consists of 25 subjects. 20 subjects are randomly selected every trials for testing, where each has 8 prototype images that is randomly selected as well. For the target domain, the template is constructed by 5 subjects with varying amount of sample image. 8000 faces from Pubfig Dataset are used as source domain. In terms of representation, it is the same as that described in Section 3.7.2.1.

3.7.4.2 Result and Discussion for Target Domain Size Evaluation

Rank 1 identification result for FERET dataset is shown in Table 3.6. Given such scarce data sets, the accuracy can still be improved at a favourable magnitude. Data multiplication, though also increase accuracy, is limited, due to insufficient information to model the new domain, as opposed to test images from Pubfig database. Therefore, higher number of individuals of the same domain contributes more the accuracy. Improvement is achieved only through training samples related to the domain. This can be observed from Table 3.6, where 80 subjects from Pubfig, though large, doesn't contribute to accuracy in this test because it is of a different domain. Identification using higher number of dimension (400) is also tested, which, as expected, produce better result. Comparison is performed with ADMCLS method [117], which outperforms our work. But it should be noted that our work does not assume any ground truth regarding pose, and that face landmarks are performed automatically. Improvement can be made through better alignment and pose specific transformation matrix. Figure 3.12 shows identification result given multiple ranks under different number of training subjects for target domain. The graph tends to converge when number of

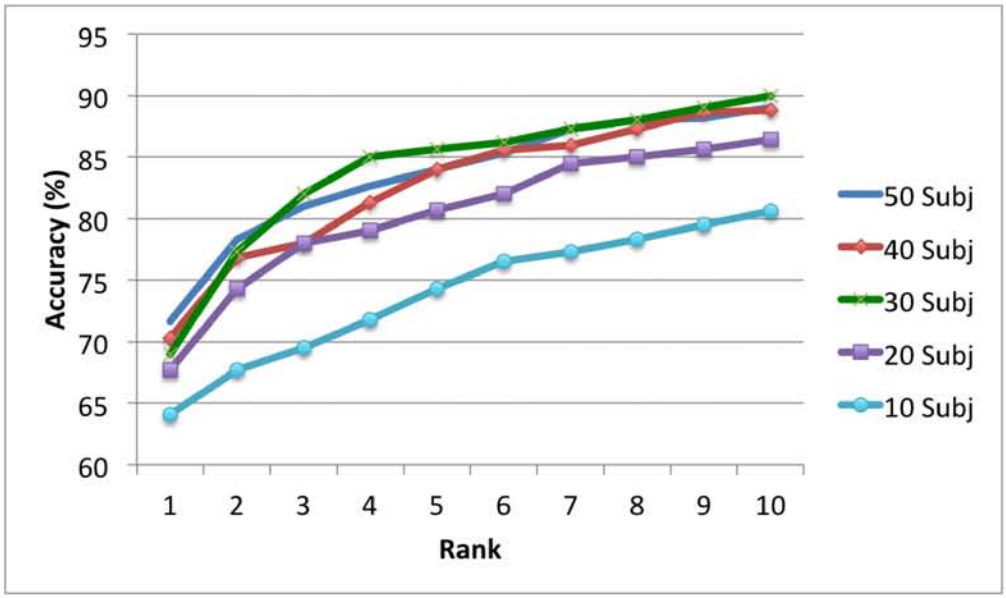


Figure 3.12: *Accuracy vs ranks test for FERET dataset*

subjects gets above 30, where noticeable improvement is negligible as rank increases.

Rank 1 identification result for ExtendedYale dataset is shown in Table 3.7. As expected, accuracy improves given more sample image per subject for target domain construction. But the performance saturates at around 74-75%. More sampling image past a certain amount is considered redundant. Subsequent future studies will be on the appropriate sample image that will maximize performance, yet at the same time, minimize redundancy. Only samples that are deemed to be able to extend the ranges of variations given an identity are stored. A suggestion is to incorporate Growing Neural Gas (GNG) into the system. More information in its application can be obtained from [20]. Due to recent success in recognition via non-metric similarity measure [112], the joint probability evaluator $r(x_1, x_2)$ can be used as a non-metric similarity measure for the GNG. In application, prototype faces are the faces used in domain template construction. GNG can be used to extract the topology and geometry of the space representing the faces, where the prototype faces will be associated to. This association with reference to GNG's constructed space can provide incremental learning without data redundancy.

3.7.4.3 Experiment on Larger Face Patches

Experiments in Section 3.7.4.2 only use image patch at the eye pair region. The use of more patches around the face can contribute more information that can be used to differentiate individuals. In this section, test is performed with a larger image patch and with higher dimension in terms of descriptor.

Table 3.6: *Identification result given target domain for FERET database*

No. of Subjects	Without mergence(%)	After mergence(%)	Accuracy gain(%)
10($\times$1)	59.3	64.6	8
10($\times$2)	59.1	66.3	12.2
10($\times$3)	59.6	65.8	10.4
10($\times$4)	60.8	65.2	7.1
10($\times$5)	59.0	65.2	10.5
20	59.3	68.2	14.9
30	60.5	70.2	16.1
40	60.2	71.4	18.6
50	59.2	72.7	22.8
80 from Pubfig	60.1	60.4	0.5
Dimension 400	60.3	80.5	33.5
ADMCLS [117]	-	86.4	-

Table 3.7: *Identification result given target domain for ExtendedYale dataset*

No. of samples per Subject	Without mergence(%)	After mergence(%)	Accuracy gain(%)
5	53.5	58.6	9.53
15	57.1	62.4	9.28
25	54.3	63.3	16.6
50	56.0	65.0	16.1
75	55.8	68.4	22.8
100	54.2	72.3	32.8
200	54.1	73.1	34.9
300	53.4	75.1	40.6
400	55.8	74.3	33.2
500	57.2	75.3	31.6
500+	54.3	74.6	37.4

Table 3.8: *Recognition test on FERET database*

Method	Accuracy
Eye Pair patch	80.5%
ADMCLS [117]	86.4%
Full Face Patch	84.3%

As described in Section 3.2, LBP is extracted from different patches of the images during preprocessing after face alignment is performed. Due to the usage of uniform LBP, histograms of 59 bins are obtained for the regions of all patches. Concatenating the histograms will produce a descriptor with a total dimension of 19175.

Unlike in Section 3.7.4.2 which uses 8000 images, 4000 images are selected from the Pubfig database to obtain the principal components. These principal basis will then reduce the overcomplete descriptor to 400 dimensions. The principal components are fixed and will not be changed throughout the recognition process, including the mean component of the PCA. Pubfig dataset will also be used as source domain for the joint probabilistic comparison to mold the baseline of face comparison.

For FERET test, the 200 subjects with controlled condition under different poses are used. Images with letter codes "ba, bb, bc . . . bk" are selected, where the pose ranges from -60 degree to 60 degree. Out of the 200 subjects, 150 are randomly selected for testing, where each subject only has one prototype image used for recognition that is randomly chosen as well. The remaining 50 subjects will be used for training, where they act as target domain. Table 3.8 shows the result. As can be observed, the method can perform reasonably under different poses. In previous test where the face region used that ranges from the eyes to nose produces an accuracy of 80.5%. Accuracy is improved to 84.3% when full face patch is included. Although ADMCLS by [117] produces a result of 86.4%, it should be emphasized that the method assumes a pose specific classifier with the pose ground truth known, whereas, for our method, no ground truth is assumed.

Face recognition test under different illumination and small pose change is performed using ExtendedYale dataset. This dataset consists of 26 subjects where each subject has an average of around 500 samples. Twenty subjects are randomly selected as test subjects, while the remaining 6 will be used as target domain. For the 20 test subjects, 32 testing and 32 prototype images are randomly selected for its repertoire of faces. The 32 prototype images will also be included as a target domain. The accuracy is 81.2%, which shows that the method performs reasonably well under different illumination and pose. It is demonstrated that higher number of target domain will further improve the accuracy up to a certain point, after which improvement slows down to a halt as shown in Section 3.7.4.2, which is due to

limited information despite huge number of samples.

3.7.5 Open Set Identification Test

More realistic open set uncontrolled dataset test is performed using Pubfig+LFW dataset. Comparison is also made between other well-established methods, which are the SRC gradient projection for sparse reconstruction (GPSR) [44], SRC Homothopy [91], Support vector machine (SVM) related methods and nearest neighbor method. GPSR is known for its high performance for open universe face identification, and SRC Homothopy is a faster version with high performance. SVM methods are known for their speed since a classifier is built for one identity for identification. This is an offline test.

3.7.5.1 Experimental Setup

For the test, 100 subjects from the Pubfig database are chosen at random. Thirty prototypes are randomly chosen for every subject.

Under joint probabilistic face method, apart from prototype and test image, source domain samples are also required to construct the template for recognition, which are obtained from identities not selected for testing. Target domain samples are also collected, where there is some overlap with the prototype, but not the test samples. There are three variants of the method, depending on whether prototype and target domain are used to merge with the source domain, which are termed as NPYT, YPNT and YPYT. Prototype is not merged for NPYT, contrary to YPNT, where prototype is used instead of target domain for mergence. For YPYT, both the prototype and the target domain are merged to the source domain.

For SRC method, the prototypes provide the necessary basis to construct the input sparsely and to minimize the residual of the reconstruction. For methods under SVM, for each identity, the positive and negative samples consist of its own 30 prototype samples and all the other samples respectively. With the randomly selected test images, an equal number of distractor images are included from LFW database to simulate an open universe problem. Distractor images are images where the identities are not known, thus, should be rejected. Distractors are obtained from LFW database that does not overlap with identities in the Pubfig database.

3.7.5.2 Result and Discussion

Results are visualized using the Precision-Recall (PR) curve and Receiver Operating Characteristic (ROC) curve as shown in Figure 3.13, which is suitable for open universe

recognition.

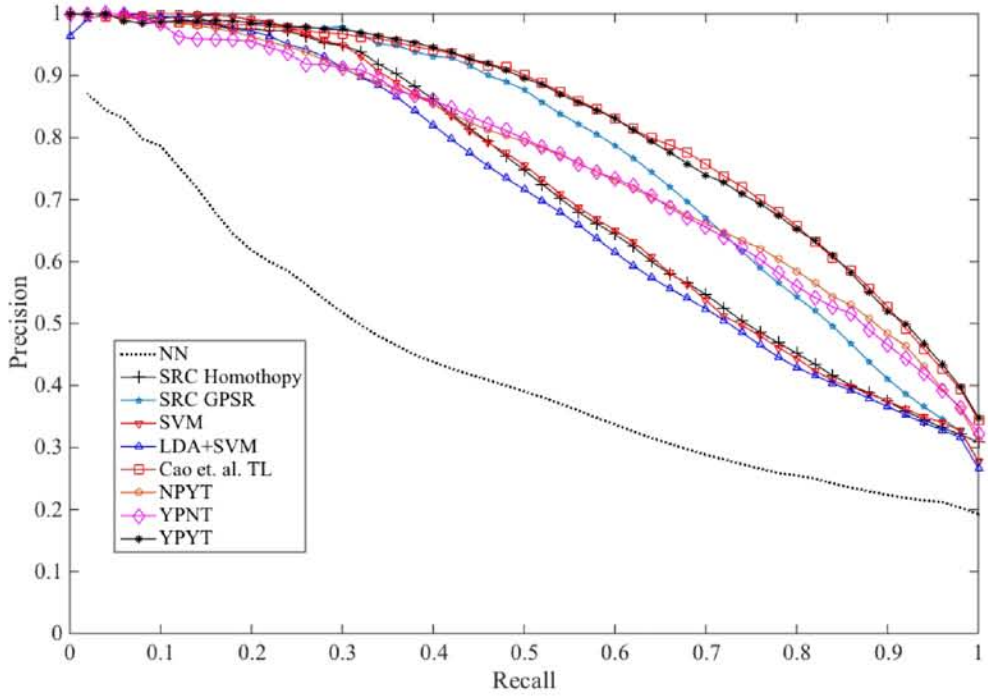
With only prototype samples or additional target domain molded into the template (NPYT and YPNT), joint probabilistic method can perform at the same level as SRC GPSR under higher recall. Under lower recall, SRC GPSR out-performs the joint probabilistic method. Yet, given the combination of both prototype and additional target domain (YPYT), the joint probabilistic method has a sharp increase in performance, exceeding the performance from the SRC GPSR. It also performs well on the ROC curve that shows how much the system can correctly recognize a person and that this person needs to be included in the database.

SRC does not work so well because of the lack of prototype samples for input reconstruction. As number of prototype samples increases, although accuracy will increase, computational load will get heavier. One can also resort to LASRC [102], which is a fast SRC method with high precision. Like wise, for SVM related methods, accuracy is expected to increase with higher number of training samples. But the downside is that it requires large amount of training data for every identity, which is the luxury we do not normally have in practical situation. Given these points, our method can achieve good performance and can support cold start.

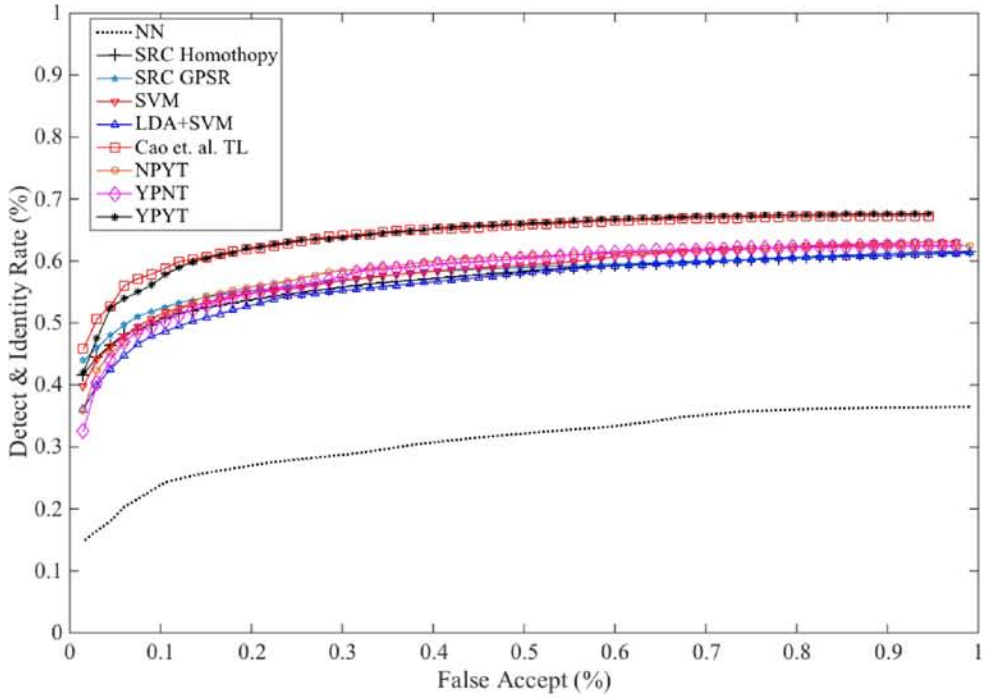
Precision drops below 90% after recall exceeds approximately 50%. Regardless, because high recall is not required for real time identification [5] given the number of input images due to streaming, therefore low precision at high recall is not of great concern. In terms or comparison between the original and the reformulated joint probabilistic face method, the same performance between applying transfer learning specified in [27] and ours is evidence that the reformulated method is fundamentally the same as the original. But [27] needs to handle all images during run-time, while our method merged the images' data before using it, which is much more efficient for execution and storage.

3.7.6 Real-time Indoor Test

Real-time uncontrolled indoor test is performed to experiment on how the recognition system fairs for normal everyday interaction in doors. The test can be divided into 2 types, which are, prototypes from the same domain and prototypes from different domain. The first type is when the subject's prototype image is captured in the same indoor environment, whereas the second type has the subject's prototype image captured in a different environment and time using a different image-capturing device. As we can yet to strictly quantify what is considered same environment or not, we will consider any images taken outside our test indoor setting as different environment. Examples of face captured in the



(a) *PR curve*



(b) *ROC curve*

Figure 3.13: *Identification test using Pubfig and LFW database*

Figure 3.14: *Image samples comparison between (above) indoor test environment and (below) samples obtained from different environment and time*

indoor environment, compared to that of a different environment are shown in Figure 3.14.

3.7.6.1 Experimental Setup

The test involves 13 individuals from Kubota laboratory (TMU)[1], where 11 are randomly chosen for testing, and the remaining 2 will be used as target domain. For every individual, streams of images on them acting naturally and uncooperatively are captured at a distance of not more than approximately 1.5 meters, to simulate individuals interacting with the robot partner. As the robot in the test is dealing with only 1 person at a time (since the purpose of the test is on the contribution of transfer learning towards accuracy improvement), T_E is set to 1 second for convenience. Further studies need to be done to find the effect of T_E through deploying the system in a crowded place where it needs to track and learn multiple faces at once. PF_{MAX} is set empirically at 0.8. In the experiment to choose PF_{MAX}, two groups of streamed photos with detected faces are created, where one group consists of a perfect stream, while the other consists of a disrupted stream (via cutting off portions of the stream or pasting erroneous portions to it). PF_{MAX} value is chosen based on how well it separates these two groups. Source domain is constructed using a subset of Pubfig and FEI database. Target domain, which is obtained from 2 remaining identities as described, are collected through spatio-temporal association but without similarity checking (meaning, all samples that passes the PF are collected). Spatio-temporal association is not performed for test identities to prevent samples in target domain from overlapping with

[1]See http://www.comp.sd.tmu.ac.jp/kubota-lab/hp/index_en.htm/

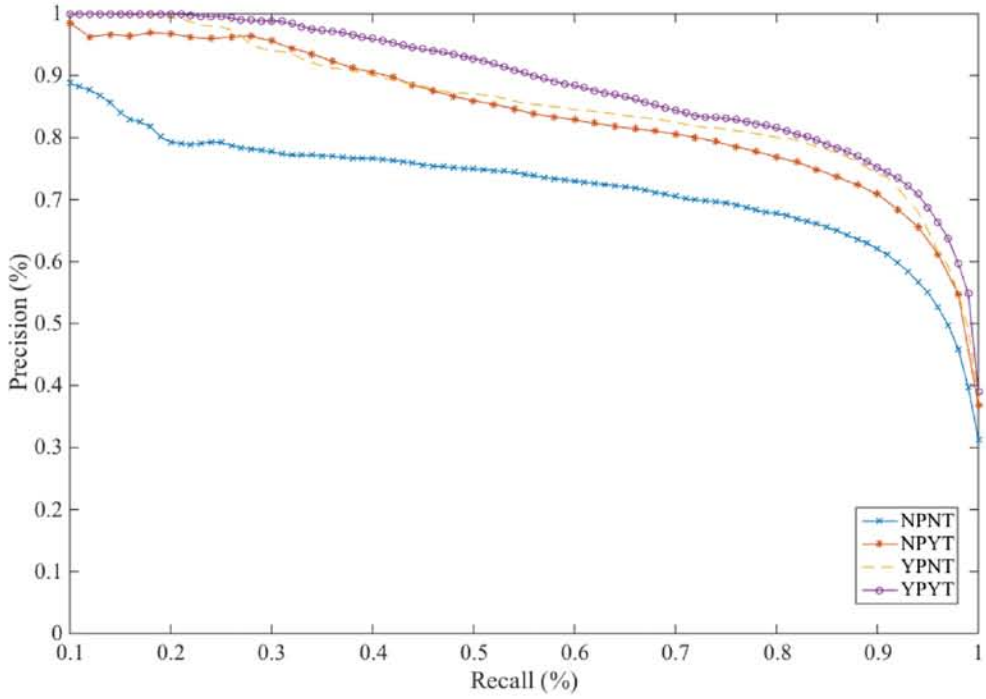
the identities for testing. We hypothesized that the improvement brought by the mergence target domain obtained from other identities not in the test can be also applied to target domain obtained from identities during probing. 500 distractor faces are randomly selected from the remaining faces from Pubfig and FEI, some members from our laboratory who do not contribute to have their face stream taken, and also from LFW database.

For test related to using prototypes from a different environment, each individual has 5 prototypes all of which differ in their environments. Like in Section 3.7.5, NPYT signifies using only the target domain to merge with the source domain, while for YPNT, prototype is merged with the prototype and not the target domain. YPYT means merging both the prototype and target domain to the source domain. NPNT means only source domain is used to construct the template, without mergence from the prototype and target domain.

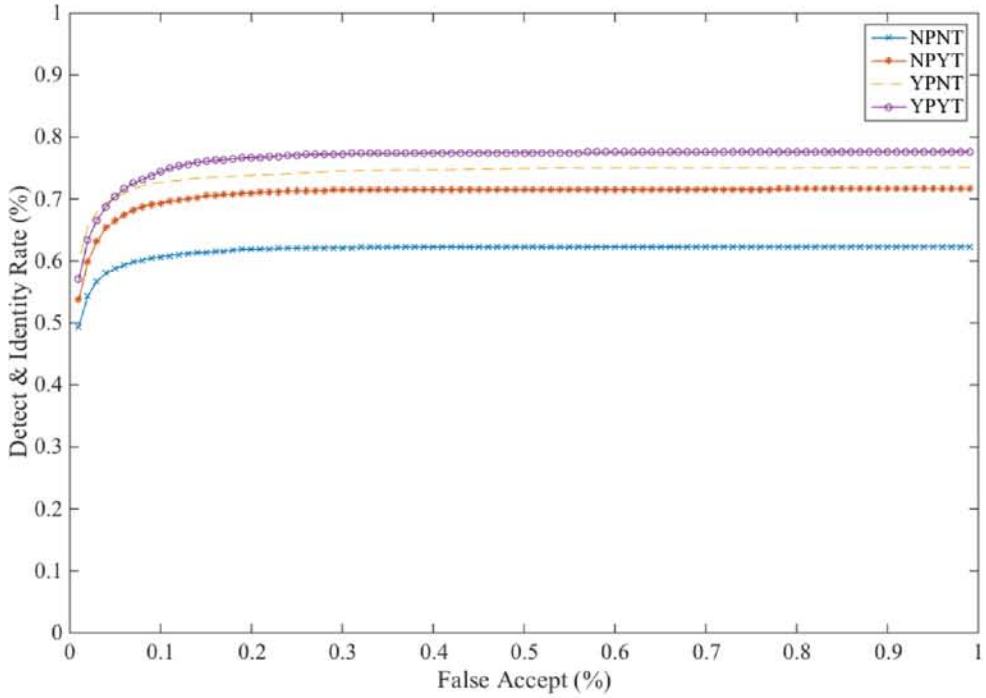
3.7.6.2 Result and Discussion

Result represented in PR and ROC curve are shown in Figure 3.15. As expected, with mergence, results improved significantly. NPNT only starts to rise to reasonable precision at very low recall. This can be interpreted as the unfamiliarity of the robot towards its own environment. Thus, we can impose stricter acceptance to limit recall in order to obtain higher precision. At higher recall, YPNT performs better than NPYT, although there are only 5 prototypes from each individual, compared to hundreds randomly chosen images from target domain individuals. A lot of target domain samples are clearly redundant. This is due to information having reached its contribution limit given target domain from the same environment, which is insufficient to transcend towards images from different environment. With the prototypes for mergence, since its environment has a wider scope, its coverage for faces is higher compared to only using images from the same environment. More studies need to perform in order to quantify the magnitude of coverage such that no redundancy occurs.

For test related to the use of prototypes of the same environment, every individual has only one prototype sample. Therefore, prototype cannot be merged with the source domain. Only target domain samples from other identities will be merged. The purpose of this test is to study the performance when the robot meets a new person who has his/her face registered on the spot. The result is shown in Figure 3.16. Performance is very good even with high recall. It can also be clearly observed that with target domain, precision will improve. Thus, on spot registration for recognition can perform well, even if no target domain or multiple prototypes are provided.

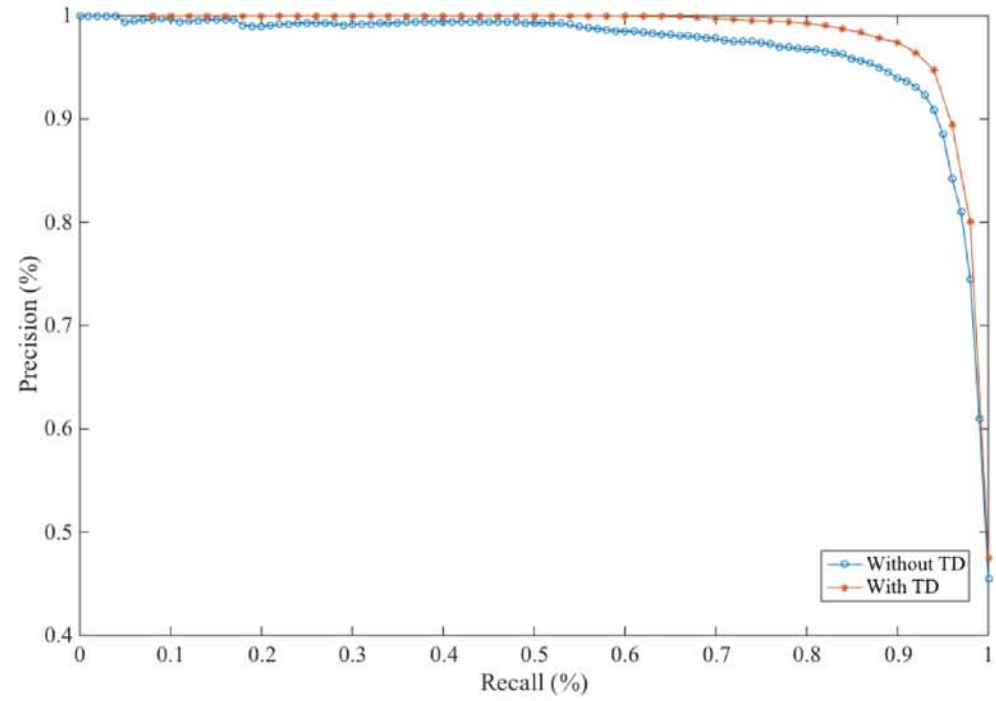


(a) *PR curve*

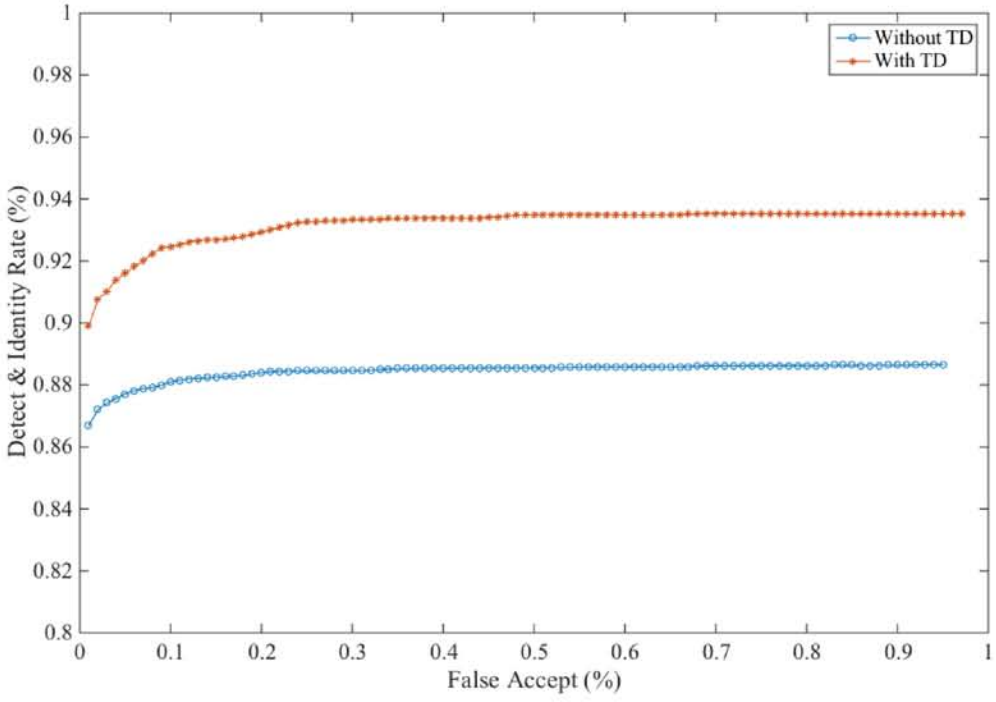


(b) *ROC curve*

Figure 3.15: *Test under different environmental setting*



(a) *PR curve*



(b) *ROC curve*

Figure 3.16: *Test under similar environmental setting*

3.7.7 Adaptive Learning Evaluation

In Section 3.6, it is mentioned that real time adaptive learning should be used to obtain the target domain for mergence, such that current environmental setting can be molded into the template for better recognition. As can be seen from previous tests in Figure 3.15 and 3.16, similar domain can help contribute in the improvement of accuracy, even though the domain samples are obtained from different identities.

Despite the improvement, test shows that a lot of target domain samples are redundant. Besides, performance might deteriorate given the distribution of intra-variation of the samples, confirmed by an independent test where target domain are obtained and accepted without condition. A closer study reveals that the trace of S_μ and S_ε diminishes, signifying a drop in dispersion. This is interpreted as a drop in invariance and rise in discrimination.

Images are collected and categorized based on spatio-temporal association as described in Section 3.6. To reduce redundancy and maintain coverage dispersion, threshold is introduced on the similarity measure to control the inflow of images to be used as target domain. Twelve identities are used as test subjects. One identity is used for real-time adaptive learning through spatio-temporal association to obtain samples as target domain. Result of accuracy improvement is shown in Figure 3.17. For the threshold, for example, "below -20" means that an image that has its comparison score with the previous image of below 20 will be accepted as a sample in target domain.

From the result, if no threshold is imposed, all samples from the image stream after spatio-temporal association are accepted. For every category size, it contributed the least improvement to accuracy. Threshold that is too high will also see less improvement due to rigidity.

In terms of number of samples per category, it can be observed that improvement will increase and saturate as number of samples increases. The implication is that only a small amount of samples is required when selecting samples for target domain. This is advantageous since storage size for robot partners is limited.

3.7.8 Computational Evaluation

The face recognition system module is written in Xcode 5.1 and OpenCV 2.4.9 for IOS, and is implemented on an iPhone 5s IOS 7.0.4. Other modules included in the iPhone are the speech recognition and utterance, servo control for robot motion and gesture recognition. During operation, face recognition module is run concurrently in a separate thread from other modules mentioned as a background process. Memory scratch pad is created specifically for face recognition module such that no memory allocation and deallocation

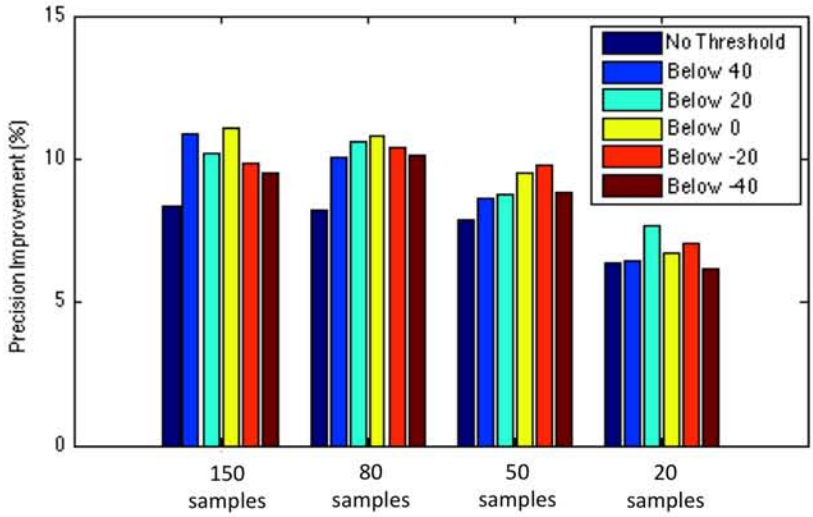


Figure 3.17: *Accuracy improvement due to similarity measure threshold and number of samples per category*

is performed during operation apart from program initialization. All mathematical operations are done in double precision number. Matrix operations are performed via functions provided by OpenCV.

The running time of the system are recorded and averaged. The result for real-time loop is shown in Table 3.9. Real-time loop consists of all the operations that will be constantly running, where each loop accommodates for each image captured. The total time is approximately 240ms. Since verification is performed independently, the time it took for this operation depends on the number of prototypes to be compared with. From Table 3.9, face detection takes up the most time, after which is dimension reduction. By concentrating on these operations, speed improvements can be made. One example is to replace the Flandmark for landmark detection with cascaded regression [26], which is shown to be quite fast and accurate. Speed can also be improved by using floating-point number, given that accuracy is not affected significantly, which is subject to further testing and optimization.

Apart from real-time loop, two more operations that need not be strictly run in real-time but which will affect the speed are the sample mergence (specified in Algorithm 1 and Figure 3.6) and EM algorithm (specified in Algorithm 2). Sample mergence occurs when the timestamp is overdue with enough samples in the category or that number of samples has already reached maximum limit. How much the mergence operation will influence real-time experience depends on similarity measure threshold and minimum number of samples per category. With 40 samples per category, the mergence operation takes 384ms to complete. On the other hand, EM algorithm is only performed when a certain number of categories

Table 3.9: *Consumed operation time for real-time operations*

Operation	Time(ms)
Face detection	79.5
Landmark detection	48.5
Face alignment	53.7
Dimension reduction	55.3
Verification (1 prototype)	0.268

are reached. For 8 iterations, which is more than enough to obtain S_μ and S_ε, it takes about 59.22 seconds to complete. As direct inversion is used for matrix division, in the future, this can be improved to further speed up the EM algorithm.

3.8 Concluding Remarks

Real-time face recognition for unconstrained condition is built, where the computational load is low enough to be programmed into the robot's onboard computer and to run with acceptable speed based on transfer learning and joint probabilistic method. To overcome the tradeoff between computational load and accuracy, as well as training data scarcity, it exploits the potential images captured in informationally structured space. Faces captured can be categorised not just into identities, but also conditions when the image is obtained (what type of camera, indoor or outdoor, day or night etc.).

Test performed in comparison with other established methods and also real-life indoor test shows promising results. Besides, adaptive learning test sheds light on the direction of future development. There is ample room for further improvement as well. Current landmark detection has too little landmarks, which pose a problem for alignment. Future work will employ the cascaded regression method [26], which can accurately pinpoint landmarks under milliseconds. The work can also be extended towards creating more descriptive face prior via Gaussian Process [88]. Apart from that, through communication, symbol grounding can help provide reinforcements for better recognition [143].

For those who reject face recognition system, the HIM should be replaced by something that they are comfortable with, where the selected approach should be capable of supporting tracking and discerning familiar people and strangers.

Chapter 4

Constraint Satisfaction Automated Planning in Service Provision

Automated planning tries to bind the devices available and execute them according to a generated plan such that they can maximize current QOLs.

The work presented in this section will deal with Automated Planner Module (APM), which deals with service composition and implementation through peripheral devices (For example, TV, sensor modules and weather web service) to achieve goals by the smart home, where command, query, and announcement are mediated between the human inhabitants and the smart home through a communication robot.

Service composition is performed through solving Constraint Satisfaction Problem (CSP), and implements them by appropriate devices connected to the smart home. But due to limitations of hard constraints from CSP, this section also extends it to weighted constraint satisfaction, where optimization and partial goal fulfillment can be achieved.

4.1 Planner Module

Figure 4.1 shows the flow chart of the planning and plans implementation process, which is central to the planning/implementation block of the Informationally Structured Space (ISS) framework. With the services that come from devices, planner will plan a sequence of actions to be implemented to fulfill goals.

Variables are instances to record knowledge (termed as knowledge variable) or act as a switch pertaining a corresponding object (termed as effect variable). An example of a knowledge variable is the variable that records room temperature. For effect variable, an example is a variable TV, which turns the TV on if it is of state 1.

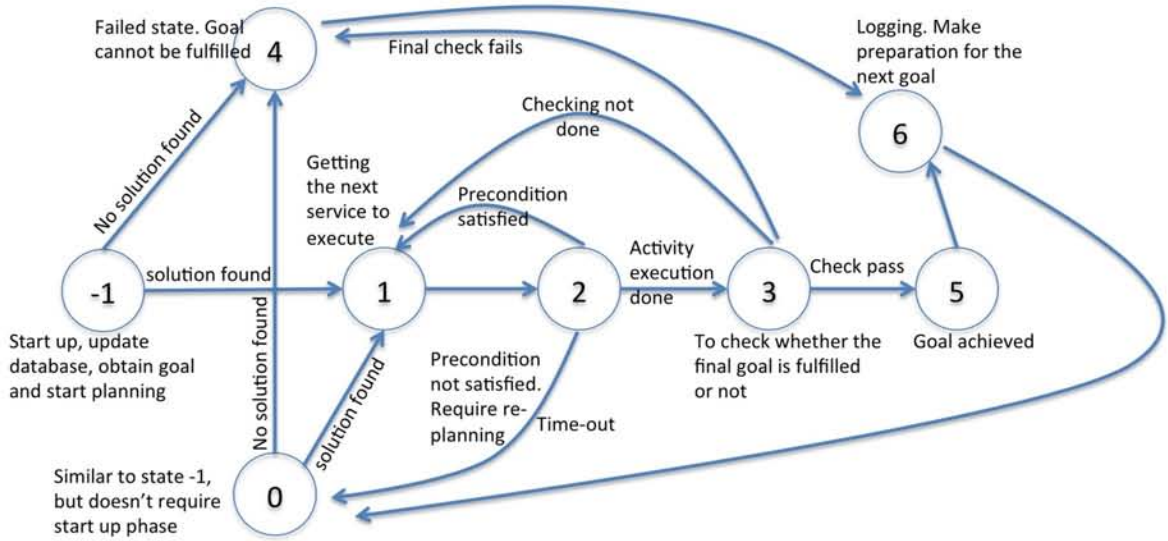


Figure 4.1: *Service composition and execution process flow*

An activity is a service that is being wrapped with appropriate parameters, effects and preconditions. For planning module, the word Activity instead of Service (as used during variable binding) to differentiate between service generation stage and planning stage. Activity's role is to manipulate the values of the variables, which have correspondence with the physical world associated during variable binding. An activity consists of an ID for the activity itself, parameters, pre-conditions and effects. ID acts as an identifier for the activity, and parameters are the required input for the activity to function. Pre-conditions are conditions that need to be true before the activity can be executed. Effects are the changes that the activity makes when the activity is finished without any glitches or unforeseen circumstances.

The following is the explanation for the process flow shown in Figure 4.1. The system starts at state -1, where initialization is performed and connection is set up. At this stage, it is required that variables are already bound and activities are ready. The system will restart at stage -1 if there are changes to the number of variables and services due to device connection or disconnection or failure. After initialization, goal is obtained and planning is implemented. If a sequence of activities is found that can fulfill the goal, the system will proceed to stage 1. Otherwise, it will proceed to stage 4, indicating a failure. Stage 1 deals with getting the next activity in sequence to be executed. After the next activity is selected, it will proceed to stage 2, which is orchestration of activities. Before execution occurs, it needs to check whether the precondition is fulfilled, upon which if not, it will proceed to stage 0 for re-planning. If the precondition is fulfilled, activity is executed by sending

command to the relevant devices. After the device finish executing the activity, the system will proceed to stage 3. Connection lost or device failure will bound to happen, which will delay or stop executing the activity altogether. Given this situation, stage 2 has a time-out threshold, upon which if the threshold is reached, the system will assume the current plan don't work, and proceed to stage 0. Although not implemented yet, this event can be logged to record number of failures a certain device faces. This can help determine the quality of service of the devices, such that it can be used as a weight when selecting services. Stage 3 checks for 2 things: whether there is any activity left for execution (if true, the system will proceed to stage 1 to get the next activity), and if there is none, it will check whether the goal is fulfilled or not. If the final goal is not fulfilled, then it will proceed to stage 4, otherwise, to stage 5, indicating a success. Given stage 4 or stage 5, the system will proceed to stage 6. Stage 6 records a log of past events, obtains a new goal, and proceeds to stage 0. Stage 0 is similar to stage -1, except that it does not need perform initialization.

4.2 Domain Description for Planning

Devices provide their function callbacks as a set of actions they are capable of running, and a set of variables associated with the actions. These actions are being wrapped up as service activities that are to be used by the planner. For example, the function call of switching channel of a TV is wrapped up as an activity that changes TV channel, with the pre-condition that the TV needs to be turned on. Nested activities can also occur, where different activities can be called under one activity, albeit indirectly. An example of nested activity is the activity of environmental checking, where activities such as lighting and temperature control activities are called. The purpose of nested activities will be explained in more detail in Section 4.5.1.

It is assumed that the pervasive block discovers the objects and their function callbacks as well as variables, and variable/activity updater module performs the wrapping up of activities of these function callbacks or creation of nested activities. Therefore, for explanation purposes, activity refers to these wrapped up function calls or nested activities. It is also being assumed that the variables contained in list of activities and goals are being declared under variable list filled up by the pervasive block and variable/activity updater block.

Domain description will be based on the work of [71]. We denote ϑ = variable set (list of variables). ϑ contains V variables, which consists of knowledge, effect and response variables confined by their own domain. Response variables represent information that can only be obtained from objects, information that comes from sources not within ϑ. During planning stage, response variables remain the same throughout all planning sequence and

represent an unknown value (thus, initialization constraint is imposed on them). They can take on whatever value to facilitate constraint satisfaction during planning that employs an optimistic approach (value taken on by response variables are considered true). Response variable usage will be made more evident in Section 4.5.1.

A state is a tuple of values to variables at a particular plan implementation sequence index t that is denoted as $X_t = (X_t^1, X_t^2 ... X_t^V)$ where $X_t^1, X_t^2 ... X_t^V \in \vartheta_t$, confined by their domains denoted by $D^1, D^2 ... D^V$. As there is a finite limit to the number of sequence per plan being planned called the planning horizon that is denoted as K, thus, $0 \leq t < K$. For the current work, domains of the variables remain unchanged over time.

α is the set of activities, where $a = (id(a), precond(a), effect(a)) \in \alpha$. $id(a)$ is the identifier of the activity. There is an additional activity in α that does nothing. It has no pre-conditions and effects, termed as Nop.

$precond(a)$ is the pre-condition that need to be met before the activity can be executed, such as the location being known as a pre-condition to implement the weather forecast web service. Precondition of an activity can be described as follows:

$$precond(a) ::= prop | precond(a) \wedge precond(a) | precond(a) \vee precond(a) | \neg precond(a) | precond(a) \rightarrow precond(a) \quad (4.1)$$

$$prop ::= var \bullet var | var \bullet val | (var \odot var) \bullet var | (var \odot var) \bullet val | Brel \quad (4.2)$$

where $var \in \vartheta$, val is a constant, $\odot \in \{+, -\}$ is a binary operator, $\bullet \in \{=, <, >, \neq, \leq, \geq\}$ is a relational operator, and $Brel$ is a Boolean relation.

$effect(a)$ is the changes that will be induced after the activity is completed. It emulates how the variables will change given the activity is run by its corresponding objects such that its logical formulation can be used to impose constraints on subsequent sequence of the plan for activity planning. It should be emphasized that the actual object manipulates variables during run-time after planning instead of the $effect(a)$ formulation (which is only used for planning). Effect of an activity can be formulated as or a combination of the following: $var_{t+1} = val$, $var_{t+1} = var_t$, $var_{t+1} = f(v_1, v_2)$ where $v_1, v_2 \in \vartheta_t$ or v_1, v_2 are constants, and f is the sum, subtraction and Boolean operation.

For simplification, when necessary, we will denote the above relations as $var_{t+1} = effects_t(a)$. This relation is read differently between the planner and the orchestrator. During planning, $var_{t+1} = effects_t(a)$ means the truth statement that: ($effects_t(a)$ includes

an effect towards the variable var) implies
$(var_{t+1} = effects_t(a))$ holds true.
On the contrary, for orchestrator, $var_{t+1} = effects_t(a)$ just indicates that the variable var is being modified according to $effects_t(a)$. Therefore, depending on whether planner or orchestrator is referred to, the correct interpretation has to be made.

Though this work only use the specified effects, more sophisticated effects, such as conditional effects, can be used as shown in [70, 69].

4.3 Planning as Constraint Satisfaction Problem

Given goals, which are represented as propositions, activity planning can be obtained to fulfill the goal by representing the problem as constraint satisfaction problem and solve it. A constraint satisfaction problem is a triple $CSP = \langle \chi, D, \zeta \rangle$, where χ is a set of variables, D is the set of domains of the variables in χ, and ζ is a set of constraints over χ. A solution to a CSP is an assignment of values to the variables in χ such that the values fall within D and all constraints in ζ are satisfied. In this work, D is unchanged throughout the activity sequence. It is considered determined when a goal is passed to the planning process flow in Figure 4.1, and will stay that way until the goal is achieved.

In terms of the planner, $\chi = \{X_1, X_2 ... X_K\} \cup \{A_1, A_2 ... A_{K-1}\} \cup R$, R is a set of response variables, and A_t is the chosen goal at sequence index t. Unlike X and A, variables in R remain the same throughout the planning sequence.

ζ consists of constraints imposed by a chosen activity at t from activity pre-conditions and effects, frame axioms, initial and final variable state and maintenance of achieved goal constraints. Initial variable state is just a constraint that dictates the values of all variables (obtained from object state module) before any planning. Final state constraint consists of the goal proposition that needs to hold at sequence index K. A solution to a CSP is an assignment of values to the variables in χ such that the values fall within D and all constraints in ζ are satisfied, which is normally obtained from backtracking methods [13].

Constraints from activity pre-conditions:
$(A_t = a) \rightarrow precond(a)$ where $\forall a \in \alpha$
Constraints from activity effects:
$(A_t = a) \rightarrow [(var_{t+1} = effects_t(a)) \wedge Fr]$ where $\forall a \in \alpha$
where Fr is the frame axiom constraint, which indicates that for every other variables var (excluding those from R) not affected by $effects_t(a)$, $var_{t+1} = var_t$.
Maintenance of achieved goal constraint:
This constraint dictates that whenever a goal is achieved at sequence index $\bar{t} < K$:

Table 4.1: *List of example activities*

Name	Precondition	Effect
gen_on	$-$	$gen := 1, gen\_Changed := 1$
gen_off	$-$	$gen := 0, gen\_Changed := 1$
$Light\_on$	$gen = 1$	$Light := 1, Light\_Changed := 1$
$Light\_off$	$-$	$Light := 0, Light\_Changed := 1$
Fan_on	$gen = 1$	$Fan := 1, Fan\_Changed := 1$
Fan_off	$-$	$Fan := 0, Fan\_Changed := 1$

$A_t = Nop$ where $\bar{t} < t < (K - 1)$
Maintenance of achieved goal constraint is just one of the goals specified in [70, 69], though it is sufficient for the current work.

Constraints are fed to a solver to obtain a sequence of A, which are the activities that need to be implemented to fulfill the given goals. Z3, which is a state of the art SMT solver, is used to obtain the plan [35]. The plan will be solved by continually increasing K until the constraints are satisfied.

As an example, consider the activities in Table 4.1, a goal of $Light = 1 \wedge gen = 1$, and initial state $gen = Light = Fan = 0$, the constraint graph for the planning problem is shown in Figure 4.2. The Precondition and Effect arrowed links of the figure shows which variables are associated with the activity. Inertia law arrow indicates the frame axiom, which means information of a variable will carry forward if that variable is not affected by the activity. To construct a plan to achieve the goal, a sequence of activities need to be found such that initial coniditon will transform to the goal after that sequence. This means, from the constraint graph in Figure 4.2, the solution is to decide, for every Activity selection block (indicated by the green square), the activity to run such that the final state will fulfill the goal. The constraint graph is created during run-time, when the activities and variables are known.

4.4 Enhancement Modifications

Previous section explains how planning is executed through solving CSP. But mere implementation without certain manipulation is inefficient and may introduce unwanted consequences although they are logically consistent as will be explained in Section 4.4.3.

4.4.1 Activity Search Space Reduction

There are a lot of times where goals only apply to certain cases, such as certain time or when certain events are triggered. An example is the goal to turn on appropriate lights

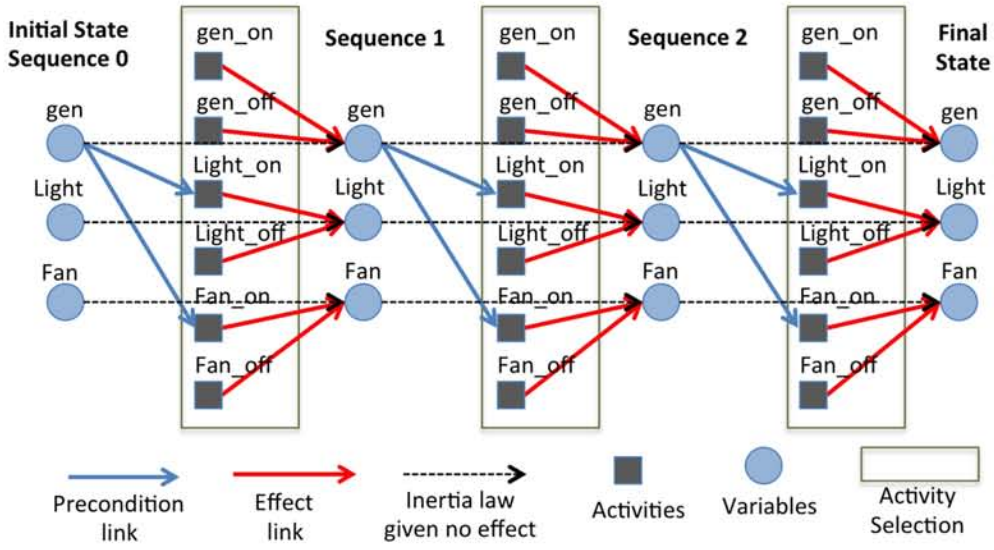


Figure 4.2: *Constraint graph for planning*

when the host is in the living room. This goal doesn't need to be considered when the host is elsewhere. Therefore, in the design of goals, every goal comes with implications $ImpGoal(Kv) \rightarrow Goal$, where if $ImpGoal(Kv)$ is met, then $Goal$ will be assigned as a goal for planning. $ImpGoal$ depends on current knowledge variable Kv it is using, but not on the effect variables. This is because knowledge variables cannot be manipulated by activities directly. To determine the truth value, satisfiability is performed on $ImpGoal$, where knowledge variables of $ImpGoal$ are fixed to the current value, and effect variables are set as free variables. Intuitively, it means, given the current knowledge variables, is there any assignment of effect variables that will make $ImpGoal$ true.

Only activities that are relevant to the given goals are selected to impose constraints. This work follows the same method used by [37]. Irrelevant activities to the goal are pruned out to reduce the search space. For every activity a_i, a list of other activities that has at least one effect that has the potential to satisfy the one of the pre-conditions of a_i are found. Backward action chain can then be computed. This chain of actions can be used to select the activities that are relevant to the imposed goal. An interesting alternative is to use dependency graph as proposed in this work [37], or to divide into effects ontology to shortlist relevant activities [33].

4.4.2 Activity Implementation and Re-planning

Given the planned activities (activity sequence $A_1 ... A_{K-1}$) that satisfy the given goals, the activity sequence will be implemented by the orchestrator, which distributes tasks to

relevant objects.

Before an activity is being carried out, orchestrator will check whether its pre-condition $precond(a)$ is satisfied. If it does, orchestrator will call the relevant object to implement the activity. Planner and orchestrator only know about an activity's pre-condition and effect, thus, whatever the object does is a blackbox to them. When the object has completed its required task, it will (or expected to) update the variable state according to that specified by its effect $effect(a)$, after which the subsequent activity of the sequence will be implemented. Objects connected to the pervasive block, which is responsible for service discovery, perform actual implementations. Pervasive block selects the appropriate services based on criteria like the Quality of Service (QoS) or past preferences and settings from repository log. This work assumes fully functional and well-behaved services as pervasive block is not the main concern.

During planning, information of the environment is incomplete. The planner needs to work through this uncertainty to come out with a viable plan. Planning through solving constraint satisfaction problem as shown poses an optimistic solution. This means that under incomplete information, the planner will fill in missing information, treating it as if the information has already been pre-specified as such, just so that the goal can be fulfilled. Due to contextual changes or feedback that differs from the expectation of the optimistic planner, pre-conditions might not be met in the midst of implementation. Under this situation, planner will use the current variable state to re-plan a new activity sequence.

4.4.3 Optimistic Planning Problem

Incomplete information can be handled through optimistic planning and re-planning. This is likened to humans who follow a plan with a certain expectation to reach a goal, while at the same time, collecting more information and re-evaluate their plans when necessary.

But due to optimistic planning considering that its assumed values to be true, the plan will be devised around this assumption. This can have negative consequences, even with re-planning, when the order of activity sequence is a requirement and that there are conspicuous activities that depend on a certain decision that can only be made during run-time. Conspicuous activities mean those activities that can be observed by human, such as, turn on of lights and robot query. Counter examples are activities such as obtaining user location, weather information and schedule updating.

An example is when the goal is to either turn on the TV or organize a game depending on what the user wants upon query. Since the user preferred activity could only be known during run-time through, for example, query through robot, during planning phase, an

activity will be assumed just to fulfill the constraints. Assuming the user would like to play a game, but the planner assumes he/she prefers to watch TV. Since the planning is optimistic, the planner might turn on the TV first before querying the user what he/she wants, because it considers its assumption to be true that the user wants to watch TV. Although re-planning can get the user what they want, but the switching on of TV is redundant and is a peculiarity. Besides, identifying the actions that have the prospect of satisfying propositions induced by the goal and, entailing relevant conspicuous effects [37] cannot solve this problem, as the wrong conspicuous activities are equally probable due to insufficient information.

One can alleviate the problem by introducing more pre-conditions to the conspicuous activities to ensure proper ordering – Turning on of TV or game preparation only after query is made. But this method beats the purpose of loose coupling and late binding of the SOA. Activities become non-reusable.

For the introduction of activation flag, every conspicuous activity can be implemented only when its corresponding activation flag is true, and set it back to false when the activity is completed. Activation flag is set by the services that require it – Depending on the query result, it will set the corresponding activation flag for turning on of TV or game preparation. The conspicuous activity uses the activation flag as a pre-condition. The downside is that one needs to know what activities are likely to be involved. This approach requires that the class of activities be known for conspicuous activities – For turning on of TV and asking about user preferred channel, these activities fall under TV Watching. This issue can be alleviated through the use of knowledge ontology, and will be subject of future work.

For imposition of a sequence of new goals, activities that can impose new goals are created, such that by fulfilling this sequence of goals, the main goal can be achieved – The query activity will impose a new goal, which is to fulfill the needs of the user's preferences. The advantage is that it is faster than activation flag approach because the goal is not as complex since it can be introduced on the go while the preceding goal is achieved. The downside is that one needs to know the sequence of goals, and that this can affect reusability of activities. Therefore, this approach is well suited for plans that are tedious but specific.

4.5 Case Studies

4.5.1 Design of Variables and Activities

Preliminary design of variables is based on the work of [71], where they consist of knowledge, effect and response variables.

Knowledge variable records a piece of information that can be referred to in the

future. An example is a variable recording the location of the human inhabitant. Every knowledge variable comes with a complementary Boolean variable that indicates whether the knowledge variable is updated or not (Intuitively, it indicates whether the situation referred by the variable is known or not). The following is an example activity to obtain user location:

$GetUserLocation:$

$Precond: UserLocation_{Known} = false$

$Effect: Userlocation = Userlocation_{res},$

$\quad Userlocation_{Known} = true$

The activity's pre-condition indicates that the activity can only be implemented if the user location is unknown, which is indicated by a false in Boolean variable $UserLocation_{Known}$. Once the user location is obtained (via transferring information from response variable $Userlocation_{res}$ to knowledge variable $Userlocation$), $UserLocation_{Known}$ is set to true to indicate the update of knowledge. Pre-condition to check whether a variable is known or not before getting values from response variables is important to prevent continual reading, which might cause unending planning/implementation loop under certain circumstances.

Effect variable records changes that need to be made to or is made by the corresponding object. An example is a variable where the value stored is used to change TV channels. Likewise, there is a complementary Boolean variable that indicates whether the original effect variable is being changed or not, and can be illustrated by the following example:

$LightON:$

$Effect: Light = 1, Light\_Changed = true$

The example shows that when the light is being switched on, the complementary Boolean variable is set to be true to indicate change.

An alternative to using complementary variables is to use time duration to determine the state of knowledge. This can be an interesting approach for study in the future. For subsequent activity statement examples, the complementary flags are not explicitly written down for simplicity.

Response variables represent information that can only be obtained from objects during runtime. Intuitively, they are the information from the outside world. They should never be used within pre-conditions as they are meant to be unknown until implemented by the corresponding object. Therefore, no complementary variables are associated with them.

Apart from the three types of variables, for this work, two more types are introduced.

They are the activity activation flag and activity completion flag. Activation flag determines whether a conspicuous activity can be run or not by placing itself as a pre-condition as shown in the following example:
$TV.ON:$
$Precond: TV\_Act = true$
$Effect: TV = 1, TV\_Changed = true$

The example shows that the conspicuous activity $TV.ON$ (responsible for turning on the TV) will only be implemented when the activation flag TV_Act is true.

Activity completion flag indicates whether an activity (or nested activity) has completed or not. The following example shows such flag:
$WakeUpCheck:$
$Precond: (SleepTime > 10) \rightarrow (Alarm = 1)$
$Effect: WakeUpCheck_{FLG} = true$

$WakeUpCheck$ activity checks whether the activities of checking on the person and waking him/her up if necessary are fulfilled or not (Note that this is different from an activity which actually wakes up the person, which is performed by $Alarm$). Activity completion flag $WakeUpCheck_{FLG}$ will be set to true if the person is woken up given that he/she sleeps more than 10 hours (Note that it will also be set to true if the person sleeps less than or equal to 10 hours and no actions are taken by the smart home via ISS). Completion flag is a useful tool to introduce goals to the planner.

Nested activities are defined by more than one activity statement. This may be due to input that will affect the plans, which will influence its finishing state, or that there are multiple output possibilities, or that it requires other activities to be implemented before it can be completed. An example is shown below:
$EnsureUPactivityRun1.SS$
$Precond: UPactivity = watchTV$
$Effect: TV\_Act = true$

$EnsureUPactivityRun2.SS$
$Precond: UPactivity = readNews$
$Effect: NewsService_{Act} = true$

$EnsureUPactivityRun.SE$
$Precond:$

$((UPactivity = watchTV) \rightarrow (TVChannel = UPChannel)) \wedge ((UPactivity = readNews) \rightarrow readNews_{FLG})$
$Effect : EnsureUPactivityRun_{FLG} = true$

Activity $EnsureUPactivityRun$ will check that the ISS has performed the necessary task to accommodate the needs of the user's preferred activity, $UPactivity$. $EnsureUPactivityRun$ is divided into three, where the first and second statement determine which activity should be performed based on user preferences, and the last statement is to check whether the appropriate tasks are implemented or not. It should be emphasized that $EnsureUPactivityRun$ does not dictate which activity should be run next, but just to activate the necessary activities to deal with activity ordering issue as discussed previously (Note that the approach used in this example is activation flag method).

A type of activity that requires special attention is the activity that deals with general information relay. Examples of such activities are the robot query and robot object fetching. Robot query activity relays information depending on context from the human to the ISS and vice versa (Example, asking their preferences, TV channel of their choosing). Robot object fetching activity fetches and moves arbitrary objects based on requirements, and by moving it around, it needs to change the location information of the attended object. Both of these examples deal with arbitrary objects. One can design or generate individual activities for every possible queries (for robot query) and fetch-able objects (for robot object fetching). We termed this as individual device activity. If the number of queries/fetch-able objects increases as well as the number of attending robots, number of activities will grow rapidly. Another approach is to create a template for query/fetch-able objects, which acts as a mediator between the robots and the query/fetch-able objects. We will use the query example for explanation since this work relies heavily on it.

Robot query is used to obtain information from user like their preferred activities and whether they want to take a shower or not, and to relay information to the user like announcing the need to bring an umbrella and unheard messages. To use individual device activity, and let's say there are two robots that are able to support the queries, for each robot and each query type, an activity needs to be generated to pass information to the relevant variables. Another approach is to use query template. The template itself is one activity, and there is one template for each robot. The type of query will be passed to the template. Depending on the query type, it will pass the returned information to the appropriate variable. An example is shown below:
$Robot1Query.SS$
$Precond : RobotSelect = Robot1, \neg Query = Null$

$Effect : QueryResult = QueryResult_{res}$

$AskUPActivity.SE$
$Precond : Query = AskUPActivity$
$Effect : UPactivity = QueryResult, Query = Null$

Activity $Robot1Query.SS$ is a query template for robot 1. It does not concern itself with the type of query being made, but only to signal the robot to implement the query. It is the robot's responsibility to check the type of query, which it will implement a suitable program for it. Nested activity $AskUPActivity$ set the query type via variable $Query$, and waits for response from the query template.

4.5.2 Smart home ISS Setup

This section will explain the setup for case study purposes. ISS provides the information structure of the smart home. In order to ensure the wellbeing of its inhabitants, the ISS needs to perform information collection and distribute tasks to relevant objects to achieve certain goals. These goals come from human command, event trigger, scheduled triggering and imposed by another activity. Goal that comes from human command is quite self-explanatory, which is a request from the human (such as requesting to prepare bath water). Event triggered goals are goals that are imposed when an event occurs, like an event when the human comes home or emergency events. Schedule triggered goals will be imposed given a pre-specified time or interval. Examples of such goals are goals that monitor the human's sleeping state, and goals that maintain a desirable living environment. It is a special case of event-triggered goal, but it is emphasized due to the special role it played for the ISS. Lastly, it is the goal imposed by the activity. This is important when dealing with goal imposition to handle activity-ordering issue.

For the evaluation, the ISS is set up with 24 connected devices. There are a total of more than 55 activities (Nested activities are considered as one) wrapped up from the atomic services of the connected objects for implementation. Relevant activities with their preconditions and effects are shown in Table 4.2 to 4.4. Out of the 24 devices, two robots, weather web service and news web service are physically implemented. The rest of the devices are run via individually designed emulators. The planner and orchestrator is run using 2.5 GHz Intel Core i5 computer with 4 GB RAM.

As one of the important elements in this research, robot partner plays important role in conducting communication with human. However, we developed robot partner not only

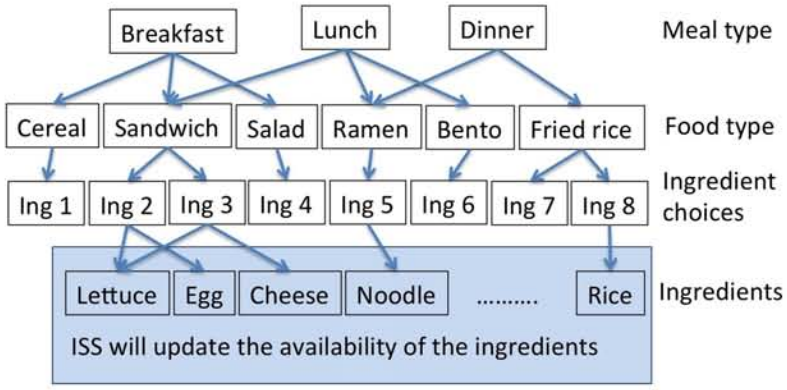


Figure 4.3: *Food ontology*

for direct communication with human, but also as a medium between human and server that control the entire system. We are using iPhone as the smartphone for robot partner called iPhonoid [128, 23]. For this work, the robot is to perform query and also to make announcements.

This work also includes a food database, which stores the ontology of food as shown in Figure 4.3. This structure can be either obtained from previous eating habits or from suggestion. At the root of the structure is a list of ingredients, which contains the information of ingredient availability. ISS will update this information assuming every consumerable products are tagged with either a barcode or RFID, or that inference can be made regarding its availability. ISS can use the food database to search and suggest the appropriate type of food based on time and availability to the user. The concept resembles that of web service discovery in [69], where backtracking is performed until the correct search result is obtained. From [69], backtracking is performed by the orchestrator when pre-condition is not met or the returned value has errors. But for this work, backtracking is part of a nested activity. Either way is just a matter of preferences.

4.5.3 Selected Cases

This section will present some selected cases that will be used for discussion in terms of subsequent development and implications. Planning and implementation for all cases are shown in Appendix A. Short description of the cases are presented respectively. For clarity, in Appendix A, when showing the actual planning and implementation in the subsequent sections, the sequence of activities under the bold **Planning** (or **Re-planning**) is the plan devised by the CSP (or the alternative plan devised after the previous plan is unsuccessfully implemented), with its processing time shown. Symbol ⇒ means "the sub-

sequent planned activity". For example, $A \Rightarrow B \Rightarrow C$ means the first planned activity is A, followed by B, and then C. It should be noted that these are just planned activity sequence before actual implementation. Under the bold **Implementation**, only the activity, which does not have its pre-condition met, is shown (which will lead to re-planning). For clarity, some activities have the same name but with different ending that is either $.SS$ (service starting) or $.SE$ (service ending), which means they are nested activities and should be considered as one activity. An example is $LeisureCheck$, which tries to ensure the user has performed necessary leisure activities, or else a log will be recorded, which is regarded as the goal. $LeisureCheck.SS$ will set the necessary activation flag such as activity to query on user preferred activity (in the case where user preferred activity is already known, $LeisureCheck.SS$ will not be run since no query is required). After proper preparation is made, $LeisureCheck.SE$ will run to set the activity completion flag $Leisure_{FLG}$ that indicates leisure activity preparation has been made. As explained, not all parts of a nested activity need to be executed. Depending on the context, only the appropriate parts are implemented, which are derived from solving the CSP. Information of activity description is shown in Table 4.2 to 4.4, where activation flag and query template are used. Modifications goal imposition approach and individual device query method is described in the cases.

Table 4.2: *Activity description 1*

Activity	Precondition	Effects
$TV\_on.SS$	$TV_{ACT} = 1$	$TV = 1, TV_{Changed} = 1, TV_{ACT} = 0$
$TV\_off.SS$	$TV_{ACT} = 1$	$TV = 0, TV_{Changed} = 1, TV_{ACT} = 0$
$Light\_on.SS$	$Light_{ACT} = 1$	$Light = 1, Light_{Changed} = 1, Light_{ACT} = 0$
$Light\_off.SS$	$Light_{ACT} = 1$	$Light = 0, Light_{Changed} = 1, Light_{ACT} = 0$
$Window\_op.SS$	$Window_{ACT} = 1$	$Window = 1, Window_{Changed} = 1, Window_{ACT} = 0$
$Window\_cl.SS$	$Window_{ACT} = 1$	$Window = 0, Window_{Changed} = 1, Window_{ACT} = 0$
$Clothesline\_op.SS$	$Clothesline_{ACT} = 1, Weather_{Known} = 1,$ $\neg(Weather = rain)$	$Clothesline = 1, Clothesline_{Changed} = 1,$ $Clothesline_{ACT} = 0$
$Clothesline\_cl.SS$	$Clothesline_{ACT} = 1, Weather_{Known} = 1,$ $Weather = rain$	$Clothesline = 0, Clothesline_{Changed} = 1,$ $Clothesline_{ACT} = 0$
$AirCon\_on.SS$	$AirCon_{ACT} = 1$	$AirCon = 1, AirCon_{Changed} = 1, AirCon_{ACT} = 0$
$AirCon\_off.SS$	$AirCon_{ACT} = 1$	$AirCon = 0, AirCon_{Changed} = 1, AirCon_{ACT} = 0$
$Fan\_on.SS$	$Fan_{ACT} = 1$	$Fan = 1, Fan_{Changed} = 1, Fan_{ACT} = 0$
$Fan\_off.SS$	$Fan_{ACT} = 1$	$Fan = 0, Fan_{Changed} = 1, Fan_{ACT} = 0$
$AlarmService.SS$	$AlarmService_{ACT} = 1$	$AlarmService_{FLG} = 1, AlarmService_{ACT} = 0$
$LightService.SS$	$LightService_{ACT} = 1$	$Light_{ACT} = 1, LightService_{ACT} = 0$
$LightService.SE$	$Light, Light_{Changed}$	$LightService_{FLG} = 1$
$WindowService.SS$	$WindowService_{ACT} = 1, Weather_{Known} = 1$	$Window_{ACT} = 1, WindowSerice_{ACT} = 0$
$WindowService.SE$	$(Weather = rain) \rightarrow (Window = 1,$ $Window_{Changed} = 1), Weather_{Known} = 1$	$WindowService_{FLG} = 1$
$Robot1Query.SS$	$Query_{ACT} = 1, Query_{Changed} = 1,$ $RobotSelect = 1, RobotSelect_{Known} = 1$	$QueryResult = QueryResult_{res},$ $QueryResult_{Known} = 1, Query_{ACT} = 0$
$Robot2Query.SS$	$Query_{ACT} = 1, Query_{Changed} = 1,$ $RobotSelect = 2, RobotSelect_{Known} = 1$	$QueryResult = QueryResult_{res},$ $QueryResult_{Known} = 1, Query_{ACT} = 0$
$AskUPActivity.SS$	$AskUPActivity_{ACT} = 1$	$Query_{Changed} = 1, AskUPActivity_{ACT} = 0$
$AskUPActivity.SE$	$QueryResult_{Known} = 1, Query = AskActivity$	$UPactivity = QueryResult, UPactivity_{Known} =$ $1, Query = NaN, QueryResult_{Known} = 0$
$AskUPChannel.SS$	$AskUPChannel_{ACT} = 1$	$Query_{ACT} = 1, Query = AskUPChannel,$ $Query_{Changed} = 1, AskUPChannel_{ACT} = 0$
$AskUPChannel.SE$	$QueryResult_{Known} = 1, Query = AskUPChannel$	$UPChannel = QueryResult,$ $UPChannel_{Known} = 1, Query =$ $NaN, QueryResult_{Known} = 0$
$AskNewsCat.SS$	$AskNewsCat_{ACT} = 1$	$Query_{ACT} = 1, Query = AskNewsCat,$ $Query_{Changed} = 1, AskNewsCat_{ACT} = 0$
$AskNewsCat.SE$	$QueryResult_{Known} = 1, Query = AskNewsCat$	$UPnewsCat = QueryResult,$ $UPnewsCat_{Known} = 1, Query = NaN,$ $QueryResult_{Known} = 0$
$AskShower.SS$	$AskShower_{ACT} = 1$	$Query_{ACT} = 1, Query = AskShower,$ $Query_{Changed} = 1, AskShower_{ACT} = 0$
$AskShower.SE$	$QueryResult_{Known} = 1, Query = AskShower$	$UPactivity = QueryResult, UPactivity_{Known} =$ $1, Query = NaN, QueryResult_{Known} = 0$
$MsgReportService.SS$	$MsgReportService_{ACT} = 1, Messages_{Known} =$ $1,$ $RobotSelect_{Known} = 1$	$Query_{ACT} = 1, Query = reportMsg,$ $Query_{Changed} = 1, MsgReportService_{ACT} = 0$

Table 4.3: *Activity description 2*

Activity	Precondition	Effects
$MsgReportService.SE$	$QueryResult_{Known} = 1, Query = reportMsg$	$MsgReportService_{FLG} = 1, Query = NaN,$ $QueryResult_{Known} = 0$
$ReadNews.SS$	$ReadNews_{ACT} = 1, UPnewsCat_{Known} = 1,$ $RobotSelect_{Known} = 1$	$Query_{ACT} = 1, Query = reportNews,$ $Query_{Changed} = 1, ReadNews_{ACT} = 0$
$ReadNews.SE$	$QueryResult_{Known} = 1, Query = reportNews$	$ReadNews_{FLG} = 1, Query = NaN,$ $QueryResult_{Known} = 0$
$WeatherWarn.SS$	$WeatherWarn_{ACT} = 1, Weather_{Known} = 1,$ $RobotSelect_{Known} = 1$	$Query_{ACT} = 1, Query = WeatherWarn,$ $Query_{Changed} = 1, WeatherWarn_{ACT} = 0$
$WeatherWarn.SE$	$QueryResult_{Known} = 1, Query = WeatherWarn$	$WeatherWarn_{FLG} = 1, Query = NaN,$ $QueryResult_{Known} = 0$
$ReadFoodSuggestion.SS$	$ReadFoodSuggestion_{ACT} = 1,$ $FoodIngdAvail_{Known} = 1, FoodIngdAvail = 1,$ $RobotSelect_{Known} = 1$	$Query_{ACT} = 1, Query = SuggestFood,$ $Query_{Changed} = 1, ReadFoodSuggestion_{ACT} = 0$
$ReadFoodSuggestion.SE$	$QueryResult_{Known} = 1, Query = SuggestFood$	$ReadFoodSuggestion_{FLG} = 1, Query = NaN,$ $QueryResult_{Known} = 0$
$FoodSuggest.SS$	$FoodSuggest_{ACT} = 1, TimeCat_{Known} = 1,$ $MealType_{Known} = 1$	$ReadFoodSuggestion_{ACT} = 1, FoodSuggest_{ACT} = 0$
$FoodSuggest.SE$	$ReadFoodSuggestion_{FLG} = 0$	$FoodSuggest_{FLG} = 1$
$RainyDayCheck.SS$	$RainyDayCheck_{ACT} = 1$	$WindowService_{ACT} = 1, Clothesline_{ACT} = 1,$ $RainyDayCheck_{ACT} = 0$
$RainyDayCheck.SE$	$WindowService_{FLG} = 1, Weather_{Known} = 1,$ $(Weather = rain) \rightarrow (\neg Clothesline,$ $Clothesline_{Changed} = 1)$	$RainyDayCheck_{FLG} = 1$
$EnvironmentCheck.SS$	$EnvironmentCheck_{ACT} = 1$	$WindowService_{ACT} = 1, LightService_{ACT} = 1,$ $TempControlService_{ACT} = 1,$ $EnvironmentCheck_{ACT} = 0$
$EnvironmentCheck.SE$	$WindowService_{FLG} = 1, LightService_{FLG} = 1,$ $TempControlService_{FLG} = 1$	$EnvironmentCheck_{FLG} = 1$
$LeisureCheck.SS$	$LeisureCheck_{ACT} = 1$	$AskUPActivity_{ACT} = 1, LeisureCheck_{ACT} = 0$
$LeisureCheck.SE$	$EnsureUPactivityRun_{FLG} = 1, (UPactivity = watchTV \vee UPactivity = readNews \vee UPactivity = goOut)$	$Leisure_{FLG} = 1$
$BackHomePrepService.SS$	$BackHomePrepService_{ACT} = 1$	$AskShower_{ACT} = 1, MsgReportService_{ACT} = 1,$ $BackHomePrepService_{ACT} = 0$
$BackHomePrepService.SE$	$EnsureUPactivityRun_{FLG} = 1,$ $MsgReportService_{FLG} = 1, UPactivity_{Known} = 1$	$BackHomePrepService_{FLG} = 1$
$TempControlService1.SS$	$TempControlService_{ACT} = 1,$ $RoomTemp_{Known} = 1 AirCon = 1$	$TempControlService_{FLG} = 1,$ $TempControlService_{ACT} = 0$
$TempControlService2.SS$	$TempControlService_{ACT} = 1,$ $RoomTemp_{Known} = 1, \neg AirCon\ Fan = 1$	$TempControlService_{FLG} = 1,$ $TempControlService_{ACT} = 0$
$TempControlService3.SS$	$TempControlService_{ACT} = 1,$ $RoomTemp_{Known} = 1, \neg AirCon, \neg Fan$	$Fan_{ACT} = 1,$ $TempControlService_{ACT} = 0$
$TempControlService3.SE$	$Fan = 1, Fan_{Changed} = 1$	$TempControlService_{FLG} = 1$
$PrepBathService.SS$	$PrepBathService_{ACT} = 1$	$PrepBathService_{FLG} = 1,$ $PrepBathService_{ACT} = 0$

Table 4.4: *Activity description 3*

Activity	Precondition	Effects
$EnsureUPactivityRun1.SS$	$UPactivity_{Known} = 1, UPactivity = watchTV$	$TV_{ACT} = 1, AskUPChannel_{ACT} = 1,$ $Light_{ACT} = 1$
$EnsureUPactivityRun2.SS$	$UPactivity_{Known} = 1, UPactivity = readNews$	$AskNewsCat_{ACT} = 1, ReadNews_{ACT} = 1$
$EnsureUPactivityRun3.SS$	$UPactivity_{Known} = 1, UPactivity = goOut$	$WeatherWarn_{ACT} = 1$
$EnsureUPactivityRun4.SS$	$UPactivity_{Known} = 1, UPactivity = Bath$	$PrepBathService_{ACT} = 1$
$EnsureUPactivityRun5.SS$	$UPactivity_{Known} = 1, UPactivity = PrepareFood$	$FoodSuggest_{ACT} = 1$
$EnsureUPactivityRun.SE$	$UPactivity_{Known} = 1,$ $UPactivity = 1 \rightarrow (TVChannel = UPChannel,$ $TVChannel_{Changed} = 1, Light = 1,$ $Light_{Changed} = 1),$ $UPactivity = 2 \rightarrow ReadNews_{FLG} = 1,$ $UPactivity = 3 \rightarrow WeatherWarn_{FLG} = 1,$ $UPactivity = 4 \rightarrow PrepBathService_{FLG} = 1,$ $UPactivity = 5 \rightarrow FoodSuggest_{FLG} = 1$	$EnsureUPactivityRun_{FLG} = 1$
$SearchFoodType.BT.N$	$MealType_{Known} = 1, \neg(FoodType = -1)$	$FoodType = FoodType_{res}, FoodType_{Known} = 1$
$SearchFoodType.BT.E$	$FoodType = -1$	$MealType_{Known} = 0, FoodType = 0,$ $FoodType_{Known} = 0$
$SearchFoodIngd.BT.N$	$FoodType_{Known} = 1, \neg(FoodIngd = -1),$ $\neg(FoodType = -1)$	$FoodIngd = FoodIngd_{res}, FoodIngd_{Known} = 1$
$SearchFoodIngd.BT.E$	$FoodIngd = -1$	$FoodType_{Known} = 0, FoodIngd = 0,$ $FoodIngd_{Known} = 0$
$FoodIngdAvailb.BT.N$	$FoodIngd_{Known} = 1, \neg(FoodIngdAvail = -1),$ $\neg(FoodIngd = -1)$	$FoodIngdAvail = FoodIngdAvail_{res},$ $FoodIngdAvail_{Known} = 1$
$FoodIngdAvailb.BT.E$	$FoodIngdAvail = -1$	$FoodIngdAvail = 0, FoodIngdAvail_{Known} = 0,$ $FoodIngd_{Known} = 1$
$GetUserGeographicLocation$	$UgeoLocation_{Known} = 0$	$UgeoLocation = UgeoLocation_{res},$ $UgeoLocation_{Known} = 1$
$GetUserLocation$	$Ulocation_{Known} = 0$	$Ulocation = Ulocation_{res}, Ulocation_{Known} = 1$
$GetWeather$	$Weather_{Known} = 0, UgeoLocation_{Known} = 1$	$Weather = Weather_{res}, Weather_{Known} = 1$
$GetRoomTemp$	$RoomTemp_{Known} = 0$	$RoomTemp = RoomTemp_{res}, RoomTemp_{Known} = 1$
$GetUstate$	$Ustate_{Known} = 0$	$Ustate = Ustate_{res}, Ustate_{Known} = 1$
$GetUactivity$	$Uactivity_{Known} = 0$	$Uactivity = Uactivity_{res}, Uactivity_{Known} = 1$
$GetSleepTime$	$SleepTime_{Known} = 0$	$SleepTime = SleepTime_{res}, SleepTime_{Known} = 1$
$GetTimeCat$	$TimeCat_{Known} = 0$	$TimeCat = TimeCat_{res}, TimeCat_{Known} = 1$
$GetMealType$	$TimeCat_{Known} = 1$	$MealType = TimeCat - 1, MealType_{Known} = 1$
$GetMessages$	$Messages_{Known} = 0$	$Messages = Messages_{res}, Messages_{Known} = 1$
$Change\_Channel$	$TV = 1, UPChannel_{Known} = 1$	$TVChannel = UPChannel, TVChannel_{Changed} = 1$
$RobotSelect\_c1$	$Ulocation_{Known} = 1, Ulocation = 1$	$RobotSelect = 1, RobotSelect\_Known = 1$
$RobotSelect\_c2$	$Ulocation_{Known} = 1, Ulocation = 2$	$RobotSelect = 2, RobotSelect_{Known} = 1$

4.5.3.1 Case 1: Leisure Check

Leisure check is a goal to ensure the human inhabitants perform leisurely activities at a scheduled time. Leisure check goal is considered achieved if activity completion flag $Leisure_{FLG}$ is true. Given their preferred activities, ISS will automatically set up the environment for the user to engage in such activities. If their preferred activity is unknown, the nearest robot will be activated to ask them about it. If the user's activity does not belong to leisure activity group (like working), a log will be recorded. 3 activities are shown in Case 1, which is, watching TV, reading news and going out for a walk, denoted by activity number 1 to 3 respectively when relevant activity is concern. Two other activities that are not shown in Case 1 are taking a bath (activity 4) and cooking (activity 5), but they will be handled in Case 2.

For clarity, for cases involving activation flag and query template, activity $EnsureUPactivityRun$ followed by activity number and $.SS$ is responsible for setting up the appropriate activation flag to make preparation for the user preferred activity. They have pre-conditions of the correct and known user preferred activity (For example, if the user wants to watch TV and that this fact is known, then pre-condition for $EnsureUPactivityRun1.SS$ is met, but not for $EnsureUPactivityRun2.SS$). For cases involving individual device query, the same applies, except $EnsureUPactivityRun$ is replaced by the name $UPactivity$. For cases involving query template and both goal imposition and activation flag approach, $EnsureUPactivityRun.SE$ checks whether the correct proper preparations are made for each user preferred activity as described in Section 4.5.1, where if it is true, the flag $EnsureUPactivityRun_{FLG}$ is set to true.

Case 1a:

User wants to watch TV. Activation flag and query template is used. The implementation starts off by preparing to ask about the user's preferences by implementing $LeisureCheck.SS$ (initiates the necessary flags for leisure checking, such as setting the activation flag that includes activity $AskUPActivity$) and $AskUPActivity.SS$ (Activity template to start query regarding user preferred activity). $GetUserLocation$ (obtains user location in the house, for example, kitchen, bedroom or living room) is run in order to find the nearest robot later. The nearest robot to perform the query is Robot 1, whereas the assumed nearest robot is Robot 2, which is indicated by a planned activity $RobotSelect\_c2$. Due to the assumed nearest robot to be wrong, re-planning is performed to select the correct robot to query the user, which is $RobotSelect\_c1$. Before query implementation, although the user wants to watch TV, it is observed that activity $GetUserGeographicLocation$ (obtaining user geographic location, like which city he/she is currently in) is implemented

(which is not relevant to preparation for TV watching activity) due to optimistic planning where the planner assumes the user wants to go out. But since they are not conspicuous activities, ordering is not an issue. Query on user preferred activity is performed by $Robot1Query.SS$, which is the actual implementation of querying by robot 1 towards the user.

Second re-planning occurs due to erroneous assumption of user activity indicated by $EnsureUPactivityRun3.SS$, which is responsible for making preparations for the user to go out, where as the user's preferred activity is to watch TV. Re-planning and implementation will proceed to turn on the TV and adjusts the lighting. Since user's preferred TV channel is unknown, the robot will proceed to query about it before changing the TV channel through query template $AskUPChannel$.

Activity $EnsureUPactivityRun$ ensures that ISS already performs what is necessary such that the user can engage in his/her preferred activity through $EnsureUPactivityRun.SE$ delivering a true statement for its activity completion flag. For explanation purpose, an example of $EnsureUPactivityRun.SE$'s pre-condition is defined as follows:
$UPactivity_{Known} \wedge ((UPactivity = watchTV) \rightarrow (TVChannel = UPChannel)) \wedge Others$
where $Others$ are the implication pre-condition for other user preferences. Assuming the user wants to watch TV (rendering $Others$ as true such that it can be ignored), the pre-condition dictates that if user prefers to watch TV, then the TV channel should be the same as user preferred channel. This statement needs to be true so that $EnsureUPactivityRun.SE$ can be implemented. In order for the statement to be true, the planner will find whatever ways to change the TV channel to user preferred channel, which involves querying and turning the TV on.

Finally, activity $LeisureCheck.SE$ will ensure the user is engaged in leisure activity by setting the flag $Leisure_{FLG}$ true. The same concept of situation checking via implication pre-condition is used for other cases as well, which includes $BackHomePrepService$ (checking that the required preparation is made when the user comes home) and $EnvironmentCheck$ (checking that all necessary adjustments on the environment are done), and $SleepCheck$ (ensure the user has enough sleep but do not oversleep).

Case1b

User wants to watch TV. Goal imposition and query template are used. Instead of using activation flag, new goal is introduced by running activity to perform the same case situation as Case 1a. Before the new goal is introduced, the initial goal is to ensure user

preferred activity is known. The new goal is then added after activity $AskUPActivity.SE$ is completed, where the goal is that activation flag $Leisure_{FLG}$ is true. It should be noted that $AskUPActivity.SE$ can also be completed without querying given that the user preferred activity is known.

Case 1c:

User wants to watch TV. Activation flag and individual device query activity are used. The case situation is the same as Case 1a, except that individual device query is used instead of a query template.

Case 1d:

User wants to read news. Activation flag and query template are used. In this case, given the ISS already knows the user's preferred activity through query, the robot will query about the preferred news category, before announcing the appropriate news to the user. Implementation resembles that of Case 1a, except that at the last re-planning, since the user wants to read news, activity $AskNewsCat$ is implemented (to prepare query on user preferred news category), after which activity $ReadNews$ is implemented (template for news announcement).

Case 1e:

User wants to read news. Goal imposition and query template are used, with the same case situation as Case 1d. Implementation resembles that of Case 1b, except the different preferred activity described in Case 1d.

Case1f:

User wants to read news. Activation flag and individual device query activity are used, with the same case situation as Case 1d.

Case 1g:

User wants to go out. Activation flag and query template are used. Given the ISS knows about this preferred activity, it will try to warn the user if there is any bad weather forecast ahead. Implementation resembles that of Case 1a, except that preferred activity is to go out. In this case, activity $WeatherWarn$ will be implemented (query template to warn the user about potential bad weather given the weather result obtained after executing activity $GetWeather$).

Case 1h:

User wants to go out. Goal imposition and query template are used, with the same case situation as Case 1g.

Case1i

User wants to go out. Activation flag and individual device query activity are used, with the same case situation as Case 1g.

4.5.3.2 Case 2: Back Home Service

When the human inhabitant comes back home, this goal will be triggered. The goal will be considered achieved when proper preparation is made upon the user's arrival indicated by the flag $BackHomePrepService_{FLG}$. The service consists of notifying and announcing to the user any unheard messages while he/she is gone through the appropriate medium. It also queries whether the user wants to take a bath (if yes, ISS will prepare the bath water). Apart from taking the bath, the user can request for other activities, where the ISS will make the necessary adjustments to the environment. For this case, all sub-cases assume the use of query template.

Case 2a:

User wants to take a bath. Activation flag is used. In this case, when the user reaches home and wants a bath, ISS will prepare bath water after query. The ISS will also announce the unheard messages through the nearest robot recorded when the user was away. At first, activity $BackHomePrepService.SS$ will be executed to set the necessary flags for bath query $AskShower$ and message announcement activity $MsgReportService$ given that there are unheard messages obtained from activity $GetMessages$. Given that the user wants a bath, bath preparation activity $PrepBathService$ will be executed, which is a nested activity to make necessary preparation like controlling water temperature and volume. When necessary preparations are made, activity $BackHomePrepService.SE$ will set the flag $BackHomePrepService_{FLG}$ to true.

Case 2b:

User wants to take a bath. Goal imposition is used, with the same case situation as Case 2a. The initial goal is to ensure user preferred activity is known as in Case 1b as well as making announcement on unheard messages. Activity $BackHomePrepService.SE$ will then ensure that the necessary preparations are made for the user preferred activity by

introducing a new goal where $EnsureUPactivityRun_{FLG}$ is true as is the case in Case 1.

Case 2c:

User wants to prepare food and it is morning time. Activation flag is used. Upon coming home, ISS will ask the user whether he/she wants to take a bath. In this case, instead of taking a bath, the user would like to prepare food. ISS will try to satisfy this preferred activity by suggesting an appropriate meal based on available ingredients at home. Since it's morning as obtained from activity $GetTimeCat$, activity $GetMealType$ will determine that the meal is of breakfast type. Food search is made through food database as depicted in Figure 4.3 (note that the backtracking during food search is also part of the activities devised during planning). Activity $SearchFoodType.BT.N$ will search through the food type under breakfast such as cereal and sandwich shown in Figure 4.3. For every food type, $SearchFoodIngd.BT.N$ will search the ingredients types. To see if sufficient materials are available to prepare such meal based on the ingredient type, activity $FoodIngdAvailb.BT.N$ is run. If there are insufficient ingredients, backtracking will occur to choose the next ingredient type. Backtracking will also occur if there are no more ingredient types for a given food type, to which it will select the next food type in line. The state where there is no available food is also considered a legitimate type, but this only occurs when there is no choice of food left after exhaustive search given a threshold time. Then, activity $FoodSuggest$ will ensure the appropriate recommendation to the user, where the query template will be prepared by $ReadFoodSuggestion$ to announce the recommendation. For this case, ISS will also announce any unheard messages.

Case 2d:

User wants to prepare food and it is morning time. Goal imposition is used, with the same case situation as Case 2c. As in Case 2b, the initial goal is to ensure user preferred activity is known and making announcement on unheard messages. The second goal is introduced where $FoodFound_{FLG}$ is true to ensure food information is found. In order to announce the meal recommendation, third goal will be introduced where $ReadFoodSuggestion_{FLG}$ is true, which is just the activity completion flag of activity $ReadFoodSuggestion$.

4.5.3.3 Case 3: Environment Check

The goal is to handle environmental adjustment for more comfort by manipulating lighting (activity $LightService$), temperature (activity $TempControlService$) and win-

dows (activity $WindowService$). All these services are activated by activity $EnvironmentCheck$, which is also responsible for setting the flag $EnvironmentCheck_{FLG}$ to true that is the initial goal of the case. For temperature, ISS adjusts the air-conditioner or the fan, depending which one is being switched on (If none is being switched on, fan will be used). For windows, the control depends on the lighting, whether the air-conditioner is on, and also whether it is raining or not (handled by activity $RainyDayCheck$). Weather forecast will also affect the clothesline (handled by activity $Clothesline\_cl$), which will retract given that there are clothes and also that there is a high chance of rain within certain period of time.

Case 3a:

There is a rainy day forecast. Activation flag is used

Case 3b:

There is a rainy day forecast. Goal imposition is used. Second imposed goal is that $RainyDayCheck_{FLG}$ is true, which is the activity completion flag for activity $RainyDayCheck$.

4.5.3.4 Case 4: Sleep Check

This goal aims to determine whether a person has overslept or not (more than a certain duration that is extracted by $GetSleepTime$), which if true, will wake the person up through alarm service (handled by activity $AlarmService$) and also turning on of lights. But if the person is sick (obtained via activity $GetUstate$, which obtains user state), or if the time is night (obtained via $GetTimeCat$, which obtains time category like morning, noon and evening), then ISS just records the situation in a log without waking up the person. User activity is extracted by $GetUactivity$. It should be emphasized that $GetUactivity$ extracts human activity (sleeping, working, eating), instead of user preferred activity (wants to watch TV, wants to eat) that is obtained through active query if unknown. Finally, activity $SleepCheck$ is to ensure that proper activities are performed given the person's state and activity.

Assuming the person is well, has sleep more than 10 hours and the time is morning, the following is the planning and implementation result.

4.5.4 Summary of Cases

Table 4.5 shows the summary of cases as described previously. The column under AF/GI means the case is either using activation flag (AF) or goal imposition (GI). For column under TM/IN, it shows whether the case uses query template (TM) or individual device query (IN) for querying and announcement purposes. The column Time indicates the total planning time in seconds.

Table 4.5: *Cases summary*

Case	AF/GI	TM/IN	Time(sec)	Remark
Case 1a	AF	TM	28.58	Leisure
Case 1b	GI	TM	2.187	Leisure
Case 1c	AF	IN	1.874	Leisure
Case 1d	AF	TM	27.96	Leisure
Case 1e	GI	TM	2.955	Leisure
Case 1f	AF	IN	1.591	Leisure
Case 1g	AF	TM	24.43	Leisure
Case 1h	GI	TM	1.852	Leisure
Case 1i	AF	IN	1.827	Leisure
Case 2a	AF	TM	9.876	Back Home
Case 2b	GI	TM	0.348	Back Home
Case 2c	AF	TM	52.35	Back Home
Case 2d	GI	TM	9.577	Back Home
Case 3a	AF	TM	16.34	Environment
Case 3b	GI	TM	5.612	Environment
Case 4	AF	TM	1.740	Sleep

4.5.5 Discussion

4.5.5.1 Activity Ordering Issue

This section shows the importance of handling ordering issue, and current solutions used by the ISS to solve it. As explained in Section 4.4.3, due to CSP's optimistic approach in planning according to the work [71], some conspicuous activities will be implemented based on a wrong assumption that is supposed to be obtained during run-time. An example case is shown below (with individual device query), where the case situation resembles Case 1a.

Planning containing redundant activities

Planning:

$GetUserLocation \Rightarrow RobotSelect\_c2 \Rightarrow$

$Robot2AskNewsCat.SS \Rightarrow Robot2ReadNews.SS \Rightarrow$

$Robot2AskActivity.SS \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 0.486085 seconds)

Implementation:

Activity $RobotSelect\_c2$ pre-condition is not satisfied

Re-planning:

$RobotSelect\_c1 \Rightarrow Robot1AskNewsCat.SS \Rightarrow$

$Robot1ReadNews.SS \Rightarrow Robot1AskActivity.SS \Rightarrow$

$LeisureCheck.SE \Rightarrow End$ (Planning Time = 0.296197 seconds)

Implementation:

// $Robot1AskNewsCat.SS$ and $Robot1ReadNews.SS$ are run! //

Activity $LeisureCheck.SE$ pre-condition is not satisfied

Re-planning:

$TV\_on.SS \Rightarrow RobotSelect\_c1 \Rightarrow$

$Robot1AskChannel.SS \Rightarrow Change\_Channel \Rightarrow$

$Light\_on.SS \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 0.422067 seconds)

Implementation:

End

From the example, the user prefers to watch TV. But before query is made, ISS assumes the user wants to read news, thus, implementing $Robot1AskNewsCat$ and $Robot1ReadNews$ activities. Even though the TV is switched to the user's preferred channel at the end of the plan, implementing conspicuous activities in reading the news is redundant. Redundant activity execution can bring negative impact on the user, especially if the service is concerning their wellbeing, as in Case 4, which tries to ensure that the user has enough sleep, and wakes him/her up at appropriate times.

4.5.5.2 Activation Flag vs Goal Imposition

Comparison between cases (Case 1a vs Case 1b, Case 1d vs Case 1e, Case 1g vs Case 1h, Case 2a vs Case 2b, Case 2c vs Case 2d, Case 3a vs Case 3b) clearly shows that goal imposition approach is a lot faster at planning stage.

Goal imposition approach translates a complex goal into smaller manageable sub-goals. This contributes to the planner in a way that a smaller number of activities are chosen, thus, improving the speed.

Activation flag approach plans the whole sequence of activities given only the desirable final state. Therefore, it requires relatively more activities to plan compared to goal imposition method for planning since, in a way, it needs to automatically generate the required sub-goals indirectly. But due to this fact, activation flag approach possesses much more generality compared to goal imposition. Contrary to goal imposition approach, which requires one to know to a certain extent a rough sequence of activities, activation flag approach requires only the pre-conditions, effects and the type of activity, but with the expense of longer planning time. It upholds the ideals of SOA, where every device should be loosely coupled. Goal imposition also has issues with reusability. To illustrate the point, for Case 1b, the goal $Leisure_{FLG} = true$ is added while the activity $AskUPActivity.SE$ is implemented. Intuitively, the first goal is to successfully obtain user activity preference, after which the second goal is to ensure that ISS will make the necessary preparations through $Leisure_{FLG}$. But, instead of making preparations, the user just rather has the ISS record his/her activity preference. Since the addition of goal is embedded inside the activity, a new activity needs to be created just for the latter purpose, because the former cannot be reused. The same situation can also be observed in Case 2d, where there are two goals being introduced, that is, to ensure a food is found, and that proper announcement is made. Due to probable cases where to ISS might just want to use the food information for meal preparation, instead of announcing it, new activity needs to be defined.

Therefore, due to goal imposition approach fast but specific planning, it can be applied to tedious and specialized tasks like the backtracking to search for food suggestion as in Case 2d. In Case 2d, there are 7 backtrackings (may be more, depending on the ingredients availability), where each requires individual planning. Using activation flag approach in this case will yield a lower performance, which requires on average 2.535 seconds (Case 2c), compared to goal imposition that only needs 0.503 seconds. Goal imposition is also suitable for plans that are composed of a sequence of sub-plans. Combining all plans during planning process will require handling larger number of activities, as in Case 3a (which consists of environmental check and rainy day check sub-plans). By setting the environmental check activity to impose new goals for rainy day check as in Case 3b, speed is significantly improved. Besides, due to the limited number of sequence per plan K as explained in Section 4.2, a complicated goal might require more than K states to fulfill.

4.5.5.3 Individual Device Query vs Template Activity

Individual device query for robot is implemented in Case 1c, 1f and 1i using activation flag approach. Compared with their template query counterparts (Case 1a, 1d and 1g) on 2 robots, planning involving individual device query is extremely fast, comparable to that of goal imposition approach. For individual device query, every query activity needs to be defined for every robot. From the cases, queries include asking user activity preference, TV channel, asking whether the user wants a shower or not, asking user news preference, announcing message, announcing news, announcing food suggestion and announcing weather warning. A unique activity has to be defined for all these queries and all the robots involved. The method might not be efficient if the ISS involved a lot of robots or querying medium and large number of query types. In this case, template query can be used, though improvement of its speed is subject of further research.

4.5.6 Remark

Case studies show the wide range of application for the CSP planner. By improving on the work in [71], activity ordering issue is handled through the use of activation flag or goal imposition and also the introduction activity completion flag. But these methods require that the class of activities need to be known for conspicuous activities, or the knowledge for rough sequence of plans. Comparison and discussion are made to establish their suitability for various applications. Due to the emphasis on robot querying, comparison is also made between template query and individual device query. Although individual device query enables faster planning, it might create a significant rise in number of activities if not managed properly when a lot of querying mediums are involved.

There are conditions that cannot be efficiently handled through current CSP planner, where the two prominent conditions are temporal rules and goal preferences. In intelligent building domain, there is a tendency towards temporal behavior [60], such as air-conditioner should not be running for more than 5 hours and do not switch off car porch light as night. For goal preferences, there are goals that are not necessary but desirable to be achieved [52]. Besides, partial goal fulfillment and planning that optimizes certain objective function cannot be performed. For such tasks, one can resort to weighted constraints as detailed in Section 4.6.

4.6 Planning as Weighted Constraint Satisfaction

CSP approach employs hard constraints, which are fed to a solver to obtain a sequence of A that are the activities that need to be implemented to fulfill the given goals. One example of state of the art solver is Z3 SMT solver, which can efficiently obtain the plan given hard constraints[35]. The plan will be solved by continually increasing K until the constraints are satisfied. Hard constraints are used, and therefore, the approach cannot (or at least very inefficiently) supports optimization and partial goal fulfillment. Partial fulfillment of goals is important because certain goals might contradict each other, or that they might be too complicated to solve in one go. For this, solving via hard constraints will conclude the goal is unsatisfiable instead of trying to fulfill as much goals as possible. Such solvers cannot handle soft constraints to perform optimization and partial fulfillment of goals, or at least not efficiently.

For example, given mother is requesting the TV to switch to channel 1, whereas her daughter requests for channel 2. These two goals contradict each other. Service composition via hard constraints will conclude that the goals are not satisfiable, instead of trying to fulfill one of their wishes (such as switching to channel 1 to fulfill the mother's wishes as the goal imposed by her is considered much more important compared to the daughter). Such problems can be solved with soft constraints by introducing weights.

This section aims to extend the CSP to Weighted Constraint Satisfaction Problem (WCSP). WCSP endows constraints with weights, which transforms them into soft constraints where optimization can be implemented. Branch and bound with depth first search is applied to solve the planning problem represented as WCSP, where the lower bound is estimated via Bacterial Memetic Algorithm (BMA). Memetic algorithms has been shown to be effective in handling weighted constraints that gives good enough solutions [46].

4.6.1 Weighted Constraint Satisfaction Problem

WCSP can be described as a tuple $\langle \chi, D, F \rangle$ [12], where $\chi = X_1, X_2 ... X_K \cup R$, D is the set of domains of the variables in χ, and F is a set of weighted constraints. One can think of ζ as F with all the constraints f having infinity as weight values $Weight(f)$. The objective function is the sum of all functions in F,

$$L = \sum_{f \in F} Weight(f) \tag{4.3}$$

The goal is to find the instantiation of all variables such that it minimizes the objective function.

Various approaches have been used to solve general WCSP, which includes search, clustering and variable elimination as explained in this paper [81]. Good heuristics used can also further hasten solving process [97]. Our problem is specific to planning with domain description described previously. Therefore, internal structures can be exploited to build a solver that is specially tailored for plan composition via solving WCSP. Branch and bound with depth first search is used due to polynomial space complexity and the ability to handle constraints of high arity and wide domain, compared to methods like variable elimination [36] (though variable elimination can be combined with search to obtain much better result [81]). Besides, using a relaxed condition of the former planning problem as shown in Section 4.6.4, lower bound can be easily obtained to prune the search graph.

4.6.2 Design of Automated Planner

Given an initial state (or initial variable instantiations), an optimized plan means a plan that can fulfill the maximum number of goals with the least number of activities (or with the least cost given the sum of the costs for all activities), where the sequence of activities need to abide to their individual precondition and effects described in Section 4.1.

Goals are constraints for the final possible variable instantiations. For example, if the goal is $A \vee B$, then final state should be $A_K = 1, B_K = 0$, or $A_K = 0, B_K = 1$, or $A_K = 1, B_K = 1$. Weights associated with every goal are imposed as costs if the goal is not fulfilled in the final state.

Constraints imposed by activities are the preconditions and effects. An activity can only be implemented if its preconditions are met. For example, a TV can only change its channel given the precondition that the TV is on. At the same time, it is considered (at least in planning phase) that effects of activities will definitely change the subsequent state (such as the act of changing TV channel will result in a change to the preferred channel). Due to uncertainty, effects might not give the desirable response during actual execution. But this is not an issue given the dynamic planning framework described in Section 4.1, which supports re-planning. That said, constraints from preconditions and effects are treated as hard constraints unlike goal imposed constraints.

Activities themselves also come with weights. But unlike weights for constraints, these weights are imposed when an activity is implemented. They act as costs for their respective activities. This is important as there are times when one prefers certain activities from others. For example, during late night, a person might prefer the dimmer light to be switched

Table 4.6: *Example activities and their weights*

Name	Precondition	Effect	Weight
generatorON	n/a	g:=1	2
lightON	g=1	L:=1	3
fanON	g=1	f:=1	2

on instead of the brighter ones. Therefore, the action to switch on the brighter light imposes more cost than its dimmer counterparts.

It is assumed the weights are determined depending on situation and time. But this work does not deal with weight settings. Our intention is that, given a set of constraints and weights, the system will try to generate a sequence of activities as plans that is optimum. In order to achieve that, it needs to minimize the following cost function:

$$A^{opt} = \underset{A_0.A_1...A_{K-1}}{argmin} ([\sum_{0 \leq t < K} C(A_t) + Pc(A_t, S_t)] + Gc(\hat{A} \circ S_0)) \tag{4.4}$$

Where $C(A_t)$ is the weight for implementing activity A_t,
$Pc(A_t, S_t)$ gives the cost given wether or not precondition for A_t is fulfilled by state S at sequence t. Due to precondition being a hard constraint, Pc will return infinity if precondition is not met.
$\hat{A} \circ S_0 = A_{K-1} \circ A_{K-2}...A_1 \circ A_0 \circ S_0$, which gives the final state S_K when activity A_0 is implemented on state S_0, followed by A_1,and so on until finally A_{K-1}.
$Gc(S) = \sum_{f \in Goal} Weight(f, S)$, which gives the total cost imposed by non-fulfilled goals. $Weight(f, S)$ is different from equation 4.3 as it only considers variables at the final state sequence, whereas equation 4.3 considers variables of all sequences.

As example, consider two goals, 1) Fan is on $f = 1$ with weight 4 2) Light is on $L = 1$ with weight 6, and initial state where generator, light and fan are all off ($g = 0, L = 0, f = 0$ respectively, $\langle 0, 0, 0 \rangle$ in short). Example activities are shown in Table 4.6.

Case 1: Activity sequence is generatorON $\Rightarrow$ lightON $\Rightarrow$ fanON (which means generatorON is executed, followed by lightON, and finally fanON), There is no precondition of generatorON, therefore, Pc component will return a 0. Effect from the activity will set $g = 1$, thus $\langle 1, 0, 0 \rangle$. Given this new state, precondition for lightON ($g = 1$) is also met, and execution proceeds until fanON. Finally state will be $\langle 1, 1, 1 \rangle$. Total cost returned by Pc is 0. Total cost of activity is $7 = C(\text{generatorON}) + C(\text{lightON}) + C(\text{fanON})$. Since the 2 goals are fulfilled, Gc returns 0 too. Total cost is 7.

Case 2: Activity sequence is generatorON $\Rightarrow$ lightON. Execution of the plan will result in final state $\langle 1, 1, 0 \rangle$. All preconditions are met, thus, Pc returns a total of 0. Total cost of

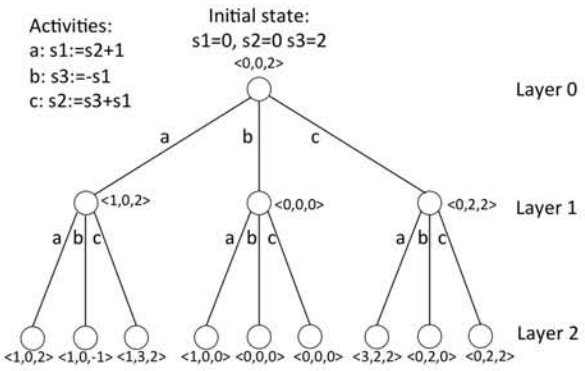


Figure 4.4: *Search tree for planning*

activity is 5. Since goal 1 is not satisfied, Gc returns 4. Total cost is 9.

Case 3: Activity sequence is lightON. Since the initial state is $\langle 0, 0, 0 \rangle$, precondition is not met. Pc returns ∞, rendering subsequent calculation trivial.

Equation 4.4 tries to find a sequence of activities that is optimal. Due to the use of logical constraints, optimization becomes complicated. Direct search is intractable, aggravated by hard constraints and its weights from Pr.

Equation 4.4 can be relaxed by eliminating Pc, and in turn, reformulated as goals. Problem reformulation is explained in Section 4.6.4. The relaxed optimization problem can then be used to calculate the lower bound for branch and bound via BMA explained in Section 4.6.5.

4.6.3 Branch and Bound Graph Search Execution

Search tree represents the possible paths to reach potential solutions. For our planning problem, every layer of the tree is a particular sequence of activity, which in our case, has a total of K layers. Every node is a particular state. Edges are the actions taken, which transform S_t to S_{t+1}. Figure 4.4 shows an example of the search tree for planning.

Depth first search branch and bound works by choosing a path that gives the best estimated cost, where the best estimated cost of the nodes for every layer except the layer of the leaf nodes is estimated via a heuristic function H, given its previous cost. H estimates the best possible cost to reach the goal state by using the relaxed optimization problem, which, combined with previous cost gives the lower bound. A potential solution is reached when the path reaches the leaf nodes. Total cost of every potential solution will be compared with the upper bound, upon which if lower, the upper bound will be updated. Branch and bound reduces the number of search paths by pruning branches of the respective nodes

where their lower bound is higher or equal to the upper bound.

Algorithm 3 shows a recursive search algorithm for optimized planning. The result of the algorithm is A^{opt}, which is the optimized sequence of plans.

Before the algorithm starts, current cost Cc and upper bound UB are set to ∞. State S is set to initial state values. The algorithm starts off by checking whether or not the last layer of the tree is reached, upon which if yes (meaning, it is a potential solution), it will update the UB if the total cost is lower. Further fine tuning can be made by checking for activity redundancy through UB_Refine, which checks whether the omission of certain activities will yield better result. Given a sequence of activities, UB_Refine will loop through each and calculate whether their omission will produce lower cost. As fast stochastic search is employed in calculating the lower bound, this comes at a price where certain redundant activity will be introduced (given activity cost don't warrant much cost relative to those imposed by goals, which is the situation for our experimentation). UB_Refine eliminates redundant activities.

If leaf nodes are not reached, current node is branched out to the subsequent layer, where each branch represent the effect from each activity. Lower bound is estimated for each node of the new layer via BMA that will be explained in Section 4.6.5. Lower bounds for activities that do not have their preconditions met are assigned ∞. Besides that, further checking is performed by LSh_Check. LSh_Check checks whether the selection of a certain activity for branching coincides with pairs listed in the local search look-up table, where the look up table is explained in Section 4.6.6. The activities are then sorted according to their lower bounds to determine which activity should be branched out further for evaluation. If UB is lower or equal to an activity's lower bound, that branch, and the subsequent branches in the sort list are all pruned out.

4.6.4 Relaxation of Optimization Problem

To obtain a larger feasible region by removing restrictions, equation 4.4 is relaxed by converting the $Pc(\bullet)$ component to goals, which gives:

$$A^{opt} = \underset{A_0.A_1...A_{K-1}}{argmin} \; (\sum_{0 \leq t < K} C(A_t) + \bar{G}c(\hat{A} \circ S_0)) \tag{4.5}$$

$\bar{G}c(\bullet)$ varies from $Gc(\bullet)$ in the sense that it not only considers cost from fulfillment of goals, but also extra sub-goals introduced by the conversion of $Pc(\bullet)$.

Conversion is done under the concept that whenever an effect of an activity takes place, its precondition has to be met. Therefore, sub-goals are constructed in such a way that if a

Result: A^{opt}
Upper bound (global variable) $UB = \infty$;
Current cost $Cc = \infty$;
Sequence index $tc = 0$;
State S initialized;
Activity sequence A set to empty;
note: Recursion starts
Function $BnB(Cc, tc, S, A)$;
if $tc \geq K - 1$ **then**
    if $CostEst(A, S) < UB$ **then**
        Update UB;
        $A^{opt} := A$;
        [UB,A^{opt}]:=UB_Refine;
    end
else
    for $f =$ *all activities* **do**
        if *Preconditions for* f *are met and LSh_Check(f)* **then**
            $Act[f] = BMA(tc, S) + C(f) + Cc$;
            note: Store the estimated lower bound if activity f is applied. $C(f)$ returns the weight of activity f;
        else
            $Act[f] = \infty$;
            note: Or else set the lower bound to ∞;
        end
    end
    $SList = sort(Act)$;
    note: Sort Act in ascending order based on their lower bound;
    for *Loop through* $SList$ **do**
        note: Looping through the sorted list;
        $LB :=$ Lower bound of current SList selection;
        AC:=Activity of current SList selection;
        SC:=State after the effect of AC;
        $Cc2$:=$Cc + C(SList)$;
        if $LB \geq UB$ **then**
            Break the loop;
        else
            $A := append(AC)$;
            $BnB(Cc2, tc + 1, SC, A)$;
        end
    end
end

Algorithm 3: Recursive search for optimized plan

$V = val$, val is a value where either $val = 0$ or 1;
V_0 is the initial value of V;
$Pstore$ set to empty;
if V *is not a response variable* **then**
  for $f =$ *all activities* **do**
    if *Found* $V := val$ *in the* f *effects* **then**
      Add f precondition in $Pstore$;
    end
  end
end
Add $(V = val \wedge V_0 \neq V) \rightarrow (\bigvee Pstore)$ as sub-goal;
note: $\bigvee Pstore$ statement is true if at least one addition of f precondition in it is true;

Algorithm 4: Conversion for direct value assignment

$V = var$;
V_0 is the initial value of V;
$Pstore$ set to empty;
if V *is not a response variable* **then**
  for $f =$ *all activities* **do**
    if *Found* $V := var$ *in the* f *effects* **then**
      Add f precondition in $Pstore$;
    end
  end
end
Add $(V = 1 \wedge var = 1 \wedge V_0 \neq V) \rightarrow (\bigvee Pstore)$ as sub-goal;
Add $(V = 0 \wedge var = 0 \wedge V_0 \neq V) \rightarrow (\bigvee Pstore)$ as sub-goal;

Algorithm 5: Conversion for variable assignment

$V = \neg var$;
V_0 is the initial value of V;
$Pstore$ set to empty;
for $f =$ *all activities* **do**
  if f *is not a response variable* **then**
    if *Found* $V := \neg var$ *in the* f *effects* **then**
      Add f precondition in $Pstore$;
    end
  end
end
Add $(V = 1 \wedge var = 0 \wedge V_0 \neq V) \rightarrow (\bigvee Pstore)$ as sub-goal;
Add $(V = 0 \wedge var = 1 \wedge V_0 \neq V) \rightarrow (\bigvee Pstore)$ as sub-goal;

Algorithm 6: Conversion for negated variable assignment

variable holds a certain value in the final state, and that it is different from its initial value, it implies that one of the preconditions of activities which subject that particular variable to hold that value has to be true. Currently, only variables of Boolean type are considered for conversion. Other types such as integer and finite domain sorts are subject to future research. Conversion of variables is limited to 4 types of effects pertaining the variable, which are, $V := 0$, $V := 1$, $V := var$ and $V := \neg var$, where var is a variable. The first two types of conversion are shown in Algorithm 4. The third type is shown in Algorithm 5, whereas the fourth is shown in Algorithm 6. The number of generated sub-goals is denoted nSG. Extra sub-goals equipped with weights reduces the effective branching factor. Optimizing the relaxed problem thus gives a better lower bound.

4.6.5 Bacterial Memetic Algorithm Lower Bound Estimation

Bacterial Memetic Algorithm (BMA) is applied for finding the optimal solution for the relaxed optimization problem discussed in Section 4.6.4. BMA is a population based stochastic optimization technique which effectively combines global and local search in order to find good quasi-optimal solution for the given problem [19]. The encoding of the individual is a sequence with $K - 1$ indexes. The evaluation of the bacteria is calculated by Eq. (4.6):

$$Cost = \sum_{0 \leq t < K} C(A_t) + \bar{G}c(\hat{A} \circ S_0) \tag{4.6}$$

The operation of the BMA starts with the generation of a random initial population containing N_{ind} individuals (see Algorithm 7). Next, until a stopping criterion is fulfilled (which is usually the number of generations, N_{gen}), we apply the bacterial mutation, gene transfer, and local search operators. Bacterial mutation creates N_{clones} number of clones (copies) of an individual, which are then subjected to random changes in their genes (see Algorithm 8). The number of genes that are modified with this mutation is a parameter of the algorithm (l_{bm}). After the bacterial mutation, the gene transfer operation is applied at population level (see Algorithm 9). It means copying genes from better individuals to worse ones. For this reason, the population is split into two halves, according to the cost values. The number of gene transfers in one generation (N_{inf}), as well as the number of genes (l_{gt}), that get transferred with each operation, are determined by the parameters of the algorithm. The last operator in each generation is the local search, which performs reparation of all individuals, using a lookup table, which describes conflicting values in the neighborhood genes of the bacterial chromosome. BMA has been successfully applied to wide range of problems. More details about the algorithm can be found in [19], [22].

Create initial population;
$generation \leftarrow 0$;
while $generation \neq N_{gen}$ **do**
    for $i \leftarrow 1$ **to** N_{ind} **do**
        BacterialMutation($bacterium_i$)
    end
    GeneTransfer;
    for $i \leftarrow 1$ **to** N_{ind} **do**
        LocalSearch($bacterium_i$)
    end
    $generation \leftarrow generation + 1$;
end

Algorithm 7: Bacterial Memetic Algorithm

Create $N_{clones} + 1$ clones of $bacterium$;
$N_{segments} \leftarrow K/l_{bm}$;
for $i \leftarrow 1$ **to** $N_{segments}$ **do**
    Select l_{bm} yet unmutated random genes;
    for $k \leftarrow 2$ **to** $N_{clones} + 1$ **do**
        Mutate the selected genes in $clone_k$
    end
    for $k \leftarrow 1$ **to** $N_{clones} + 1$ **do**
        Evaluate $clone_k$ using Eq. (4.6)
    end
    Select the best clone;
    Best clone transfers the mutated genes to all clones;
end
$bacterium \leftarrow$ best clone

Algorithm 8: Bacterial Mutation

for $i \leftarrow 1$ **to** N_{inf} **do**
    Ascending order of the population according to Eq. (4.6);
    $SourceBacterium \leftarrow Random(0, \ldots, N_{ind}/2 - 1)$
    $DestinationBacterium \leftarrow Random(N_{ind}/2, \ldots, N_{ind})$ Select random consecutive l_{gt} genes;
    Transfer the selected genes from $SourceBacterium$ to $DestinationBacterium$
end

Algorithm 9: Gene Transfer

4.6.6 Local Search

Combinations of activities that produce trivial effects should be avoided. Examples of such combinations are turning on and then off the lights, and turning the light on two consecutive times.

In this work, we only assume combination of two. To obtain a list of such trivial pairs, all activities are paired with other activities, including themselves. Given an initial state, a pair of activities is considered trivial if the state after their execution is the same as the initial state. The list is generated according to the activities available before the search commences.

This list is used by BMA to perform local search and better reparation. The local search list is also used in branch and bound search through LSh_Check. It determines whether branching out of a particular activity is redundant given the previous activity, thus, reducing search space.

4.6.7 Experiments and Discussion

4.6.7.1 Setup

Examples of wrapped atomic activities are shown in Table 4.7. Preconditions and effects are shown as first order logic (FOL) to save space. In actual implementation, the formulas are all grounded.

Activity Nop has weight $wNop = 1$. Every other activities have weights greater than that of Nop. This means, activity Nop is preferred over others after the goal is achieved. Sub-goals (not goals) have weight $wSG = 1$. Every goal (not sub-goals) have weight $= K \times wNop + nSG \times wSG$. This puts goals as having higher priority than activities and sub-goal fulfillments.

Since the purpose of this work is not on ways to set weights, but to come out an optimized plan around pre-defined weights, weights are set for the purpose of experiments. The same case applies to goals. Goals, initial conditions and weights (termed configurations) are specially designed such that optimal plans consist of 4 to 16 steps of activities. These assortments of configurations will be randomly chosen given the number of steps of the optimal plan during test.

Table 4.7: *Examples of relevant activities (as FOL)*

Name	Precondition	Effect	Wt
generator1ON	$true$	$g := 1, g_{changed} := 1$	3
generator1OFF	$true$	$g := 0, g_{changed} := 1$	3
generator2ON	$true$	$g2 := 1, g2_{changed} := 1$	2
generator2OFF	$true$	$g2 := 0, g2_{changed} := 1$	2
lightDim'n'ON	$BrightLight(n) \rightarrow false \wedge \exists_x Generator(x) = 1$	$L(n) := 1, L_{changed}(n) := 1$	$Cost(n)$
lightDim'n'OFF	$true$	$L(n) := 0, L_{changed}(n) := 1$	$Cost(n)$
lightBright'n'ON	$\neg BrightLight(n) \rightarrow false \wedge \exists_x Generator(x) = 1$	$L(n) := 1, L_{changed}(n) := 1$	$Cost(n)$
lightBright'n'OFF	$true$	$L(n) := 0, L_{changed}(n) := 1$	$Cost(n)$
fan'n'ON	$\exists_x Generator(x) = 1$	$f(n) := 1, f_{changed}(n) := 1$	$Cost(n)$
fan'n'OFF	$true$	$f(n) := 0, f_{changed}(n) := 1$	$Cost(n)$
door'n'Open	$Obstruction(n) = 0$	$D(n) := 1, D_{changed}(n) := 1$	2
door'n'Close	$Obstruction(n) = 0$	$D(n) := 0, D_{changed}(n) := 1$	2
window'n'Open	$Swing(n) \wedge Curtain(n, c) \rightarrow Curtain(c) = 1$	$W(n) := 1, W_{changed}(n) := 1$	$Cost(n)$
window'n'Close	$true$	$W(n) := 0, W_{changed}(n) := 1$	$Cost(n)$
curtain'n'Open	$true$	$Cr(n) := 1, Cr_{changed}(n) := 1$	$Cost(n)$
curtain'n'Close	$Curtain(w, c) \rightarrow window(w) = 0$	$Cr(n) := 0, Cr_{changed}(n) := 1$	$Cost(n)$
TVON	$true$	$Tv := 1, Tv_{changed} := 1$	2
TVOFF	$true$	$Tv := 0, Tv_{changed} := 1$	2
SwitchChannel	$UPChannel_{known} = 1 \wedge TV := 1$	$TVchan := UPchannel, TVchan_{changed} := 1$	2
GetUPchannel	$true$	$UPChannel := UPChannel_{response}, UPChannel_{known} := 1$	2
GetWeather	$true$	$Weather := Weather_{response}, Weather_{known} := 1$	2

Table 4.8: *Example activities*

Name	Precondition	Effect	Wt
generator1ON	$true$	$g1 := 1$	3
generator2ON	$true$	$g2 := 1$	2
light'1'ON	$g1 = 1 \vee g2 = 1$	$L1 := 1$	4
light'2'ON	$g1 = 1 \vee g2 = 1$	$L2 := 1$	2
light'3'ON	$g1 = 1 \vee g2 = 1$	$L3 := 1$	3
fan'1'ON	$g1 = 1 \vee g2 = 1$	$f1 := 1$	3
fan'2'ON	$g1 = 1 \vee g2 = 1$	$f2 := 1$	2

As an example, assume $K = 10$. Table 4.8 shows the activities in this example. All formulas are grounded. Boolean is treated as an integer of 1 and 0.
Goals are as follows:
$2 = L1 + L2 + L3$
$f1 = 1$
$f2 = 1$
$f2 \neq f1$

Initial state is $g1 = 0, g2 = 0, L1 = 0, L2 = 0, L3 = 0, f1 = 0, f2 = 0$

The optimal solution is:
generator2ON⇒light'3'ON⇒light'2'ON⇒fan'2'ON

The optimal cost is:
14(activity cost)+0(sub-goal cost)+10(goal cost)=24

Therefore, the optimal plan is 4 steps with optimal cost 24.
If instead, the planner gives the following solution:
generator2ON⇒light'3'ON⇒light'2'ON⇒fan'1'ON
Its cost is now 25. The difference from optimum cost is thus 4.17%.

4.6.7.2 Parameter Selection

Parameters for BMA are number of generations N_{gen}, number of bacteria N_{ind}, number of clones N_{clones}, mutation segment length l_{bm}, number of infections N_{inf}, and infection segment length l_{gt}. For speed and simplicity, l_{bm}, N_{inf} and l_{gt} are all set to 1.

Parameter N_{gen}, N_{ind}, and N_{clones} are selected via a test run with total sequence K of

Table 4.9: *Planning cost for various N_{gen}, N_{ind}, and N_{clones} values*

N_{gen}	N_{clones}	N_{ind}				
		3	4	5	6	7
4	2	19.7	20.2	18.6	19.1	18.8
6	2	19.3	18.5	18.7	17.9	18.2
8	2	18.4	18.1	18.2	17	16.2
10	2	17.3	17.2	16.8	17.2	16.3
4	3	18.2	19.5	17.7	18	17.6
6	3	18	17.6	17.6	17.5	17.4
8	3	17.7	17	16.6	17	16.7
10	3	16.4	16.8	16.3	16.2	16.2
4	4	19	18	17.7	17.8	17.4
6	4	17.2	17.1	16.8	17	16.9
8	4	16.6	16.8	16.7	16.7	16.6
10	4	16.7	16.2	16.3	16.4	16.1

10, with optimal solution having 4 activity sequence. UB_Refine, LSh_Check and local search are not applied in this test. Table 4.9 shows the planning cost and Table 4.10 shows the training time for various values of N_{gen}, N_{ind}, and N_{clones}. Only the relevant parameter settings are shown in the tables.

From the tables, N_{gen} of 8 and 10 has a more consistent result with increasing N_{ind}. Besides, general performance is better compared to the lower N_{gen} values. The downside is their time consumption in order to obtain the solution. For the subsequent tests, $N_{gen} = 10$, $N_{ind} = 3$, and $N_{clones} = 3$. From the tables, with $N_{gen} = 10$ and $N_{clones} = 3$, the performance is more consistent and nearer to the optimal solution. N_{ind} will remain at 3 due to time consumption constraint.

4.6.7.3 Planning Performance without Time Threshold

Planning test is performed with UB_Refine, LSh_Check and BMA local search. As mentioned in Section 4.6.3, UB_Refine will loop through the activity sequence and checks whether their omission will produce better results. LSh_Check will check whether branching out of an activity is redundant or not based on the local search look up explained in Section 4.6.6. Total sequence K is set to 20. A set of goals is designed such that the number of activities of their optimal plan lies between 4 and 8. Test is performed on 4, 6 and 8 optimal plan sequence (OPS), where the designed goals are randomly selected.

For each OPS, 5 sets of modifications are tested, which are, 1) planning without UB_Refine, LSh_Check and local search 2) planning with UB_Refine and without

Table 4.10: *Planning time (seconds) for various N_{gen}, N_{ind}, and N_{clones} values*

N_{gen}	N_{clones}	N_{ind}				
		3	4	5	6	7
4	2	5.1	5.3	7.5	8.8	10.3
6	2	7.8	8.5	10.8	14.9	14.5
8	2	8.1	12.9	16.5	19.2	20.3
10	2	13.2	11.8	33.2	19.2	32.9
4	3	6.6	7.4	10.5	11.8	14.1
6	3	8.7	12.2	11.6	23	28.9
8	3	11.9	22.8	19.9	26	32.3
10	3	13.9	24.4	30.2	22.9	25.3
4	4	9	9.1	12.1	13.3	14.6
6	4	10.3	18.7	26.7	20.4	34.2
8	4	19.2	19.9	28.2	25.3	39.4
10	4	20.5	38.5	30.2	34.5	36.5

LSh_Check and local search 3) planning with LSh_Check and without UB_Refine and local search 4) planning with both UB_Refine and LSh_Check but without local search, and 5) planning with UB_Refine, LSh_Check and BMA local search. Modification 1 to 4 can be seen as Bacterial Evolutionary Algorithm, while Modification 5 is the BMA. Each modification is run 20 times. There is no time threshold, therefore, for every test run, the planner will proceed until full search is completed. Figure 4.5 shows the result on distance from optimum cost, with their planning time shown in Figure 4.6.

It can be observed that inclusion of UB_Refine and LSh_Check can help improve performance in terms of approaching the optimum cost, although LSh_Check has a wider distribution compared to UB_Refine if applied individually. With the inclusion of local search, the performance is further enhanced.

In terms of consumption time, as expected, the more activities are required, the slower the planning time. But the inclusion of local search greatly reduces planning time, as this may due to the fact that BMA has a higher chance of obtaining good combinations given reparation from the local search.

4.6.7.4 Planning Performance with Time Threshold

Previous test shows that time consumption is a significant issue, where planning time may reach several minutes. To increase OPS further becomes impractical.

Alternatively, instead of waiting for the planner to finish, the best possible solution within a certain time limit can be used instead. This is likened to the planner generating

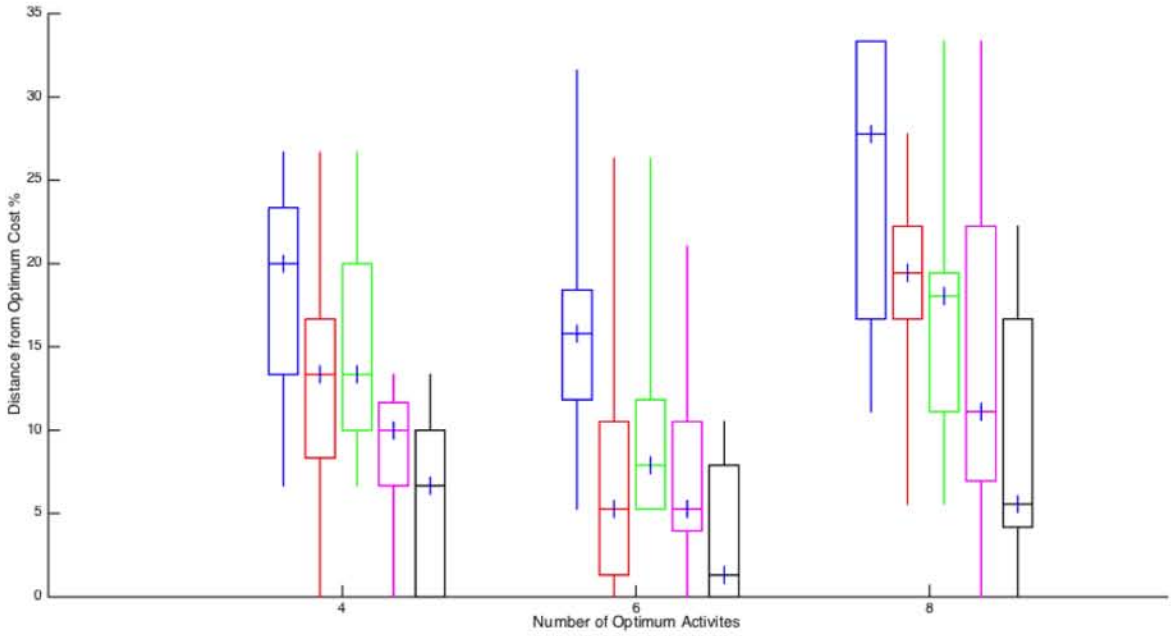


Figure 4.5: *Distance from optimum cost (in percentage) Test result for planning without time threshold. Modification from 1 to 5 (explained in Section 4.6.7.3) are indicated by color blue, red, green, magenta and black respectively. The top of the box indicates the 3rd quartile, and the bottom of the box indicates 1st quartile. The cross indicates the median. Line extension shows the minimum and maximum of the collected test samples. Same indicators apply to subsequent graph.*

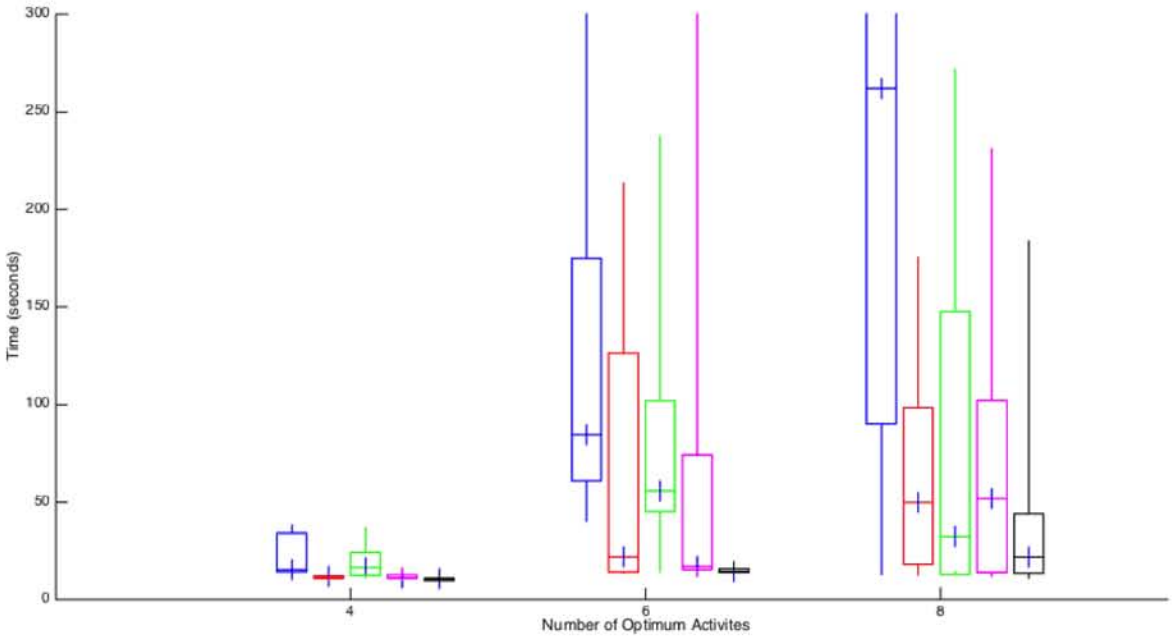


Figure 4.6: *Planning time test result for planning without time threshold*

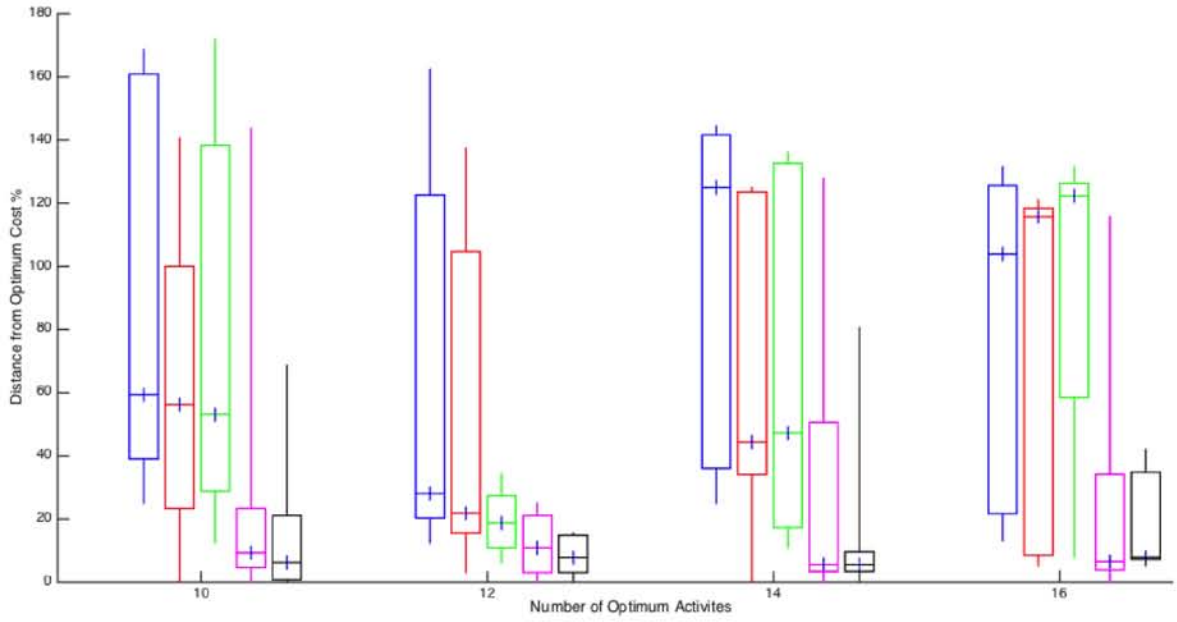


Figure 4.7: *Distance from optimum cost (in percentage) test result for planning with time threshold of 1 minute*

the first few moves for execution, before proceeding further to eventually complete the whole process.

To test on this idea, a time threshold on 1 minute is used to obtain plans of OPS from 10 to 16. If planning process doesn't end after 1 minute, the best solution is obtained instead. Experimental procedure is the same as in Section 4.6.7.3. Result of distance from optimal cost is shown in Figure 4.7. Consumption time is not examined as most of the planning process exceeds threshold time.

UB_Refine and LSh_Check don't fair so well by themselves. From time consumption result shown in Figure 4.6, both of them when applied individually, requires longer searches for OPS of 6 and 8. Cutting them short by introducing a time threshold of 1 minute forces the current best solution to be used, which is sub-optimal. Combining them, coupled by the local search enables the planner to obtain significantly better plans. Spread of distribution of performance is also reduced at a significant magnitude, indicating a more consistent performance given local search is employed for gene reparation in BMA.

The performance test from Figure 4.7 is conducted with the assumption that the optimum plan is to be obtained given only 1 planning being done. The bad performance of the distribution is mainly due to unfulfilled goals (since goals impose the most weights). One planning is fairly restrictive, given that in real life, planning and execution can be done sequentially. This means, planning only needs to come up with partial plan to be executed,

Table 4.11: *Performance for continuous planning for modifications 1 to 5*

Modification	1	2	3	4	5
Difference from Optimum	30.3%	4.5%	15.4%	2.2%	2.0%

before proceeding to the next plan, until all goals are fulfilled, which is obviously more efficient. Such continuous planning is tested with 16 steps of optimal plan. Planning (and execution, which is just the manipulation of state variables) is run until no plans can be devised (which means all the activities in the $K - 1$ sequence of activities are all Nop). Threshold is 1 minute for each planning process. Performance is shown in Table 4.11. As can be observed, performance drastically increases, which shows that although devised plan given one planning is done may give worse result, this is due to the plan being an incomplete plan. Given multiple implementation of the planner (with execution), it will approach the optimal solution. The bad result of modification 1 and 3 is due to redundant activities that contribute more costs. For other modifications, the main reason why they can't reach 0% difference from optimum is due to goals that are related to conditions on integer (For example, $3 \leq L1 + L2 + L3 + L4$, where there are specific activities to set $L1$ to $L4$ 1 or 0 with different weights). As our current sub-goals doesn't take integer into account, the planner doesn't know whether it is getting nearer or further away from a solution related to integer. It only consider whether the constraint is fulfilled or not. In this case, it may choose activities with higher costs to fulfill the goal. Future work will involve generating sub-goals for constraints involving integers to improve branching factor.

4.6.8 Remark

The planning problem itself is a mixture of soft and hard constraints. By exploiting the structure of the planning problem itself, it can be represented as a branch and bound search problem. Lower bound is calculated given a relaxed version of the original problem by exploiting the precondition and effects of the activities of the smart home. The relaxed optimization problem can be readily handled by evolutionary methods, such as the BMA in our case. BMA provides a lot of room for future research especially in the realm of its local search.

Results obtained from planning with and without time threshold demonstrate the applicability of our approach. Besides, it also shows the strength of applying local search for more efficient reparation of potential solutions in the BMA.

Yet, the work is still far from complete. At the moment, conversion of sub-goals of the relaxed optimization problem only deals with Boolean data type. More efficient branching

factor can be achieved if other data types are taken into account. Apart from that, faster searches and better heuristics should be devised to achieve better planning performance. Given the structure of the problem, search space can also be reduced by applying bucket elimination [80], where functions of low arity are eliminated before passing the constraint graph to search methods. Possible works such as weighted CSP [81] and partial fulfillment of objective used in [86] can shed some light in this respect.

4.7 Concluding Remarks

In this chapter, automated planning of the smart home through solving its representation of CSP is laid out. Dynamic planning is also implemented to deal with uncertain situation. Given the optimistic nature of CSP planning, certain measures need to be taken to prevent it.

Yet there are some limitations in CSP approach, which are, it is unable to deal with partial goal fulfillment and optimization. Therefore, extension to Weighted CSP is built to handle these issues. Results for up to 16 steps plan ahead shows favorable results.

At the moment, planning via CSP requires rules to be manually written down. No intelligence and reasoning is endowed. In the next chapter, planning is equipped with ontology and reasoner engines to exhibit intelligence service provision.

Chapter 5

Semantic Reasoning in Service Provision

Knowledge representation is required to provide smart home with reasoning capability. This is crucial such that object semantics and their relationships can be inferred. This is also important for device binding which is required in generating rules for automated planning.

This chapter aims to extend the smart home and automated planning by endowing reasoning capability to it, which is handled by Semantic Reasoner Module (SRM). The main focus will be on the construction of the building ontology, such that semantics and relationships between objects in a building can be inferred to assist in service provision. Automatic variable binding based on their semantics will also be dealt with such that planning operators can be generated for automated planning.

Section 5.1 shows the applicability of OWL in knowledge representation for buildings such that reasoning can be achieved. Section 5.2 shows how planning operators are generated from the knowledge constructed by OWL for automated planning. To demonstrate that the system can be extended to support robot planning and reasoning capability as part of the smart devices, Section 5.3 describes how a home service robot can be realized that utilizes the knowledge base of the house.

5.1 Building Ontology

The building ontology defines the relationships between entities and their concepts that are within the building environment. It is stored in the knowledge base and is described according to Web Ontology Language (OWL) ontology that contains structural, type and device information. Room layout, locations and permanent installations such as doors and

windows as well as their properties are stored as graphs. Devices with their functionalities are also encoded. OWL description logic is used as knowledge modeling language due to it being decidable and that it is endorsed by the World Wide Web consortium (which means there are abundant tools and compatibility support).

Apart from mere storage, knowledge representation represented as graphs can be used to generate additional information based on certain ontology schema. Knowledge generation is based on description logic used via inference engine. With this, user does not need to specify every knowledge graph as they can be automatically inferred given appropriate ontology schema. Detailed information on OWL description logic and their modeling method can be obtained from [75, 4].

The knowledge will be used to give smart home intelligence. With intelligence, smart home is able to perform reasoning in making the appropriate decisions. The knowledge will also be used to generate planning operators, which are wrapped up atomic services from devices connected to the smart home.

5.1.1 Case Study: Fire Emergency Planning and Support

This section demonstrates the use of OWL in constructing knowledge representation to provide intelligence and reasoning capability that is decidable for a building. The case is for fire emergency planning and support.

Information system has been widely used in the field of maintenance, designing and manufacturing for factories [98]. In a large building complex with large number of rooms and object distribution, situation can be chaotic given a disaster situation such an earthquake, which can lead to fire in multiple locations of the building. Given the layout of the building and arrangement of crucial objects like those that are combustible, and also depending on the state of doors and windows, locating danger zone and escape outlet can be hard. The spread of smoke and gases, which closely depends on whether certain doors are opened/closed, further complicate the situation. Information on the possible danger zone in times of emergency is thus crucial. Given a building that is equipped with sensors to realize ambient intelligence and knowledge structure of the building, such information can be obtained instantaneously through description logic reasoning. Objects of the building can be assigned with their semantic meanings. This includes the description of the rooms, and their relationships with every other room, such as their adjacency. Description logic is used to define the building as a graph, where objects are classified into classes, with properties to connect between them. The graph generated via description logic can be used for further inference for more data, and also be used for querying.

By modeling a building as a semantic graph described via description logic, syntactic risks of every part of the building (given every part are treated as an individual object) can be inferred using an inference engine or reasoner. With the reasoner, inference can be made in real-time with current situation. This is important as the world is dynamic, such as doors opening and closing or locked, and hazardous materials being moved around.

As situations and the effects of fire are highly unpredictable, axioms obtained from description logic are of course not sufficient to model the real environment. Building fire emergencies need to take into account not just the surrounding, but also the flow-of-influence for object states and distributions, which requires logical constructs. One can resort to densely modeled rules (which is extremely tedious and difficult to expand) or inferred from probabilistic or analogous models like probabilistic graphical model and SVM respectively (though they cannot directly support logical constructs and notion of causality). Our intention is to provide the latter with representations that are generated from the logical constructs of the description logic. Case studies done are to demonstrate the influence of flow due to object relationships inferred via OWL DL, where it is decidable. Learning models can use the stable representation to acquire more accurate depiction of the environment.

Ontology is setup such that knowledge can be represented according to its schema, which can support reasoning on possible dangers and escape outlets. In this work, emphasis is given to dangers from fire emergency. In such a case, building layout and location of combustible objects as well as possible escape outlet are important in determining the magnitude of danger of different locations of the building.

Therefore, the building ontology is built according to the building layout (rooms, doors, windows), and the classification of objects (package, combustible materials). Ontology is built based on the OWL DL language. Due to the limitation imposed on OWL in order to ensure that inference can always terminate, certain crucial functions such as implication rules cannot be directly modeled using the language. In this case, rules are normally modeled through SWRL. In this work, we use the query language SPARQL (which is the query language endorsed by W3C) to perform the work of SWRL. SPARQL is a descriptive query language that is able to support the implementation of rules and assertion of new axioms in the knowledge base [4]. But care should be exercised when using SWRL or SPARQL to avoid inconsistency of the ontology. For ease of notation in this section, property, which is part of the Resource Description Framework (RDF) triple, will sometimes be represented in predicate form. For example, an RDF triple *"Adam isA Boy"* can be represented as predicate *"isA(Adam, Boy)"*. This predicate will return true if the triple *"Adam isA Boy"* exists in the knowledge base, and false otherwise. Besides that, a class can be shown as concept assertions, where triple *"Dog rdf:type Animal"* can be represented as Animal(Dog).

Also, for new triple assertion, where SPARQL is used, First Order Predicate is represented for ease of reading. For example, given the following formula *isA(?a,Boy)→isA(?a,Human)*, which means that if an instance is related to *Boy* via *isA*, then it is also related to *Human* via *isA*. This formula represents SPARQL Construct query shown below:
CONSTRUCT {*?a :isA :Human.*}
WHERE {*?a :isA :Boy*}
More information on the modeling of ontology can be obtained from [4].

5.1.1.1 Object Classification

For object classification, objects are assigned to classes that are relevant to their properties. It is assumed that the objects involved contain RFID tags with their unique ID that can be easily read. Given objects RFID are detected, they are then represented as instance, and are assigned to a class via *rdf:type* property. Given this, objects can be easily assigned to their attributes (combustible, explosive, acidic etc.), functions (product2, fire extinguisher), locations (room1, kitchen) and package group. For simplicity, only relationships that are relevant to the case studies are described.

Every object is represented as a node in the semantic network graph, which is also an instance. Objects will be assigned IDs via property *hasID*. Certain objects may come in groups with other objects, such as materials in a luggage or a container. Since the group is also an object, it has its own reified node. Objects that belong to the group are related to it through the property *inGroup*. Group objects have their own unique class to encase all the objects that belongs in it. Therefore, for every group object, a class is created, equipped with the restriction (in Turtle):
[a owl:Restriction;
owl:onProperty :inGroup;
owl:hasValue G].
where *G* is the instance of the group. This means, any instance that is related to the instance *G* through property *inGroup* will be assigned under the unique class corresponding to *G*.

Objects will also be assigned attributes. Attributes that are relevant to the subsequent case studies are being combustible, explosive and acidic. These attributes are represented as classes, where objects with such attributes will be assigned to the relevant class. Note that these classes may overlap. The three attributes (*combustible, explosive, acidic*) are subsumed by the class *dangerousMaterials*. To represent the location of objects, every object instance is related to a location instance through property *locatedAt*. Care needs to be taken when

specifying locations for objects in groups, such that all of them, including the group itself, are pointing to the same location node. It is assumed that there are means to detect humans in the rooms, either from direct human detection via camera or RFID. A room is assigned under *haveHuman* class if humans are detected. Knowing whether there are humans and their locations in the building is important for rescuers to take further actions, as well as determining whether the humans are under trapped position or not. For the later, it can be represented by the following rule:
$haveHuman(?r) \wedge \neg validEscape0(?r) \rightarrow rescue(?r)$
The rule states that if a room has humans and that it is not an escape path, then it requires rescue by assigning the room under *rescue* class. Details on *validEscape0* are given in 5.1.1.3.

5.1.1.2 Building Layout Modeling

For building layout, ontology needs to be built such that room adjacency and their relationships with any other rooms can be obtained, termed as location ontology. These relationships should also be influenced by the opening/closing of doors, location of escape outlet and where dangers are located.

A door must be associated with two rooms. Its relationship with a room is via property *isDoor*. A door belongs to class *doorOpen* if it is open. Therefore, before inference begins, axioms should be asserted for every door that is open. Doors that are not assigned to this class are considered closed. Two rooms are adjacent if at least one of the doors between them is open. Adjacency between two rooms is defined by the property *isAdjacent*, which has a symmetric property (if room 1 is adjacent to room 2, so is the case for the other way round). Adjacency axiom can be asserted by:
$isDoor(?d,?r1) \wedge isDoor(?d,?r2) \wedge doorOpen(?d) \rightarrow isAdjacent(?r1,?r2)$
Another alternative to the above mentioned rule is to use restriction from cardinality on the property *isAdjacent*. This eliminates the need to explicitly assert axioms of adjacency as it can be taken care of by the inference engine, but preliminary test shows that this approach is very slow. Property *isConnected* is defined by the rule:
$isAdjacent(?r1,?r2) \rightarrow isConnected(?r1,?r2)$

Property *isConnected* is not equivalent to *isAdjacent*. *isConnected* is endowed with transitive property. Whereas *isAdjacent* only links two rooms that are next to each other given that there is a path between them, *isConnected* relates a room (denoted as RoomA) to any rooms that are related to any other rooms that are related to RoomA via *isConnected*.

A room's relationship with windows is made through the property *isWindow*. In this

work, no windows between rooms are assumed, except for windows the links rooms with the outside. There is a node that reifies the state of being outside of the building, which is the outside node. It is treated the same as any other rooms, except that it provides the point of an escape outlet. Windows and doors that leads to outside of the building, given that they are open, will cause the corresponding room to be adjacent to the outside, which is described as:

$isDoor(?d,outside) \wedge isDoor(?d,?r) \wedge doorOpen(?d) \rightarrow isAdjacent(?r,outside)$

$isWindow(?w,r) \wedge windowOpen(?w) \rightarrow isAdjacent(?r,outside)$

Given the above rules, since outside provides the point of an escape outlet, the following rule describes this property:

$isAdjacent(?r,outside) \rightarrow escape5(?r)$

escape5 indicates that the room assigned under it is a potential escape outlet, but not under full certainty, as danger like fire may occur in the same room. The details of the class *escape5* will be given in Section 5.1.1.3.

5.1.1.3 Influence from Adjacency

Given two rooms are adjacent, certain influence of danger or determining whether the current position will lead to an escape outlet can be inferred from the information of the adjacent room. In this sub-section, two influences from adjacency will be explained, which are 1) the spread of smoke 2) Determination of escape outlet

In a building, it is assumed that there are sensors in place that detects fire and smoke. Any room with danger detected will be assigned under the class *roomDanger*. Danger can be known from the sensors on fire and level of smoke, and also whether there are hazardous materials around. The remaining rooms will be assigned *nonDanger* by the following rule:

$room(?r) \wedge notAssigned(roomDanger,?r)) \rightarrow nonDanger(?r)$

where *notAssigned(a,b)* means if *b* is not assigned under *a*.

Smoke level consists of multiple levels, depending on the severity. The number of levels depends on the individual choosing. 6 levels are assigned, starting from level 0 to level 5, where level 5 has the highest severity. A room with a certain level of smoke is assigned to the corresponding class, denoted by *levelClass'n'*. Since there are 6 levels, the classes are *levelClass0, levelClass1, levelClass2, levelClass3, levelClass4* and *levelClass5*. It can be safely assumed that a class of lower level subsumes the higher class. For example, a room with smoke level of 5 will have at least a smoke level of 4, and so on.

Given this assumption, flow of influence due to adjacency can be modeled by imposing restriction on the level class (in Turtle):

levelClass'n' owl:equivalentClass
[a owl:Restriction;
owl:onProperty :isAdjacent;
owl:someValuesFrom levelClass'n+1'].
In this case, the smoke level of a room can cause rooms adjacent to it to be assigned to the class of lower smoke level.

As mentioned before, determining escape outlet can also be influenced by information of the adjacent room. Like wise to smoke level, we will assign 6 levels to signify how near a particular room is to the nearest escape outlet, where the level class is denoted as *escape'n'*, namely *escape0, escape1, escape2, escape3, escape4* and *escape5*. The higher the level, the nearer the room is to an escape outlet. Also, lower level class subsumes the higher level class. But the flow of influence is not as direct as that in the case of smoke. A room that is deemed dangerous should not propagate influence regarding escape outlet. That means, flow of influence only works for those rooms assigned under the class *nonDanger*. Lets denote the intersection between class *escape'n'* and *nonDanger* as *validEscape'n'*. Restrictions are then applied to this class instead as (in Turtle):
escape'n' owl:equivalentClass
[a owl:Restriction;
owl:onProperty :isAdjacent;
owl:someValuesFrom validEscape'n+1'].

5.1.1.4 Multistage Inference

Inference for the state of the building cannot be done in one stage, where certain axioms can only be asserted given previous axioms, such as the determination of escape outlet that needs information on the *nonDanger* class. In this case study, the number of stages is limited to two.

In the first stage, the core ontology is asserted, which includes the symmetric property of *isAdjacent*, restrictions on *levelClass'n'*, subsumption for *levelClass'n'*, etc. The core assertions for the first stage are shown in Table 5.1. In this stage, axioms asserted from sensors and assertion of adjacency from opened doors are also performed, which are:
1) $isDoor(?d,?r1) \wedge isDoor(?d,?r2) \wedge doorOpen(?d) \rightarrow isAdjacent(?r1,?r2)$
2) *Assigning rooms to roomDanger if sensors pick up any danger signal*
For the second stage:
$room(?r) \wedge notAssigned(roomDanger,?r)) \rightarrow nonDanger(?r)$
$haveHuman(?r) \wedge \neg validEscape0(?r) \rightarrow rescue(?r)$

Table 5.1: *First Stage Assertions*

No.	Assertion
1	Restriction to $levelClass4$ with $owl:someValuesFrom$ on property $isAdjacent$ to $levelClass5$
2	Restriction to $levelClass3$ with $owl:someValuesFrom$ on property $isAdjacent$ to $levelClass4$
3	Restriction to $levelClass2$ with $owl:someValuesFrom$ on property $isAdjacent$ to $levelClass3$
4	Restriction to $levelClass1$ with $owl:someValuesFrom$ on property $isAdjacent$ to $levelClass2$
5	Restriction to $levelClass0$ with $owl:someValuesFrom$ on property $isAdjacent$ to $levelClass1$
6	$validEscape5$ as intersection of $escape5$ and $nonDanger$
7	$validEscape4$ as intersection of $escape4$ and $nonDanger$
8	$validEscape3$ as intersection of $escape3$ and $nonDanger$
9	$validEscape2$ as intersection of $escape2$ and $nonDanger$
10	$validEscape1$ as intersection of $escape1$ and $nonDanger$
11	$validEscape0$ as intersection of $escape0$ and $nonDanger$
12	Restriction to $escape4$ with $owl:someValuesFrom$ on property $isAdjacent$ to $validEscape5$
13	Restriction to $escape3$ with $owl:someValuesFrom$ on property $isAdjacent$ to $validEscape4$
14	Restriction to $escape2$ with $owl:someValuesFrom$ on property $isAdjacent$ to $validEscape3$
15	Restriction to $escape1$ with $owl:someValuesFrom$ on property $isAdjacent$ to $validEscape2$
16	Restriction to $escape0$ with $owl:someValuesFrom$ on property $isAdjacent$ to $validEscape1$
17	Restriction to $escape0$ with $owl:someValuesFrom$ on property $isAdjacent$ to $validEscape0$
18	$escape5 \subseteq escape4 \subseteq escape3 \subseteq escape2 \subseteq escape1 \subseteq escape0$
19	$levelClass5 \subseteq levelClass4 \subseteq levelClass3 \subseteq levelClass2 \subseteq levelClass1 \subseteq levelClass0$
20	$Combustible \cup explosive \cup acidic \subseteq dangerousMaterials$
21	$owl:SymmetricProperty$ for $isAdjacent$
22	$owl:TransitiveProperty$ for $isConnected$

are run. Inference after this will show possible escape outlet for every room.

5.1.1.5 Design of Case

Case studies are performed to demonstrate the applicability of the designed building ontology to handle emergency situation. The main focus is on the detection of danger zones and determining escape outlet given the point of disaster, distribution of objects, state of doors and possible escape outlets. For this case study, FaCT++ [132] is used as the description logic inference engine. Rules and axiom assertions are implemented through SPARQL, which, in this work, is uniquely written to work with FaCT++.

Figure 5.1 shows the building layout used for the case studies, where blue rectangle indicates doors, red rectangle indicates windows. The figure shows rooms (indicated by 'R'), doors (in blue indicated by 'd') and window (in red indicated by 'w'). Given the windows and doors allocation, room R1, R2, R3, R6, R8, R11, R15, R18, R20 and R22 have the potential of being escape outlets if the doors or windows are opened.

Three configurations are demonstrated.

Configuration 1:

Fire occurs in corridor R10. There is an object that is combustible in room R13. Doors d11, d12, d13, d14 and d23 are opened. All windows are closed.

Configuration 2:

Fire occurs in corridor R12 and R11. There is acidic material in room R4. Doors d1, d2, d3,

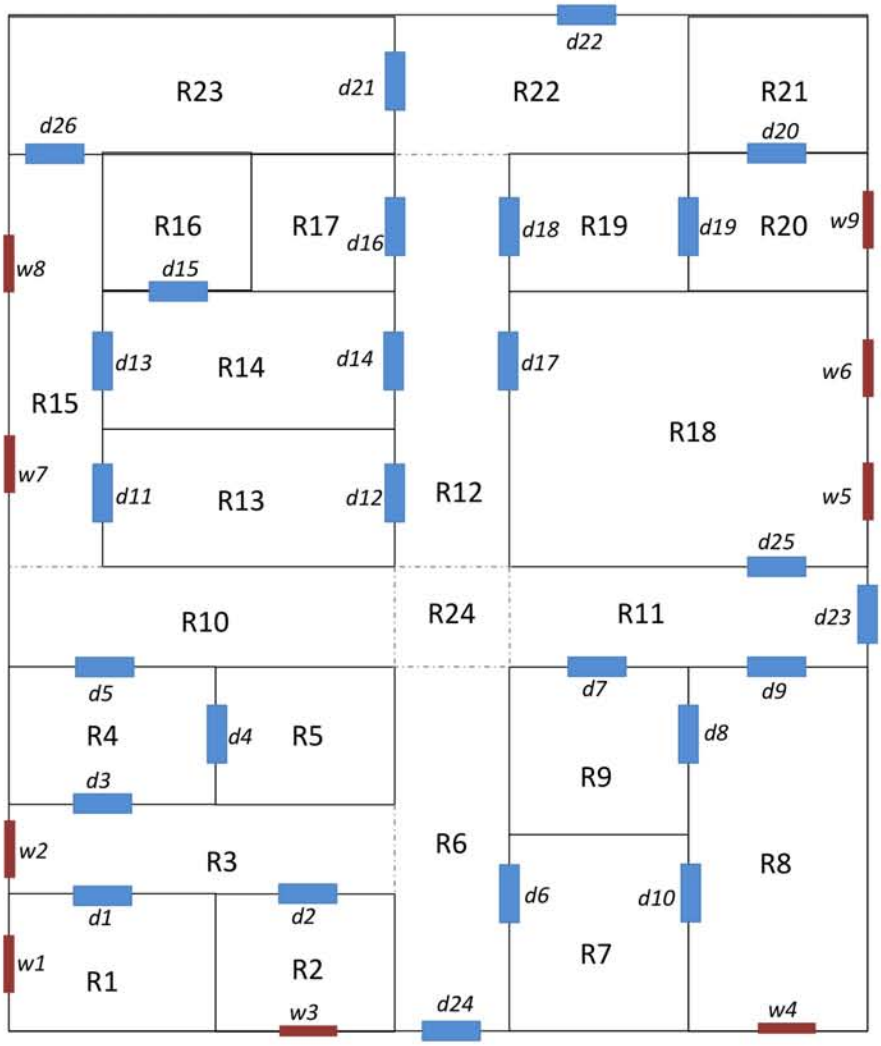


Figure 5.1: *Building layout*

d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d21, d22, d23 and d26 are opened. Window w6 is open.
Configuration 3:
Fire occurs in R3 and R12. All doors are open, and all windows are closed. There are humans in R1, R21 and R22.

5.1.1.6 Smoke Hazard Visualization

In times of fire, it is generally useful to know how the smoke will spread given the condition of doors. This can help escapees and rescuers evaluate the danger of the particular section of the building. Test is run on the three configurations stated in Section 5.1.1.5. Figure 5.2 shows the visualization of smoke spread obtained from inference. Smoke spread information can be obtained at the first stage of multistage inference.

For configuration 1, fire starts at R10. Smoke is at the highest level in R10, thus, R10 is assigned under class *levelClass5*. Smoke spread according to the state of opened doors. Given the adjacent rooms, smoke level is inferred for rooms surrounding the point of fire. Rooms that are unmarked by colors indicating different levels in the figure are considered safe from the smoke.

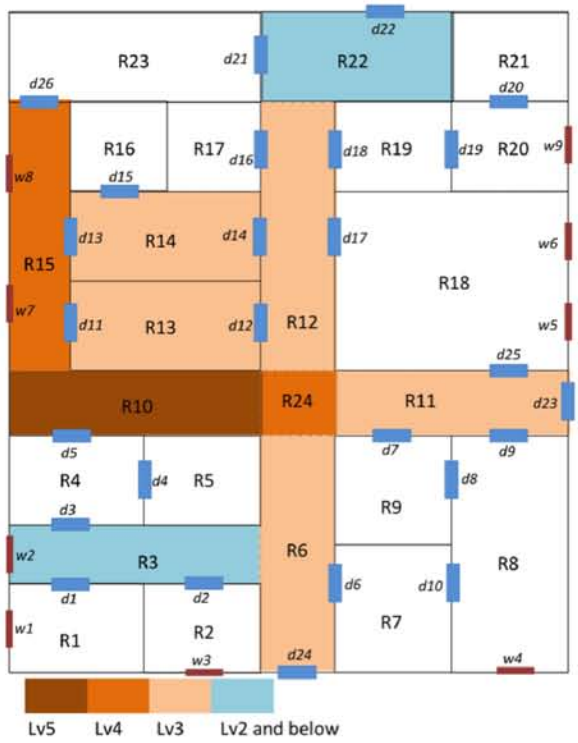
For configuration 2, fire starts in R12 and R11. Configuration 2 has more number of opened doors compared to configuration 1, thus, the spread of smoke is further. In this configuration, the spread of smoke is important in determining which rooms require more assistance. This can be observed from room R23, R15 and R10, where there is a constant level of inferred smoke level. Whereas, for rooms R1, R2, R3, R4 and R5, the smoke level is significantly lower. In this case, this information is important for both the escapees and rescuers to evaluate the overall situation in terms of smoke propagation.

Since all doors in configuration 3 are opened, visualization shows that the smoke will spread throughout the building.

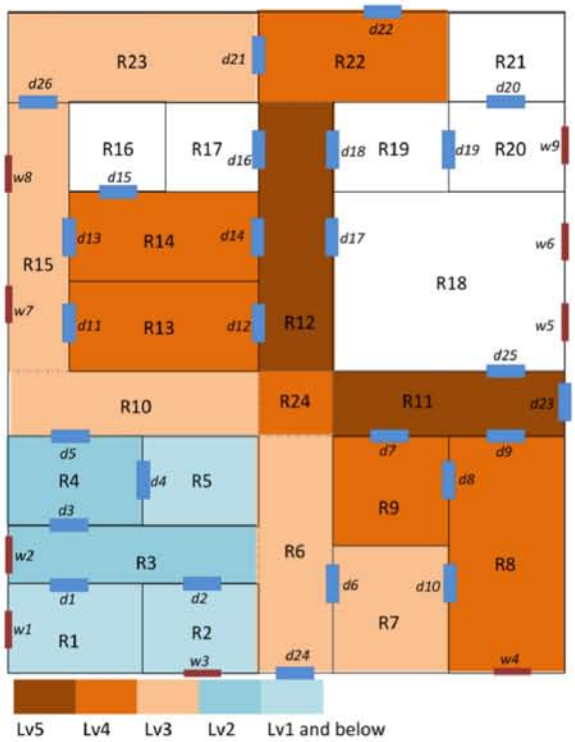
5.1.1.7 Escape Outlet Visualization

After the second stage of the multistage inference, escape outlet can be obtained. Figure 5.3 shows the visualization of escape path obtained from inference. Rooms that are not shaded are considered not part of the escape path.

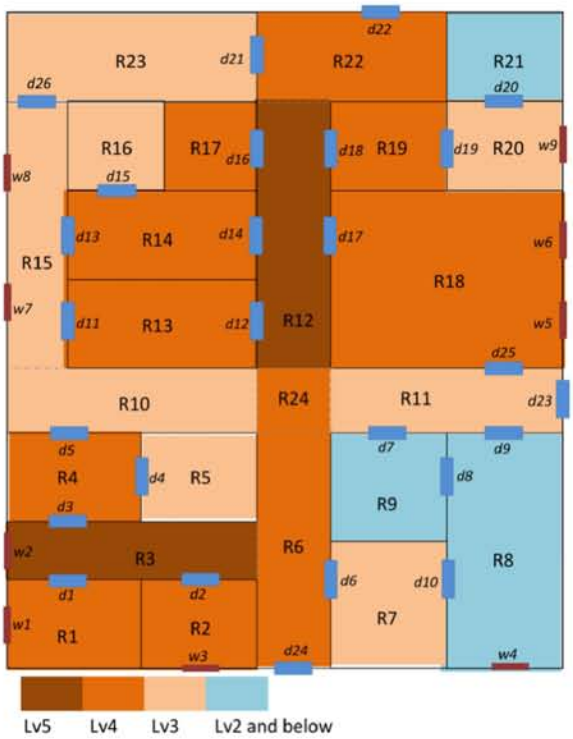
For smoke spread information obtained in Section 5.1.1.6, smoke propagation does not give details of where a person should be headed to. Besides, distribution of hazardous materials will complicate the situation, causing people to take inefficient or wrong routes to safety. Information on such path, which takes into account points of danger, is important



(a) *Configuration 1*

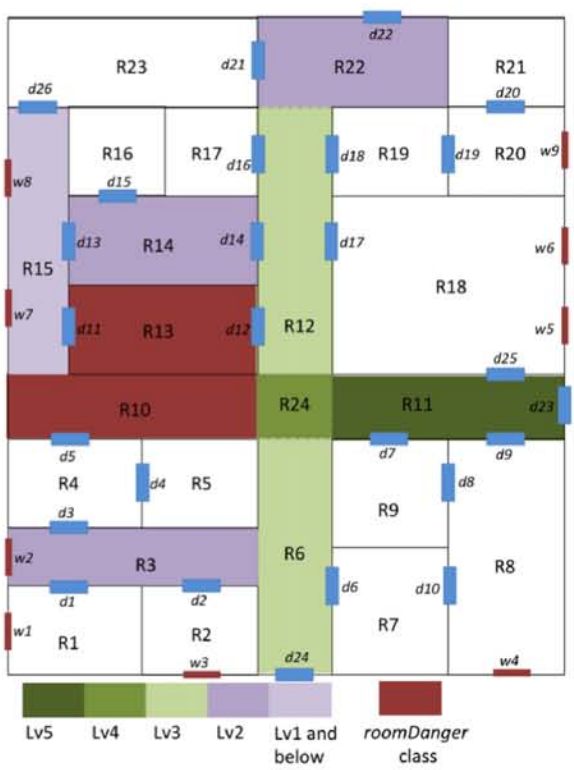


(b) *Configuration 2*

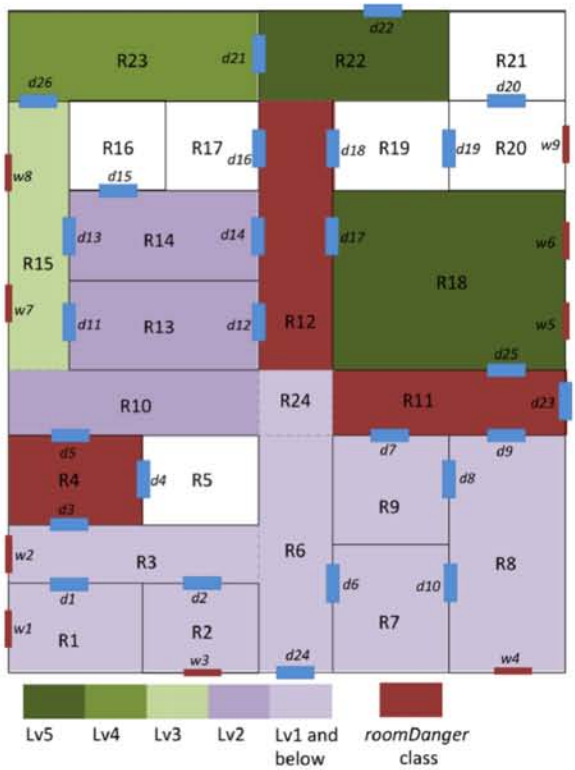


(c) *Configuration 3*

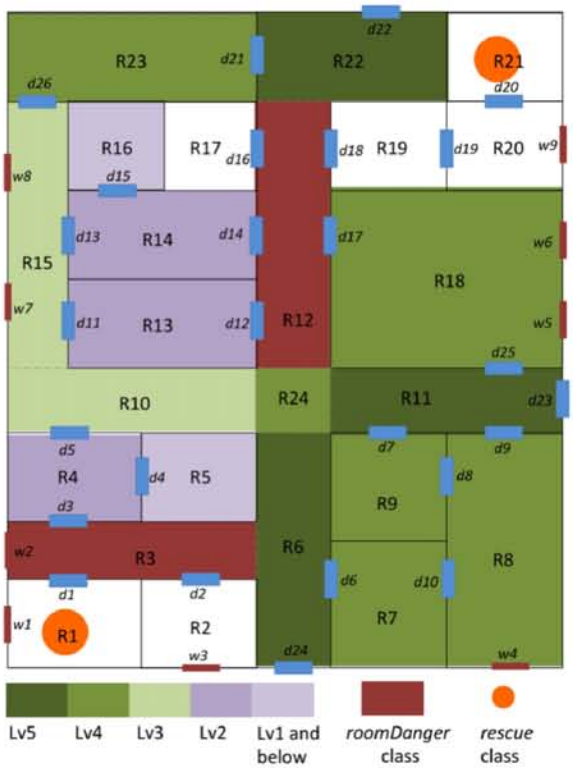
Figure 5.2: *Visualization of smoke spread*



(a) *Configuration 1*



(b) *Configuration 2*



(c) *Configuration 3*

Figure 5.3: *Visualization of escape outlet*

as it assists in deciding directions to take in order to reach a potential escape outlet.

Configuration one only has d23 opened, leading to the outside. Therefore, this configuration only has one escape outlet. Due to fire occurring in R10, and the presence of hazardous materials in R13, the figure clearly shows the desirable path for people in R15 is to go through R14 instead, which can be inferred via building ontology.

Configuration two shows two escape outlet of level 5, which is R22 (has a door to the outside) and R18 (has a window). Although d23 is opened, which makes R11 potentially a level 5 escape outlet, but due to fire occurring in R11, this information is not propagated further. For people in rooms other than R18, their only outlet is through d22 in R22. From the result, human may know that they need to approach R15 in order to reach the escape outlet safely, as other ways are blocked. Path via R4 is not being considered a path as there are hazardous materials inside, thus, people from R1, R2 and R3 should approach R6 to R24 and finally R10 to reach R15 to ensure their safety.

Result from configuration two clearly illustrates the point as stated earlier that smoke spread does not provide sufficient information on the escape route. From Figure 5.2b, it seems R1, R2, R3, R4 and R5 are the places to be to avoid smoke. But these locations are dead end, and therefore, from Figure 5.3b, it is wiser to follow a safe route to the nearest escape outlet.

Configuration three has humans in rooms R1, R21 and R22. From the result, R1 and R21 is assigned under rescue class because both these rooms are not escape paths. R1 is blocked by the fire in R3, and R21 is blocked by the fire in R12. Humans in both these rooms cannot easily reach any potential escape outlets. This information is crucial for rescuers to assess the situation and reach the victims. For humans in R22, since the room is an escape path, it is not assigned under rescue class.

5.1.1.8 Remark

Two-stage inference of building ontology is built via description logic to support human in times of fire emergency for escapees and rescuers. Given the state of doors and windows and also the distribution of hazardous materials, as well as the general layout of the building, human judgments are negatively affected that is aggravated by the chaos of the situation. Building ontology can provide information on smoke propagation and determination of escape outlet, which is obtained through multistage inference.

The case demonstrates the effectiveness of knowledge representation via OWL, where intelligence can be endowed to support further inference as well as querying.

5.2 Semantic-based Variable Binding for Rule Generation

This section aims to extend the Informationally Structured Space (ISS) of the smart home by integrating CSP used for dynamic planning with description logic applied in knowledge representation, such that reasoning capability is enabled to generate planning operators based on existing devices for service composition. It automatically generates the required rules for service composition derived from individual devices that are connected to the home, where their semantic information is represented in accordance to appropriate ontology schema.

5.2.1 Variable Binding Preliminaries

Figure 5.4 shows a subset of the ontology, where the superclass is $Entity$ with 3 direct subclasses dealing with type, device and building layout information. Ontology branching from $type$ represents the ontology schema for semantic representation dealing with categories of objects. $Layout$ is the ontology schema that relates the different rooms and installations. Ontology extended from the $Device$ class consists of controllable (devices that can be directly controlled by the ISS) and non-controllable (devices that cannot be directly controlled by the ISS) objects. Under non-controllable class, there is the requestable (objects that can be requested to perform certain task such as requesting human to bring out the garbage) and non-requestable (objects that are static like cups) class. Every object under $Device$ class is associated with a location. Devices under controllable class are associated with services that the ISS can call.

Every device that is within the class $Controllable$ shown in Figure 5.4 that is connected to the ISS is represented as a structure shown in Figure 5.5. These devices possess services called atomic services that can be controlled directly by ISS to fulfill goals.

Automatic service composition requires preconditions and effects to work. All atomic service belongs to class $classService$. Apart from variables from preconditions and effects, other parameters that the service requires for proper functioning but is not included as precondition and effect variables for planning purposes are also crucial. Service composition is implemented via constraint processing, which is explained in Chapter 4.

All variables related to the preconditions, effects and other parameters of atomic services belonging to a device will be associated with type classes. These type classes provide a way to determine the semantic meaning of the variables. All variables either belong to $stateVariable$ or $deviceVariable$ class. $stateVariable$ is a class of variables that can be

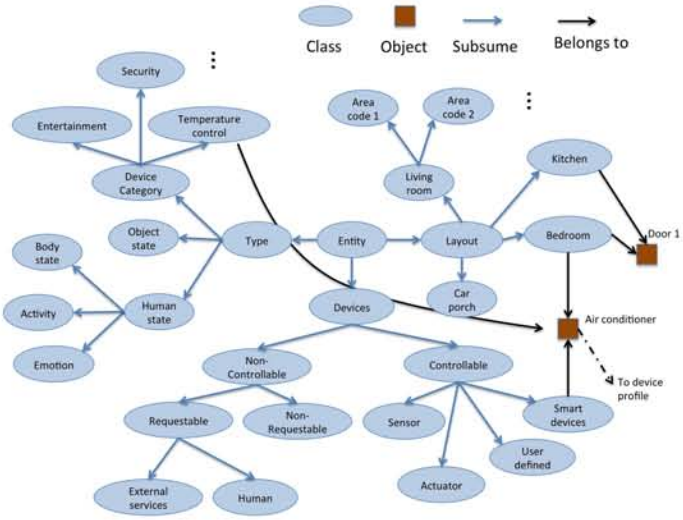


Figure 5.4: *Building ontology*

controlled by services from other devices. $deviceVariable$ is a class of variables that can only be changed by its own device. $classVariable$ is a disjoint union of $deviceVariable$ and $stateVariable$, as specified in the axioms shown in Table 5.2.

As the precondition and effect variables are required for service composition, which is done through logic programming, due to the fact that individual devices know neither the functionalities nor the variables of other devices, it is the task of the ISS to bind the variables by inferring their meaning through the concepts they are associated with. Binding can also be done through special queries between variables of different devices.

A subset of important axioms of the building ontology for variable binding is shown in Table 5.2. These axioms are used to infer additional facts about whether certain services can be used by ISS or not. Services that have some of their precondition variable not bounded to any fixed variable (variables that belong to the class $classVariableFixed$) cannot be used by ISS since the value of unbounded variable can never be determined during run-time, otherwise, the service will belong to class $classServiceFixed$. Any variable that is an effect variable of services in $classServiceFixed$ belongs to class $classVariableFixed$. There are no axioms that converts fixed variable to unbounded variable, thus, the process of finding fixed service will always terminate.

Table 5.2: *Axioms for variable binding*

Axiom Syntax	Description
$isServiceTo \equiv hasService^{-}$	Property $isServiceTo$ is inverse of $hasService$
$isPreconTo \equiv hasPrecon^{-}$	Property $isPreconTo$ is inverse of $hasPrecon$
$isEffectTo \equiv hasEffect^{-}$	Property $isEffectTo$ is inverse of $hasEffect$
$\exists isServiceTo.\top \sqsubseteq classService$	Restrict domain of $isServiceTo$ to $classService$
$\exists isEffectTo.\top \sqsubseteq effectVariable$	Restrict domain of $isEffectTo$ to $effectVariable$
$\exists isPreconTo.\top \sqsubseteq preconVariable$	Restrict domain of $isPreconTo$ to $preconVariable$
$classVariable \equiv deviceVariables \sqcup stateVariable$	$classVariable$ is a union of $deviceVariable$ and $stateVariable$
$deviceVariable \equiv \neg stateVariable \sqcap classVariable$	$deviceVariable$ and $stateVariable$ are disjoint
$constant \sqsubseteq classvariableFixed$	A variable C is a fixed variable if it is of type $constant$. During planning stage, every C will be given a unique number. Every C is identified by $C\_$ or $Ca\_$ as the first few letters of the variable during planning stage
$classServiceFixed \equiv \forall hasPrecon.classVariableFixed$	Universal restriction on $classServiceFixed$ that all instances related through property $hasPrecon$ has to be in $classVariableFixed$. Intuitively, it means for a service to be considered usable, all the variables associated with its preconditions and parameters need to be fixed variables
$classVariableFixed \equiv \exists isEffectTo.classServiceFixed$	Existential restriction on $classVariableFixed$ that some instances related through property $isEffectTo$ has to be in $classServiceFixed$. Intuitively, it means a variable is considered fixed if there is at least one usable service where the variable is the effect of that service

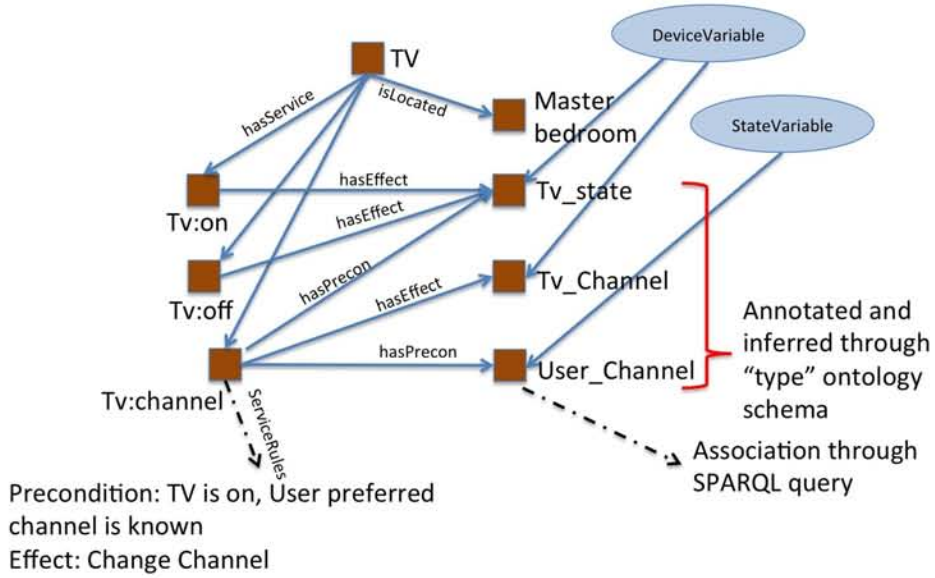


Figure 5.5: *Graph representation of device and its service*

5.2.2 Variable Binding Procedure

Figure 5.6 shows the process flow to bind variables. After the knowledge base is initialized, building ontology is loaded into the knowledge base by the first module.

The second module generates necessary data from the devices before being stored in the knowledge base. Devices dealt here are all under class $Controllable$. Every device comes with a specification that describes their services, variables and their semantics, possibly the device location and query information for further association. In the specification, every service also have a set of logical description that is coherent with the specification itself, which is used for constraint programming. Figure 5.5 shows an example specification for a TV. The TV has atomic services for turning on and off the TV, and a service for changing the channel. Services for switching the TV on and off only has effect variable, Tv_state, which is the variable indicating the on and off state of the TV. For service to change channel, it has precondition variables to check whether the user preferred channel is known and that the TV is on. TV_state and $Tv\_channel$ are both $deviceVariable$ since no other device can turn the TV on and off and change its channel other than itself. $User\_channel$ is under $stateVariable$ since the variable can be set by other services that can provide user preferred channel. Given the devices' specification (as obtained from object repository), all devices, their services and variables are given unique IDs, which the ISS will address. The original variable name from their own specification file is not used because two different variables from different devices may have the same name, due to the fact that devices don't know

the existence of each other.

But, there are some situations where two devices might need to co-operate with each other, where this knowledge of co-operation might be learnt or manually set by the user. An example is a casement window and curtain, where both have motorized control. If the curtain is installed for the casement window, when the window is to be opened, it needs to have an additional precondition, which is, the curtain needs to be open. Adding inter-device link performs such inclusion of additional precondition. Inter-device link is a file containing information of how the rules and variables from different devices should be linked, which is defined by user at the current stage. One can refer to [7] as a possible candidate for learning such links. Given the link, additional precondition will be added into the corresponding service. From the example of the casement window, a precondition of curtain being open is added into the window's service of opening the window.

Variables from different devices that are of type $stateVariables$ and is related to the devices' service by property $isEffectTo$ can be combined at the $Combine\ Similar\ Variable\ module$ if they have the exact same combination of class types (this includes the service type and device type of the variable). A crude method of comparison is done, where no reasoning is used to determine whether these variables are similar or not.

With the axioms generated from previous modules being fed into the knowledge base, association of precondition variables under $stateVariables$ is performed through querying. Querying is done by SPARQL. Using the previous example shown in Figure 5.5, $User\_channel$ by itself is of no use, unless it is associated with a variable (on user preferences) under the class $classVariableFixed$. Its potential association is written as a SPARQL command. For example, a variable $AudioFeed$ from an audio device $Audio$ is to be associated with an audio output of a device of the same location as the audio device with format X. The query can be written as:

SELECT ?$AudioFeed$ WHERE {
?a $hasService$?$AudioFeed$.
?a $isLocated$?b.
?a a X.
$Audio\ isLocated$?b.}

The variable will be associated with returned result. For every queried variable that is associated with the precondition variable, a new service is generated for every variable. For example, given that there are 5 equipment that can provide an audio device audio feed and are in the same room as it, given the above example SPARQL query, 5 duplication of the

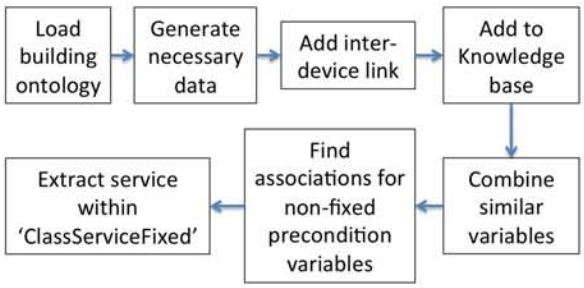


Figure 5.6: *Variable binding process flow*

service will be generated, where each service is associated to a particular device. Although this will cause an increase in the number of services, result in Figure 5.8 shows that this approach is a lot faster compared to constraining the number of services by introducing a mediator service, given that the number of duplicated services isn't too large. Mediator service is a service that deals with routing different variables to one variable that they are associated to.

Finally, services that belongs to $classServiceFixed$ through inference will be extracted. The inferences required are done through the axioms shown in Table 5.2. With the generated services, planning can proceed as how it is described in Chapter 4.

5.2.3 Example for Rule Generation for Planning

This sub-section illustrates how the system works using a simple example of robot announcing the weather. Figure 5.7 shows the devices involved and the relevant services and variables. State variables are $Weather$, $WeatherResult$, $Geolocation$, $GeographicLocation$, $C\_weather$ and $WeatherConstant$. Others are all device variables.

The following are the preconditions and effects for the services (for planning):

$GetWeather$
Precondition: $GeographicLocation_{Known} = true$
Effect: $WeatherResult = WeatherResult_{res}$,
$WeatherResult_{Known} = true$

$Speak$
Precondition: $InPort_{Known} = true, InCat_{Known} = true$
Effect: $SpeakCat = InCat$

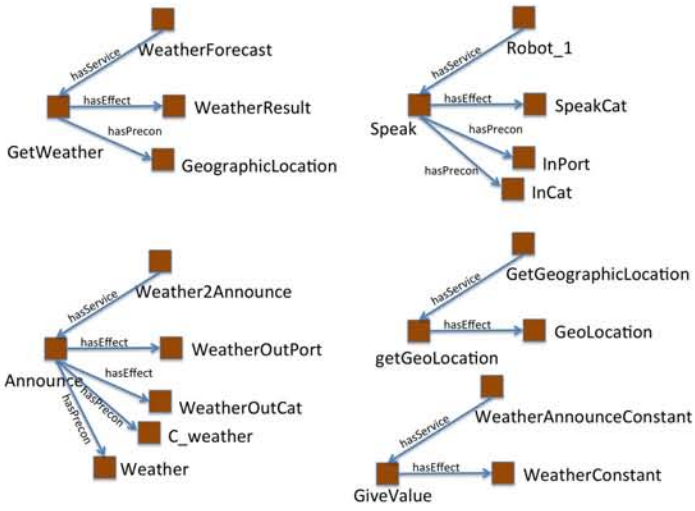


Figure 5.7: *Example graph of devices*

$Announce$
Precondition: $Weather_{Known} = true$
Effect: $WeatherOutPort = WeatherOutPort_{res}$,
$WeatherOutCat = C\_Weather$

$getGeoLocation$
Precondition: None
Effect: $GeoLocation = GeoLocation_{res}$

$GiveValue$
Precondition: None
Effect: $WeatherConstant =$ Unique ID

Assuming annotation are done for all variables and services, $Weather$ will be associated with $WeatherResult$, and $Geographiclocation$ will associated with $GeoLocation$. $C\_weather$ is a constant, and is associated with $WeatherConstant$. Also, assume Variable $WeatherOutPort$ is of type $WeatherAnnouncePort$ class subsumed by $Port$, and $WeatherOutCat$ belongs to class $AnnounceCategory$. For input variable $InPort$ and $InCat$, they have SPARQL query that associates them with variables belonging to $Port$ and $AnnounceCategory$, and that both variables need to be from the same service.

Service $getGeoLocation$ is fixed service because it doesn't have preconditions, making $GeoLocation$ a fixed variable. With $GeoLocation$ being fixed, this turns service $GetWeather$ into fixed service, which in turns convert $WeatherResult$ to fixed vari-

able. All these makes service $Announce$ a fixed service. Finally, for service $Speak$, $InPort$ is associated with $WeatherOutPort$ because $WeatherOutPort$ belongs to $WeatherAnnouncePort$. Since $WeatherAnnouncePort$ is subsumed by $Port$ (which is what $InPort$ is querying), association will be made. For $InCat$, association is made with $AnnounceCategory$, which belongs to the same service as $WeatherOutPort$. With the association, the names $InCat$ and $InPort$ in the preconditions for planning is replaced by $AnnounceCategory$ and $WeatherOutPort$ respectively.

During planning, in order for the robot to announce the weather, all one needs to define for the goal is $SpeakCat = WeatherCat$. With planning via solving the CSP, a sequence of plans is arranged in order for this goal to be fulfilled. A possible solution given by solving the CSP is: $GiveValue \Rightarrow getGeoLocation \Rightarrow GetWeather \Rightarrow Announce \Rightarrow Speak \Rightarrow Ends$

5.2.4 Case Studies

Current home automation system either covers simple service activation with knowledge representation, service composition that requires expert design for complex planning, or service composition without exploiting reasoning. This work uses the method described in Section 4.3 for complex service composition and endows it with reasoning capability. The main concern of this thesis regarding knowledge representation is for variable binding for planning purposes. Relevant devices with their variables, services and semantic markups are shown in Appendix C. Variable binding process uses this specification to link the variables between devices, which is crucial for rule generation for service composition.

5.2.4.1 Case Study Setup

Relevant services related to the case studies are shown in Table 5.3 and 5.4. These services consists of wrapped-up atomic services from the devices, web services and user defined services. The specifications of the devices are shown in Appendix C. Since the main focus of this thesis is not on the middleware, emulator is built to support transfer of messages in the smart home. Reasoner FaCT++ [132] is used for building ontology reasoning, and Z3 [35] is applied for constraint programming. A separate module is also built to support necessary SPARQL querying capability on building ontology supported by FaCT++. The whole system is run on an 2.5GHz Intel Core i5 computer with 4GB RAM.

Modifications is made on response variable and how activities respond to it to solve optimistic planning problem compared to those described in Section 4.4.3. Instead of having response variable fixed throughout the planning process and choosing its value however it

wants fit, we treat response variables just like knowledge and effect variable. The difference is that response variables are initialized to a special value that cannot be possibly returned by the activity dealing with the response variable. We call this special value Nan. Since every response variable has an associated knowledge variable, for every activity that takes this knowledge variable as a precondition argument, upon encountering Nan, the goal is considered achieve (we termed this as Nan response). Now, since Nan is a value that cannot be returned by the associated activity, during execution, Nan response can never be achieved. The planner is forced to re-plan, but this tile, in the absence of the response variable, since the knowledge variable already contains the necessary information.

To illustrate this, for example, the robot wants to ask the user whether he wants to play games or watch TV. Lets assume $UserAnswer$ is the knowledge variable storing the choice of the user, and $resUserAnswer$ is the response variable. During planning (and this is before ask the user), $resUserAnswer = Nan$, which causes the planner to composite a very short plan to pass Nan from $resUserAnswer$ to $UserAnswer$, thus, $UserAnswer = Nan$, upon which the plan ends. Now, during actual execution, $resUserAnswer$ can never be Nan. Therefore, after passing values from $resUserAnswer$ to $UserAnswer$, precondition for default response is not met, forcing the planner to replan again, only this time, $UserAnswer$ contains actual user choice. Intuitively, this approach forces the planner to obtain values for all response variables before starting the actual planning. The downside is that if too much knowledge variables associated with response variables are unknown, the planner will need to replan every time just to obtain their values before engaging in actual construction of plans. This problem can be alleviated by reduction of activity search space as explained in 4.4.1. Constant updating of knowledge variable can also improve the performance.

5.2.4.2 Duplication VS Mediation

Every variable related to the service via $isPreconTo$ will be associated with other variables via pre-specified SPARQL queries. The question is, should a new service be created for every association found? This means, duplications of the same service are made, where all differs in having the original variable replaced by the associated ones. The other alternative is to create an additional mediator service with mediating variable to pass values from different variables to the same service.

Test is performed to compare the planning speed of both approaches and to gain insights. Planning speed comparison for both approaches are shown in Figure 5.8. Speed is compared against the number of mediators used when mediation approach is employed.

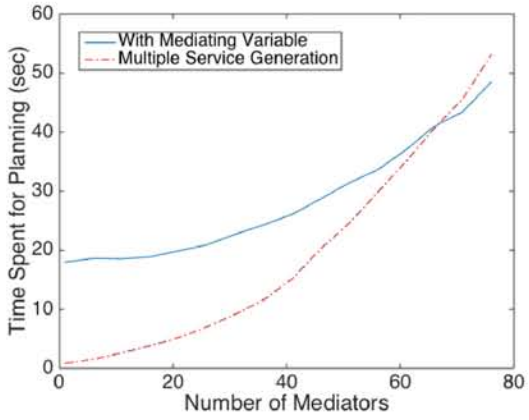


Figure 5.8: *Planning speed*

Mediating variable approach is slower than duplications when number of mediating variables are small. This is because mediating variables put more constraints on the search for possible solutions. But as the number of associations grows, duplication approach will start to slow down dramatically as services start to multiply due to the possible associations. For current system, we use duplication approach as the number of associations is less than 10 associations.

5.2.4.3 Case Description

Case studies are performed to study the effectiveness of the approach in handling complex service composition intelligently. The studies provide insights into their implications and directions for further development.

Here, we provide a story for a more integrated view on the cases. Imagine when the user comes home in the evening, the home should light up the appropriate light and announce unheard messages when he/she is away. If the home doesn't know what the user wants to do, it may query, upon which, it will make the necessary preparations (in the case study, the user wants to - though not limited to - take a shower or watching TV). While the user is at home watching TV, the house detected some movements at the garden. This triggers the security camera at the camera to capture and transfer video to the TV. Appropriate lights are turned on as well to provide sufficient illumination. If the situation is under control, the user can just tell the home to ignore it. Smoke break out in the wet kitchen while the user is in the living room. The home will activate the vent. It will also close the appropriate doors to separate the user from the location where the smoke occurs.

From the story, cases will involve a)preparation made by home automation after query-

ing the user b)device signal connection c)choosing the right device and the number of devices d)device control based on building state and semantics. The purpose of a) is to ensure that certain services are only provided after the home knows about user preferences from querying. b) is dealing with signal routing between different devices. Sometimes, the specific device to fulfill a task is not that important compared to the number of such devices running, which is dealt by c). For d), its purpose is to deal with services that depend on the knowledge representation and state of the home. a) and b) is handled in Case 1, b) and c) is handled in Case 2, and finally, d) is handled in Case 3. Simple service composition and device interoperation is not explicitly shown as they are subsumed by Cases 1 to 3. As stated, the motivation is to acquire crucial insights for further development.

Three case studies are shown in this section, namely 1)Event triggered/Scheduled with query 2)Device Selection 3)Complex rules handling. As discussed in Section **??**, goals for planning are introduced depending on event, schedule and command. We treat schedule triggered goals as a special case of event triggered, and is shown in Case 1. It also involves user querying by ISS to gain more information on user needs. Case 2 aims to get devices connected and right number of devices to work with more elaborate logic rules. Case 3 is on service composition to fulfill complex rules. This work involves knowledge representation for variable binding such that complex service composition can take place. Given knowledge representation described by description logic, semantic information can be inferred and be used in rules. Although reasoning via rules is not directly dealt with in this section, in Case 3, we show that the system is capable of composing services for goals based on complex rules.

The relevant wrapped services for planning is shown in Table 5.3 and 5.4. Names for the services and their variables are renamed for more easy reading. In actual case, all of these are identified by unique ID numbers generated during variable binding as shown in Appendix B. From Table 5.3 to 5.4, there are some services which have the same name. These services are duplicates. Take for example, $Robot1Ask$ has a duplicate, where the only difference is the value being fed to $Robot1AnsCat$, which is, $C\_UPactivity$ and $C\_UPchannel$. These two variables are actually constants to identify the category type of information that the robot has to query. The number of duplicates will grow with more categories. Inter-device link file is written for service $RobotSelect$, $Robot1Ask$, $Robot2Ask$, $Robot1Announce$ and $Robot2Announce$ as robots are selected according to service $RobotSelect$. To save space, we write $Light'n'$ and $Door'n'$, where n is the device number, because there are multiple lights and doors.

Table 5.3: *Relevant services for the cases 1*

Service name	Precondition	Effect
$Robot1Ask$	$RobotChoice = 1 \wedge RobotChoice_{Known}$	$Robot1Ans := Robot1Ans_{res}$, $Robot1Ans_{Known} := true$, $Robot1AnsCat := C\_UPactivity$
$Robot2Ask$	$RobotChoice = 2 \wedge RobotChoice_{Known}$	$Robot2Ans := Robot2Ans_{res}$, $Robot2Ans_{Known} := true$, $Robot2AnsCat := C\_UPactivity$
$Robot1Ask$	$RobotChoice = 1 \wedge RobotChoice_{Known}$	$Robot1Ans := Robot1Ans_{res}$, $Robot1Ans_{Known} := true$, $Robot1AnsCat := C\_UPchannel$
$Robot2Ask$	$RobotChoice = 2 \wedge RobotChoice_{Known}$	$Robot2Ans := Robot2Ans_{res}$, $Robot2Ans_{Known} := true$, $Robot2AnsCat := C\_UPchannel$
$Robot1Announce$	$RobotChoice = 1 \wedge RobotChoice_{Known}$, $MessagePort_{Known} \wedge MessageCat_{Known}$	$Robot1Cat := MessageCat$
$Robot2Announce$	$RobotChoice = 2 \wedge RobotChoice_{Known}$, $MessagePort_{Known} \wedge MessageCat_{Known}$	$Robot2Cat := MessageCat$
$MsgService$	$MessagesNum > 0 \wedge MessagesNum_{Known}$	$MessagePort_{Known} := true$, $MessagePort := C\_MessagePort$, $MessageCat_{Known} := true$, $MessageCat := C\_MessageCat$ $MessageNum := MessageNum_{res}$, $MessageNum_{Known} := true$
$GetMessages$	$true$	$UPactivity := Robot1Ans, UPactivity_{Known} := true$
$GetResult1$	$Robot1Ans_{Known} \wedge Robot1AnsCat = C\_UPactivity$	$UPactivity := Robot2Ans, UPactivity_{Known} := true$
$GetResult2$	$Robot2Ans_{Known} \wedge Robot2AnsCat = C\_UPactivity$	$UPactivity := Robot1Ans, UPchannel_{Known} := true$
$GetResult1$	$Robot1Ans_{Known} \wedge Robot1AnsCat = C\_UPchannel$	$UPactivity := Robot2Ans, UPchannel_{Known} := true$
$GetResult2$	$Robot2Ans_{Known} \wedge Robot2AnsCat = C\_UPchannel$	$BathOperation := true$
$BathOp$	$true$	$EnsureActivity_{FLG} := true$
$EnsureActivity$	$UPactivity_{Known} \wedge (UPactivity = C_bath \rightarrow BathOperation) \wedge (UPactivity = C_WatchTV \rightarrow (TVchannel = UPchannel \wedge UPchannel_{Known}) \wedge$other activities)	
$Light\_on$	$true$	$Light := true, Light_{Changed} := true$

Table 5.4: *Relevant services for the cases 2*

Service name	Precondition	Effect
TV_on	$true$	$TV := true, TV_{Changed} := true$
TV_Stream	$TV \wedge TV\_Changed \wedge CAMCarPorchPort\_Known \wedge CAMCarPorchCat\_Known$	$TVCat := CAMCarPorchCat, TVCat\_Known := true$
TV_Stream	$TV \wedge TV\_Changed \wedge CAMGardenPort\_Known \wedge CAMGardenCat\_Known$	$TVCat := CAMGardenCat, TVCat\_Known := true$
$ChangeChannel$	$TV \wedge UPchannel_{Known}$	$TVchannel := UPchannel$
$GetUserLocation$	$true$	$Ulocation := Ulocation_{res}, Ulocation_{Known} := true$
$RobotSelect$	$Ulocation_{Known}$	$RobotChoice := RobotChoice_{res}, RobotChoice_{Known} := true$
$CAMCarporchPort$	$CamCarPorch \wedge CamCarPorch\_Changed$	$CAMCarPorchPort := C\_CAMCarPorchPort, CAMCarPorchPort\_Known := true, CAMCarPorchCat := C\_CAMCarPorchCat, CAMCarPorchCat\_Known := true$
$CAMGardenPort$	$CamGarden \wedge CamGarden\_Changed$	$CAMGardenPort := C\_CAMGardenPort, CAMGardenPort\_Known := true, CAMGardenCat := C\_CAMGardenCat, CAMGardenCat\_Known := true$
$CamCarporchOn$	$true$	$CamCarPorch := true, CamCarPorch\_Changed := true$
$CamGardenOn$	$true$	$CamGarden := true, CamGarden\_Changed := true$
$Vent$	$true$	$Vent := true, Vent\_Changed := true$
$Light'n'$	$true$	$Light'n' := true, Light'n'\_Changed := true$
$Door'n'$	$true$	$Door'n' := true, Door'n'\_Changed := true$

5.2.4.4 Case 1: Event Triggered/Scheduled

Goals that are triggered by event or schedule play a big part in the smart home. Examples are temperature control given an uncomfortable room temperature, emergency response of an accident, and ventilation activation when there is smoke.

For this case, the goal is triggered when the user comes home (thus, event triggered). The goal imposed after the trigger is

$$Light \wedge EnsureSctivity_{FLG} \wedge (MessagesNum > 0 \rightarrow (Robot1Cat = C\_MessageCat \vee Robot2Cat = C\_MessageCat))$$

Upon arriving home, when there are any unheard phone messages, the home will select the speaker or robot nearest to the user to announce it. The home will also ask what the user would like to do. Upon answering (or not answering, where the response will be null), the home automation will determine whether it can assist in preparation for the human to engage in such activity. If it is able to, proper preparation will be made. To cases are presented.

In the first case, there are no phone messages, and that upon coming back the user would like to take a bath.

Case 1.1

Planning:

$GetUserLocation \Rightarrow RobotSelect \Rightarrow Nan\_Response \Rightarrow End$ (Planning Time = 0.053616 seconds)

Implementation:

Activity $Nan\_Response$ pre-condition is not satisfied

Re-planning:

$GetMessages \Rightarrow MsgService \Rightarrow Robot1Ask \Rightarrow Light\_on \Rightarrow GetResult1 \Rightarrow EnsureActivity \Rightarrow Robot1Announce \Rightarrow End$ (Planning Time = 0.320875 seconds)

Implementation:

Activity $MsgService$ pre-condition is not satisfied

Re-planning:

$Robot1Ask \Rightarrow GetResult1 \Rightarrow Light\_on \Rightarrow EnsureActivity \Rightarrow End$ (Planning Time = 0.085359 seconds)

Implementation:

Activity $EnsureActivity$ pre-condition is not satisfied

Re-planning:

$BathOp \Rightarrow EnsureActivity \Rightarrow End$ (Planning Time = 0.026977 seconds)

Implementation:

End

The planning started off by getting the user location via $GetUserLocation$, which is required for $RobotSelect$ service to select the nearest robot. Due to Nan response discussed earlier, the planning will end. Implementation will proceed until it meets $Nan\_Response$. Condition is not satisfied because Nan response can never be achieved during actual execution. But because user location is already known from $GetUserLocation$, re-planning can proceed through Robot 1, which is the nearest robot. The second re-planning started off by getting messages through $GetMessages$. $MsgService$ will then use this information and relay it to the robot, given there is any messages. Currently, it assumes there are messages. Robot 1 will then proceed to ask the user what activity he/she wants to do via $Robot1Ask$. Light is turned on after that via $Light\_on$. $GetResult1$ is a service which extracts the answer from $Robot1Ask$. Since there is no information of what the user wants to do, nothing will happen with checking service $EnsureActivity$. Finally, $Robot1Announce$ is run to announce the unheard messages. Due to the fact that there is no messages, implementation will stop at $MsgService$. Re-planning is performed again, this time, without $Robot1Announce$ because no messages are to be announced. During the third implementation, the user told the robot he/she wants a bath upon asking (which is an activity the ISS can help by preparing bath). Therefore, $EnsureActivity$ is not fulfilled because it assumes no activity or an activity where ISS can do nothing about. Final re-planning proceeds by running the bath preparation operation $BathOp$. Implementation completes successfully.

For the second case, There are some unheard messages, and the user like to watch TV.

Case 1.2

Planning:

$GetUserLocation \Rightarrow RobotSelect \Rightarrow Nan\_Response \Rightarrow End$ (Planning Time = 0.053297 seconds)

Implementation:

Activity $Nan\_Response$ pre-condition is not satisfied

Re-planning:

$GetMessages \Rightarrow MsgService \Rightarrow Robot1Ask \Rightarrow Light\_on \Rightarrow GetResult1 \Rightarrow EnsureActivity \Rightarrow Robot1Announce \Rightarrow End$ (Planning Time = 0.301082 seconds)

Implementation:

Activity $EnsureActivity$ pre-condition is not satisfied

Re-planning:

$Robot1Ask \Rightarrow TV\_on \Rightarrow Robot1Announce \Rightarrow GetResult1 \Rightarrow changeChannel \Rightarrow EnsureActivity \Rightarrow End$ (Planning Time = 0.232423 seconds)

Implementation:

End

The planning and implementation have a similar start as the first case. At the second implementation, checking service $EnsureActivity$ is not met as the user would like to watch TV. For the last re-planning, $Robot1Ask$ will ask the user the channel he/she wants to watch. It proceeds to turning on the TV via TV_on. It will also announce the unheard messages through $Robot1Announce$. $GetResult1$ will then obtain the result from previous query on TV channel, which will be used by $changeChannel$ to switch to the preferred channel. Implementation is completed successfully.

There are some planning sequences that seem counter-intuitive. For example, from the last re-planning of the second case, we have $Robot1Ask \Rightarrow TV\_on \Rightarrow Robot1Announce \Rightarrow GetResult1 \Rightarrow ...$, where the robot query result is only recorded after turning the TV on and announcement made by the same robot. But, the sequence is acceptable. Besides, one can imagine these services are running in parallel. Parallelism is further explored by [71]. Besides that, more research has to be done to optimize the generation of services. There are some cases where generation of services will lead to an explosion in service number, when multiple services are referencing on each other.

5.2.4.5 Case 2: Device Selection

In certain cases, we only need a certain amount of devices to be activated at one time, where the choice of the devices also depends on certain rules. To exhibit such cases, for this case study, we assume a trigger is activated at the home garden, which leads to the security camera video being shown in the home TV. At the same time, two lights at the garden should be turned on. The goals imposed is as follows:

Goal 1:

$(TVCat = C\_CAMGardenCat) \wedge TVCat\_Known$

Goal 2:

$\exists x \exists y (x \neq y \wedge Type(x, light) \wedge Type(y, light) \wedge State(x, ON) \wedge State(y, ON) \wedge Location(x, Garden) \wedge Location(y, Garden))$

This first goal dictates that the garden camera is to be connected to the TV. The second goal states that there should be at least two lights located in the garden that needs to be turned on (Due to the way planning is performed, 2 lights will be turned on although the goal states "at least 2"). In detail, it states that there exists x and y that are distinct from each other, that are of type $light$ determined by function $Type$, where both are ON obtained from function $State$, and that both are locatied at the $Garden$ indicated by

$Location$. The following shows the planning and implementation:

Planning:

$CamGardenOn \Rightarrow CamGardenPort \Rightarrow TV\_on \Rightarrow TV\_Stream \Rightarrow Light4\_On \Rightarrow Light2\_On \Rightarrow End$ (Planning Time = 0.044648 seconds)

Implementation:

End

From the plan, garden camera is turned on by $CamGardenOn$. A signal port will be obtained from the camera by $CamGardenPort$. Before connection is made, the TV needs to be turned on via TV_On. The camera signal is then connected to the TV through TV_Stream. Finally, two lights at the garden are turned on via $Light2\_On$ and $Light4\_On$. There are altogether 4 lights in the garden (Light 2 to Light 5), but due to Goal 2, only 2 lights are turned on.

Currently, the logic functions ($Type$, $State$ and $Location$) are only available at the goal. Further enhancements can be made such that these functions can be embedded into the services themselves. But in doing so, variable binding will be made more complicated that require further research.

5.2.4.6 Case 3: Complex Rules Handling

Case 3 will demonstrate that the system can perform services to fulfill complex rules. Rules can be built using information from the building ontology. This shows the potential of the system to extend to more intelligent service composition.

In this case, given that there is smoke detected in the wet kitchen, and the user is in the living room, the home needs to disconnect the user from the smoke. It will also open the ventilation system in the wet kitchen to vent out the smoke. Goals will be to have at least one vent (in the wet kitchen) being turned on. Information of room adjacency can be obtained from the building ontology, and be used to build rules as goals for service planning. Figure 5.10 shows the example layout used for this case. Detailed description of the layout is explained in Section 5.3.2.1. The numbered parts are doors, and are assumed to be all controllable devices. Two rooms are considered connected if there is a path between the two rooms without crossing walls and closed doors. The following shows the rules to determine whether two rooms as considered connected:

$$Belongs(door, room1) \wedge Belongs(door, room2) \wedge is(door, open) \rightarrow connected(room1, room2)$$

$$connected(room1, room2) \wedge connected(room2, room3) \rightarrow connected(room1, room3)$$

Since current variable binding assumes propositional logic, the first order logic shown are converted to propositional logic via generating all possible combinations called grounding. Of course first order logic with existential and universal quantifiers can be directly coded. But this will be subject of future work, where variable binding will take into account more complex rules. To set the goal where the user will be cut off from the smoke, the propositional logic value for the connection between the living room and the wet kitchen is set false, which is:
$connected(LivingRoom, WetKitchen) = false$

When the planner is run, it activates the vent in the wet kitchen. It also closes door 19 or door 23 since either one will cut the user off from the smoke. This shows that planner can working with knowledge representation to perform intelligent executions.

More work will be done to provide a more expressive and standardized formulation of the rules with inference from the ontology.

5.2.5 Issues to be Addressed

Case studies demonstrate that the system can perform proper variable binding to be used in automated service composition for complex tasks. It also shows the ability of using complex rules derived from reasoning (via knowledge representation used for variable binding) as goals. Various issues require addressing. In this section, issues that require further development are listed down.

Inter-device link (as shown in the third module of the sequence in Figure 5.6) is crudely implemented at the moment. It involves writing a separate file that dictates the variable linkages between two participating devices. This module also deals with linking variables with logic functions used as goal during planning (like the $Type$, $Location$ and $State$ function in Case 2) that uses the same method as the former. More automated linking, or at least a more user-friendly and simple implementation, is needed.

Services for devices have to be manually defined by the user or manufacturers. But there may be cases where additional rules can be included into the services to aid planning

and deliver more complex service sequence. An example is to be able to know a light sensor will be activated if a nearby light is turned on. Learning is possible to reduce manual efforts as shown in [107]. The system should support such learning capabilities, or at least provide the possibility to include them.

Explosion of number of generated services is an issue given badly designed service parameters. Currently, services are duplicated if multiple variable associations of variables are found. This is made worse if the effect variables of such duplicated services are associated with other precondition variables. The problem is temporarily alleviated by combining similar effect variables (as shown in the fifth module of the sequence in Figure 5.6). But it is impractical to assume the variables have the exact annotations (which is assumed by the module), thus, more research need to be done in this respect.

To include intelligence, knowledge and reasoning have to be incorporated into service composition, which is shown in Case 3. As explained, current system require manual programming to extract goals derived from reasoning for service composition. The next step is to automate this process.

Nan response is used to avoid redundant implementation of conspicuous services due to optimistic planning issue. The downside of it is that all unknown variables that will trigger Nan response will have to be determined first during planning. This is inefficient if certain variables can only be determined given a certain amount of delay. This problem can be alleviated by constantly updating variables prior to planning.

5.2.6 Remark

Through variable binding, variables and services provided by individual devices can be related to each other, after which service wrapping is performed to convert the services into propositional rules for service composition. Via representing the problem as CSP given goals, solution obtained is the sequence of services that needs to be implemented to fulfill the goals.

Case study shows the system is capable of composing and executing complex plans. Besides, it shows potential in composing services for complex rules derived from the building ontology. At this stage, devising these rules requires manual programming. A more standardized automation should be developed.

5.3 Extension to Robot Agent Planning for Home

Smart homes and ad-hoc home automations are on the rise to provide support for the elderly. But currently, installation and integration of the system requires technical knowledge. Besides, large variety of vendors and standards brings more confusion instead of peace to the elderly. But we by no means say that smart home and home automation fail to deliver. In fact, they are necessary in the near future to provide the assistance we need. At the current stage, it is more appropriate to devise a physical agent that can dispense service at a home yet to be supported or supported minimally by smart devices. This agent, or we shall call, service robot, should provide security, assistance, communication and companionship to its human occupants.

Various service robots have been developed over the years. An example is the HSR robot developed by Toyota [145]. Robot manipulation and mobility is a mature field. But these are low level actions, where they require high level activity planning in order to perform complex tasks. Service planning is able to provide them with the high level overview of how their minute actions will be carried out overtime.

In the field of robotics, Markov Decision Process (MDP) is a popular approach for robot motion planning. The downside of this approach is that it lacks the complexity required for service composition, which can be delivered by logic-based approach [40]. But due to MDP being a statistical method, this gives them an edge in this respect in dealing with uncertainty over logic-based approach. Therefore, both statistical and logical approach can be considered tackling different scope of the same problem. In this section, emphasis is given to service composition performed by automated reasoning. It handles the broad service composition such that it also provides room for statistical or other tedious methods to figure out the minute details.

This section presents the work where robot planning problem is represented as constraint problem. Sequence of plans is generated for the robot to execute to fulfill predefined goals. Through this approach, robot individual services become loosely coupled, thus, enabling more flexibility and enhancing reusability in service design. FOL is used, where the functions and predicates are constructed from the semantic information obtained according to the schema of the home ontology.

5.3.1 Robot Agent Planner Design

Planning and execution of plans are performed by a single mobile robot. The plans involve high level plans like where the robot should go to, opening doors, picking up certain objects and so on. It does not involve minute details such as how the robot should open

doors through the coordination of servos and visual input. In this work, when a robot is expected to execute a plan, the robot is already in an environment that is conducive or possible to execute such plans. The planner and the orchestrator will execute a sequence of plans to prepare such an environment. For example, when the robot needs to switch on a light, the planner will first tell the robot to go near the corresponding switch before switching it on with its manipulators.

A few basic movements of the robot are assumed, which are, mobility, opening and closing, switching on and off, picking and putting objects. It is assumed that the robot can hold something while performing open/close actions. Planning and execution will revolve around these basic movements to achieve certain goals. Planner module is responsible for the planning and passing commands to the robot for its execution. Given a certain goal, planner module will compose a sequence of plans for the robot to execute such that after the plans are completed, the goal is achieved. For example, given that the goal is to place a canned soft drink in the fridge and that all cabinet needs to be closed, the robot will first find where the soft drink is. If there is one that happens to be in the kitchen cabinet, the robot will approach it, open the cabinet and pick up the soft drink, and then close it. It will subsequently approach and open the fridge, after which it will place the canned drink before closing it.

Modications of the CSP planner module from Section 4.3 is made as shown in Figure 5.9. In this constraint graph, there is an additional variable called the dynamic response variable. Unlike response variable, dynamic response variable exists in every sequence just like normal variables. It differs from normal variables in that it does not abide to simple frame axiom. Its value needs to abide to logical rules, where these rules apply to previous and current variables. Future variables can also be used, but we will concentrate on the past and present variables.

5.3.1.1 Design of Terms

The robot in this work is intended to fetch and place objects, as well as opening/closing and switching on/off switches. It needs to be able to move around to enable it to perform the tasks. There are 3 types of objects, which are, movable objects, non-movable objects and location.

Movable objects consist of objects that can be moved around by the robot like towel and cans. Non-movable objects are objects that are fixed, but they may be able to be operated by the robot. Examples of non-movable objects are bed, doors, cabinet and switches. The robot itself is also considered a non-movable object for convenience during planning which will

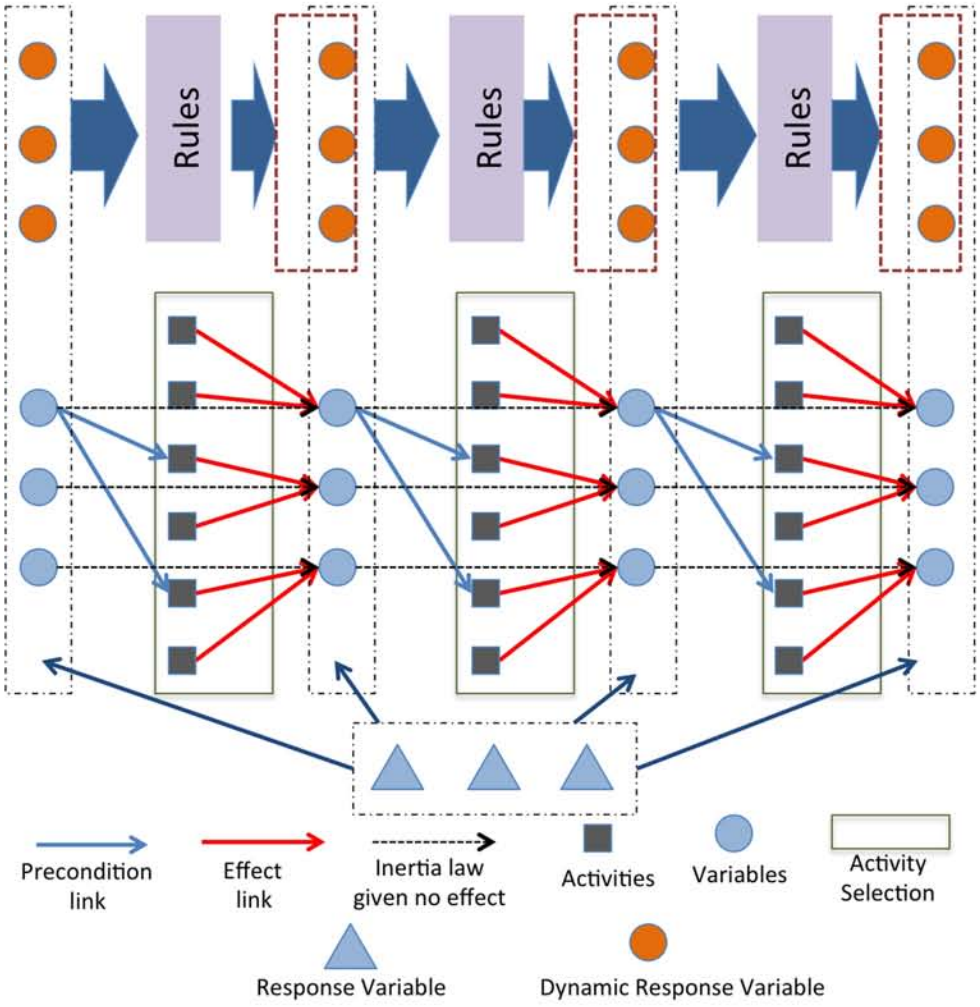


Figure 5.9: *Constraint graph for robot planning*

be shown later on. Human is also considered a non-movable object as the assumption is that the human remain stationary during planning. But if the human moves, due to the ability to replan as discussed previously, the issue can be easily solved. Location object consists of regions on the home such as the living room, bedroom and kitchen.

Apart from integer and Boolean datatype, 5 new data types are introduced to support the mentioned objects, namely, $NOType$, $MOType$, $Location$, $NOStateType$, $MOStateType$, $NOID$ and $MOID$. $NOType$ defines the type (ex: cabinet, door, switch) of non-movable object with ID defined in data type $NOID$. Likewise, $MOType$ defines the type (towel, paper, cup) of movable object with ID defined in data type $MOID$. $Location$ is the datatype of location.$MOStateType$ and $NOStateType$ are the datatype that defines the state of a movable and non-movable object respectively.

There are 2 static functions (functions where the output values remain the same throughout all planning sequence), which are, $FNOType : NOID \rightarrow NOType$ and $FMOType : MOID \rightarrow MOType$. A static predicate is $NOLocation(NOID, Location)$, an a constant is $HumanLocation$.

$FNOType$ maps a particular non-movable object to its type (For example, mapping an object as a door). Like wise, the same applies to $FMOType$, but that it applies to movable object. The predicate $NOLocation(a, b)$ states the truth value whether a non-movable object a is in location b. This is especially important for objects like doors. $HumanLocation$ stores where the human is at in datatype $Location$

The subsequent terms explained in this subsection can have their values changed in the course of planning sequence. This means, given a term A, for a plan with K sequence, there will be $A \times K$ number of variable A, each for every sequence, where each has a unique definition $A_1, A_2...A_K$.

The function $DYStateNO_t : NOID \rightarrow NOStateType$ outputs the state of the non-movable object. $DYStateNO_t(a) = b$ states that a is in a state of b at sequence t. The same applies to $DYStateNO_t : MOID \rightarrow MOStateType$, which is for movable objects.

The function $DYTune_t : NOID \rightarrow Int$ outputs the numerical value (in integer Int type) associated with the non-movable object at sequence t.

As movable objects will always be placed at a non-movable object (ex: cup on a table, book with a human), the function $DYAtMO_t : MOID \rightarrow NOID$ maps a movable object to a non-movable object it is placed at.

Besides the functions, 3 variables are included, namely, $RobotLocation_t$, $RobotApproach_t$ and $RobotHold_t$. $RobotLocation_t$ stores the location of the robot (with datatype $Location$) at sequence t. $RobotApproach_t$ records the non-movable object the robot is approaching. $RobotHold_t$ is a Boolean variable determining whether the robot

is holding something or not.

For proper functionality, 3 dynamic response variables are introduced, which are, $Approachres_t$, $MOIDres_t$ and $RobotHoldres_t$. The role of these dynamic response variables will be made more evident during the discussion of the services.

5.3.1.2 Design of Services

This subsection describes, but not limited to, 3 types of services the robot is expected to perform. Although only 3 types are listed, more services can be included depending on the application. The 3 types are open/closing service, mobility, and put/pick service.

Open/closing service consists of two services, which are opening and closing. These two services applies to opening/closing of doors and switching on/off switches.

It has a precondition:
$NOLocation(RobotApproach_t, RobotLocation_t) = true$
and effect:

$DYStateNO_{t+1}(RobotApproach_t) = Open/Close$ respectively

The precondition has to make sure the robot is approaching object stored in $RobotApproach_t$ and that the robot is in location $RobotLocation$, before it can open/close the non-movable object by manipulating $DYStateNO$ at sequence $t + 1$.

Tuning up/down service deals with numerical manipulation that consists of 1) tuning up (increasing by 1), and 2) tuning down (decreasing by 1).
It has a precondition:
$NOLocation(RobotApproach_t, RobotLocation_t) = true$
and effect:
$DYTune_{t+1}(RobotApproach_t) := DYTune_t(RobotApproach_t) + / - 1$ respectively

Put/Pick service consists of robot fetching and placing a movable object.

The preconditions for picking up objects are:
$DYStateNO_t(RobotApproach_t) = Open$
$DYAtMO_t(MOIDres_t) = RobotApproach_t$
$NOLocation(RobotApproach_t, RobotLocation_t) = true$

$RobotHold_t = false$

and effects:
$DYAtMO_{t+1}(MOIDres_t) = NRobot$
$RobotHold_{t+1} = true$

The preconditions make sure that the movable object the robot is trying to fetch is in located in a non-movable object with state $Open$ through $DYStateNO_t(RobotApproach_t)$,and that the robot is not holding anything via $RobotHold_t = false$. They also make sure the object the robot is trying to fetch ($MOIDres_t$) is located in non-movable object $RobotApproach_t$. $MOIDres_t$ is a dynamic response variable, where its value is freely determined by the planner to aid optimistic planning which is inherent in planning via solving CSP. If the preconditions are met, the robot can pick object $MOIDres_t$ through setting $DYAtMO_{t+1}(MOIDres_t)$ to $NRobot$, where $NRobot$ is the non-movable object ID under datatype $NOID$ for the robot. $RobotHold$ is set to true to indicate that the robot is holding something.

The preconditions for putting objects are:
$DYStateNO_t(RobotApproach_t) = Open$
$DYAtMO_t(RobotHoldres_t) = NRobot$
$NOLocation(RobotApproach_t, RobotLocation_t) = true$
$RobotHold_t = true$

and effects:
$DYAtMO_{t+1}(RobotHoldres_t) = RobotApproach_t$
$RobotHold_{t+1} = false$

For putting objects, the preconditions also specify that the intended non-movable object be opened. It requires the robot to hold movable object $RobotHoldres_t$, which is indicated by having $NRobot$ as the output for function $DYAtMO_t$. Just like $MOIDres_t$, $RobotHoldres_t$ is a dynamic response variable that stores the current object the robot is holding. $RobotHold_t = true$ is the constraint where the robot is holding something. With the preconditions met, $RobotHoldres_t$ will be placed at $RobotApproach_t$ through $DYAtMO_{t+1}$.

Mobility service consists of two services, that is, movement within a room, and movement between rooms.

Movement within a room has the following precondition:
$NOLocation(Approachres_t, RobotLocation_t) = true$

and effect:
$RobotApproach_{t+1} = Approachres_t$

The robot will always be going towards a non-movable object, as it is practically meaningless to go towards nothing. Therefore, the precondition makes sure that a non-movable object the robot is going to is within the current room, where the dynamic response variable $Approachres_t$ stores the object the robot is approaching.

Movement between rooms is required for every door. One can write a precondition as the following:
$\exists z(NOLocation(z, RobotLocation_t) \wedge NOLocation(z, res_t) \wedge (res_t \neq RobotLocation_t) \wedge DYStateNO_t(RobotApproach_t) \wedge (RobotApproach_t = z)(FNOType(z) = Door))$

But from preliminary tests, this precondition with existential quantifier takes a longer time to plan. Another alternative is to build a service for every doors. The precondition is shown as follows:

$DYStateNO_t(V1) = true$
$RobotLocation_t = V2$

and effects:
$RobotLocation_{t+1} = V2$

Services with above mentioned preconditions and effects are built for every door and for every side. It means that if there are 2 doors altogether, there are a total of 4 such services, where each door takes up two for the robot to move through the door in both directions. From the preconditions and effects, $V1$ is the door object ID and $V2$ is the location the robot goes to after passing the door.

5.3.2 Experiments and Discussion

This section presents the case study of the planner for robot service execution. The intention is to show the applicability of the approach, while at the same time, provides insights into future developments.

Case study will show how the robot, given the standard activities (mobility, open/close and pick/put), can fulfill goals that can provide support for its human inhabitants. It is done through simulation built via Open Dynamics Engine (ODE). The study is run on 2.5GHz Intel Core i5 computer.

5.3.2.1 House Setup

Case study is performed on a simulated home via ODE. Figure 5.10 shows the layout of the house. It contains 36 non-movable objects (excluding human and the robot), 7 movable objects, 10 locations, and 10 doors. The numbers in white shown in the figure are the non-movable object IDs. Blue parts are switches, red lines are windows, orange parts are cabinets, Purple part is a fridge, green part is the table, pink part is the bed and brown part is the washing machine. The shapes with swinging curves are doors.

Out of all the non-movable object, the bed (ID 1) and table (ID 34) cannot be manipulated. Doors, windows, fridge, cabinets and washing machine can be opened/closed. Apart from doors, all non-movable objects only have one associated location (indicated by predicate $NOLocation$). Of all the non-movable objects, only N37 is associated with $DYTune$, as N37 is considered a volume control for the living room fan. The locations are, Master bedroom, Living room, Kitchen, Wet Kitchen, WC1, WC2, Bed room 2, Bed room 3, Car Porch and the Garden.

Numbers preceded by 'subloc' are sublocations for the objects. Since the robot need to be near enough to an object before it can manage it, the sublocations provide such spots. Sublocations are not included in the planning, but their information with their corresponding non-movable objects are stored in the knowledge base such that they can be used during execution. This information is crucial during path planning. For this work, due to the simple grid-like layout, Floyd-Warshall algorithm is used for path planning.

Details of the movable objects are shown in Table 5.5, which shows their type and which non-movable objects they are placed at. Information for these movable objects are also extracted from knowledge base of the home. For faster planning, one can limit the information to only deal with the objects required.

For clarity, all IDs for non-movable object are preceded by a capitalized N, and all movable object will be preceded by M during the case studies. For example, the door with ID

Table 5.5: *Movable object details*

Object ID	MOType	NOID
1	Canned drink	22
2	Canned drink	20
3	Canned drink	12
4	Paper	34
5	Towel	3
6	Towel	31
7	Book	11

6 will be identified as N6, while the towel with ID 5 in the washing machine N31 will be identified by M5.

5.3.2.2 Open Dynamics Engine Environment

Model of the house is set up via Open Dynamics Engine (ODE) [1], including the movable and non-movable objects. For the robot, a simple mobile robot with arms is built.

The robot is able to move around the modeled environment subject to physical laws. The robot arms are used for opening/closing and picking/putting activities. Figure 5.11 shows the modeled robot. In this simulation, no explicit gripper is modeled, as the emphasis is on planning. For picking up objects, the intended object will be attached to the arm instead when the arm is of the same coordinate as the object. In order to operate an object, it needs to be within the reach of the robot arm (which, of course, has been taken into account by the planner), which, if this is the case, it is up to the robot to decide how it is going to operate it. This is in line with the work's motivation, which is to provide high level planning to get the robot to a situation suitable for it to perform low level actions.

Orchestrator from the planning module constantly communicates with the robot. It sends service instructions one at a time for the robot to execute, while the robot will respond given every executed service.

5.3.2.3 Speed Comparison on Different Service Design

Service design will have significant effect on how search is performed to obtain solution. In this section, test is performed to select whether to utilize a general service that covers wide number of objects, or to duplicate the services to handle these objects individually.

That said, we will concentrate on two types of services, that is, moving between locations (S1) and opening/closing (S2) services. Both of these service types covers a wide range

[1] See http://www.ode.org/

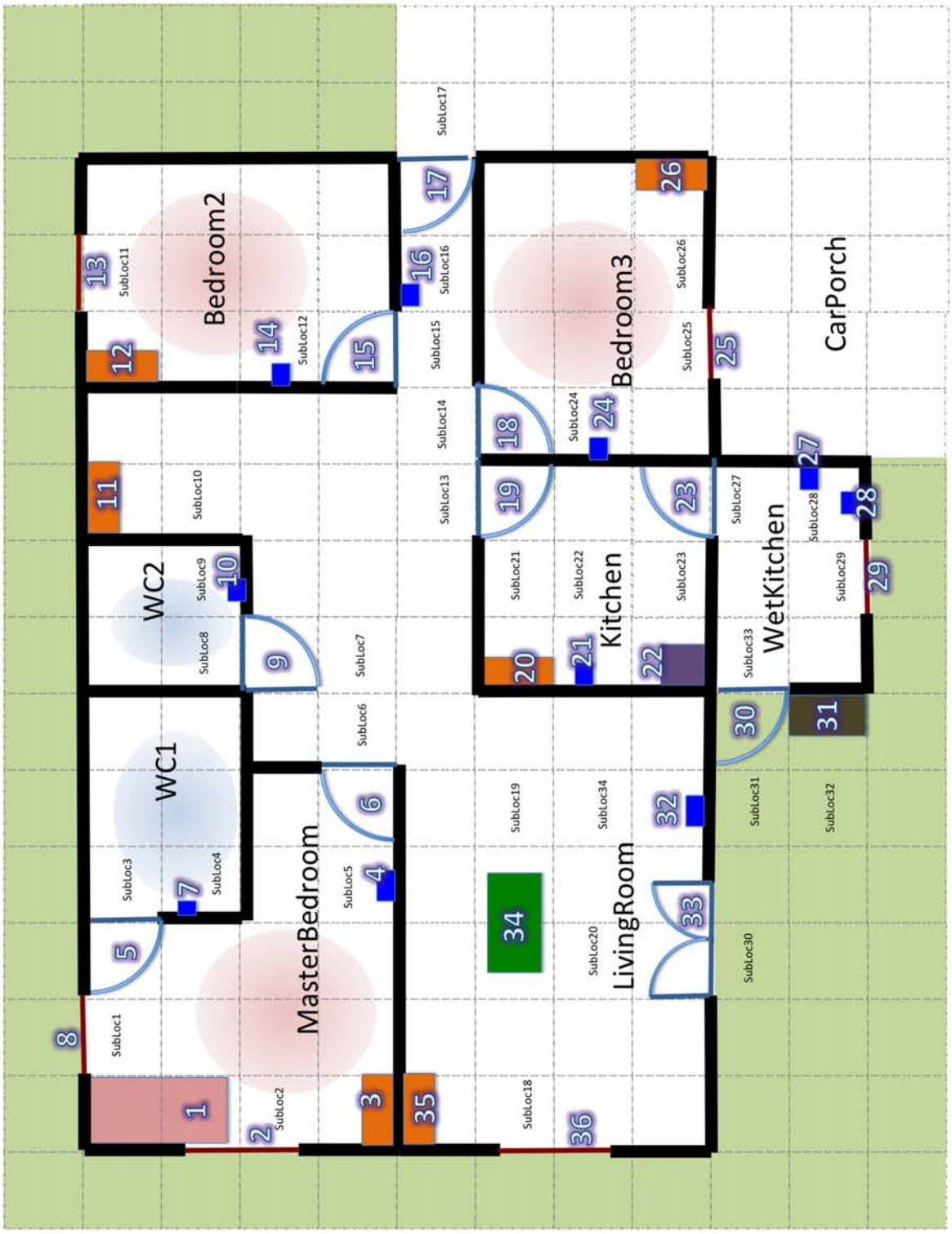


Figure 5.10: *House layout*

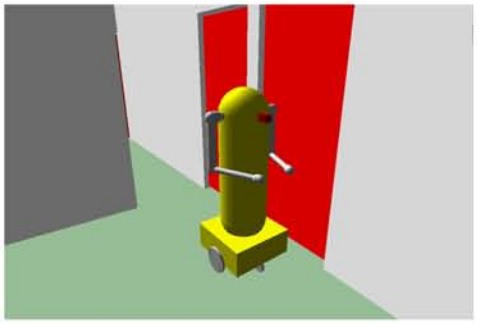


Figure 5.11: *Model robot: Simple mobile robot with arms for opening/closing and picking/putting operations*

Table 5.6: *Test on different service design*

Test no.	Description of the goal	Command	Steps	A(sec)	B(sec)	C(sec)
1	Approach N1	explicit	4	0.431	0.219	0.294
2	Open all windows in master bedroom	implicit	7	1.357	0.938	1.346
3	Move M2 to N20 (same location)	explicit	9	2.27	1.03	1.76
4	Move any canned drink to M34	implicit	9	3.36	2.74	5.21
5	Move M5 to N31 (different location)	explicit	13	18.36	14.94	11.36
6	Switch on N10, N12, N14, N28	explicit	22	655.40	272.67	609.35

of objects, thus is important to select the optimal approach to implement them to obtain the best speed.

We denote type A test as general service for S1 and S2, type B test as general S1 and duplicated S2, and finally type C test as duplicated S1 and general S2. For clarity, explicit command means the goal is explicitly specified (ex: moving object A to B), while implicit command means the goal is implied (ex: move any object with property N out of D). Type A contains 6 services, type B contains 25 services and type C has 91 services. Table 5.6 shows the test result on the speed of planning for all 3 types of services.

It can be observed that type B (general S1 and duplicated S2) achieves the best result most of the time. Type A is the slowest most of the time despite it having only 6 services to choose and search from, which shows generality comes at a price on speed. Type C, having the most services at 91, beats type A in planning speed. But its performance is insignificant compared to type B, due to its high number of duplicated services. Therefore, the subsequent case studies will utilize type B as design approach for the services. Table 5.7 shows the details of all type B services.

Table 5.7: *Service preconditions and effects*

No.	Precondition	Effect
1	$NOLocation(RobotApproach_t, RobotLocation_t)$	$DYStateNO_{t+1}(RobotApproach_t) := true$
2	$NOLocation(RobotApproach_t, RobotLocation_t)$	$DYStateNO_{t+1}(RobotApproach_t) := false$
3	$NOLocation(Approachres_t, RobotLocation_t)$	$RobotApproach_{t+1} := Approachres_t$
4	$DYStateNO_t(N6) \wedge RobotLocation_t = MasterBedroom$	$RobotLocation_{t+1} := LivingRoom$
5	$DYStateNO_t(N6) \wedge RobotLocation_t = LivingRoom$	$RobotLocation_{t+1} := MasterBedroom$
6	$DYStateNO_t(N5) \wedge RobotLocation_t = MasterBedroom$	$RobotLocation_{t+1} := WC1$
7	$DYStateNO_t(N6) \wedge RobotLocation_t = WC1$	$RobotLocation_{t+1} := MasterBedroom$
8	$DYStateNO_t(N9) \wedge RobotLocation_t = WC2$	$RobotLocation_{t+1} := LivingRoom$
9	$DYStateNO_t(N9) \wedge RobotLocation_t = LivingRoom$	$RobotLocation_{t+1} := WC2$
10	$DYStateNO_t(N15) \wedge RobotLocation_t = Bedroom2$	$RobotLocation_{t+1} := LivingRoom$
11	$DYStateNO_t(N15) \wedge RobotLocation_t = LivingRoom$	$RobotLocation_{t+1} := Bedroom2$
12	$DYStateNO_t(N17) \wedge RobotLocation_t = LivingRoom$	$RobotLocation_{t+1} := CarPorch$
13	$DYStateNO_t(N17) \wedge RobotLocation_t = CarPorch$	$RobotLocation_{t+1} := LivingRoom$
14	$DYStateNO_t(N18) \wedge RobotLocation_t = LivingRoom$	$RobotLocation_{t+1} := Bedroom3$
15	$DYStateNO_t(N18) \wedge RobotLocation_t = Bedroom3$	$RobotLocation_{t+1} := LivingRoom$
16	$DYStateNO_t(N19) \wedge RobotLocation_t = LivingRoom$	$RobotLocation_{t+1} := Kitchen$
17	$DYStateNO_t(N19) \wedge RobotLocation_t = Kitchen$	$RobotLocation_{t+1} := LivingRoom$
18	$DYStateNO_t(N23) \wedge RobotLocation_t = Kitchen$	$RobotLocation_{t+1} := WetKitchen$
19	$DYStateNO_t(N23) \wedge RobotLocation_t = WetKitchen$	$RobotLocation_{t+1} := Kitchen$
20	$DYStateNO_t(N30) \wedge RobotLocation_t = WetKitchen$	$RobotLocation_{t+1} := Garden$
21	$DYStateNO_t(N30) \wedge RobotLocation_t = Garden$	$RobotLocation_{t+1} := WetKitchen$
22	$DYStateNO_t(N33) \wedge RobotLocation_t = LivingRoom$	$RobotLocation_{t+1} := Garden$
23	$DYStateNO_t(N33) \wedge RobotLocation_t = Garden$	$RobotLocation_{t+1} := LivingRoom$
24	$DYStateNO_t(RobotApproach_t) \wedge$ $DYAtMO_t(MOIDres_t) = RobotApproach_t \wedge$ $NOLocation(RobotApproach_t, RobotLocation_t) \wedge$ $\neg RobotHold_t$	$DYAtMO_{t+1}(MOIDres_t) := NRobot \wedge$ $RobotHold_{t+1} := true$
25	$DYStateNO_t(RobotApproach_t) \wedge$ $DYAtMO_t(RobotHoldres_t) = NRobot \wedge$ $NOLocation(RobotApproach_t, RobotLocation_t) \wedge$ $RobotHold_t$	$DYAtMO_{t+1}(RobotHoldres_t) := RobotApproach_t \wedge$ $RobotHold_{t+1} := false$
26	$NOLocation(RobotApproach_t, RobotLocation_t)$	$DYTune_{t+1}(RobotApproach_t) :=$ $DYTune_t(RobotApproach_t) + 1$
27	$NOLocation(RobotApproach_t, RobotLocation_t)$	$DYTune_{t+1}(RobotApproach_t) :=$ $DYTune_t(RobotApproach_t) - 1$

5.3.2.4 Summary of Cases

This section briefly describes the 6 case studies used to demonstrate the capabilities of the planning system.

Case 1: "Simple Object Fetch for Human" is used as a starting example on fulfilling an explicit goal of robot fetching a book for its human master. It is also used to show a difference between implicit and explicit goal.

Case 2: "Dynamic Planning under Uncertain Situation" shows how the planner deals with uncertain situation that causes inconsistencies. Uncertain situation may cause information that is crucial for planning to change without a prior update on the planner's knowledge. This case demonstrates dynamic re-planning to tackle such issues.

Case 3: "Inferences for Making Choices" demonstrates more clearly (compared to Case 1) the use of implicit goals for the planner to make choices based on the properties of objects, which is also an extended capability from previous work [71].

Case 4: "Reasoning and Planning with Numbers" demonstrates how the CSP planner treats numerical reasoning and manipulation as part of the planning process.

Case 5: "Reusability of Activities" shows how the robot activities can easily co-operate with the activity of other smart devices installed in the smart home. It reuses the activity definition of the smart device without having to make further manipulations.

Case 6: "Complex Goals" shows planning under more complex implicit goals and its implications in time and enhancements.

5.3.2.5 Case 1: Object Fetch

A service robot need to be able to approach human as well as fetching objects or them or picking or putting objects according to command, without the user having to dealt into too much detail of how it is done. This case study on object fetch showcases how the robot is able to find the book (from knowledge of the knowledge base) and deliver it to the user, who is in the bedroom. This case covers the aforementioned requirements of the robot.

Case 1 assumes simple goals:

$DYAtMO_K(M7) = NHuman$

$\forall z((FNOType(z) = Cabinet) \rightarrow \neg DYStateNO_k(z))$

Where the first goal states that the book with ID M7 will be given to the human, and the second goal requires all cabinet to be closed at the end of the plan. The robot initial location is at non-movable object N17.

Case 1
Approach N6⇒ Open N6⇒ Approach N11⇒ Open N11⇒ Pick up M7 from N11⇒ Close N11⇒ Pass Door N6 to MasterBedroom⇒ Approach NHuman⇒ Place M7
Planning time= 1.335 seconds

Figure 5.12 shows the robot's plan execution. Given the generated plan, the robot will first approach the door N6 to open it. It will then go to cabinet N11 to pick up the book. After that, the robot move to the bedroom to deliver the book the the human. A peculiarity that occurs is the fact that the robot opens door N6 first, and then returns to Cabinet N11, before going back to N6 to enter the Master bedroom. This is because the planner doesn't know about the cost of paths. It only judge the shortest number of activities based on the basic services aforementioned.

An implicit goal of fetching the book can also be carried out. Case 1 explicitly states which book to deliver to the human. One can also set up an implicit goal as follows:
$\exists z((FMOType(z) = Book) \wedge (DYAtMO_K(z) = NHuman))$

Where the planner will find any book to be delivered to the human. Sequence of plans remains the same, except that the average time of planning is 1.941 seconds, as it covers a wider search space.

5.3.2.6 Case 2: Dynamic Planning under Uncertain Situation

Planning needs to take into account the fact that the environment will change in the course of planning or plan execution. The planner explained in Section "Planner Module" supports dynamic re-planning, which means, if inconsistency is met, re-planning can take place, taking into account current information.

In this case study, the robot needs to fetch the human (who is in the living room) a canned drink. At the start of the experiment, as shown in Table 1, 3 canned drinks are present in the home. During the experiment, we simulate the condition where a person takes away the canned drink that the robot is after just before it reaches for it, except for the last (or 3rd) canned drink. Case 2 goal is simple as follows:
$\exists z((FMOType(z) = CanDrink) \wedge (DYAtMO_K(z) = NHuman))$

The generated plan (and re-planning) is shown as follows:

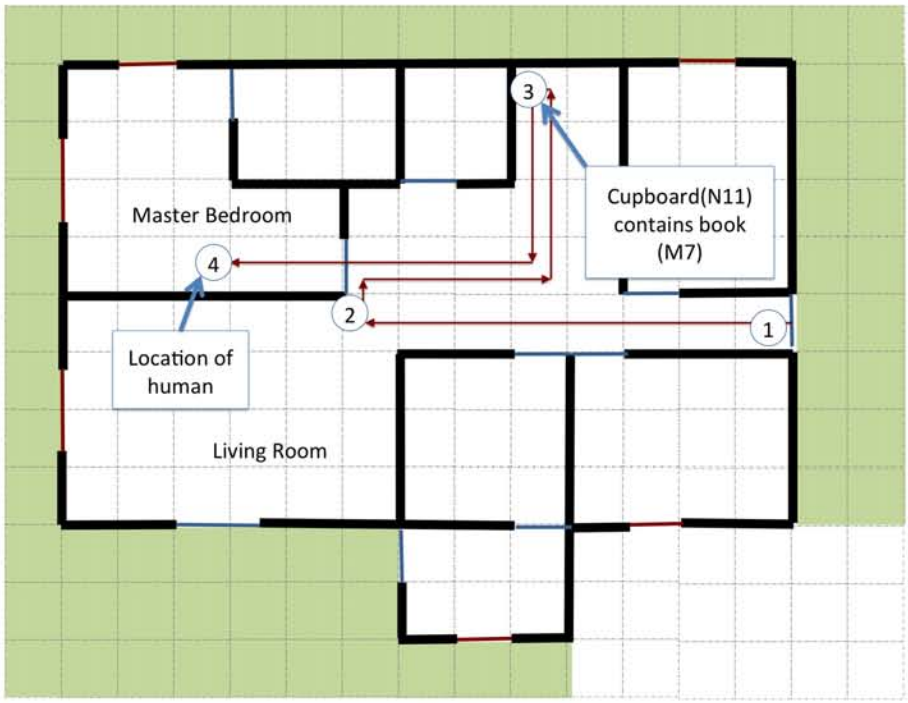


Figure 5.12: *Case 1: Step 1 is the starting point. The robot proceeds to master bedroom door(N6) and opened it in Step 2. It then goes to the cupboard(N11), opens it, and gets a book, and closed the cupboard in Step 3. Finally, in Step 4, it goes into the master bedroom and passes the book to the human.*

Case 2
Approach N19⇒ Open N19⇒ Pass Door N19 to the Kitchen⇒ Approach N22⇒ Open N22⇒ Pick up M1 from N22⇒ Pass Door N19 to the Living Room⇒ Approach NHuman⇒ Place M1 at NHuman
Planning time= 1.82 seconds

Inconsistency during execution at "Pick up M1 from N22". Re-planning is performed...

Approach N20⇒ Open N20⇒ Pick up M2 from N20⇒ Pass Door N19 to the Living Room⇒ Approach NHuman⇒ Place M2 at NHuman
Planning time= 1.14 seconds

Inconsistency during execution at "Pick up M2 from N20". Re-planning is performed...

Pass Door N19 to the Living Room⇒ Approach N15⇒ Open N15⇒ Pass Door N15 to Bedroom 2⇒ Approach N12⇒ Open N12⇒ Pick up M3 from N12⇒ Pass Door N15 to the Living Room⇒ Approach NHuman⇒ Place M3 at NHuman
Planning time= 2.17 seconds

Figure 5.13 shows the robot execution to fulfill the goal. At the initial stage, there are three canned drinks, two in the kitchen (with MOID M1 and M2) and one in bedroom 2 (with MOID M3). The robot's initial plan is to fetch M1 from the fridge in the kitchen. Right before it can fetch the drink, someone takes it. While trying to pick an object, a precondition for the activity is $DYAtMO_t(MOIDres_t) = RobotApproach_t$ that requires the object the robot wants to fetch to be in the location the robot approaches. Therefore, a missing M1 means the precondition cannot be fulfilled, and thus, will lead to re-planning. Re-planning occurs until the robot manages to get a canned drink and passes it to the human.

5.3.2.7 Case 3: Inferences for Making Choices

As an example, when human gives command to turn on the light in the living room, considering there are multiple lights, he/she would convey something like "Turn on a dim light in the living room", instead of "Turn on light A32". In this example, it doesn't matter

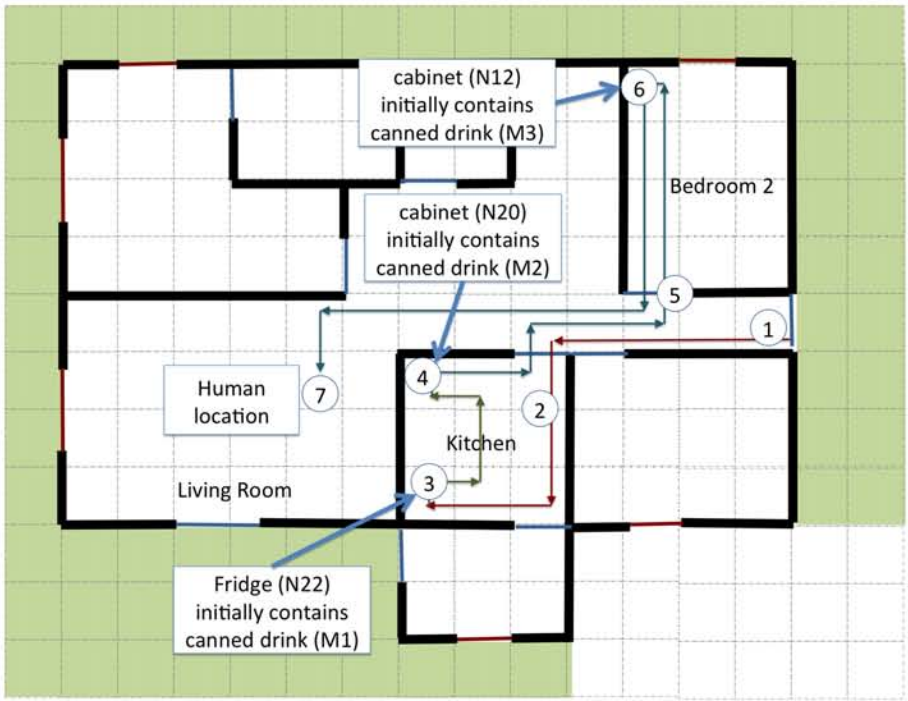


Figure 5.13: *Case 2: Different colors of path trajectories shows different plans, where red represents executed the first plan, which subsequently leads to green (due to missing M1), after which leads to the blue (due to missing M2). From Step 1 to 3, the robot tries to fetch M1 from the fridge N22. In Step 3, due to missing M1 (due to someone taking it away), re-planning is performed, which leads the robot to fetch M2 from the cabinet(N20) in Step 4. As M2 is missing too, re-planning is performed, which leads to the robot fetching M3. It goes to bedroom 2 in Step 5, and fetch M3 in the cabinet in Step 6. Finally, it brings it to the human in the living room in Step 7.*

which light is being turned on, as long as it is a dim light. Therefore, properties of objects are much more crucial. This case demonstrates the capability of the planner to make intelligent choices based on the properties of objects it needs to handle, instead of explicit definition of objects.

The goal is to fetch an object of type "Towel" and located in the garden, and transfer it to cabinet N11. In our case study, our home has two towels M5 (in the master bedroom) and M6 (in the garden). In this goal, no explicit definition of the object ID is specified. Instead the planner needs to select these objects based on the stated properties. The goals are shown as follows:
$\exists x, y((FMOType(x) = Towel) \wedge (DYAtMO_0(x) = y) \wedge NOLocation(y, Garden) \wedge (DYAtMO_K(x) = N11))$

The generated plan is as follows:

Case 3
Approach N33⇒ Open N33⇒ Pass Door N33 to the Garden⇒ Approach N31⇒ Open N31⇒ Pick up M6 from N31⇒ Pass Door N33 to the Living Room⇒ Approach N11⇒ Open N11⇒ Place M6 at N11
Planning time= 1.36 seconds

Executing the plan, the robot will transfer towel M6 in the washing machine to N11. To get towel from the master bedroom, all one needs to do is to change the location to MasterBedroom in the goal as follows:
$\exists x, y((FMOType(x) = Towel) \wedge (DYAtMO_0(x) = y) \wedge NOLocation(y, MasterBedroom) \wedge (DYAtMO_K(x) = N11))$
This demonstrates the ability to make intelligent choices, where making inferences becomes part of the planning process, which is an extension from [70] (where choices need to be explicitly made). ASP approach [146] can provide such descriptive power, but currently it is still new in the field of planning, and it cannot directly handle numbers as shown in Case 4.

5.3.2.8 Case 4: Reasoning and Planning with Numbers

In a lot of cases, manipulation and reasoning with numbers are required, such as the case in volume control and triggers from numerical constraints. CSP planner is able to

cover these domains under one declarative language. This case study will demonstrate this capability through tuning the fan volume based on the temperature. Given the following goal:
$(Temperature > 30) \rightarrow (DYTune_K(N37) > 4)$

N37 is a volume tuner and has an initial state 2. Variable "Temperature" records the temperature of the house, which can be easily embedded into the CSP planner. The states that if the temperature rises above 30, N37 needs to be tuned to more than 4.

Given the current temperature to be 32. The following shows the plan:

Case 4
Approach N37⇒ Tune up N37 by 1⇒ Tune up N37 by 1⇒ Tune up N37 by 1
Planning time= 0.25 seconds

Since the temperature is more than 20, N37 needs to be more than 4. With an initial value of 2, the robot needs to find ways to realize this with the activities it has. The robot will first approach N37, and then uses 1-increment tuner activity 3 times to get N37 to 5.

Currently, from literature review, there is no approach that can accommodate planning and number manipulation and reasoning under the same declarative language, except introducing an extension to it. To be able to describe them in one declarative way has two advantages, which are 1) Work on generalization of robot planning can be done 2) Optimization method can be easily studied to improve planning.

5.3.2.9 Case 5: Reusability of Activities

As stated in the introduction, robot is useful in the evolution of the smart home, where it can dispense services that are yet to be supported by available smart devices. Since the number of smart devices will continue to grow, the robot is required to co-operate with these devices.

The services of these smart devices can be easily constructed with the precondition/effect definition [70], which can be used by the CSP planner. In this case, we consider the door to bedroom 2 (N15) to be installed a motor, which will open/close the door automatically. The activity definition is very simple, where it doesn't have a precondition, and the effect is just $DYStateNO_t(N15) = Open$ or $DYStateNO_t(N15) = Close$.

Lets consider the goal where the robot needs to move M3 (which is in bedroom 2) to

the living room table(N34) as follows:
$DYAtMO_K(M3) = N34$

The generated plan is as follows:

Case 5
N15 opens (Automatic)⇒ Pass Door N15 to Bedroom 2⇒ Approach N12⇒ Open N12⇒ Pick up M3 from N12⇒ Pass Door N15 to the Living Room⇒ Approach N34⇒ Place M3 at N34
Planning time= 0.73 seconds

As shown in the plan, the activity for the automatic door can simply be executed to aid the robot. This can be performed without having to redefine sub-goals or ontologies, or making additional rules regarding additional alternatives. CSP planner can execute them where it sees fit, which shows the benefit of reusability.

5.3.2.10 Case 6: Complex Goals

Case 2 showcase the ability to handle complex goals. In case 1, goals are considered direct commands from the user. But in a lot of cases, goals are imposed by environmental situations and implicit rules, such as the opening/closing of the appropriate doors to cut off rooms to prevent smoke from spreading, turning on the right amount of light when the surveillance camera is activated, and so on.

For this case, assume the user wants to transfer a cold drink out of the fridge(N22) while he/she is attending something else in the wet kitchen, and that there is an inherent rule where the fridge must always be stocked with canned drinks. Besides, there is a little smoke in the wet kitchen, which triggers a goal to request for window in the wet kitchen to be opened. The goals, both from human commands and trigger are listed below:
Goal 1:
$\exists x, y((FMOType(x) = CanDrink) \wedge (DYAtMO_0(x) = N22) \wedge (DYAtMO_K(x) = y) \wedge (FNOType(y) = Cabinet) \wedge (DYStateNO_K(N22) = Close) \wedge (DYStateNO_K(y) = Close))$

Goal 2:
$\exists z((FMOType(z) = CanDrink) \wedge (DYAtMO_K(z) = N22))$

Goal 3:
$\exists x((FNOType(x) = Window) \wedge (DYStateNO_K(x) = Open) \wedge NOLocation(x, WetKitchen))$

The first goal states that there is a canned drink,x ,that should be initially in the fridge and at the final stage, should be in an object, y, that is of type 'Cabinet'. This y should be closed at the final stage, and, like wise for the fridge. The second goal states that there is a canned drink, z, which should be in the fridge at final stage. The third goal states that an object x which is of type 'Window' and located in the Wet Kitchen should be opened. This goal is the smoke triggered goal.

The goals require the planner to make choices on the objects it need to fetch as well as the windows it needs to open, which demonstrates implicit goals.

The following shows the plan:

Case 6
Approach N19⇒ Open N19⇒ Pass Door N19 to the Kitchen⇒ Approach N22⇒ Open N22⇒ Pick up M1 from N22⇒ Approach N20⇒ Place M1 at N20⇒ Close N20⇒ Pick up M2 from N20⇒ Close N20⇒ Approach N22⇒ Place M2 at N22⇒ Close N22⇒ Approach N23⇒ Open N23⇒ Pass Door N23 to the Wet Kitchen⇒ Approach N29⇒ Open N29
Planning time= 82.6 seconds

The simulation is shown in Figure 5.15 and activity flow shown in Figure 5.14. The robot will enter the kitchen and approach the fridge(N22) to obtain the cold drink(M1). It then chooses to place it in the kitchen cabinet(N20) due to the fact that all movable objects should be placed at a non-movable object like the cabinet. Objects will not be placed at non movable objects like doors as restrictions are made. Since N20 has another canned drink inside, the robot will transfer that to the fridge, after which all cabinet and fridge doors are closed. The robot then proceeds to the wet kitchen to open the window N29.

Although the goals are achieved, the time it took for planning is more than 1 minute, which is considered too long, especially in times of emergency. Therefore, methods to distribute the goals into manageable sub-goals are required. But care should be taken as this may introduce Sussman Anomaly. For example, if Goal 1 and 2 are separated, the achievement of Goal 1 (bringing out M1 from the fridge) may be undone when trying to fulfill Goal

2 separately.

Besides, knowledge base can be further exploited by reducing as much uncertainty as possible. The use of existential quantifiers will increase search space, thus, slowing down the planning process. With more uncertainty as to which object to deal with, use of existential quantifiers can be reduced.

We would like to think of the intelligence of smart home to consist of the planner and knowledge base (which consists of database and inference engine). The CSP planner is capable of performing inference to choose the appropriate object to attend to, yet, such inference can be made separately from the knowledge base. Therefore, there is a gray area of who should be dealing with that. In this thesis, we are not ready to answer this question as it depends on the full design of the smart home, but the implication can be seen from how the goal is constructed as in Case 6. Given the 3 goals above (which are implicit), the planner needs to perform inference to choose the objects. But the 3 goals above can also be simplified to explicit direct goals, such as:
$(DYAtMO_K(M1) = N20)$, which means canned drink M1 should be in cabinet N20 as goal.
To have such simplified goal, knowledge base needs to come up with direct answers of what objects to attend to. This also means the making of choices is separated from the planner, and other complications might ensue. If the inference engine is run according to OWL description logic, there is yet another issue of it taking on the open world assumption, which requires further work to combine them.

Therefore, one can use solely the planner by providing it with well-constructed implicit goals (but may be complicated), and which, may take up more planning time, or, the knowledge base can come provide direct answers to the choices, but risk further complications due to different logic used.

5.3.3 Remark

Service composition provides robot with a higher level plan of execution. Service composition for a service robot is developed via representing and solving planner problem in terms of CSP. The few basic type of services and their corresponding terms are optimized for faster planning. Simulation shows that the system is able to perform tasks bounded by complex logical rules, yet no pre-programming is required to combine different services as they are loosely bounded.

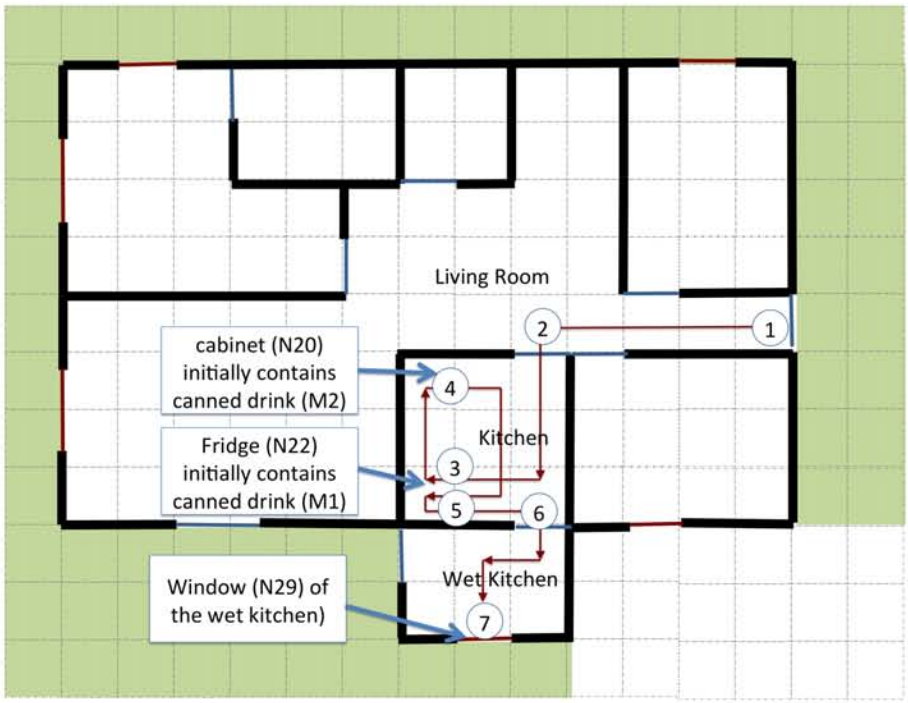


Figure 5.14: *Case 6: The robot proceeds to the kitchen at Step 2 and goes toward the fridge(N22) to fetch a canned drink(M1) in Step 3. It subsequently goes to the cabinet(N20) to place M1. N20 contains another canned drink(M2). The robot will fetch M2 and close N20 at the end of Step 4. It proceeds to the fridge to place M2 and closed it in Step 5. After that, the robot approaches the wet kitchen in Step 6 and opens the window in Step 7.*

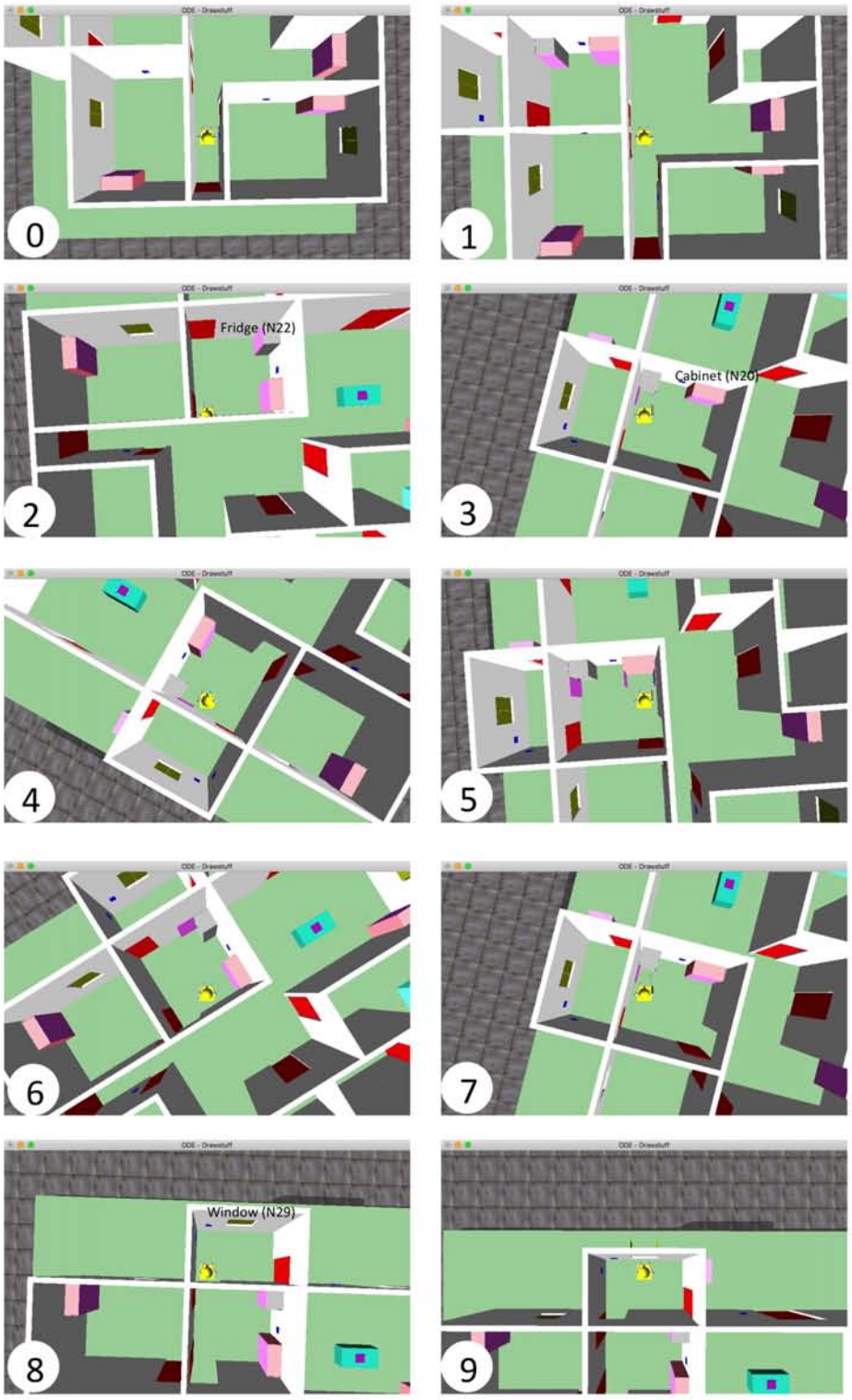


Figure 5.15: *Robot planning simulation*

Chapter 6

Human-centric Implementation for Personalized Service Provision

Previous chapters explain the various modules in realizing human-centric system. This chapter describes the implementation of human-centric system in a prototype smart home to demonstrate the applicability of the approach.

Human-centric system will try to maximize the fulfillment of personal constraints (known as QOLs) by using the devices it can control. Human Identification Module (HIM) identifies (and possibly tracks) people who are within the vicinity of control such that their personalized data can be extracted, like their personal info and QOL. Identification is required to be open set because it needs also to return whether it knows the person or not. If the person is unfamiliar, default QOL is assigned to him/her. This work emphasizes individual goal fulfillment, thus, a simpler method for tracking using clothing colors is used as described in the home setup of Section 6.2.2.1. With the devices connected to the smart home and the house layout, Semantic Reasoner Module (SRM) will wrap up their services to become planning operators (or activities) fit for planning. Automated Planner Module (APM) will then coordinate the devices to maximize fulfillment of QOL. For clarity, we will use "activity" to refer to service operators for planning, unlike "service", which is of a much general use. Service composition through service mash-ups that are generated by semantic reasoners is not discussed in this architecture. We assume the right services are already composed that can be used for planning.

6.1 Building Ontology

The building ontology defines the relationships between entities and their concepts that are within the building environment, described according to OWL DL. Two ontologies are defined for the building ontology, which are, ontology for human and for devices. This section gives the gist of the ontology, where specific structures will be illustrated in Section 6.2.2.

6.1.1 Human Ontology

As human-centric system focuses on fulfilling the needs of human, it needs to have a syntactic representation of the human where whatever decisions and manipulations can be made from and on it. That said, every human that is detected will be assigned a node, where the graph associated with the node is shown in Figure 6.1. The node is indicated by 1 and it falls under class $Human$, which consists of classes of different privileges $priv$, where larger privileges subsumes lower privileges (For example, $priv2 \subseteq priv3$). Privileges with lower numbers indicate a higher privilege compared to those with higher numbers. For example, the host's $Human$ node falls under $priv1$, and the friends fall under $priv2$. The $Human$ node is connected to its information via property $hasInfo$, which directs to further or raw information of the person.

Preference variables (shown as node 2 and 3)are related to $Human$ node through property $hasVar$. Examples of preference variables are alarm clock setting, meal preferences and checkups per month. Although some are not considered preferences but a requirement set by others (such as checkups per month) or might be inferred through sensor networks such as inferred activities or goal recognition, but for simplicity, we will termed them as preferences. Every preference variables are associated with a Boolean knowledge variable, which is related to it via property $hasKVar$, where they are set to 1 if the preference variables they are associated to are considered known. Preference variables have names and values, which are associated through property $hasName$ and $hasValue$ respectively.

Human location is constantly being updated based on the room he/she is in. Human location can be obtained via property $hasLocation$. There is a special variable with the name of 'Location' (shown as node 6) that records the value of whatever being assigned to the human's location.

$Human$ node is connected to QOL (shown as node 4) through property $hasQOL$. These are personal constraints that the automated planner needs to optimize. QOLs are considered manually assigned by the host ($priv1$) for his/her friends ($priv2$) and strangers ($priv3$). All constraints in QOL have individual costs denoted as $w(q, h)$, where q is the constraint index

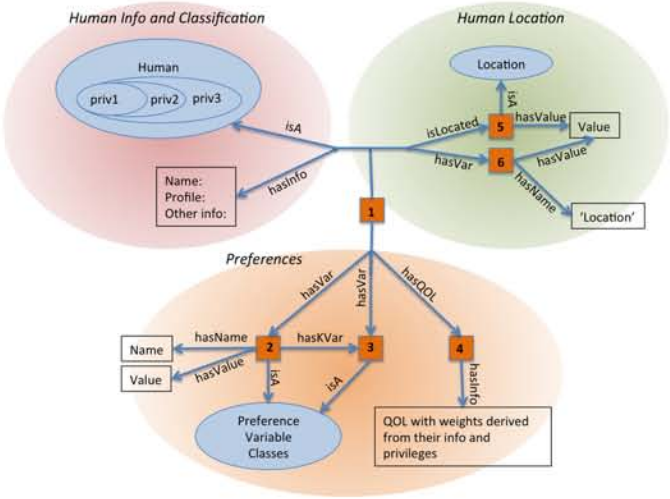


Figure 6.1: *Ontology for human. Square indicates an instance, ellipse indicates a class, arrow indicates a property and square indicates values or node that leads to information. These indications apply to future graphs.*

in the QOL and h is the human identifier. The cost is the penalty if the constraint from the QOL is not met. At the current stage of work, costs are manually assigned. Total cost of each constraint is obtained via $w(q, h) + priv(h)$, where $priv(h)$ is the privilege cost of person h (The cost is 50, 30 and 10 for $priv1$, $priv2$ and $priv3$ respectively, which means, penalty is higher if constraints are not met for people with higher privileges). Besides, as stated, currently, we consider QOLs to be assigned manually by the host, who needs to have knowledge on the devices in his/her home, or have knowledge to set implicit constraints. More work has to be done for more intelligent and automatic assignment of costs.

6.1.2 Device Ontology

Device ontology is built for every controllable device to enable device and service discovery, interoperation, composition and configuration, which is a heavily researched field [3, 92]. The associations made are to connect services to compatible and available devices or services, where services are dispensed based on policies. Semantic mark-ups and ontologies provide the means for service composition through the use of semantic reasoners. Here, we introduce the ontology extension to accommodate planning operators.

Besides the device type and services provided by the device, planning operators also requires information on the variables. For example, for a dim lamp, apart from assigning a type to the light as dim light, information on what service should be executed to turn on/off the light is needed, as well as an indicator (or variable) to indicate that the light is on/off. Device graph is shown in Figure 6.2. As in [3], devices (node 1) are associated with

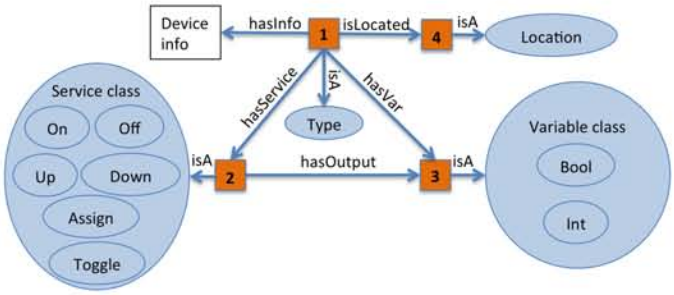


Figure 6.2: *Ontology for device*

a type class and location. The type classes used in our case studies are shown in Figure 6.4. This can support service composition, but in order to extend to planning, services (node 2) and their corresponding variables (node 3) are included as well, both having their own type classes respectively. Variables can be either $Bool$ or Int. The service referred here is called the atomic service, which has not been wrapped up with planning rules such as preconditions. Service operators for planning (which is equipped with preconditions and effects) requires wrapping up atomic services via rules, which, at the moment, are user-specified. The rules associate the nodes to the preconditions, effects and other parameters of the planning operators. The rules used for the case studies are shown in Table 6.5 in Section 6.2.2.3, where detailed explanations are given.

Currently, atomic service can be assigned to 6 types of classes: Class On/Off are for services that can assign $1/0$ to variables under $Bool$, which is connected via property $hasOutput$. Class $Up/Down$ are for services that can increase/decrease values in variables under Int. Class $Assign$ are for services that assign values to variables under Int or $Bool$ that is connected via property $hasOutput$ from a constant or from other variables that are connected via property $hasInput$ (not shown in the figure). Class $Toggle$ performs negation on variable under $Bool$.

6.2 System Implementation

Current home automation system either covers simple service activation with knowledge representation, service composition that requires expert design for complex planning, or planning without optimization. This section demonstrates the human-centric system that supports plan generation and optimization to maximize personalized constraints.

Reasoner FaCT++ [132] is used for building ontology reasoning, and Z3 [35] is applied for satisfiability test. A separate module is also built to support necessary SPARQL querying capability (to emulate SWRL) on building ontology supported by FaCT++. For the weighted

constraint satisfaction planner, we set the maximum time threshold to 10 seconds for planning. This means if the planner has not finish searching the optimum solution within 10 seconds, the current best solution is chosen instead. The whole system is run on an 2.5GHz Intel Core i5 computer with 4GB RAM.

Parameters for BMA are number of generations N_{gen}, number of bacteria N_{ind}, number of clones N_{clones}, mutation segment length l_{bm}, number of infections N_{inf}, and infection segment length l_{gt}. For speed and simplicity, l_{bm}, N_{inf} and l_{gt} are all set to 1. The other parameters are set from preliminary tests on result consistency and time consumption, which are, $N_{gen} = 10$, $N_{ind} = 3$, and $N_{clones} = 3$.

6.2.1 Planning Evaluation

6.2.1.1 Problem Relaxation Comparison Test

As explained in Section 4.6.4, original planning problem is relaxed by eliminating preconditons, and converting them to sub-goals with costs.

In this section, comparison is performed to evaluate the improvement contributed by the conversion of precondition to sub-goals in problem relaxation for branch and bound, instead of relaxation through eliminating preconditions only (termed normal relaxation). The test is performed to compare the time they take to obtain the optimal solution. The experiment is conducted with maximum $K = 9$, 11 and 13 horizon steps of activity. Goals are designed, and are randomly assigned during experiment, such that they need at least K steps to reach the optimal solution. Number of activities is not reduced. Table 6.1 is the comparison result. Search will still proceed even when the optimal solution is reached (as it doesn't know that it's the best solution until full search is completed). Result for time consumption to complete full search (for $K = 9$, as higher K takes more than 3 minutes) is shown in Table 6.2.

It can be observed that, with sub-goal conversion, optimal solution can be obtained very quickly relative to without conversion. On the other hand, search time is only slightly improved. We need the algorithm to obtain the optimal solution fast, and place less emphasis on the total search time as there is a time threshold of 10 seconds. Therefore, relaxations through sub-goal conversion are used due to their superior performance.

6.2.1.2 Evaluation on Time Consumption on Constraint and Activity Numbers

Test is performed to evaluate time consumption to reach 50% improvement from non-action with different number of constraints (each constraint consists of grounded terms

Table 6.1: *Time consumption (seconds) to reach optimal solution for different relaxation approach*

Description	K	Median	1st Quartile	3rd Quartile	Min	Max
Normal Relaxation	9	30.0	17.1	57.1	8.7	93.2
	11	55.17	19.6	74.4	11.2	110.6
	13	85.6	31.2	151.0	37.0	291.4
With Sub-goal Conversion	9	8.4	7.7	30.7	7.3	55.9
	11	16.6	11.3	31.2	10.4	97.4
	13	34.1	30.7	66.4	23.6	133.0

Table 6.2: *Time consumption (seconds) for full search ($K = 9$) for different relaxation approach*

Description	Median	1st Quartile	3rd Quartile	Mean	Std	Min	Max
Normal relaxation	88.0	59.0	103.7	83.1	28.9	37.9	123.2
With Sub-goal Conversion	74.7	54.2	103.4	75.1	38.2	13.0	122.6

(for finite domain data type) and /or integers, and, Boolean and/or Algebraic relations), where $K = 9$, and number of activities is 50. Non-action means the total cost of not doing anything. The reason optimum solution is not required is because the planner can replan to reach optimal or near optimal solution, which is demonstrated in Case 1 in Section 6.2.3.1. No activity search space reduction and goal reduction are performed. Result is shown in Table 6.3. It can be observed that time consumption rises quite linearly with the number of constraints, which means it scales well with rising constraints.

Evaluation is also performed on different number of activities (without reduction), with 20-50 constraints. Other test conditions are similar to previous settings. Results are shown in Table 6.4. It can be observed that time consumption rises linearly with number of activities as well, though performance consistency drops. Therefore, activity search reduction proposed in Section 4.4.1 is crucial in ensuring scalability in terms of activity numbers.

6.2.2 Home and Device Setup

6.2.2.1 Home Setup

An experimental home is set up as shown in Figure 6.3, where various snapshots from different camera views of the actual setup are taken. Figure 6.3a shows the information of the home layout and the devices for each room, and also the direction of camera view for the snapshots. The information shown here can be used to construct ontology that dictates the class and adjacency of the rooms, which is denoted as location ontology.

Table 6.3: *Time consumption with different constraint numbers*

No. of Constraints	20	200	2,000	20,000
Median	8.6	16.3	142.0	1729.3
1st Quartile	8.0	15.4	138.1	1645.3
3rd Quartile	11.4	18.8	147.0	2105.7

Table 6.4: *Time consumption with different activity numbers*

No. of Activities	50	100	150	200
Median	10.6	31.1	52.7	75.2
1st Quartile	10.2	30.8	50.1	72.8
3rd Quartile	10.8	31.6	58.5	126.8

There are also a clock and temperature sensor than updates variable $time$ and $GeneralTemperature$ respectively. Besides that, a fire event sensor is installed in the kitchen, which set the Boolean variable $KitchenFireEvent$ to 1 is there is a fire, and 0 otherwise. There is also a Boolean variable $PlanTimer$, which is be set to 1 after certain duration. After one planning cycle is finished, it will be set back to 0. It usage as implication for constraints is explained in Section 6.2.2.4.

There are three pressure sensors located in the bed, the toilet and the floor next to the stove in the kitchen, where their activation indicate sleeping, in the toilet and cooking respectively. Currently, their activation manipulates the variable $HumanState$ (more details on the variable in Section 6.2.2.4).

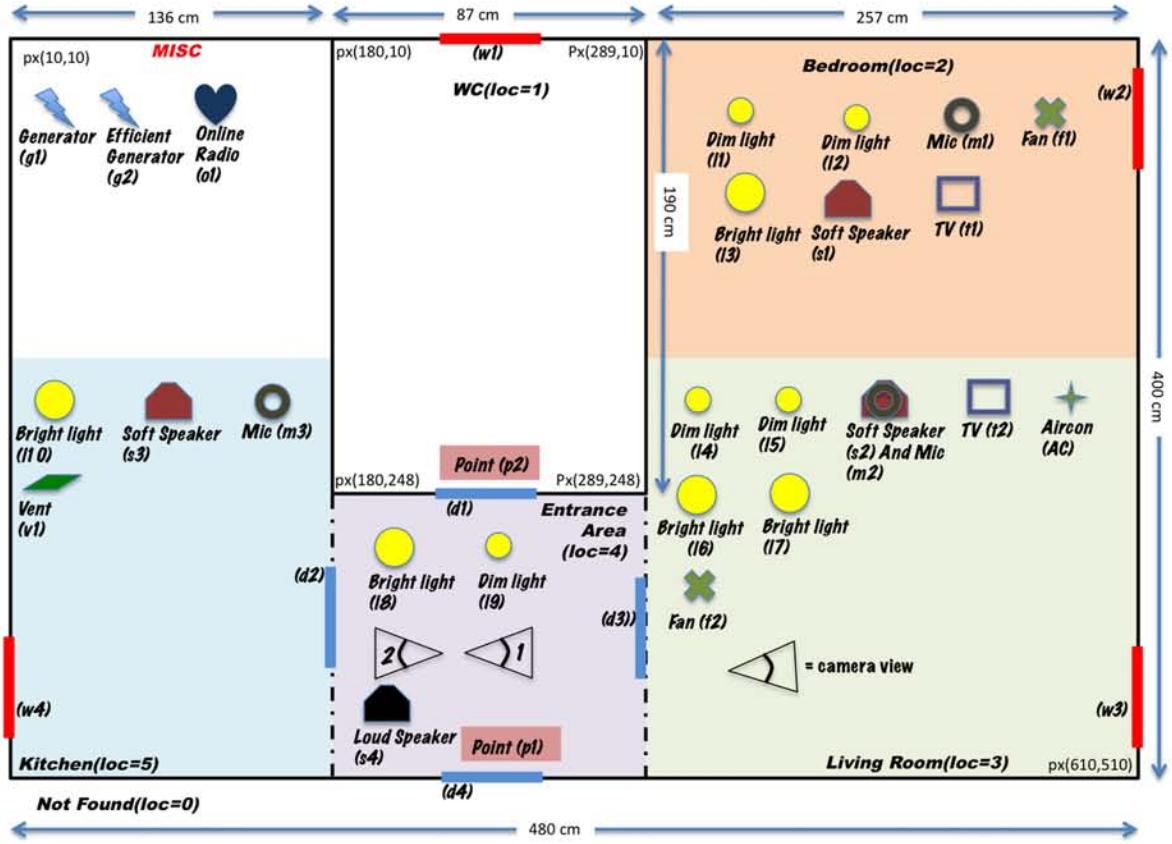
Two Kinect sensors are used to localize human in the house. To enable multi-human tracking and to provide them with appropriate privileges, we use color information of the clothes the person wears. Black shirt is the host, blue shirt is the friend and the others are strangers. Information on human localization will then be used to determine which room the person is in, thus, assigning the location node to the $Human$.

6.2.2.2 Device Setup

Figure 6.4 shows the device class type and the assignment of devices to the classes, where the device node of each device will be assigned as shown in Figure 6.5. Microphone 2 and speaker 2 are shown as a single object because this is realized by a communication robot we developed called iPhonoid. More information on the robot can be found in [23]. Speakers 1, 2 and 3 are soft speakers, meaning they can only be heard when the person is in the same room as them. Speaker 4 is a loud speaker that can be heard throughout the house.

2 generators, $g1$ and $g2$, are shown as thunder-shaped object. They don't exist in a particular room, thus, drawn in the $MISC$ area. $g2$ is a lower powered generator compared to $g1$, but it is considered much more efficient. All devices under light, tv and fan can run given generator 1 and 2, except the air-conditioner, which needs $g1$. This requirement is set as extended constraints and will be explained in Section 6.2.2.4. Another component that don't exist in any particular room is the online radio $o1$ (heart-shaped).

The devices described above are all realized through an emulator, except the speaker and



(a) *Home Layout and Devices Diagram*

(b) *Camera View 1*

(c) *Camera View 2*

Figure 6.3: *Home layout and devices*

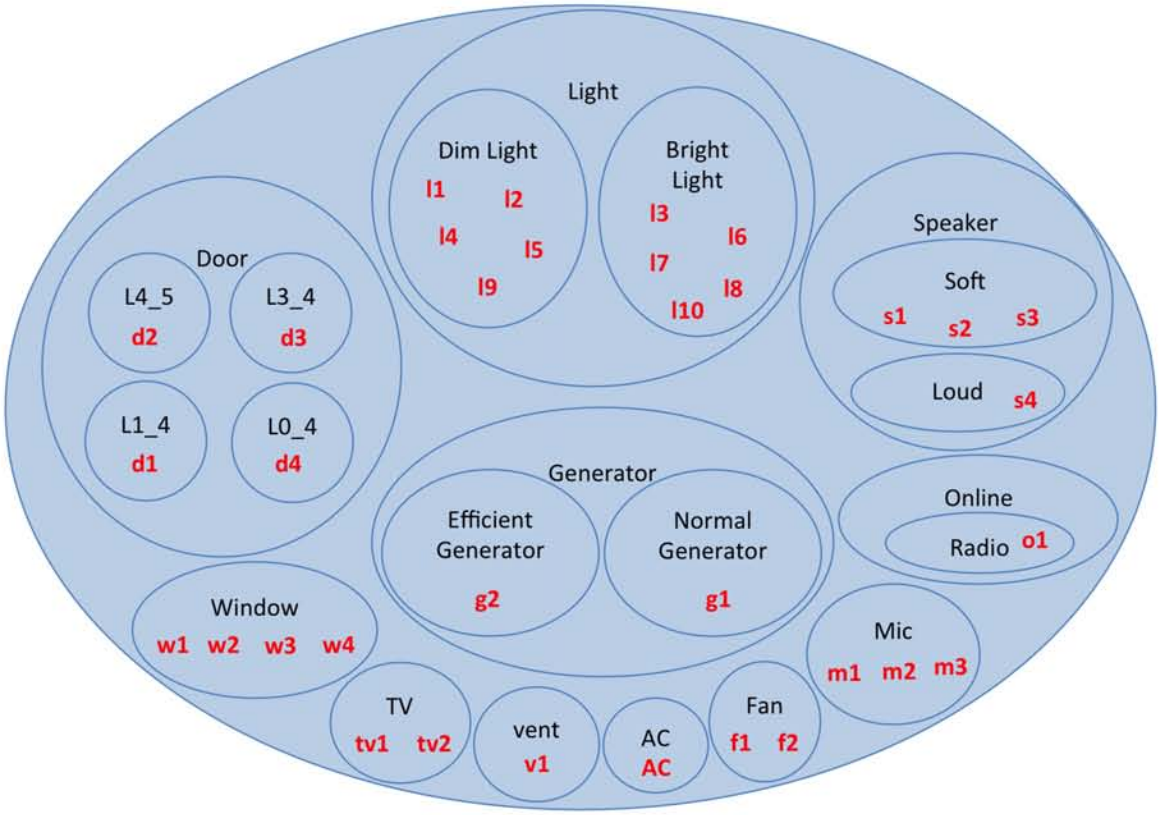


Figure 6.4: *Device class type - IDs from Figure 6.3a*

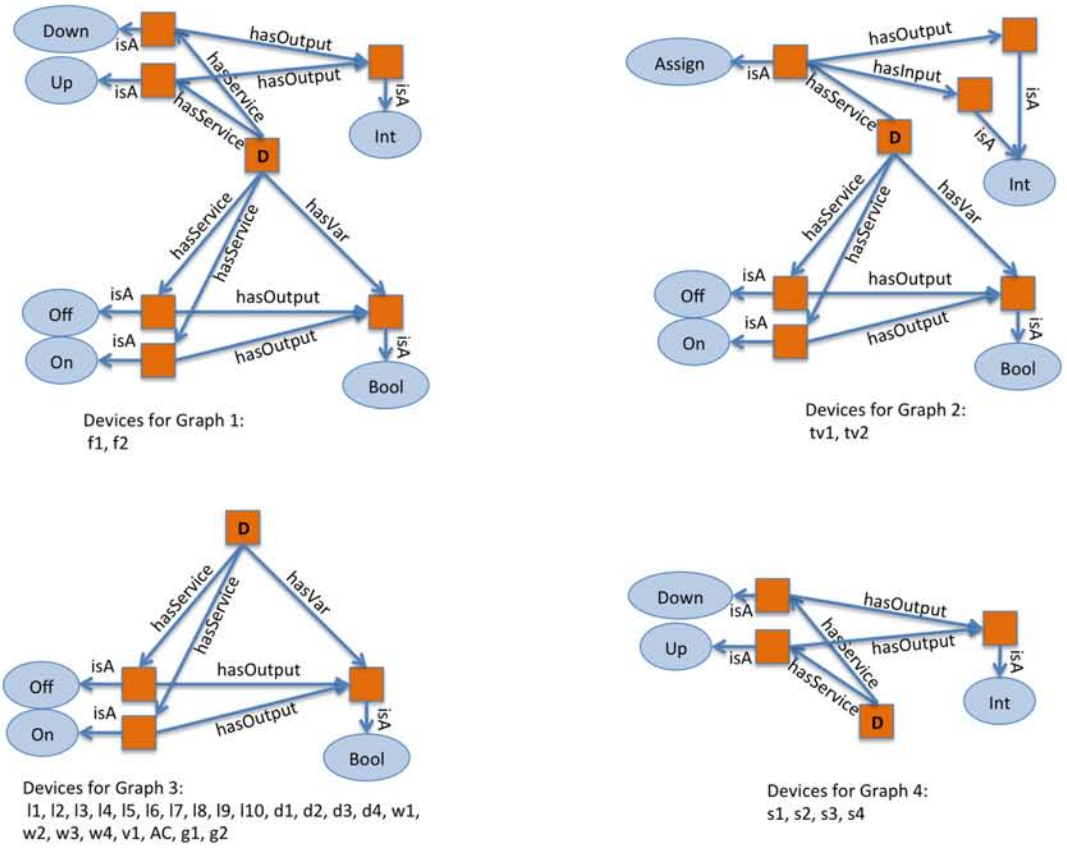


Figure 6.5: *Semantic graphs for the devices. D indicates the device node. Device IDs are listed below their corresponding graph*

the microphones. We believe there will be no loss of generality in using emulated devices as we assume these devices don't affect events. For example, we don't take into account the fact that the turning on of light will cause certain light sensors to pick that up and induce further goals. Such cases are subject of future work. That said, speaker and microphone should be implemented in hardware as they concern interactions with humans.

Semantic graphs (which follows the description in Section 6.1) for the devices are shown in Figure 6.5. Fan $f1$ and $f2$ applies Graph 1 because they can be turned on/off and be tuned up/down. TV $tv1$ and $tv2$ uses Graph 2 as they can be turned on/off and change the channels by transferring information from the input variable. Speaker $s1$, $s2$, $s3$ and $s4$ is always on, but their volume can be controlled, thus, applies Graph 4. The rest of the devices (except the microphones) can only be turned on/off, which applies Graph 3.

6.2.2.3 Generating Planning Activities from Atomic Services

As explained in Section 6.1, devices only provide atomic services. To enable planner to use them, they need to be wrapped up by associating them to planning services through rules shown in Table 6.5. $nonManaged$ is to check whether the objects of its argument have been dealt with or not.

The generated activities are shown in Table 6.6. For easier reading, in the table, the variables are given relevant names. In actuality, all variables are just nodes that have their own IDs. To shorten the table, we replace a list of activities by n. For example, l'n' means $l1, l2, l3...l10$, which are the IDs for the lights. Variables with subscript res is the response variable. Variable $PrefChan$ is preceded by $Host$ and $Frnd$, which indicates that the variable comes from the Host and Friend $Human$ node respectively. The same applies to Human location variable shown as Loc. $t1chan$ and $t2chan$ are variables under $assign$ type for $tv1$ and $tv2$ respectively.

From Table 6.6, activities 1 to 50, 63 and 64 are generated from Association 1 and 2 of Table 6.5, activity 67 to 78 by Association 3 and 4, activity 51 to 53 and 55 to 57 by Association 5, activity 54 and 58 by Association 6, activity 59 to 62 by Association 7, and, 65 and 66 by Association 8 and 9.

Given the generated activities, the costs for the activities are manually assigned based on preferences. For clarity, we explain the motivation behind the cost that are set: Default cost is the minimum at 2. Activities concerning generator $g1$ is set to a higher value of 3 as the use of $g2$ is preferable compared to $g1$. Activities concerning AC is set high at 4 as it is not desirable to turn it on. Activities concerning speaker $s4$ is set at 4 because its loudness is not desirable.

6.2.2.4 Variables, QOL and Goals

Preference variables (shown in Figure 6.1) store states and information pertaining the human that is crucial for planning. The relevant preference variables used in the case studies are $WatchTV$ (inferred state that the human wants to watch TV), $PrefChan$ (preferred tv channel), $PrefTemperature$ (preferred thermostat temperature) and $HumanState$ (State of human such as sick, well, sleeping etc. each with their own IDs). The following are the relevant values for $HumanState$: 0 is healthy and sleeping, 1 is healthy and awake, 2 is sick and awake, 3 is sick and sleeping, 4 is cooking, 5 is in the toilet.

Environment variables store states to describe its state. The relevant environment variables are $time$, $GeneralTemperature$ and $KitchenFireEvent$ as laid out in Section 6.2.2.1. Description for the state of $time$ is as follows: 0 is early morning,1 is morning,

Table 6.5: *Rules for atomic service wrap-up*

No	Association Rule	Precondition and Effects
1	$Device(?D) \wedge hasService(?D,?S) \wedge On(?S) \wedge hasOutput(?S,?V) \wedge nonManaged(?D) \rightarrow: assign(?D,?S,?V)$	Precondition: none Effect: $?V_{t+1} = true$
2	$Device(?D) \wedge hasService(?D,?S) \wedge Off(?S) \wedge hasOutput(?S,?V) \wedge nonManaged(?D) \rightarrow: assign(?D,?S,?V)$	Precondition: none Effect: $?V_{t+1} = false$
3	$Device(?D) \wedge hasService(?D,?S) \wedge Up(?S) \wedge hasOutput(?S,?V) \wedge nonManaged(?D) \rightarrow: assign(?D,?S,?V)$	Precondition: none Effect: $?V_{t+1} = ?V_t + 1$
4	$Device(?D) \wedge hasService(?D,?S) \wedge Down(?S) \wedge hasOutput(?S,?V) \wedge nonManaged(?D) \rightarrow: assign(?D,?S,?V)$	Precondition: none Effect: $?V_{t+1} = ?V_t - 1$
5	$Human(?H) \wedge Speaker(?P) \wedge Mic(?M) \wedge isLocated(?H,?L) \wedge hasVar(?H,?LC) \wedge hasName(?LC, Location) \wedge isLocated(?P,?L) \wedge isLocated(?M,?L) \wedge Priv2(?H) \wedge hasVar(?H,?PC) \wedge hasName(?PC, Channel\_Preference) \wedge hasKVar(?PC,?KPC) \wedge nonManaged(?H,?P,?M,?L) \rightarrow: assign(?H,?P,?M,?L,?LC,?PC,?KPC)$	Precondition: $?LC_t = ?L_t$ Effect: $?PC_{t+1} = Variable_{Res}$, $?KPC_{t+1} = true$
6	$Human(?H) \wedge Loud\_Speaker(?P) \wedge Mic(?M) \wedge isLocated(?P,?L) \wedge isLocated(?M,?L) \wedge Priv2(?H) \wedge hasVar(?H,?PC) \wedge hasName(?PC, Channel\_Preference) \wedge hasKVar(?PC,?KPC) \wedge nonManaged(?H,?P,?M) \rightarrow: assign(?H,?P,?M,?PC,?KPC)$	Precondition: none Effect: $?PC_{t+1} = Variable_{Res}$, $?KPC_{t+1} = true$
7	$Human(?H) \wedge Priv2(?H) \wedge TV(?D) \wedge hasService(?D,?S1) \wedge hasService(?D,?S2) \wedge Assign(?S1) \wedge On(?S2) \wedge hasOutput(?S1,?O) \wedge hasOutput(?S2,?T) \wedge hasInput(?S1,?I) \wedge hasVar(?H,?PC) \wedge hasName(?PC, Channel\_Preference) \wedge hasKVar(?PC,?KPC) \wedge nonManaged(?D,?H) \rightarrow: assign(?D,?H,?O,?I,?T,?PC,?KPC)$	Precondition: $(?T_t = true) \wedge (?KPC_t = true)$ Effect: $?I_{t+1} = PC_t$
8	$Loud\_Speaker(?D1) \wedge Radio(?D2) \wedge hasService(?D2,?S) \wedge On(?S) \wedge hasOutput(?S,?V) \wedge nonManaged(?D1,?D2) \rightarrow: assign(?D1,?D2,?S,?V)$	Precondition: none Effect: $?V_{t+1} = true$
9	$Loud\_Speaker(?D1) \wedge Radio(?D2) \wedge hasService(?D2,?S) \wedge Off(?S) \wedge hasOutput(?S,?V) \wedge nonManaged(?D1,?D2) \rightarrow: assign(?D1,?D2,?S,?V)$	Precondition: none Effect: $?V_{t+1} = true$

2 is noon, 3 is afternoon, 4 is evening, 5 is night, 6 is midnight.

Constraints in QOL are specified by individuals. Given that there are two persons (a host and his friend) that the smart home knows, thus, there are two sets of QOL from the human in the case studies. Relevant constraints of the QOLs and their costs are shown in Table 6.7. The costs shown include costs from privileges, where Host is $priv1$ and Friend is $priv2$, thus, 50 and 30 respectively. Constraints 1 to 14 are the constraints from Host's QOL, and constraints 15 to 26 are the constraints from Friend's QOL. The remaining constraints are set by the host for the home, and are not tied particularly to any person.

For simplicity, when referring to a variable x that belongs to a device of certain device type and its value, instead of writing $Light(D) \wedge hasVar(D,x) \wedge hasValue(x,1)$, in Table 6.7, it will be written as $Light(x) \wedge (x = 1)$ (or $Light(x) \wedge x$ if x is Boolean). Constraints 27 to 36 applies to every lights of the house, and constraint 37 applies to every vents.

Predicate $connected(a,b)$ returns whether location a and b are connected or not. Two rooms are connected if one can walk to the room without obstructions in between such as closed doors. The truth value of the predicate can be easily obtained given the state of doors using simple SAT solvers or from multi-stage description logic [17].

Constraints 11 and 12 can be generated from constraints 9 and 10, and constraints 23 to 26 can be generated from constraints 21 and 22. This is because the fulfillment of the original constraints implies the fulfillment of the generated constraints, which enables better

Table 6.6: *Generated activities*

No	Precondition	Effects	Cost
1	none	$g1_{t+1} = true$	3
2	none	$g1_{t+1} = false$	2
3	none	$g2_{t+1} = true$	2
4	none	$g2_{t+1} = false$	2
5-14	none	$l`n'_{t+1} = true$	2
15-24	none	$l`n'_{t+1} = false$	2
25-26	none	$f`n'_{t+1} = true$	2
27-28	none	$f`n'_{t+1} = false$	2
29	none	$AC_{t+1} = true$	4
30	none	$AC_{t+1} = false$	2
31-32	none	$t`n'_{t+1} = true$	2
33-34	none	$t`n'_{t+1} = false$	2
35-38	none	$d`n'_{t+1} = true$	2
39-42	none	$d`n'_{t+1} = false$	2
43-46	none	$w`n'_{t+1} = true$	2
47-50	none	$w`n'_{t+1} = false$	2
51	$Host\_Loc_t = 2$	$Host\_PrefChan_{t+1} = Host\_PrefChan_{res}$, $KHost\_PrefChan_{t+1} = true$	2
52	$Host\_Loc_t = 3$	$Host\_PrefChan_{t+1} = Host\_PrefChan_{res}$, $KHost\_PrefChan_{t+1} = true$	2
53	$Host\_Loc_t = 5$	$Host\_PrefChan_{t+1} = Host\_PrefChan_{res}$, $KHost\_PrefChan_{t+1} = true$	2
54	none	$Host\_PrefChan_{t+1} = Host\_PrefChan_{res}$, $KHost\_PrefChan_{t+1} = true$	4
55	$Frnd\_Loc_t = 2$	$Frnd\_PrefChan_{t+1} = Frnd\_PrefChan_{res}$, $KFrnd\_PrefChan_{t+1} = true$	2
56	$Frnd\_Loc_t = 3$	$Frnd\_PrefChan_{t+1} = Frnd\_PrefChan_{res}$, $KFrnd\_PrefChan_{t+1} = true$	2
57	$Frnd\_Loc_t = 5$	$Frnd\_PrefChan_{t+1} = Frnd\_PrefChan_{res}$, $KFrnd\_PrefChan_{t+1} = true$	2
58	none	$Frnd\_PrefChan_{t+1} = Frnd\_PrefChan_{res}$, $KFrnd\_PrefChan_{t+1} = true$	4
59	$(t1_t = true) \wedge (KHost\_PrefChan_t = true)$	$t1chan_{t+1} = Host\_PrefChan_t$	2
60	$(t1_t = true) \wedge (KFrnd\_PrefChan_t = true)$	$t1chan_{t+1} = Frnd\_PrefChan_t$	2
61	$(t2_t = true) \wedge (KHost\_PrefChan_t = true)$	$t2chan_{t+1} = Host\_PrefChan_t$	2
62	$(t2_t = true) \wedge (KFrnd\_PrefChan_t = true)$	$t2chan_{t+1} = Frnd\_PrefChan_t$	2
63	none	$v1_{t+1} = true$	2
64	none	$v1_{t+1} = false$	2
65	none	$o1_{t+1} = true$	2
66	none	$o1_{t+1} = false$	2
67-70	none	$s`n'vol_{t+1} = s`n'vol_t + 1$	2
71-74	none	$s`n'vol_{t+1} = s`n'vol_t - 1$	2
75-76	none	$f`n'vol_{t+1} = f`n'vol_t + 1$	2
77-78	none	$f`n'vol_{t+1} = f`n'vol_t - 1$	2

optimization by generating more goals. Since such goals can be checked in linear time, it imposes negligible effect.

Constraints 27 to 38 depends on $PlanTimer$, which is periodically set to one. Therefore, these constraints will be periodically considered. The purpose of these constraints is to maintain default state such as turning the lights off if there is nobody there.

Constraints $E1$ and $E2$ are extended constraints. Unlike other constraints that need to be fulfilled at the end of the plan, extended constraints are required to be met throughout the whole planning sequence. $E1$ states that if lights, tv or Fan are turned on, at least one device under generator class needs to be on, regardless of whether its an efficient or normal generator. $E2$ states that is any air-conditioner is on, normal generator needs to be turned on.

6.2.3 Demonstration Cases

Given the problems stated in Section 1.3, automatic device binding is achieved as shown in Table 6.6. This section will present the case studies to demonstrate the system's capability in handling problems on complex plan generation and over-constrained situation.

The cases also show the system's capability to extend to more complicated goals, namely goals from plug-in functions and extended goals and constraints that need to be fulfilled throughout the planning process. Reason behind plug-in function extension demonstration is that there are a lot of goals that cannot be efficiently logically represented in the QOL list. Adding a plug-in function into the QOL list can help extend the QOL to such goal, such as goals that depend on structured reasoning. As for extended goals, normal goals in QOL only require their fulfillment after the whole plan is completely executed. Extended goal goes a bit further to ensure that it is achieved throughout the plan.

Case 1 is on continuous planning and optimization, where planning is done in multi-stages to reach the optimum solution due to time threshold and plans that are too complex to generate and execute at one go. Case 2 is on reasoning and dealing with conflicting constraints. Case 3 demonstrates the simplicity to extend the planner to support plug-in functions. Case 4 demonstrates the use of extended goals that maintains certain condition throughout the plan, which is crucial if sequence ordering is important.

6.2.3.1 Case 1: Continuous Planning and Optimization

Case 1 demonstrates the ability of the system to continually perform planning and optimization with uncertain information. In this case, the time is set at noon, where the temperature is 24 degrees. Fans are off and their initial volume is 1. The host and the friend

Table 6.7: *Constraints from QOLs*

No	Constraints	Cost
1	$(Host\_WatchTV \wedge (Host\_Loc = 2)) \rightarrow (KHost\_PrefChan \wedge t1chan = Host\_PrefChan)$	100
2	$(Host\_WatchTV \wedge (Host\_Loc = 3)) \rightarrow (KHost\_PrefChan \wedge t2chan = Host\_PrefChan)$	100
3	$(Host\_WatchTV) \rightarrow (KHost\_PrefChan \wedge ((t1chan = Host\_PrefChan) \vee (t2chan = Host\_PrefChan)))$	90
4	$((Host\_Loc = 2) \wedge (Host\_HumanState = 1) \wedge ((time = 5) \vee (time = 6))) \rightarrow (\exists x(Dim\_Light(x) \wedge isLocated(x, 2) \wedge (x = 1)) \wedge \forall y(Bright\_Light(y) \wedge isLocated(y, 2) \wedge (y = 0)))$	100
5	$((Host\_Loc = 3) \wedge (Host\_HumanState = 1) \wedge ((time = 5) \vee (time = 6))) \rightarrow (\exists x(Dim\_Light(x) \wedge isLocated(x, 3) \wedge (x = 1)) \wedge \forall y(Bright\_Light(y) \wedge isLocated(y, 3) \wedge (y = 0)))$	100
6	$((time = 1) \wedge (Host\_HumanState = 1) \wedge \neg(Host\_Loc = 0)) \rightarrow music$	60
7	$((time = 1) \wedge (Host\_HumanState = 1) \wedge \neg(Host\_Loc = 0)) \rightarrow (s4vol > 5)$	60
8	$(((Host\_HumanState = 0) \vee (Host\_HumanState = 2) \vee (Host\_HumanState = 3)) \wedge \neg(Host\_Loc = 0)) \rightarrow \neg music$	100
9	$((Host\_Loc = 2) \wedge (GeneralTemperature > 25)) \rightarrow (f1 \wedge (f1vol > 2))$	100
10	$((Host\_Loc = 3) \wedge (GeneralTemperature > 25)) \rightarrow (f2 \wedge (f2vol > 2))$	100
11	$((Host\_Loc = 2) \wedge (GeneralTemperature > 25)) \rightarrow (f1 \wedge (f1vol > 1))$	100
12	$((Host\_Loc = 3) \wedge (GeneralTemperature > 25)) \rightarrow (f2 \wedge (f2vol > 1))$	100
13	$(\neg(Host\_Loc = 0) \wedge (GeneralTemperature > 32)) \rightarrow AC$	100
14	$(\neg(Host\_Loc = 0) \wedge (GeneralTemperature > 25) \wedge ((Host\_Loc = 2) \vee (Host\_Loc = 3))) \rightarrow (\neg(f1 = AC) \wedge \neg(f2 = AC))$	100
15	$(Frnd\_WatchTV \wedge (Frnd\_Loc = 2)) \rightarrow (KFrnd\_PrefChan \wedge t1chan = Frnd\_PrefChan)$	80
16	$(Frnd\_WatchTV \wedge (Frnd\_Loc = 3)) \rightarrow (KFrnd\_PrefChan \wedge t2chan = Frnd\_PrefChan)$	80
17	$(Frnd\_WatchTV) \rightarrow (KFrnd\_PrefChan \wedge ((t1chan = Frnd\_PrefChan) \vee (t2chan = Frnd\_PrefChan)))$	70
18	$((Frnd\_Loc = 2) \wedge (Frnd\_HumanState = 1) \wedge ((time = 5) \vee (time = 6))) \rightarrow (\exists x(Bright\_Light(x) \wedge isLocated(x, 2) \wedge (x = 1)))$	80
19	$((Frnd\_Loc = 3) \wedge (Frnd\_HumanState = 1) \wedge ((time = 5) \vee (time = 6))) \rightarrow ((l4 + l5 + l6 + l7) > 1)$	80
20	$(((Frnd\_HumanState = 0) \vee (Frnd\_HumanState = 2) \vee (Frnd\_HumanState = 3)) \wedge \neg(Frnd\_Loc = 0)) \rightarrow \neg music$	80
21	$((Frnd\_Loc = 2) \wedge (GeneralTemperature > 23)) \rightarrow (f1 \wedge (f1vol > 3))$	80
22	$((Frnd\_Loc = 3) \wedge (GeneralTemperature > 23)) \rightarrow (f2 \wedge (f2vol > 3))$	80
23	$((Frnd\_Loc = 2) \wedge (GeneralTemperature > 23)) \rightarrow (f1 \wedge (f1vol > 2))$	80
24	$((Frnd\_Loc = 3) \wedge (GeneralTemperature > 23)) \rightarrow (f2 \wedge (f2vol > 2))$	80
25	$((Frnd\_Loc = 2) \wedge (GeneralTemperature > 23)) \rightarrow (f1 \wedge (f1vol > 1))$	80
26	$((Frnd\_Loc = 3) \wedge (GeneralTemperature > 23)) \rightarrow (f2 \wedge (f2vol > 1))$	80
27-36	$PlanTimer \rightarrow (\forall x(Light(x) \wedge x = 0))$	10
37	$PlanTimer \rightarrow (\forall x(Vent(x) \wedge x = 0))$	10
38	$PlanTimer \rightarrow (\exists x(Generator(x) \wedge x = 1) \rightarrow (g1 + g2 = 1))$	10
39	$KitchenFireEvent \rightarrow (\forall x((x = Host\_Loc) \rightarrow \neg connected(5, x)))$	50
40	$KitchenFireEvent \rightarrow (\forall x((x = Frnd\_Loc) \rightarrow \neg connected(5, x)))$	50
41	$KitchenFireEvent \rightarrow v1$	50
E1	$\exists x((Light(x) \vee TV(x) \vee Fan(x)) \wedge x = 1) \rightarrow (\exists y(Generator(y) \wedge y = 1))$	50
E2	$\exists x(AC(x) \wedge x = 1) \rightarrow (\exists y(Normal\_Generator(y) \wedge y = 1))$	50

are in the living room, where both want to watch TV. Their $PrefChan$ variables are not known.

The case starts off by host and friend requesting the robot (s2 and m2) that they want to watch TV as shown in sequence A of Figure 6.6. Communication with the robot starts off by stating the identity, followed by a keyword that the robot understands. In this case, the host and friend both request for watching TV, which the robot will set variable $WatchTV$ to 1 for both of them. The fact that they are in the living room and have their $WatchTV$ set means constraints 2 and 14 in Table 6.7 are considered as goals for the planner since they have their implications met (Recall that this helps reduce activity search space to increase speed as explained in Section4.4.1).

Given the current state, the planner will generate the following sequence of activities (number in the parenthesis is the activity ID from Table 6.6):

First plan:

Turn on $g2$(3) ⇒ Turn on $f2$(26) ⇒ Turn on $tv2$(32) ⇒ Ask Host about channel(52) ⇒ Increase $f2$ volume by 1(76) ⇒ Ask Friend about channel(56) ⇒ Change $tv2$ channel(61) ⇒ End

Second plan:

Turn on $tv1$(31) ⇒ Change $tv1$ channel(60) ⇒ Increase $f2$ volume by 1(76) ⇒ Increase $f2$ volume by 1(76) ⇒ End

---

The plan starts off by turning on generator $g2$. Since $g2$ has a lower cost to turn on compared to $g1$, it is selected by the planner. Fan $f2$ is then turned on, followed by $tv2$ in the living room. The robot then proceeds to ask the host about his preferred channel shown in sequence C in Figure 6.6. The plan also attempts to increase the volume of the fan from initial volume of 1. This is due to constraints 22, 24 and 26. After that, it proceeds to asking the friend of his preferred channel shown in sequence D. Finally, the $tv2$ channel is switched (to host's preference). The device activation flow can be observed in Figure 6.7 from sequence 1 to 4.

From Figure 6.6, the host and friend has different preferences for channel. $tv2$ has been used to fulfill the host's constraints due to it having a higher cost.

But all is not lost for the friend because of the constraint 17 in Table 6.7, which states that any television will do fine, although at a slightly lower cost. At the same time, the fan volume is at 2, which does not fulfill constraints 22 and 24.

The second planning therefore proceeds to continue the optimization. Since it already knows about the friend's preference, it turned on $tv1$ in the bedroom and switched to the

Figure 6.6: *Robot communication flow for information extraction in Case 1*

preferred channel. The plan is completed by increasing the fan volume to 4.

6.2.3.2 Case 2: Making Intelligent Choices under Conflicting Constraints

Case 2 demonstrates the capability to make intelligent choices of devices under conflicting constraints given the dynamic situation in the home. The case is set at night with temperature at 22 degrees. Initially, the friend is in the bedroom and the host is in the living room as shown in sequence A of Figure 6.8.

The initial state dictates that constraint 5 and 18 of Table6.7 are considered, where host wants at least one dim light to be on and all bright lights to be off if he is in the living room, and, the friend wants at least one bright light to be on if he is in the bedroom, respectively. Planning will generate the following plans:

Turn on $g2(3) \Rightarrow$ Turn on Light $l3(7) \Rightarrow$ Turn on Light $l5(9) \Rightarrow$ End

---

Light $l3$ (bright light in the bedroom) and light $l5$ (dim light in the bedroom) are switched on, where the sequence is shown in Figure 6.9 from 1 to 4.

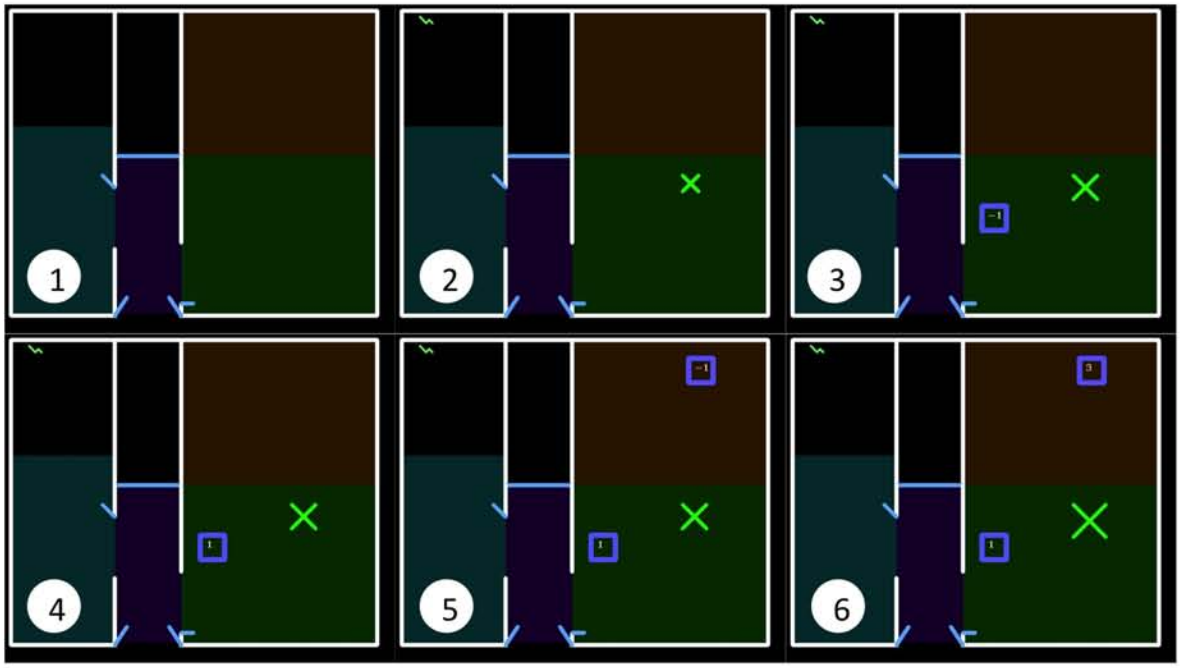


Figure 6.7: *Device visualization for Case 1: Blue square indicates TV that is turned on, where the number within represent channel. Green thunder indicates the efficient generator. Green cross indicates the fan, where its size changes based on the volume.*

After a short while, the host starts to walk to the bedroom. Constraint 5 is no longer considered, and constraint 4 will be considered, which is conflicting with constraint 18 (from the friend). But since host possess higher cost, home automation will try to fulfill his wishes by a newly generated plan as follows:

Turn on Light $l2(6)$ ⇒ Turn off Light $l3(17)$ ⇒ End

Bright light $l3$ that is previously turned on for the friend is turned off, and dim light $l2$ is turned on, shown in sequence 5 of Figure 6.9.

Constraints 27 to 36 will be considered every once in a while due to periodic setting of $PlanTimer$. When these constraints are considered, it requires all light to be turned off. But it will not turn off $l2$, as the cost of fulfilling constraint 4 is higher. But $l5$ in the living room is still turned on as it won't turn off by itself, since no constraints are violated even when the host moves to the bedroom. The act of turning it off will introduce cost of 2. But with constraints 27 to 36 imposing costs on lights that are turned on, the planner will proceed by turning off $l5$ as shown in sequence 6 of Figure 6.9.

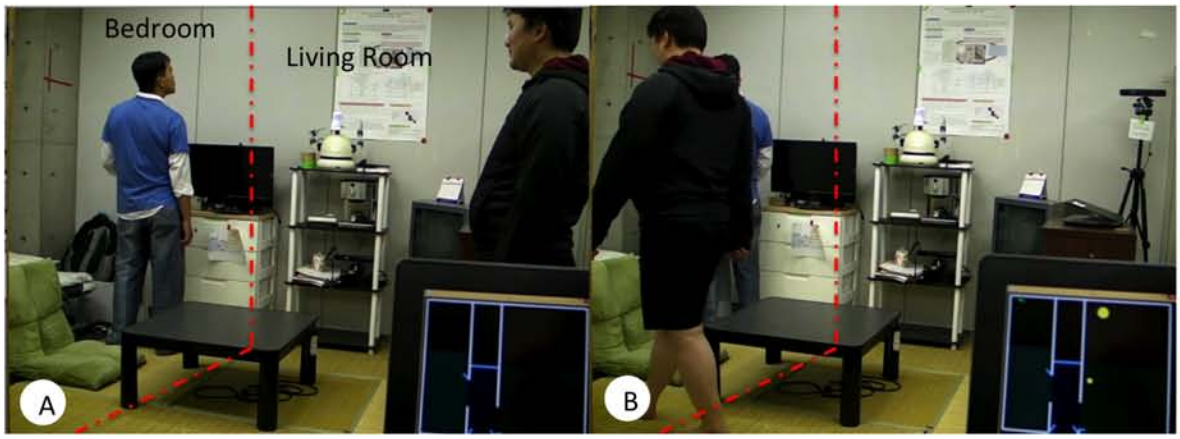


Figure 6.8: *Human movements between rooms in Case 2*

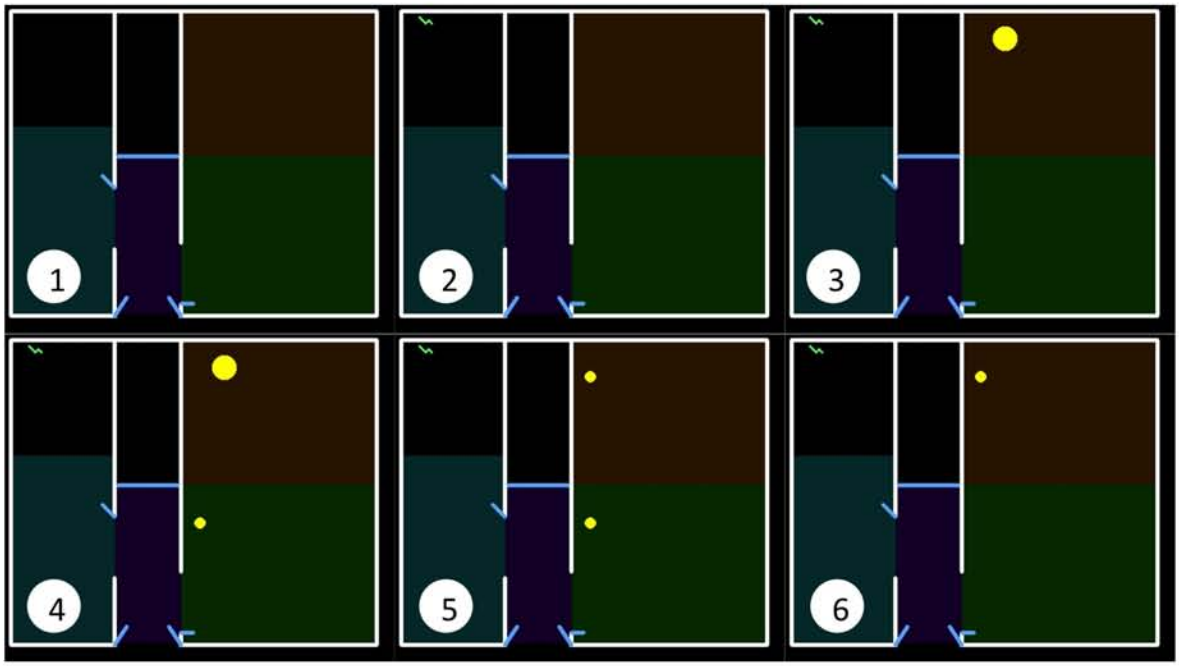


Figure 6.9: *Device visualization for Case 2: Small yellow circle is a dim light, and large yellow circle is a bright light*

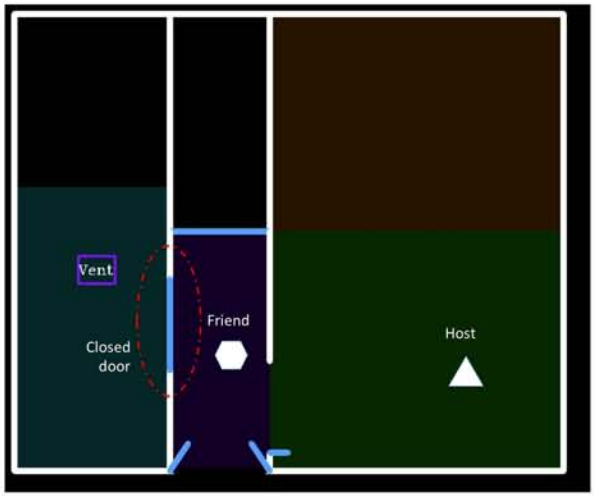


Figure 6.10: *Device visualization for Case 3: Blue bars indicate doors. Here, door $d2$ is closed*

6.2.3.3 Case 3: Plug-in Function Demonstration

Case 3 demonstrates the use of plug-in function, where the result is used to determine intelligent plans, as CSP does not concern itself with the data structure and function used, as long as truth values can be obtained.

This case is used on fire emergency in the kitchen when both the host and friend are in the kitchen. The function is $connected$, as shown in constraints 39 and 40, where it determines whether two rooms are directly connected or not. It makes decision based on the door states and layout of the house. In times of fire, where there is smoke, one tries to disconnect people from the smoke by whatever means necessary.

When fire occurs, $KitchenFireEvent$ is triggered, causing constraint 39, 40 and 41 to be considered. Constraint 41 causes the vent to be turned on. During the fire, the friend runs to the entrance area and the host runs to the living room. Given this situation, both person can be disconnected from the fire by closing door $d2$ to fulfill constraint 39 and 40. Thus, the planner will proceed with this plan. The final device state is shown in Figure 6.10. The doors will not be closed if they are in the kitchen as $connected$ cannot be true. Therefore, the best plan is to do nothing, where no additional activity costs are imposed. In this case, the smart home can be thought of as waiting for them to get out of the kitchen before it proceeds to closing the door.

One can add numerous plug-in functions as goal evaluation can be done extremely fast, which is limited by the complexity of the functions themselves.

6.2.3.4 Case 4: Demonstration of Extended Goals

We demonstrate the use of extended constraints in this case. There are two extended constraints as shown in Table 6.7, which require at least one generator to be on if light, fan or tv are to be turned on, and that normal generator needs to be turned on if air-conditioner is to be turned on. One can consider extended constraints as additional preconditions for every activity.

The case starts off with the host and friend in the living room at night. Sequence 1 of Figure 6.11 shows that dim light $l5$ and efficient generator $g2$ us turned on to maximize constraint fulfillment. The planner turns on $g2$ before switching $l5$ to on as per required by extended constraint $E1$.

The host then request for air-conditioner to be turned on. As AC require normal generator to be on as per required by $E2$, $g1$ is switched on as shown in sequence 2, after which AC is switched on in sequence 3.

Periodic constraint 38 states that at most only one generator can be switched on. This will cause $g2$ to be switched off as shown in sequence 4. Previous constraints are not violated as lights, fan and tv can also run under $g1$.

6.3 Remarks

Intelligent plan composition and optimization is implemented on the smart home through using building ontology and solving weighted CSP. Building ontology is used to provide a schema of knowledge representation, including knowledge of building layout and devices connected to the smart home. Through rule associations for service wrap-up, variables and services provided by individual devices can be related to each other. Via representing the problem as weighted CSP given weighted goals, solution obtained is the sequence of activities that tries to fulfill as much personalized constraints as possible. Case study shows the system is capable of composing and executing optimized plans for conflicting constraints. Besides, it shows potential in composing services for complex rules derived from the building ontology.

Currently, cost for activities and cost for not fulfilling constraints are manually set by user. Improvements should be made such that costs can be automatically set, or at least provide a simpler alternative for users to choose from. Another issue related to cost is the danger of being over-written. If there are conflicting goals between people in higher privileges (like the host) and people of lower privileges (like the friend or strangers), the planner will try to maximize the constraints that manifest itself as trying hard to fulfill

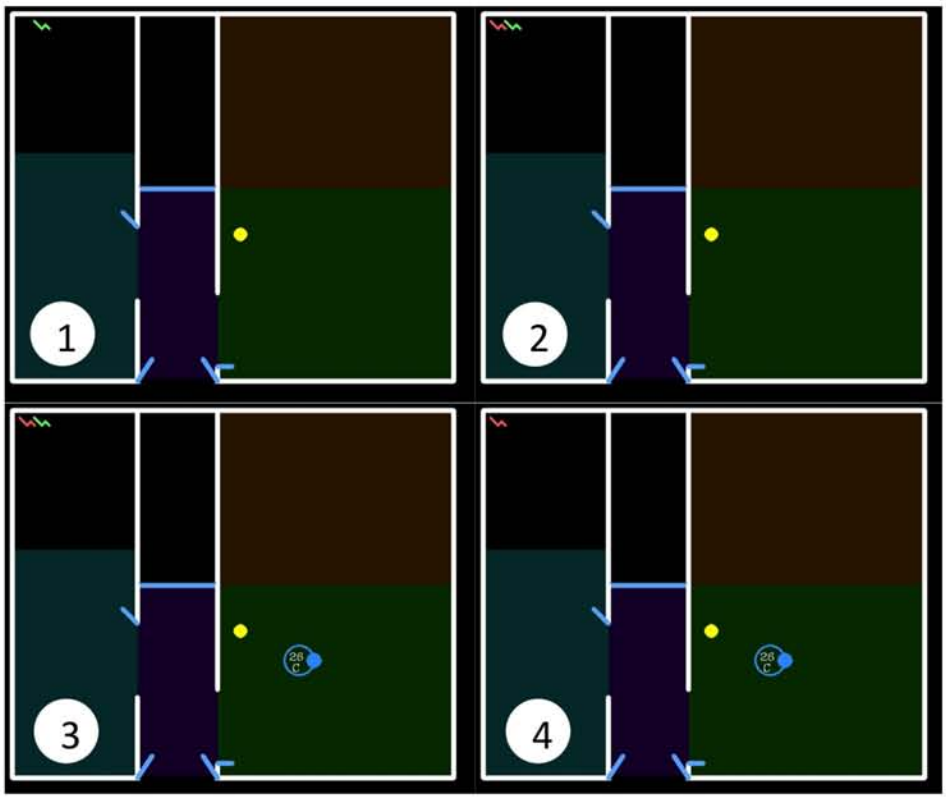


Figure 6.11: *Device visualization for Case 4: Blue circle is the air-conditioner, red thunder at the top left indicates normal generator*

constraints for the higher privileged people. Danger occurs if a group of strangers have the same constraints that are conflicting with the host's. The total cost contributed by the group of strangers will cause optimization to occur in their favor.

Chapter 7

Conclusion

7.1 Summary

The goal of this thesis is to realize a human-centric smart home with reasoning and optimization. Human-centric system tries to maximize the fulfillment of personal constraints (called QOL) using the available device connected to the smart home. This is performed through the integration of 3 modules, which are Human Identification Module (HIM), Semantic Reasoner Module (SRM) and Automated Planner Module (APM).

Personal constraints are obtained after HIM. The proposed face identification system uses joint probabilistic face method, which can run in real-time. Transfer learning is realized to handle problem of low training data. At the same time, reformulation leads to more efficient data storage and faster transfer learning. The system is also extended to real-time face learning. Tests show that the approach is very competitive compared to other state of the art methods.

The human-centric system is endowed with intelligence using methods endorsed by the World Wide Web consortium for compatibility. Knowledge for devices and the home, as well as humans, are modeled through OWL language, where new knowledge can be extracted using inference engine. The knowledge is used by SRM to generate planning operators and rules for automated planning and reasoning.

Constraint processing is then used to implement automated planning in APM, where planning is done through representing it as constraint satisfaction problem (CSP) and solving it. The advantage of CSP is the ability to handle larger domains more efficiently. Extension is made towards weighted CSP to deal with over-constrained as well as contradictory constraints. The planner is also extended to realize planning with a service robot, where case studies demonstrate a wide range of applications.

Actual implementation is done on a prototype smart home. Optimality, time consumption and scalability test is performed, which shows that the system will never produce redundant plans, and that it scales well given increasing number of devices connected to the smart home.

The proposed system is considered a high-level implementation for smart home control. In this work, it is built into the previous work of information structure of the smart home for practical implementation purpose. That said, as long as there is a low level structure that connects the devices and manage information passing, this system can be built on top of it, since it is independent of the hardware and information format.

To be used by the general public given that the service is provided by a company, the company can provide the middleware that interfaces the system with the low-level structure of individual smart home. With the portability issue settled, the system can be used as described.

In terms of service design, the company can provide the overhead needed for the user to start using the system by providing default design and consultation. This is a continuous process, where, over time, the user can start to take control of the system and able to configure it according to their needs.

Therefore, the system can be used when the company can provide the middleware to connect the system with the low-level structure of the smart home, and a continuous process of consultation and nurturing for the users are provided for them to eventually construct a smart home that is personalized to them.

7.2 Limitations

The main limitation of the proposed system is that if no objective encoding can be made on the subjective factors, then it will be outside its scope. The system is unable to handle goals that cannot be directly encoded objectively, or doesn't have a direct pragmatic way of achieving them.

The system might also make decisions that, although is rational given the constraints and goals, might be counter-intuitive for humans, because it lacks common sense (which is a shared set of axioms within a group where members of the group expect each other to have). Although one can use common sense reasoning tools like event calculus, the work is currently still very hard and can only handle simple problems.

Currently, QOLs specifications and weights should be manually set by the users. This can be cumbersome and careful planning is required given user experience is at stake. This situation is aggravated if non-intelligent users are considered, such as pets and babies,

where the guardians need to specify all the rules to ensure their pet's or baby's welfare.

Besides that, the current system tries to optimize goal fulfillment using the devices it can use. It can't lead the user to perform an action if there are no devices that can co-operate to perform such task. For example, in times of fire, the system can only open the necessary doors such that a path to safety is established, but it can't lead the user from door to door to safety.

7.3 Future Work

Current work provides room for further extension and research work. At the moment, association rules to generate planning operators should be manually defined. One can reduce such manual effort by introducing learning to obtain associations between the cause and effect for devices [107].

Closely related to previous enhancement is the creation of an ecosystem of support between the company that provides the services and the users through technology, where semi-automatic way of setting QOLs can be established. The company can provide goals and constraints based on user information, and the users can select and give preferences to assist the company in providing the best services. Given non-disclosure and privacy agreements, the company can also manage the database for the users. As the company is dealing with wide range of information and resources, they can help the users make better-informed decisions in their design. Support software with GUI can also assist in this respect.

In terms of weight setting, a separate CSP system can be set up, where the users (normally hosts) specify the constraints for the weights (for example, the host user can specify that his QOL weights should always be higher than the combined weights of the other detected users in the house). This way, the CSP can generate the appropriate weights that abide to these rules.

QOL weights will also change depending on context. For example, priority in who gets to watch TV channel of their preferences changes with the time of the day. Such dynamic weights can be achieved by encoding them directly to the QOL, but it comes at a cost of higher processing load. Future work involves personalized functions that can set the weights of their QOLs. There is no loss in generality as automated planner does not consider how the weights are set.

Besides, automated planner can also be extended to consider non-determinism. As effects induced by activities of devices might deliver a range of possibilities, being robust to it will improve its ability to deal with non-determinism more efficiently.

Apart from that, the system should anticipate that certain services might give a negative

response compared to what is expected. Given such negative return, the planner should take a different approach in fulfilling goals. Such capability can enable the system to lead users to do something (by treating the user as a device that might return negative response) if there are no available devices that can perform such task.

Finally, the vision of this work is to be able to implement human-centric system to a community or city. Devices installed by anyone who participated in the human-centric system will be shared in the sense that all these devices will try to cooperate to maximize the QOL of the community.

Given this vision, scalability to support large number of devices has to be addressed. In the current work, search space reduction is employed to only select devices that are relevant. To improve scalability, metric distance of the device should also be used to determine their relevancy.

As human's QOL is used by the human-centric system, which might contain personal information, in a community-centric scope, security should be intensified. Security does not just cover communication, but also on data encryption and the design of services.

References

[1] B. Adair, K. Miller, E. Ozanne, R. Hansen, A.J. Pearce, N. Santamaria, L. Viegas, M. Long, and C.M. Said. Smart-home technologies to assist older people to live well at home. *Journal of Aging Science*, 2013, 2013.

[2] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006.

[3] M. Alaya, Y. Banouar, T. Monteil, C. Chassot, and K. Drira. Om2m: Extensible etsi-compliant m2m service platform with self-configuration capability. *Procedia Computer Science*, 32:1079–1086, 2014.

[4] D. Allemang and J. Hendler. *Semantic web of the working ontologist.* Morgan Kaufman, 2012.

[5] N. Apostoloff and A. Zisserman. Who Are You? - Real-time Person Identification. In *British Machine Vision Conference*, pages 1–10, 2007.

[6] T.C. Au, U. Kuter, and D. Nau. Web service composition with volatile information. In Y. Gil, E. Motta, V.R. Benjamins, and M. Musen, editors, *The Semantic Web ISWC 2005*, volume 3729 of *Lecture Notes in Computer Science*, pages 52–66. 2005.

[7] S. Bandini, A. Mosca, and M. Palmonari. Common-sense spatial reasoning for information correlation in pervasive computing. *Applied Artificial Intelligence*, 21(5):405–425, 2007.

[8] B.C. Becker and E.G. Ortiz. Evaluating open-universe face identification on the web. In *Proc. of the 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 904–911, 2013.

[9] T. Berg and P.N. Belhumeur. Tom-vs-Pete classifiers and identity-preserving alignment for face verification. In *British Machine Vision Conference*, volume 2, pages 1–7, 2012.

[10] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, pages 29–37, 2001.

[11] L. Bevilacqua, A. Furno, V.S. di Carlo, and E. Zimeo. A tool for automatic generation of WS-BPEL compositions from OWL-S described services. In *Proc. of 5th International Conference on Software, Knowledge Information, Industrial Management and Applications*, pages 1–8, 2011.

[12] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM (JACM)*, 44(2):201–236, 1997.

[13] J.R. Bitner and E.M. Reingold. Backtrack programming techniques. *Commun. ACM*, 18(11):651–656, 1975.

[14] V. Blanz and T. Vetter. Face recognition based on fitting a 3D morphable model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1063–1074, 2003.

[15] D. Bonino, E. Castellina, and F. Corno. The DOG gateway: enabling ontology-based intelligent domotic environments. *IEEE Transactions on Consumer Electronics*, 54(4):1656–1664, 2008.

[16] D. Bonino and F. Corno. Dogont - ontology modeling for intelligent domotic environments. In Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan, editors, *The Semantic Web - ISWC 2008*, volume 5318 of *Lecture Notes in Computer Science*, pages 790–803. Springer Berlin Heidelberg, 2008.

[17] D. Bonino and F. Corno. Rule-based intelligence for domotic environments. *Automation in Construction*, 19(2):183 – 196, 2010.

[18] K. Bonnen, B. F. Klare, and A. K. Jain. Component-based representation in automated face recognition. *IEEE Transactions on Information Forensics and Security*, 8(1):239–253, 2013.

[19] J. Botzheim, C. Cabrita, L. T. Kóczy, and A. E. Ruano. Fuzzy rule extraction by bacterial memetic algorithms. *International Journal of Intelligent Systems*, 24(3):312–339, March 2009.

[20] J. Botzheim and N. Kubota. Growing neural gas for information extraction in gesture recognition and reproduction of robot partners. In *Micro-NanoMechatronics and Human Science (MHS), 2012 International Symposium on*, pages 149–154, 2012.

[21] J. Botzheim, D. Tang, B. Yusuf, T. Obo, N. Kubota, and T. Yamaguchi. Extraction of daily life log measured by smart phone sensors using neural computing. *Procedia Computer Science*, 22:883–892, 2013.

[22] J. Botzheim, Y. Toda, and N. Kubota. Bacterial memetic algorithm for offline path planning of mobile robots. *Memetic Computing*, 4(1):73–86, 2012.

[23] J. Botzheim, Jinseok Woo, N.N.W. Tay, N. Kubota, and T. Yamaguchi. Gestural and facial communication with smart phone based robot partner using emotional model. In *World Automation Congress*, pages 644–649, 2014.

[24] A. Bronstein, M. Bronstein, and R. Kimmel. Robust expression-invariant face recognition from partially missing data. In Ale Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, volume 3953 of *Lecture Notes in Computer Science*, pages 396–408. 2006.

[25] A.M. Bronstein, M.M. Bronstein, and R. Kimmel. Expression-invariant face recognition via spherical embedding. In *Proc. of the 2005 IEEE International Conference on Image Processing*, volume 3, 2005.

[26] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. *International Journal of Computer Vision*, 107(2):177–190, 2014.

[27] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun. A practical transfer learning algorithm for face verification. In *Proc. of the 2013 IEEE International Conference on Computer Vision*, pages 3208–3215, 2013.

[28] P. Chahuara, A. Fleury, F. Portet, and M. Vacher. Using markov logic network for on-line activity recognition from non-visual home automation sensors. In *Ambient intelligence*, pages 177–192. Springer, 2012.

[29] X. Chai, S. Shan, X. Chen, and W. Gao. Local linear regression (LLR) for pose invariant face recognition. In *Proc. of the 7th International Conference on Automatic Face and Gesture Recognition*, pages 631–636, 2006.

[30] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun. Bayesian face revisited: A joint formulation. In *Proc. of the 2012 European Conference on Computer Vision*, pages 566–579. 2012.

[31] D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *Proc. of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3025–3032, 2013.

[32] F. Corno and F. Razzak. Intelligent energy optimization for user intelligible goals in smart home environments. *Smart Grid, IEEE Transactions on*, 3(4):2128–2135, 2012.

[33] F. Corno and F. Razzak. Real-time monitoring of high-level states in smart environments. *Journal of Ambient Intelligence and Smart Environments*, 7(2):133–153, 2015.

[34] C. Cruz, L.E. Sucar, and E.F. Morales. Real-time face recognition for human-robot interaction. In *Proc. of the 8th IEEE International Conference on Automatic Face Gesture Recognition*, pages 1–6, 2008.

[35] L. De Moura and N. Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. 2008.

[36] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1):41–85, 1999.

[37] V. Degeler and A. Lazovik. Dynamic constraint reasoning in smart environments. In *Proc. of 25th IEEE International Conference on Tools with Artificial Intelligence*, pages 167–174, 2013.

[38] T. Di Noia, E. Di Sciascio, and F.M Donini. Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Journal of Artificial Intelligence Research*, 29:269–307, 2007.

[39] D. Dietrich, D. Bruckner, G. Zucker, and P. Palensky. Communication and computation in buildings: A short introduction and overview. *IEEE transactions on industrial electronics*, 57(11):3577–3584, 2010.

[40] P. Domingos and D. Lowd. Markov logic: An interface layer for artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–155, 2009.

[41] L. Durkop, H. Trsek, J. Otto, and J. Jasperneite. A field level architecture for reconfigurable real-time automation systems. In *Proc. of 10th IEEE Workshop on Factory Communication Systems*, pages 1–10, 2014.

[42] F. Englert, T. Schmitt, S. Kößler, A. Reinhardt, and R. Steinmetz. How to auto-configure your smart home?: high-resolution power measurements to the rescue. In *Proceedings*

of the fourth international conference on Future energy systems, pages 215–224. ACM, 2013.

[43] P. Espinace, T. Kollar, N. Roy, and A. Soto. Indoor scene recognition by a mobile robot through adaptive object detection. *Robotics and Autonomous Systems*, 61(9):932 – 947, 2013.

[44] M.A.T. Figueiredo, R.D. Nowak, and S.J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, 2007.

[45] J. Corman G. Linday, B. Woods. Smart homes and the internet of things. Technical report, Atlantic Council, Brent Scowcroft Center on International Security, 3 2016.

[46] J.E. Gallardo, C. Cotta, and A.J. Fernández. Solving weighted constraint satisfaction problems with memetic/exact hybrid algorithms. *Journal of Artificial Intelligence Research*, pages 533–555, 2009.

[47] C. García-Rodríguez, R. Martínez-Tomás, J.M. Cuadra-Troncoso, M. Rincón, and A. Fernández-Caballero. A simulation tool for monitoring elderly who suffer from disorientation in a smart home. *Expert Systems*, 32(6):676–687, 2015.

[48] A.S. Georghiades, P.N. Belhumeur, and D. Kriegman. From few to many: generative models for recognition under variable pose and illumination. In *Proc. of the 4th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 277–284, 2000.

[49] I. Georgievski. Planning for coordination of devices in energy-smart environments. In *Proc. of the Doctoral Consortium of the 23rd International Conference on Automated Planning and Scheduling*, 2013.

[50] I. Georgievski and M. Aiello. An overview of hierarchical task network planning. *arXiv preprint arXiv:1403.7426*, 2014.

[51] I. Georgievski, T.A. Nguyen, and M. Aiello. Combining activity recognition and ai planning for energy-saving offices. In *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing*, pages 238–245, 2013.

[52] A. Gerevini and D. Long. Preferences and soft constraints in PDDL3. In *ICAPS workshop on planning with preferences and soft constraints*, pages 46–53, 2006.

[53] M. Ghallab, D. Nau, and P. Traverso. The actors view of automated planning and acting: A position paper. *Artificial Intelligence*, 208:1–17, 2014.

[54] D. Gonzalez-Jimenez and J.L. Alba-Castro. Toward pose-invariant 2D face recognition through point distribution models and facial symmetry. *IEEE Transactions on Information Forensics and Security*, 2(3):413–429, 2007.

[55] P. Grother and P.J. Phillips. Models of large population recognition performance. In *Proc. of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 68–75, 2004.

[56] S. Han and M. Philipose. The case for onloading continuous high-datarate perception to the phone. In *Proc. of HotOS*, 2013.

[57] S.N. Han, G.M. Lee, and N. Crespi. Semantic context-aware service composition for building automation system. *IEEE Transactions on Industrial Informatics*, 10(1):752–761, 2014.

[58] O. Hatzi, D. Vrakas, N. Bassiliades, D. Anagnostopoulos, and I. Vlahavas. Semantic awareness in automated web service composition through planning. volume 6040 of *Lecture Notes in Computer Science*, pages 123–132. 2010.

[59] O. Hatzi, D. Vrakas, M. Nikolaidou, N. Bassiliades, D. Anagnostopoulos, and L. Vlahavas. An integrated approach to automated semantic web service composition through planning. *IEEE Transactions on Services Computing*, 5(3):319–332, 2012.

[60] T. Heider and T. Kirste. Smart environments and self-organizing appliance ensembles. *Mobile Computing and Ambient Intelligence*, 5181, 2005.

[61] M. Helmert. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173(5):503–535, 2009.

[62] S. Van Hoecke, R. Verborgh, D. Van Deursen, and R. Van de Walle. Samus: service-oriented architecture for multisensor surveillance in smart homes. *The Scientific World Journal*, 2014, 2014.

[63] J. Hoffmann, I. Weber, and F. Kraft. Sap speaks PDDL. In *24th National Conference of the American Association for Artificial Intelligence*, 2010.

[64] I.C. Hsu. Extensible access control markup language integrated with semantic web technologies. *Information Sciences*, 238:33–51, 2013.

[65] G.B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[66] Huawei White Paper by Carol Wan and Don Low. Capturing next generation smart home users with digital home. `http://www1.huawei.com/en/static/HW-275915.pdf`, 2013.

[67] I. Iida and T. Morita. Overview of human-centric computing. *Fujitsu Scientific and Technical Journal*, 48(2):124–128, 2012.

[68] V.R. Jakkula and D.J. Cook. Detecting anomalous sensor events in smart home data for enhancing the living experience. *Artificial intelligence and smarter living*, 11:07, 2011.

[69] E. Kaldeli, A. Lazovik, and M. Aiello. Continual planning with sensing for web service composition. In *Proc. of 25th AAAI Conference on Artificial Intelligence*, 2011.

[70] E. Kaldeli, E.U. Warriach, J. Bresser, A. Lazovik, and M. Aiello. Interoperation, composition and simulation of services at home. In P. Maglio, M. Weske, J. Yang, and M. Fantinato, editors, *Service-Oriented Computing*, volume 6470 of *Lecture Notes in Computer Science*, pages 167–181. 2010.

[71] E. Kaldeli, E.U. Warriach, A. Lazovik, and M. Aiello. Coordinating the web of services for a smart home. *ACM Trans. Web*, 7(2):10:1–10:40, 2013.

[72] J.E. Kim, G. Boulos, J. Yackovich, T. Barth, C. Beckel, and D. Mosse. Seamless integration of heterogeneous devices and access control in smart homes. In *Intelligent Environments (IE), 2012 8th International Conference on*, pages 206–213. IEEE, 2012.

[73] B.F. Klare and A.K. Jain. Heterogeneous face recognition using kernel prototype similarities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1410–1422, 2013.

[74] M. Klusch, A. Gerber, and M. Schmidt. Semantic web service composition planning with OWLS-XPLAN. In *Proc. of the 2005 AAAI Fall Symposium on Semantic Web and Agents*, 2005.

[75] M. Krotzsch, F. Simancik, and I. Horrocks. A description logic primer. *arXiv preprint arXiv:1201.4089*, 2013.

[76] N. Kubota and K. Nishida. Cooperative perceptual systems for partner robots based on sensor network. *International Journal of Computer Science and Network Security*, 6(11):19–28, 2006.

[77] N. Kubota, T. Obo, and T. Fukuda. An intelligent monitoring system based on emotional model in sensor networks. In *Proc. of the 18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 346–351, 2009.

[78] N. Kubota and Y. Toda. Information visualization in intelligent navigation for multiple mobile robots. In Fumio Kojima, Futoshi Kobayashi, and Hiroyuki Nakamoto, editors, *Simulation and Modeling Related to Computational Science and Robotics Technology*, pages 120–139. 2012.

[79] N. Kumar, A.C. Berg, P.N. Belhumeur, and S.K. Nayar. Attribute and simile classifiers for face verification. In *Proc. of the 12th IEEE International Conference on Computer Vision*, pages 365–372, 2009.

[80] J. Larrosa and R. Dechter. Boosting search with variable elimination in constraint optimization and constraint satisfaction problems. *Constraints*, 8(3):303–326, 2003.

[81] J. Larrosa, E. Morancho, and D. Niso. On the practical use of variable elimination in constraint optimization problems:'still-life'as a case study. *Journal of Artificial Intelligence Research*, pages 421–440, 2005.

[82] G. Leitner. The future home is wise, not smart. In Richard Harper, editor, *The Future Home is Wise, Not Smart*, Computer Supported Cooperative Work. 2015.

[83] P. Li, Y. Fu, U. Mohammed, J.H. Elder, and S.J.D. Prince. Probabilistic models for inference about identity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):144–157, 2012.

[84] W. Li, Y.H. Lee, W.T. Tsai, J. Xu, Y.S. Son, J.H Park, and K.D. Moon. Service-oriented smart home applications: composition, code generation, deployment, and execution. *Service Oriented Computing and Applications*, 6(1):65–79, 2012.

[85] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: Fast feature extraction and SVM training. In *Proc. of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1689–1696, 2011.

[86] G. Loseto, F. Scioscia, M. Ruta, and E. Di Sciascio. Semantic-based smart homes: a multi-agent approach. In *Proc. of the 13th Workshop on objects and agents (WOA 2012)*, volume 892, pages 49–55, 2012.

[87] A. Lotfi, C. Langensiepen, S.M. Mahmoud, and M.J. Akhlaghinia. Smart homes for the elderly dementia sufferers: identification and prediction of abnormal behaviour. *Journal of ambient intelligence and humanized computing*, 3(3):205–218, 2012.

[88] C. Lu and X. Tang. Learning the face prior for bayesian face recognition. In *Proc. of the 2014 European Conference on Computer Vision*, pages 119–134. 2014.

[89] C. Lu and X. Tang. Surpassing human-level face verification performance on LFW with gaussianface. *arXiv preprint arXiv:1404.3840*, 2014.

[90] C. De Maio, G. Fenza, M. Gallo, V. Loia, and S. Senatore. Formal and relational concept analysis for fuzzy-based automatic semantic annotation. *Applied Intelligence*, 40(1):154–177, 2013.

[91] D.M. Malioutov, M. Cetin, and A.S. Willsky. Homotopy continuation for sparse signal representation. In *Proc. of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 733–736, 2005.

[92] S. Mayer, R. Verborgh, M. Kovatsch, and F. Mattern. Smart configuration of smart environments. *IEEE Transactions on Automation Science and Engineering*, PP(99):1–9, 2016.

[93] S. McIlraith and T.C. Son. Adapting golog for composition of semantic web services. *KR*, 2:482–493, 2002.

[94] T. Mendes, R. Godina, Ed. Rodrigues, J. Matias, and J. Catalão. Smart home communication technologies and applications: Wireless protocol assessment for home area network resources. *Energies*, 8(7):7279–7311, 2015.

[95] S. Milborrow and F. Nicolls. Locating facial features with an extended active shape model. In *Computer Vision–ECCV 2008*, pages 504–513. Springer, 2008.

[96] B. Moghaddam, T. Jebara, and A. Pentland. Bayesian face recognition. *Pattern Recognition*, 33(11):1771–1782, 2000.

[97] J.H. Moreno-Scott, J.C. Ortiz-Bayliss, H. Terashima-Marın, and S.E. Conant-Pablos. Experimental matching of instances to heuristics for constraint satisfaction problems. *Computational Intelligence and Neuroscience*, 2016:1–15, 2016.

[98] A. Nakajima, T. Onishi, and T. Sawaragi. Building interoperable system enabling failure information knowledge sharing between designing/manufacturing and maintenance departments focusing on information architecture differences. *E-Journal of Advanced Maintenance*, 2:120–134, 2010.

[99] Q. Ni, A.B. García Hernando, and I.P. de la Cruz. The elderlyfis independent living in smart homes: A characterization of activities and sensing infrastructure survey to facilitate services development. *Sensors*, 15(5):11312–11362, 2015.

[100] N. Noury and T. Hadidi. Computer simulation of the activity of the elderly person living independently in a health smart home. *Computer methods and programs in biomedicine*, 108(3):1216–1228, 2012.

[101] T. Obo, N. Kubota, and B.H. Lee. Localization of human in informationally structured space based on sensor networks. In *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, pages 1–7. IEEE, 2010.

[102] E.G. Ortiz and B.C. Becker. Face recognition for web-scale datasets. *Computer Vision and Image Understanding*, 118:153–170, 2014.

[103] F.P. Pai, I.C. Hsu, and Y.C. Chung. Semantic web technology for agent interoperability: A proposed infrastructure. *Applied Intelligence*, 44(1):1–16, 2015.

[104] M.P. Papazoglou and W.J. Heuvel. Service oriented architectures: Approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415, 2007.

[105] P.J. Phillips, H. Wechsler, J. Huang, and P.J. Rauss. The FERET database and evaluation procedure for face-recognition algorithms. *Image and Vision Computing*, 16(5):295–306, 1998.

[106] J. Puttonen, A. Lobov, and J.L. Martinez Lastra. Semantics-based composition of factory automation processes encapsulated by web services. *IEEE Transactions on Industrial Informatics*, 9(4):2349–2359, 2013.

[107] K. Rasch. *Smart assistants for smart homes.* PhD thesis, 2013.

[108] A. Razzaq, K. Latif, H.F. Ahmad, A. Hur, Z. Anwar, and P.C. Bloodsworth. Semantic security against wen application attacks. *Information Sciences*, 254:19–38, 2014.

[109] S. Richter and M. Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1):127–177, 2010.

[110] J. Ruiz-del Solar, R. Verschae, and M. Correa. Recognition of faces in unconstrained environments: A comparative study. *EURASIP Journal on Advances in Signal Processing*, pages 1–19, 2009.

[111] M. Ruta, F. Scioscia, E. Di Sciascio, and G. Loseto. Semantic-based enhancement of ISO/IEC 14543-3 EIB/KNX standard for building automation. *IEEE Transactions on Industrial Informatics*, 7(4):731–739, 2011.

[112] W.J. Scheirer, M.J. Wilber, M. Eckmann, and T.E. Boult. Good recognition is non-metric. *Pattern Recognition*, 47(8):2721–2731, 2014.

[113] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *arXiv preprint arXiv:1503.03832*, 2015.

[114] N. Shadbolt. Ambient intelligence. *IEEE Intelligent Systems*, 18:2–3, 2003.

[115] N. Shadbolt, T. Berners-Lee, and W. Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[116] Z.U. Shamszaman, S.S Ara, I. Chong, and Y.K. Jeong. Web-of-objects (WoO)-based context aware emergency fire management systems for the internet of things. *Sensors*, 14(2):2944–2966, 2014.

[117] A. Sharma, M. Al Haj, J. Choi, L.S. Davis, and D.W. Jacobs. Robust pose invariant face recognition using coupled latent space discriminant analysis. *Computer Vision and Image Understanding*, 116(11):1095–1110, 2012.

[118] Q. Z. Sheng, X. Qiao, A.V. Vasilakos, C. Szabo, S. Bourne, and X. Xu. Web services composition: A decade's overview. *Information Sciences*, 280:218–238, 2014.

[119] Y. Shimizu, S. Yoshida, J. Shimazaki, and N. Kubota. An interactive support system for activating shopping streets using robot partners in informationally structured space. In *Proc. of the 2013 IEEE Workshop on Advanced Robotics and its Social Impacts*, pages 70–75, 2013.

[120] K. Simonyan, O. Parkhi, A. Vedaldi, and A. Zisserman. Fisher vector faces in the wild. In *British Machine Vision Conference*, 2013.

[121] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. HTN planning for web service composition using SHOP2. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):377–396, 2004.

[122] P. Staroch. *A weather ontology for predictive control in smart homes.* Master Thesis, 2013.

[123] Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. *arXiv preprint arXiv:1406.4773*, 2014.

[124] Y. Taigman and L. Wolf. Leveraging billions of faces to overcome performance barriers in unconstrained face recognition. *arXiv preprint arXiv:1108.1122*, 2011.

[125] Y. Taigman, Ming Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proc. of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[126] D. Tang, J. Botzheim, N. Kubota, and T. Yamaguchi. Estimation of human transport modes by fuzzy spiking neural network and evolution strategy in informationally structured space. In *Genetic and Evolutionary Fuzzy Systems (GEFS), 2013 IEEE International Workshop on*, pages 36–43, 2013.

[127] D. Tang and N. Kubot. Human localization by fuzzy spiking neural network based on informationally structured space. In *Neural Information Processing. Theory and Algorithms*, pages 25–32. Springer, 2010.

[128] D. Tang, B. Yusuf, J. Botzheim, N. Kubota, and C.S. Chan. A novel multimodal communication framework using robot partner for aging population. *Expert Systems with Applications*, 42(9):4540–4555, 2015.

[129] N.N.W. Tay, J. Botzheim, C.K. Loo, and N. Kubota. Robust face recognition via transfer learning for robot partner. In *Proc. of the 2014 IEEE Symposium on Robotic Intelligence In Informationally Structured Space*, pages 1–8, 2014.

[130] C.E. Thomaz and G.A. Giraldi. A new ranking method for principal components analysis and its application to face image analysis. *Image and Vision Computing*, 28(6):902–913, 2010.

[131] Y. Toda, Y. Kodai, E. Hiwada, and N. Kubota. Human motion tracking for cognitive rehabilitation in informationally structured space based on sensor networks. In *Proc. of 2011 IEEE International Conference on Fuzzy Systems*, pages 1459–1465, 2011.

[132] D Tsarkov and I Horrocks. FaCT++ description logic reasoner: System description. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, pages 292–297. Springer Berlin Heidelberg, 2006.

[133] S. Tulyakov and V. Govindaraju. Identification model with independent matching scores. In *Biometrics Consortium Conference*, 2005.

[134] M. Uřičář, V. Franc, and V. Hlaváč. Detector of facial landmarks learned by the structured output SVM. *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 12:547–556, 2012.

[135] E. Vezzetti and F. Marcolin. 3D human face description: landmarks measures and geometrical features. *Image and Vision Computing*, 30(10):698–712, 2012. 3D Facial Behaviour Analysis and Understanding.

[136] E. Vezzetti, F. Marcolin, and G. Fracastoro. 3D face recognition: An automatic strategy based on geometrical descriptors and landmarks. *Robotics and Autonomous Systems*, 62(12):1768–1776, 2014.

[137] P. Viola and M.J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[138] W3C by M.L. Richard Cyganiak and D. Wood. Rdf 1.1 concepts and abstract syntax. `http://www.w3.org/TR/rdf11-concepts/`, 2015.

[139] A. Wagner, J. Wright, A. Ganesh, Z. Zhou, H. Mobahi, and Y. Ma. Toward a practical face recognition system: Robust alignment and illumination by sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):372–386, 2012.

[140] M. Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.

[141] L. Wolf, T. Hassner, and Y. Taigman. Effective unconstrained face recognition by combining multiple descriptors and learned background statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):1978–1990, 2011.

[142] J. Woo and N. Kubota. Interactive categorization of living space based on simultaneous localization and mapping. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–7. IEEE, 2012.

[143] J. Woo and N. Kubota. Recognition of indoor environment by robot partner using conversation. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 17(5):753–760, 2013.

[144] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.

[145] H. Yaguchi, K. Sato, M. Kojima, K. Sogen, Y. Takaoka, M. Tsuchinaga, T. Yamamoto, and M. Inaba. Development of 3d viewer based teleoperation interface for human support robot hsr. *ROBOMECH Journal*, 1(1):1–12, 2014.

[146] F. Yang, P. Khandelwal, M. Leonetti, and P. Stone. Planning in answer set programming while learning action costs for mobile robots. In *AAAI Spring 2014 Symposium on Knowledge Representation and Reasoning in Robotics (AAAI-SSS)*, 2014.

[147] D. Yi, Z. Lei, and S.Z. Li. Towards pose robust face recognition. In *Proc. of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[148] Q. Yin, X. Tang, and J. Sun. An associate-predict model for face recognition. In *Proc. of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 497–504, 2011.

[149] S. Zhang, M. Sridharan, and J.L. Wyatt. Mixed logical inference and probabilistic planning for robots in unreliable worlds. *IEEE Transactions on Robotics*, 31(3):699–713, 2015.

[150] X. Zhu, Z. Lei, J. Yan, D. Yi, and S.Z. Li. High-fidelity pose and expression normalization for face recognition in the wild. In *Proc. of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, pages 787–796, 2015.

[151] Z. Zhu, P. Luo, X. Wang, and X. Tang. Recover canonical-view faces in the wild with deep neural networks. *arXiv preprint arXiv:1404.3543*, 2014.

[152] A. Zipperer, P. Aloise-Young, S. Suryanarayanan, R. Roche, L. Earle, D. Christensen, P. Bauleo, and D. Zimmerle. Electric energy management in the smart home: Perspectives on enabling technologies and consumer behavior. *Proceedings of the IEEE*, 101(11):2397–2408, 2013.

Acknowledgement

First and foremost, I would like to dedicate this thesis to my parents for their undying support and believe in my undertaking.

I would like to extend my special thanks to my supervisor, Prof. Naoyuki Kubota for his kind and strong supports, inspiring advices, and wonderful guidance. I could learn and experience a lot of things from him in the past 3 years.

I would also want to thank my reviewers, Prof. Yoshiki Shimomura, Prof. Yasufumi Takama and Prof. Takashi Yoshimi, for their valuable suggestions and comments to improve my thesis, as well as giving me insights regarding where my future direction should be.

Besides that, I like to thank to Prof. Janos Botzheim, whose advice and support gave me the opportunity to learn much in Japan. For Dr. Takahiro Takeda, I would like to thank for his advice and unrelenting support during my pursuit. For Prof. Loo Chu Kiong, I would like to show my appreciation for his suggestion and help as a friend.

I would also like to thank my close friends Yusuf Bakhtiar, Azhar Saputra, Woo Jinseok, Dalai Tang, Juli Széles, Mutsumi Iwasa, Ivan Ucherdzhiev and Yoshihara Yuri for their help, suggestion and advice in making my research better.

Last but not least, I like to extend my appreciation to the members of the Kubota laboratory who provide the support for my study. No one is too young or too old to learn from, given all the different experiences from all walks of life.

Appendix A: Planning and Implementation of Cases of Section 6.2.3

Case 1a

Planning:

$LeisureCheck.SS \Rightarrow AskUPActivity.SS \Rightarrow GetUserGeographicLocation \Rightarrow GetUserLocation \Rightarrow RobotSelect\_c2 \Rightarrow Robot2Query.SS \Rightarrow AskUPActivity.SE \Rightarrow EnsureUPactivityRun3.SS \Rightarrow GetWeather \Rightarrow WeatherWarn.SS \Rightarrow Robot2Query.SS \Rightarrow WeatherWarn.SE \Rightarrow EnsureUPactivityRun.SE \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 21.146608 seconds)

Implementation:

Activity $RobotSelect\_c2$ pre-condition is not satisfied

Re-planning:

$RobotSelect\_c1 \Rightarrow Robot1Query.SS \Rightarrow AskUPActivity.SE \Rightarrow EnsureUPactivityRun3.SS \Rightarrow GetWeather \Rightarrow$
$WeatherWarn.SS \Rightarrow Robot1Query.SS \Rightarrow WeatherWarn.SE \Rightarrow EnsureUPactivityRun.SE \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 4.582176 seconds)

Implementation:

Activity $EnsureUPactivityRun3.SS$ pre-condition is not satisfied

Re-planning:

$EnsureUPactivityRun1.SS \Rightarrow Light\_on.SS \Rightarrow TV\_on.SS \Rightarrow AskUPChannel.SS \Rightarrow Robot1Query.SS \Rightarrow$
$AskUPChannel.SE \Rightarrow Change\_Channel \Rightarrow EnsureUPactivityRun.SE \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 2.856030 seconds)

Implementation:

End

Case 1b

Planning:

$GetUserLocation \Rightarrow AskUPActivity.SS \Rightarrow RobotSelect\_c2 \Rightarrow Robot2Query.SS \Rightarrow AskUPActivity.SE \Rightarrow End$ (Planning Time = 0.514916 seconds)

Implementation:

Activity $RobotSelect\_c2$ pre-condition is not satisfied

Re-planning:

$RobotSelect\_c1 \Rightarrow Robot1Query.SS \Rightarrow AskUPActivity.SE \Rightarrow End$ (Planning Time = 0.214231 seconds)

Implementation:

A new goal $Leisure_{FLG} = 1$ is added from $AskUPActivity.SE$

Re-Planning:

$EnsureUPactivityRun1.SS \Rightarrow TV\_on.SS \Rightarrow AskUPChannel.SS \Rightarrow Robot1Query.SS \Rightarrow Light\_on.SS \Rightarrow$
$AskUPChannel.SE \Rightarrow Change\_Channel \Rightarrow EnsureUPactivityRun.SE \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 1.458679 seconds)

Implementation:

End

Case 1c

Planning:

$GetUserLocation \Rightarrow RobotSelect\_c2 \Rightarrow LeisureCheck.SS \Rightarrow Robot2AskActivity.SS \Rightarrow UPactivity2.act \Rightarrow$
$RobotSelect\_c2 \Rightarrow Robot2AskNewsCat.SS \Rightarrow Robot2ReadNews.SS \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 0.859543 seconds)

Implementation:

Activity $RobotSelect\_c2$ pre-condition is not satisfied

Re-planning:

$RobotSelect\_c1 \Rightarrow LeisureCheck.SS \Rightarrow Robot1AskActivity.SS \Rightarrow RobotSelect\_c1 \Rightarrow UPactivity2.act \Rightarrow$
$Robot1AskNewsCat.SS \Rightarrow Robot1ReadNews.SS \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 0.704468 seconds)

Implementation:

Activity $UPactivity2.act$ pre-condition is not satisfied

Re-planning:

$UPactivity1.act \Rightarrow Robot1AskChannel.SS \Rightarrow TV\_on.SS \Rightarrow Light\_on.SS \Rightarrow Change\_Channel \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 0.310479 seconds)

Implementation:

End

Case 1d

Planning:

$LeisureCheck.SS \Rightarrow AskUPActivity.SS \Rightarrow GetUserGeographicLocation \Rightarrow GetUserLocation \Rightarrow RobotSelect\_c2 \Rightarrow Robot2Query.SS \Rightarrow AskUPActivity.SE \Rightarrow EnsureUPactivityRun3.SS \Rightarrow GetWeather \Rightarrow WeatherWarn.SS \Rightarrow Robot2Query.SS \Rightarrow WeatherWarn.SE \Rightarrow EnsureUPactivityRun.SE \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 20.888720 seconds)

Implementation:

Activity $RobotSelect\_c2$ pre-condition is not satisfied

Re-planning:

$RobotSelect\_c1 \Rightarrow Robot1Query.SS \Rightarrow AskUPActivity.SE \Rightarrow EnsureUPactivityRun3.SS \Rightarrow GetWeather \Rightarrow$
$WeatherWarn.SS \Rightarrow Robot1Query.SS \Rightarrow WeatherWarn.SE \Rightarrow EnsureUPactivityRun.SE \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 4.223950 seconds)

Implementation:

Activity $EnsureUPactivityRun3.SS$ pre-condition is not satisfied

Re-Planning:

$EnsureUPactivityRun2.SS \Rightarrow AskNewsCat.SS \Rightarrow Robot1Query.SS \Rightarrow AskNewsCat.SE \Rightarrow ReadNews.SS \Rightarrow Robot1Query.SS \Rightarrow ReadNews.SE \Rightarrow EnsureUPactivityRun.SE \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 2.842578 seconds)

Implementation:

End

Case 1e

Planning:

$GetUserLocation \Rightarrow AskUPActivity.SS \Rightarrow RobotSelect\_c2 \Rightarrow Robot2Query.SS \Rightarrow AskUPActivity.SE \Rightarrow End$ (Planning Time = 0.510371 seconds)

Implementation:

Activity $RobotSelect\_c2$ pre-condition is not satisfied

Re-planning:

$RobotSelect\_c1 \Rightarrow Robot1Query.SS \Rightarrow AskUPActivity.SE \Rightarrow End$ (Planning Time = 0.212980 seconds)

Implementation:

A new goal $Leisure_{FLG} = 1$ is added from running activity $AskUPActivity.SE$

Re-planning:

$EnsureUPactivityRun2.SS \Rightarrow AskNewsCat.SS \Rightarrow Robot1Query.SS \Rightarrow AskNewsCat.SE \Rightarrow ReadNews.SS \Rightarrow Robot1Query.SS \Rightarrow ReadNews.SE \Rightarrow EnsureUPactivityRun.SE \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 2.231900 seconds)

Implementation:

End

Case 1f

Planning:

$GetUserLocation \Rightarrow RobotSelect\_c2 \Rightarrow LeisureCheck.SS \Rightarrow Robot2AskActivity.SS \Rightarrow UPactivity2.act \Rightarrow$
$RobotSelect\_c2 \Rightarrow Robot2AskNewsCat.SS \Rightarrow Robot2ReadNews.SS \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 0.891432 seconds)

Implementation:

Activity $RobotSelect\_c2$ pre-condition is not satisfied

Re-planning:

$RobotSelect\_c1 \Rightarrow LeisureCheck.SS \Rightarrow Robot1AskActivity.SS \Rightarrow RobotSelect\_c1 \Rightarrow UPactivity2.act \Rightarrow$
$Robot1AskNewsCat.SS \Rightarrow Robot1ReadNews.SS \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 0.699647 seconds)

Implementation:

End

Case 1g

Planning:

$LeisureCheck.SS \Rightarrow AskUPActivity.SS \Rightarrow GetUserGeographicLocation \Rightarrow GetUserLocation \Rightarrow RobotSelect\_c2 \Rightarrow Robot2Query.SS \Rightarrow AskUPActivity.SE \Rightarrow EnsureUPactivityRun3.SS \Rightarrow GetWeather \Rightarrow WeatherWarn.SS \Rightarrow Robot2Query.SS \Rightarrow WeatherWarn.SE \Rightarrow EnsureUPactivityRun.SE \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 19.802629 seconds)

Implementation:

Activity $RobotSelect\_c2$ pre-condition is not satisfied

Re-planning:

$RobotSelect\_c1 \Rightarrow Robot1Query.SS \Rightarrow AskUPActivity.SE \Rightarrow EnsureUPactivityRun3.SS \Rightarrow GetWeather \Rightarrow$
$WeatherWarn.SS \Rightarrow Robot1Query.SS \Rightarrow WeatherWarn.SE \Rightarrow EnsureUPactivityRun.SE \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 4.628042 seconds)

Implementation:

End

Case 1h

Planning:

$GetUserLocation \Rightarrow AskUPActivity.SS \Rightarrow RobotSelect\_c2 \Rightarrow Robot2Query.SS \Rightarrow AskUPActivity.SE \Rightarrow End$ (Planning Time = 0.524003 seconds)

Implementation:
Activity $RobotSelect\_c2$ pre-condition is not satisfied
Re-planning:
$RobotSelect\_c1 \Rightarrow Robot1Query.SS \Rightarrow AskUPActivity.SE \Rightarrow End$ (Planning Time = 0.225442 seconds)
Implementation:
A new goal $Leisure_{FLG} = 1$ is added from running activity $AskUPActivity.SE$
Re-planning:
$EnsureUPactivityRun3.SS \Rightarrow GetUserGeographicLocation \Rightarrow GetWeather \Rightarrow WeatherWarn.SS \Rightarrow Robot1Query.SS \Rightarrow WeatherWarn.SE \Rightarrow EnsureUPactivityRun.SE \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 1.102279 seconds)
Implementation:
End

Case 1i
Planning:
$GetUserLocation \Rightarrow RobotSelect\_c2 \Rightarrow LeisureCheck.SS \Rightarrow Robot2AskActivity.SS \Rightarrow UPactivity2.act \Rightarrow$
$RobotSelect\_c2 \Rightarrow Robot2AskNewsCat.SS \Rightarrow Robot2ReadNews.SS \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 0.925987 seconds)
Implementation:
Activity $RobotSelect\_c2$ pre-condition is not satisfied
Re-planning:
$RobotSelect\_c1 \Rightarrow LeisureCheck.SS \Rightarrow Robot1AskActivity.SS \Rightarrow RobotSelect\_c1 \Rightarrow UPactivity2.act \Rightarrow$
$Robot1AskNewsCat.SS \Rightarrow Robot1ReadNews.SS \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 0.689016 seconds)
Implementation:
Activity $UPactivity2.act$ pre-condition is not satisfied
Re-planning:
$UPactivity3.act \Rightarrow GetUserGeographicLocation \Rightarrow GetWeather \Rightarrow Robot1WeatherWarn.SS \Rightarrow LeisureCheck.SE \Rightarrow End$ (Planning Time = 0.211590 seconds)
Implementation:
End

Case 2a
Planning:
$BackHomePrepService.SS \Rightarrow GetUserLocation \Rightarrow RobotSelect\_c2 \Rightarrow GetMessages \Rightarrow$

$MsgReportService.SS \Rightarrow Robot2Query.SS \Rightarrow MsgReportService.SE \Rightarrow AskShower.SS \Rightarrow Robot2Query.SS \Rightarrow AskShower.SE \Rightarrow$
$EnsureUPactivityRun.SE \Rightarrow BackHomePrepService.SE \Rightarrow End$ (Planning Time = 5.891321 seconds)

Implementation:

Activity $RobotSelect\_c2$ pre-condition is not satisfied

Re-planning:

$RobotSelect\_c1 \Rightarrow AskShower.SS \Rightarrow Robot1Query.SS \Rightarrow GetMessages \Rightarrow AskShower.SE \Rightarrow MsgReportService.SS \Rightarrow EnsureUPactivityRun.SE \Rightarrow Robot1Query.SS \Rightarrow MsgReportService.SE \Rightarrow BackHomePrepService.SE \Rightarrow End$ (Planning Time = 2.976437 seconds)

Implementation:

Activity $EnsureUPactivityRun.SE$ pre-condition is not satisfied

Re-planning:

$EnsureUPactivityRun4.SS \Rightarrow PrepBathService.SS \Rightarrow EnsureUPactivityRun.SE \Rightarrow Robot1Query.SS \Rightarrow$
$MsgReportService.SE \Rightarrow BackHomePrepService.SE \Rightarrow End$ (Planning Time = 1.007859 seconds)

Implementation:

End

Case 2b

Planning:

$BackHomePrepService.SS \Rightarrow AskShower.SS \Rightarrow GetMessages \Rightarrow GetUserLocation \Rightarrow RobotSelect\_c1 \Rightarrow$
$Robot1Query.SS \Rightarrow AskShower.SE \Rightarrow MsgReportService.SS \Rightarrow Robot1Query.SS \Rightarrow MsgReportService.SE \Rightarrow BackHomePrepService.SE \Rightarrow End$ (Planning Time = 0.326603 seconds)

Implementation:

A new goal $EnsureUPactivityRun_{FLG} = 1$ is added from running activity $BackHomePrepService.SE$

Re-planning:

$EnsureUPactivityRun4.SS \Rightarrow PrepBathService.SS \Rightarrow EnsureUPactivityRun.SE \Rightarrow End$ (Planning Time = 0.021350 seconds)

Implementation:

End

Case 2c

Planning:

$BackHomePrepService.SS \Rightarrow GetUserLocation \Rightarrow RobotSelect\_c2 \Rightarrow GetMessages \Rightarrow$

$MsgReportService.SS \Rightarrow Robot2Query.SS \Rightarrow MsgReportService.SE \Rightarrow AskShower.SS \Rightarrow Robot2Query.SS \Rightarrow AskShower.SE \Rightarrow$
$EnsureUPactivityRun.SE \Rightarrow BackHomePrepService.SE \Rightarrow End$ (Planning Time = 5.862680 seconds)

Implementation:

Activity $RobotSelect\_c2$ pre-condition is not satisfied

Re-planning:

$RobotSelect\_c1 \Rightarrow AskShower.SS \Rightarrow Robot1Query.SS \Rightarrow GetMessages \Rightarrow AskShower.SE \Rightarrow MsgReportService.SS \Rightarrow EnsureUPactivityRun.SE \Rightarrow Robot1Query.SS \Rightarrow MsgReportService.SE \Rightarrow BackHomePrepService.SE \Rightarrow End$ (Planning Time = 2.878341 seconds)

Implementation:

Activity $EnsureUPactivityRun.SE$ pre-condition is not satisfied

Re-planning:

$Robot1Query.SS \Rightarrow MsgReportService.SE \Rightarrow GetTimeCat \Rightarrow GetMealType \Rightarrow SearchFoodType.BT.N \Rightarrow$
$SearchFoodIngd.BT.N \Rightarrow EnsureUPactivityRun5.SS \Rightarrow FoodSuggest.SS \Rightarrow FoodIngdAvailb.BT.N \Rightarrow$
$ReadFoodSuggestion.SS \Rightarrow Robot1Query.SS \Rightarrow ReadFoodSuggestion.SE \Rightarrow FoodSuggest.SE \Rightarrow$
$EnsureUPactivityRun.SE \Rightarrow BackHomePrepService.SE \Rightarrow End$ (Planning Time = 22.425772 seconds)

Implementation:

Not enough ingredients

Activity ReadFoodSuggestion.SS pre-condition is not satisfied

//A total of 7 backtrackings are performed to find a suitable breakfast suggestion where the ingredients are available based on ISS's knowledge. Average time spent is 2.535 seconds//

Re-planning:

$SearchFoodIngd.BT.E \Rightarrow SearchFoodType.BT.N \Rightarrow SearchFoodIngd.BT.N \Rightarrow FoodIngdAvailb.BT.N \Rightarrow$
$ReadFoodSuggestion.SS \Rightarrow Robot1Query.SS \Rightarrow ReadFoodSuggestion.SE \Rightarrow FoodSuggest.SE \Rightarrow$
$EnsureUPactivityRun.SE \Rightarrow BackHomePrepService.SE \Rightarrow End$ (Planning Time = 3.477976 seconds)

Implementation:

End

Case 2d

Planning:

$BackHomePrepService.SS \Rightarrow AskShower.SS \Rightarrow GetMessages \Rightarrow GetUserLocation \Rightarrow RobotSelect\_c1 \Rightarrow$

$Robot1Query.SS \Rightarrow AskShower.SE \Rightarrow MsgReportService.SS \Rightarrow Robot1Query.SS \Rightarrow MsgReportService.SE \Rightarrow BackHomePrepService.SE \Rightarrow End$ (Planning Time = 3.728780 seconds)

Implementation:

A new goal $EnsureUPactivityRun_{FLG} = 1$ is added from running activity $BackHomePrepService.SE$

Re-planning:

$GetTimeCat \Rightarrow GetMealType \Rightarrow EnsureUPactivityRun5.SS \Rightarrow FoodSuggest.SS \Rightarrow EnsureUPactivityRun.SE \Rightarrow End$ (Planning Time = 0.766165 seconds)

Implementation:

A new goal $FoodFound_{FLG} = 1$ is added from running activity $FoodSuggest.SS$

Re-planning:

$SearchFoodType.BT.N \Rightarrow SearchFoodIngd.BT.N \Rightarrow FoodIngdAvailb.BT.N \Rightarrow FoodFound.SS \Rightarrow End$ (Planning Time = 0.515529 seconds)

Implementation:

Not enough ingredients

Activity $FoodFound.SS$ pre-condition is not satisfied

//A total of 7 backtrackings are performed to find a suitable breakfast suggestion where the ingredients are available based on ISS's knowledge. Average time spent is 0.503 seconds//

Re-planning:

$SearchFoodIngd.BT.E \Rightarrow SearchFoodType.BT.N \Rightarrow SearchFoodIngd.BT.N \Rightarrow FoodIngdAvailb.BT.N \Rightarrow$
$FoodFound.SS \Rightarrow End$ (Planning Time = 0.771768 seconds)

Implementation:

A new goal $ReadFoodSuggestion_{FLG} = 1$ is added from running activity $FoodFound.SS$

Re-planning:

$ReadFoodSuggestion.SS \Rightarrow Robot1Query.SS \Rightarrow ReadFoodSuggestion.SE \Rightarrow End$ (Planning Time = 0.273511 seconds)

Implementation:

End

Case 3a

Planning:

$WindowService.SE \Rightarrow GetUserGeographicLocation \Rightarrow EnvironmentCheck.SS \Rightarrow LightService.SS \Rightarrow GetWeather \Rightarrow GetRoomTemp \Rightarrow Light\_on.SS \Rightarrow LightService.SE \Rightarrow RainyDayCheck.SE \Rightarrow TempControlService3.SS \Rightarrow$
$Fan\_on.SS \Rightarrow TempControlService3.SE \Rightarrow EnvironmentCheck.SE \Rightarrow End$ (Planning Time = 14.548797 seconds)

Implementation:

Activity $RainyDayCheck.SE$ pre-condition is not satisfied

Re-planning:
$RainyDayCheck.SS \Rightarrow Clothesline\_cl.SS \Rightarrow TempControlService3.SS \Rightarrow Fan\_on.SS \Rightarrow RainyDayCheck.SE \Rightarrow TempControlService3.SE \Rightarrow EnvironmentCheck.SE \Rightarrow End$ (Planning Time = 1.793523 seconds)
Implementation:
End

Case 3b
Planning:
$EnvironmentCheck.SS \Rightarrow LightService.SS \Rightarrow WindowService.SE \Rightarrow Light\_on.SS \Rightarrow LightService.SE \Rightarrow$
$GetRoomTemp \Rightarrow TempControlService3.SS \Rightarrow Fan\_on.SS \Rightarrow TempControlService3.SE \Rightarrow EnvironmentCheck.SE \Rightarrow End$ (Planning Time = 4.888223 seconds)
Implementation:
A new goal $RainyDayCheck_{FLG} = 1$ is added from running activity $EnvironmentCheck.SE$
Re-planning:
$GetUserGeographicLocation \Rightarrow GetWeather \Rightarrow RainyDayCheck.SE \Rightarrow End$ (Planning Time = 0.367750 seconds)
Implementation:
Activity $RainyDayCheck.SE$ pre-condition is not satisfied
Re-planning:
$RainyDayCheck.SS \Rightarrow Clothesline\_cl.SS \Rightarrow RainyDayCheck.SE \Rightarrow End$ (Planning Time = 0.355830 seconds)
Implementation:
End

Case 4
Planning:
$GetTimeCat \Rightarrow GetUstate \Rightarrow GetUactivity \Rightarrow GetSleepTime \Rightarrow SleepCheck.SE \Rightarrow End$ (Planning Time = 0.701171 seconds)
Implementation:
Activity $SleepCheck.SE$ pre-condition is not satisfied
Re-planning:
$SleepCheck.SS \Rightarrow LightService.SS \Rightarrow Light\_on.SS \Rightarrow AlarmService.SS \Rightarrow LightService.SE \Rightarrow SleepCheck.SE \Rightarrow End$ (Planning Time = 1.039245 seconds)
Implementation:
End

Appendix B: Generated Activities for Cases in Section 5.2.4

Activity	Precondition	Effect
$RobotAskact : 1$	$(RobotChoice : 10 = 1) \wedge RobotChoice_{known} : 9$	$RobotAns : 12 := res\_RobotAns : 12, RobotAns_{known} : 13 := true, RobotAnsCat : 15 := Ca\_UPactivity$
$RobotAskcha : 2$	$(RobotChoice : 10 = 1) \wedge RobotChoice_{known} : 9$	$RobotAns : 12 = res\_RobotAns : 12, RobotAns_{Known} : 13 := true, Robot2AnsCat : 15 := Ca\_UPactivity$
$RobotAnnounce : 3$	$MessagePort_{k}nown : 8 \wedge MessageCat_{k}nown : 6$	$RobotCat : 4 := MessageCat : 5$
$servicetvon : 17$	$true$	$TV : 20 := true, TV_{changed} : 21 := true$
$servicetvoff : 18$	$true$	$TV : 20 := false, TV_{changed} : 21 := true$
$servicechan-gechannel : 19$	$TV : 20 \wedge UPchannel_{known} : 24$	$TVchannel : 22 := UPchannel : 23$
$RobotAskact : 26$	$(RobotChoice : 35 = 2) \wedge RobotChoice_{k}nown : 34$	$RobotAns : 37 := res\_RobotAns : 37, RobotAns_{known} : 38 := true, RobotAnsCat : 40 := Ca\_UPactivity$
$RobotAskcha : 27$	$(RobotChoice : 35 = 2) \wedge RobotChoice_{known} : 34$	$RobotAns : 37 := res\_RobotAns : 37, RobotAns_{known} : 38 := true, RobotAnsCat : 40 := Ca\_UPchannel$
$RobotAnnounce : 28$	$MessagePort_{known} : 33 \wedge MessageCat_{known} : 31$	$RobotCat : 29 := MessageCat : 30$
$RobotSelect : 42$	$Ulocation_{known} : 44$	$RobotChoice : 45 := res\_RobotChoice : 45, RobotChoice_{known} : 46 := true$
$GetUserLocation : 48$	$true$	$Ulocation : 49 := res\_Ulocation : 49, Ulocation_{known} : 50 := true$
$CamGardenOn : 52$	$true$	$CamGarden : 53 := true, CamGarden_{changed} : 54 := true$

Activity	Precondition	Effect
$CamCarporchOn : 56$	$true$	$CamCarPorch : 57 := true, CamCarPorch_{changed} : 58 := true$
$GetResult\ UPactivity : 60$	$RobotAns_{known} : 38 \wedge (RobotAnsCat : 40 = Ca\_UPactivity)$	$UPactivity : 65 := RobotAns : 37, UPactivity_{known} : 66 := true$
$GetResult\ UPactivity : 60$	$RobotAns_{known} : 13 \wedge (RobotAnsCat : 15 = Ca\_UPactivity)$	$UPactivity : 65 := RobotAns : 12, UPactivity_{known} : 66 := true$
$GetResult\ UPchannel : 61$	$RobotAns_{known} : 38 \wedge (RobotAnsCat : 40 = Ca\_UPchannel)$	$UPchannel : 67 := RobotAns : 37, UPchannel_{known} : 68 := true$
$GetResult\ UPchannel : 61$	$RobotAns_{known} : 13 \wedge (RobotAnsCat : 15 = Ca\_UPchannel)$	$UPchannel : 67 := RobotAns : 12, UPchannel_{known} : 68 := true$
$CamGardenPort : 70$	$CamGarden : 71 \wedge CamGarden_{changed} : 72$	$CAMGardenPort : 73 := C\_CAMGardenPort, CAMGardenPort_{known} : 74 := true, CAMGardenCat : 75 := C\_CAMGardenCat, CAMGardenCat_{known} : 76 := true$
$CAMCarporchPort : 78$	$CamCarPorch : 79 \wedge CamCarPorch_{changed} : 80$	$CAMCarPorchPort : 81 := C\_CAMCarPorchPort, CAMCarPorchPort_{known} : 82 := true, CAMCarPorchCat : 83 := C\_CAMCarPorchCat, CAMCarPorchCat_{known} : 84 := true$
$GetMessages : 86$	$true$	$MessagesNum : 87 := res\_MessagesNum : 87, MessagesNum_{known} : 88 := true$
$MsgService : 90$	$(MessagesNum : 91 > 0) \wedge MessagesNum_{known} : 92$	$MessagePort_{known} : 94 := true, MessagePort : 93 := C\_MessagePort, MessageCat_{known} : 96 := true, MessageCat : 95 := C\_MessageCat$
$EnsureActivity : 98$	$UPactivity_{known} : 101 \wedge ((UPactivity : 100 = C\_bath) \rightarrow BathOperation : 107) \wedge ((UPactivity : 100 = C\_WatchTV) \rightarrow ((TVchannel : 102 = UPchannel : 104) \wedge UPchannel_{known} : 103))$	$FLG\_EnsureActivity : 105 := true$
$PrepBathService : 99$	$UPactivity_{known} : 101 \wedge ((UPactivity : 100 = C\_bath) \rightarrow BathOperation : 107)$	$FLG\_PrepBathService : 106 := true$
$BathOp : 109$	$true$	$BathOperation : 110 := true$
$GetUserGeographic Location : 112$	$true$	$UgeoLocation : 113 := res\_UgeoLocation : 113, UgeoLocation_{known} := true$
$TV\_Stream : 115$	$TV : 20 \wedge TV_{changed} : 21 \wedge CAMCarPorchPort_{known} : 82 \wedge CAMCarPorchCat_{known} : 84$	$TVCat : 122 := videoCat : 120, TVCat_{known} : 123 := true$
$TV\_Stream : 115$	$TV : 20 \wedge TV_{changed} : 21 \wedge CAMGardenPort_{known} : 74 \wedge CAMGardenCat_{known} : 76$	$TVCat : 122 := videoCat : 120, TVCat_{known} : 123 := true$

Appendix C: Device Specification for Cases in Section 5.2.4

The following shows the assigned semantic annotations for the devices relevant to the case studies. Every assignment ends with a semicolon (;). Assignment starts with property relations, followed by objects and/or classes. For example, $nt : hasservice, deviceRobot, RobotAskact$; means $deviceRobot$ is related to $RobotAskact$ by $nt : hasservice$. There are two special properties, which are, $: query$ and $nt : CSP$.

$: query$ means its following description is a query for the purpose of variable association. First, the variable and its query symbol is described. After the keyword $query$, are the queries similar to SPARQL. For example, for $: query, A, ?a, query, ?a, isa, C;$, first, variable A has a query symbol $?a$. After the $query$ keyword, $?a, isa, C$ is similar to $SELECT\ ?a\ WHERE\{?a\ isa\ C\}$. A will then be associated with the returned query result.

$nt : CSP$ means its following description is the activity description used for planning. It consists of Activity name, preconditions and effect. Precondition is enclosed in parentheses, while effect description starts after $effect$ keyword.

Device 1:Robot1

$nt : hasservice, deviceRobot, RobotAskact;$
$nt : hasservice, deviceRobot, RobotAskcha;$
$nt : hasservice, deviceRobot, RobotAnnounce;$
$nt : hasprecon, RobotAskact, RobotChoice;$
$nt : hasprecon, RobotAskact, RobotChoice_{known};$
$nt : haseffect, RobotAskact, RobotAns;$
$nt : haseffect, RobotAskact, RobotAns_{known};$
$nt : haseffect, RobotAskact, RobotAnsCat;$
$nt : hasprecon, RobotAskcha, RobotChoice;$
$nt : hasprecon, RobotAskcha, RobotChoice_{known};$
$nt : haseffect, RobotAskcha, RobotAns;$
$nt : haseffect, RobotAskcha, RobotAns_{known};$
$nt : haseffect, RobotAskcha, RobotAnsCat;$
$nt : hasprecon, RobotAnnounce, MessagePort_{known};$
$nt : hasprecon, RobotAnnounce, MessageCat_{known};$
$nt : hasprecon, RobotAnnounce, MessageCat;$
$nt : haseffect, RobotAnnounce, RobotCat;$
$: query, RobotChoice, ?a, query, ?a, a, tb : RobotChoice;$
$: query, RobotChoice_{known}, ?a, query, ?a, a, tb : RobotChoice;$
$: query, MessageCat, ?a, ?a, a, tb : MessageCat;$
$: query, MessageCat_known, ?a, ?a, a, tb : MessageCat;$
$: query, MessagePort, ?a, ?a, a, tb : MessagePort;$
$: query, MessagePort_{known}, ?a, ?a, a, tb : MessageCat;$
$rdf : type, MessageCat, c : statevariable;$
$rdf : type, MessageCat_{known}, c : statevariable;$
$rdf : type, MessageCat, tb : MessageCat;$
$rdf : type, MessageCat_{known}, tb : MessageCat;$
$rdf : type, RobotChoice_{known}, c : statevariable;$
$rdf : type, RobotChoice, c : statevariable;$
$rdf : type, RobotChoice_{known}, c : statevariable;$
$rdf : type, RobotChoice, tb : RobotChoice;$
$rdf : type, RobotChoice_{known}, tb : RobotChoice;$
$rdf : type, RobotAns, c : statevariable;$
$rdf : type, RobotAns, tb : RobotAns;$
$rdf : type, RobotAns_{known}, c : statevariable;$
$rdf : type, RobotAns_{known}, tb : RobotAns;$
$rdf : type, res\_RobotAns, c : devicevariable;$
$rdf : type, res\_RobotAns, tb : res\_RobotAns;$
$rdf : type, RobotAnsCat, c : statevariable;$
$rdf : type, RobotAnsCat, tb : RobotAnsCat;$
$nt : CSP, RobotAskact, ((RobotChoice = 1) \wedge RobotChoice_{known}), effect, RobotAns := res\_RobotAns, RobotAns_{known} := true, RobotAnsCat := Ca\_UPactivity;$
$nt : CSP, RobotAskcha, ((RobotChoice = 1) \wedge RobotChoice_{known}), effect, RobotAns := res\_RobotAns, RobotAns_{known} := true, RobotAnsCat := Ca\_UPchannel;$
$nt : CSP, RobotAnnounce, (MessagePort_{known} \wedge MessageCat_{known}), effect, RobotCat := MessageCat;$

Device 2:TV

$nt : hasservice, devicetv, servicetvon;$

$nt : hasservice, devicetv, servicetvoff;$
$nt : hasservice, devicetv, servicechangechannel;$
$nt : haseffect, servicetvon, TV;$
$nt : haseffect, servicetvon, TV_{changed};$
$nt : haseffect, servicetvoff, TV;$
$nt : haseffect, servicetvon, TV_{changed};$
$nt : hasprecon, servicechangechannel, UPchannel_{known};$
$nt : hasprecon, servicechangechannel, TV;$
$nt : haseffect, servicechangechannel, TVchannel;$
$nt : haseffect, servicechangechannel, UPchannel;$
$nt : haseffect, servicechangechannel, UPchannel_{known};$
$rdf : type, TV, c : devicevariable;$
$rdf : type, TV_{changed}, c : devicevariable;$
$rdf : type, TVchannel, c : devicevariable;$
$rdf : type, UPchannel, c : statevariable;$
$rdf : type, UPchannel_{known}, c : statevariable;$
$rdf : type, TV, tb : tv;$
$rdf : type, TV, tb : bool;$
$rdf : type, TV_{changed}, tb : tv;$
$rdf : type, TV_{changed}, tb : bool;$
$rdf : type, TVchannel, tb : tvchannel;$
$rdf : type, TVchannel, tb : int;$
$rdf : type, UPchannel, tb : UPchannel;$
$rdf : type, UPchannel, tb : int;$
$rdf : type, UPchannel_{known}, tb : UPchannel;$
$rdf : type, UPchannel, tb : int;$
$nt : islocatedat, devicetv, home : livingroom;$
$nt : CSP, servicetvon, effect, TV := true,$
$TV_{changed} := true;$
$nt : CSP, servicetvoff, effect, TV := false,$
$TV_{changed} := true;$
$nt : CSP, servicechangechannel,$
$(TV \wedge UPchannel_{known}),$
$effect, TVchannel := UPchannel;$
$: query, UPchannel_{known}, ?a, query, ?a, a,$
$tb : UPchannel;$

Device 3:Robot2

$nt : hasservice, deviceRobot, RobotAskact;$
$nt : hasservice, deviceRobot, RobotAskcha;$
$nt : hasservice, deviceRobot, RobotAnnounce;$
$nt : hasprecon, RobotAskact, RobotChoice;$
$nt : hasprecon, RobotAskact, RobotChoice_{known};$
$nt : haseffect, RobotAskact, RobotAns;$
$nt : haseffect, RobotAskact, RobotAns_{known};$
$nt : haseffect, RobotAskact, RobotAnsCat;$
$nt : hasprecon, RobotAskcha, RobotChoice;$
$nt : hasprecon, RobotAskcha, RobotChoice_{known};$
$nt : haseffect, RobotAskcha, RobotAns;$
$nt : haseffect, RobotAskcha, RobotAns_{known};$
$nt : haseffect, RobotAskcha, RobotAnsCat;$
$nt : hasprecon, RobotAnnounce, MessagePort_{known};$
$nt : hasprecon, RobotAnnounce, MessageCat_{known};$
$nt : hasprecon, RobotAnnounce, MessageCat;$
$nt : haseffect, RobotAnnounce, RobotCat;$
$: query, RobotChoice, ?a, query,$
$?a, a, tb : RobotChoice;$
$: query, RobotChoice_{known}, ?a, query, ?a, a,$
$tb : RobotChoice;$
$: query, MessageCat, ?a, ?a, a, tb : MessageCat;$
$: query, MessageCat_{known}, ?a, ?a, a,$
$tb : MessageCat;$
$: query, MessagePort, ?a, ?a, a,$
$tb : MessagePort;$
$: query, MessagePort_{known}, ?a, ?a, a,$
$tb : MessageCat;$
$rdf : type, MessageCat, c : statevariable;$
$rdf : type, MessageCat_{known}, c : statevariable;$
$rdf : type, MessageCat, tb : MessageCat;$
$rdf : type, MessageCat_{known}, tb : MessageCat;$
$rdf : type, RobotChoice_{known}, c : statevariable;$
$rdf : type, RobotChoice, c : statevariable;$
$rdf : type, RobotChoice_{known}, c : statevariable;$
$rdf : type, RobotChoice, tb : RobotChoice;$
$rdf : type, RobotChoice_{known},$
$tb : RobotChoice;$
$rdf : type, RobotAns, c : statevariable;$
$rdf : type, RobotAns, tb : RobotAns;$
$rdf : type, res\_RobotAns, c : devicevariable;$
$rdf : type, res\_RobotAns, tb : res\_RobotAns;$
$rdf : type, RobotAnsCat, c : statevariable;$
$rdf : type, RobotAnsCat, tb : RobotAnsCat;$
$nt : CSP, RobotAskact,$
$((RobotChoice = 2) \wedge RobotChoice_{known}),$
$effect, RobotAnsres\_RobotAns,$
$RobotAns_{known}true,$
$RobotAnsCatCa\_UPactivity;$
$nt : CSP, RobotAskcha,$
$((RobotChoice = 2) \wedge RobotChoice_{known}),$
$effect, RobotAnsres\_RobotAns,$
$RobotAns_{known}true,$
$RobotAnsCatCa\_UPchannel;$
$nt : CSP, RobotAnnounce,$
$(MessagePort_{known} \wedge MessageCat_{known}), effect,$
$RobotCat := MessageCat;$

Device 4:Robot Select sub-service

$nt : hasservice, deviceRobotselect, RobotSelect;$
$nt : hasprecon, RobotSelect, Ulocation;$
$nt : haseffect, RobotSelect, RobotChoice;$
$nt : haseffect, RobotSelect, RobotChoice_{known};$
$rdf : type, Ulocation, c : statevariable;$
$rdf : type, Ulocation, tb : Ulocation;$
$rdf : type, Ulocation_{known}, c : statevariable;$
$rdf : type, Ulocation_{known}, tb : Ulocation;$
$rdf : type, RobotChoice, c : statevariable;$
$rdf : type, RobotChoice_{known}, c : statevariable;$
$rdf : type, RobotChoice, tb : RobotChoice;$

$rdf : type, RobotChoice_{known},$
$tb : RobotChoice;$
$: query, Ulocation, ?a, query,$
$?a, a, tb : Ulocation;$
$: query, Ulocation_{known}, ?a, query, ?a, a,$
$tb : Ulocation;$
$nt : CSP, RobotSelect, (Ulocation_{known}),$
$effect, RobotChoice := res\_RobotChoice,$
$RobotChoice_{known} := true;$

Device 5: User Localization
$nt : hasservice, devicegetuserlocation, GetUserLocation;$
$nt : haseffect, GetUserLocation, Ulocation;$
$nt : haseffect, GetUserLocation, Ulocation_{known};$
$rdf : type, Ulocation, c : statevariable;$
$rdf : type, Ulocation, tb : Ulocation;$
$rdf : type, Ulocation_{known}, c : statevariable;$
$rdf : type, Ulocation_{known}, tb : Ulocation;$
$nt : CSP, GetUserLocation, effect,$
$Ulocation := res\_Ulocation,$
$Ulocation_{known} := true;$

Device 6: Garden Camera
$nt : hasservice, deviceCamGarden, CamGardenOn;$
$nt : haseffect, CamGardenOn, CamGarden;$
$nt : haseffect, CamGardenOn, CamGarden_{changed};$
$rdf : type, CamGarden, c : devicevariable;$
$rdf : type, CamGarden, tb : CamGarden;$
$rdf : type, CamGarden_{changed},$
$c : devicevariable;$
$rdf : type, CamGarden_{changed},$
$tb : CamGarden;$
$nt : CSP, CamGardenOn, effect,$
$CamGarden := true,$
$CamGarden_{changed} := true;$

Device 7: Car Porch Camera
$nt : hasservice, deviceCamcarporch,$
$CamCarporchOn;$
$nt : haseffect, CamCarporchOn, CamCarPorch;$
$nt : haseffect, CamCarporchOn,$
$CamCarPorch_{changed};$
$rdf : type, CamCarPorch, c : devicevariable;$
$rdf : type, CamCarPorch, tb : CamCarPorch;$
$rdf : type, CamCarPorch_{changed},$
$c : devicevariable;$
$rdf : type, CamCarPorch_{changed},$
$tb : CamCarPorch;$
$nt : CSP, CamCarporchOn, effect,$
$CamCarPorch := true,$
$CamCarPorch_{changed} := true;$

Device 8: User Response
$nt : hasservice, serviceGetResult, GetResultUPactivity;$
$nt : hasservice, serviceGetResult, GetResultUPchannel;$
$nt : hasprecon, GetResultUPactivity, RobotAns;$
$nt : hasprecon, GetResultUPactivity, RobotAns_{known};$
$nt : hasprecon, GetResultUPactivity, RobotAnsCat;$
$nt : haseffect, GetResultUPactivity, UPactivity;$
$nt : haseffect, GetResultUPactivity, UPactivity_{known};$
$nt : hasprecon, GetResultUPchannel, RobotAns;$
$nt : hasprecon, GetResultUPchannel, RobotAns_{known};$
$nt : hasprecon, GetResultUPchannel, RobotAnsCat;$
$nt : haseffect, GetResultUPchannel, UPchannel;$
$nt : haseffect, GetResultUPchannel, UPchannel_{known};$
$rdf : type, RobotAns, c : statevariable;$
$rdf : type, RobotAns, tb : RobotAns;$
$rdf : type, RobotAns_{known}, c : statevariable;$
$rdf : type, RobotAns_{known}, tb : RobotAns;$
$rdf : type, RobotAnsCat, c : statevariable;$
$rdf : type, RobotAnsCat, tb : RobotAnsCat;$
$rdf : type, UPactivity, c : statevariable;$
$rdf : type, UPactivity, tb : UPactivity;$
$rdf : type, UPactivity_{known}, c : statevariable;$
$rdf : type, UPactivity_{known}, tb : UPactivity;$
$rdf : type, UPchannel, c : statevariable;$
$rdf : type, UPchannel, tb : UPchannel;$
$rdf : type, UPchannel_{known}, c : statevariable;$
$rdf : type, UPchannel_{known}, tb : UPchannel;$
$: query, RobotAnsCat, ?a, query, ?a, a,$
$tb : RobotAnsCat;$
$: query, RobotAns, ?a, query,$
$?a, a, tb : RobotAns;$
$: query, RobotAns_{known}, ?a, query,$
$?a, a, tb : RobotAns;$
$nt : CSP, GetResultUPactivity, (RobotAns_{known} \wedge$
$(RobotAnsCat \wedge Ca\_UPactivity)), effect,$
$UPactivity := RobotAns,$
$UPactivity_{known} := true;$
$nt : CSP, GetResultUPchannel, (RobotAns_{known} \wedge$
$(RobotAnsCatz \wedge Ca\_UPchannel)), effect,$
$UPactivity := RobotAns,$
$UPactivity_{known} := true;$

Device 9: Video Port of Garden Camera
$nt : hasservice, deviceCamGardenPort,$
$CamGardenPort;$
$nt : hasprecon, CamGardenPort, CamGarden;$
$nt : hasprecon, CamGardenPort, CamGarden_{changed};$
$nt : haseffect, CamGardenPort, CAMGardenPort;$
$nt : haseffect, CamGardenPort,$
$CAMGardenPort_{known};$
$nt : haseffect, CamGardenPort, CAMGardenCat;$
$nt : haseffect, CamGardenPort, CAMGardenCat_{known};$
$rdf : type, CamGarden, c : statevariable;$
$rdf : type, CamGarden_{changed},$
$c : statevariable;$
$rdf : type, CAMGardenPort, c : devicevariable;$

$rdf : type, CAMGardenPort_{known},$
$c : devicevariable;$
$rdf : type, CAMGardenCat, c : statevariable;$
$rdf : type, CAMGardenCat_{known},$
$c : statevariable;$
$rdf : type, CamGarden, tb : CamGarden;$
$rdf : type, CamGarden_{changed},$
$tb : CamGarden;$
$rdf : type, CAMGardenPort,$
$tb : CAMGardenPort;$
$rdf : type, CAMGardenPort_{known},$
$tb : CAMGardenPort;$
$rdf : type, CAMGardenPort, tb : VideoPort;$
$rdf : type, CAMGardenPort_{known},$
$tb : VideoPort;$
$nt : isFlag, CAMGardenPort_{known}, CAMGardenPort;$
$rdf : type, CAMGardenCat,$
$tb : CAMGardenCat;$
$rdf : type, CAMGardenCat_{known},$
$tb : CAMGardenCat;$
$rdf : type, CAMGardenCat, tb : VideoCat;$
$rdf : type, CAMGardenCat_{known},$
$tb : VideoCat;$
$nt : isFlag, CAMGardenCat_{known}, CAMGardenCat;$
$: query, CamGarden, ?a, query,$
$?a, a, tb : CamGarden;$
$: query, CamGarden_{changed}, ?a, query, ?a, a,$
$tb : CamGarden;$
$nt : CSP, CamGardenPort,$
$(CamGarden \wedge CamGarden_{changed}), effect,$
$CAMGardenPort := C\_CAMGardenPort,$
$CAMGardenPort_{known} := true,$
$CAMGardenCat := C\_CAMGardenCat,$
$CAMGardenCat_{known} := true;$

Device 10: Video Port of Car Porch Camera
$nt : hasservice, deviceCAMCarporchPort,$
$CAMCarporchPort;$
$nt : hasprecon, CAMCarporchPort, CamCarPorch;$
$nt : hasprecon, CAMCarporchPort,$
$CamCarPorch_{changed};$
$nt : haseffect, CAMCarporchPort,$
$CAMCarPorchPort;$
$nt : haseffect, CAMCarporchPort,$
$CAMCarPorchPort_{known};$
$nt : haseffect, CAMCarporchPort,$
$CAMCarPorchCat;$
$nt : haseffect, CAMCarporchPort,$
$CAMCarPorchCat_{known};$
$rdf : type, CamCarPorch, c : statevariable;$
$rdf : type, CamCarPorch_{changed},$
$c : statevariable;$
$rdf : type, CAMCarPorchPort,$
$c : devicevariable;$
$rdf : type, CAMCarPorchPort_{known},$
$c : devicevariable;$
$rdf : type, CAMCarPorchCat, c : statevariable;$
$rdf : type, CAMCarPorchCat_{known},$
$c : statevariable;$
$rdf : type, CamCarPorch, tb : CamCarPorch;$
$rdf : type, CamCarPorch_{changed},$
$tb : CamCarPorch;$
$rdf : type, CAMCarPorchPort,$
$tb : CAMCarPorchPort;$
$rdf : type, CAMCarPorchPort_{known},$
$tb : CAMCarPorchPort;$
$rdf : type, CAMCarPorchPort, tb : VideoPort;$
$rdf : type, CAMCarPorchPort_{known},$
$tb : VideoPort;$
$rdf : type, CAMCarPorchCat,$
$tb : CAMCarPorchCat;$
$rdf : type, CAMCarPorchCat_{known},$
$tb : CAMCarPorchCat;$
$rdf : type, CAMCarPorchCat, tb : VideoCat;$
$rdf : type, CAMCarPorchCat_{known},$
$tb : VideoCat;$
$: query, CamCarPorch, ?a, query, ?a, a,$
$tb : CamCarPorch;$
$: query, CamCarPorch_{changed}, ?a, query, ?a, a,$
$tb : CamCarPorch;$
$nt : CSP, CAMCarporchPort, (CamCarPorch \wedge$
$CamCarPorch_{changed}), effect,$
$CAMCarPorchPort := C\_CAMCarPorchPort,$
$CAMCarPorchPort_{known} := true,$
$CAMCarPorchCat := C\_CAMCarPorchCat,$
$CAMCarPorchCat_{known} := true;$

Device 11: Message Store
$nt : hasservice, deviceMessage, GetMessages;$
$nt : haseffect, GetMessages, MessagesNum;$
$nt : haseffect, GetMessages,$
$MessagesNum_{known};$
$rdf : type, MessagesNum, c : statevariable;$
$rdf : type, MessagesNum_{known},$
$c : statevariable;$
$rdf : type, MessagesNum, tb : MessagesNum;$
$rdf : type, MessagesNum_{known},$
$tb : MessagesNum;$
$nt : CSP, GetMessages, effect,$
$MessagesNum := res\_MessagesNum,$
$MessagesNum_{known} := true;$

Device 12: Message Service
$nt : hasservice, deviceMsgService, MsgService;$
$nt : hasprecon, MsgService, MessagesNum;$
$nt : hasprecon, MsgService, MessagesNum_{known};$
$nt : haseffect, MsgService, MessagePort;$
$nt : haseffect, MsgService, MessagePort_{known};$

$nt : haseffect, MsgService, MessageCat;$
$nt : haseffect, MsgService, MessageCat_{known};$
$rdf : type, MessagesNum, c : statevariable;$
$rdf : type, MessagesNum_{known},$
$c : statevariable;$
$rdf : type, MessagePort, c : devicevariable;$
$rdf : type, MessagePort_{known},$
$c : devicevariable;$
$rdf : type, MessageCat, c : statevariable;$
$rdf : type, MessageCat_{known}, c : statevariable;$
$rdf : type, MessagesNum, tb : MessagesNum;$
$rdf : type, MessagesNum_{known},$
$tb : MessagesNum;$
$rdf : type, MessagePort, tb : MessagePort;$
$rdf : type, MessagePort_{known},$
$tb : MessagePort;$
$rdf : type, MessageCat, tb : MessageCat;$
$rdf : type, MessageCat_{known}, tb : MessageCat;$
$: query, MessagesNum, ?a, query, ?a, a,$
$tb : MessagesNum;$
$: query, MessagesNum_{known}, ?a, query, ?a, a,$
$tb : MessagesNum;$
$nt : CSP, MsgService,$
$((MessagesNum > 0) \wedge MessagesNum_{known}),$
$effect, MessagePort_{known} := true,$
$MessagePort := C\_MessagePort, MessageCat_{known} :=$
$true, MessageCat := C\_MessageCat;$

Device 13: Activity Selection Service

$nt : hasservice, serviceActivity, EnsureActivity;$
$nt : hasservice, serviceActivity, PrepBathService;$
$nt : hasprecon, EnsureActivity, UPactivity;$
$nt : hasprecon, EnsureActivity, UPactivity_{known};$
$nt : hasprecon, EnsureActivity, TVchannel;$
$nt : hasprecon, EnsureActivity, UPchannel_{known};$
$nt : hasprecon, EnsureActivity, UPchannel;$
$nt : haseffect, EnsureActivity, EnsureActivity_{FLG};$
$nt : hasprecon, PrepBathService, UPactivity;$
$nt : hasprecon, PrepBathService, UPactivity_{known};$
$nt : hasprecon, PrepBathService, BathOperation;$
$nt : haseffect, PrepBathService,$
$PrepBathService_{FLG};$
$rdf : type, UPactivity, c : statevariable;$
$rdf : type, UPactivity_{known}, c : statevariable;$
$rdf : type, TVchannel, c : statevariable;$
$rdf : type, UPchannel_{known}, c : statevariable;$
$rdf : type, UPchannel, c : statevariable;$
$rdf : type, EnsureActivity_{FLG}, c : statevariable;$
$rdf : type, PrepBathService_{FLG},$
$c : statevariable;$
$rdf : type, BathOperation, c : statevariable;$
$rdf : type, UPactivity, tb : UPactivity;$
$rdf : type, UPactivity_{known}, tb : UPactivity;$
$rdf : type, TVchannel, tb : tvchannel;$
$rdf : type, UPchannel_{known}, tb : UPchannel;$
$rdf : type, UPchannel, tb : UPchannel;$
$rdf : type, EnsureActivity_{FLG},$
$tb : EnsureActivity_{FLG};$
$rdf : type, PrepBathService_{FLG},$
$tb : PrepBathService_{FLG};$
$rdf : type, BathOperation,$
$tb : BathOperation_{FLG};$
$: query, UPactivity, ?a, query, ?a, a,$
$tb : UPactivity;$
$: query, UPactivity_{known}, ?a, query, ?a, a,$
$tb : UPactivity;$
$: query, TVchannel, ?a, query,$
$?a, a, tb : tvchannel;$
$: query, UPchannel, ?a, query,$
$?a, a, tb : UPchannel;$
$: query, UPchannel_{known}, ?a, query, ?a, a,$
$tb : UPchannel;$
$: query, BathOperation, ?a, query, ?a, a,$
$tb : BathOperation;$
$nt : CSP, EnsureActivity,$
$(UPactivity_{known} \wedge ((UPactivity = C\_bath) \rightarrow$
$BathOperation) \wedge$
$((UPactivity = C\_WatchTV) \rightarrow ((TVchannel$
$= UPchannel) \wedge UPchannel_{known}))), effect,$
$EnsureActivity_{FLG} := true;$
$nt : CSP, PrepBathService, (UPactivity_{known} \wedge$
$((UPactivity = C\_bath) \rightarrow BathOperation)),$
$effect, PrepBathService_{FLG} := true;$

Device 14: Bath Preparation Service

$nt : hasservice, operationBathOp, BathOp;$
$nt : haseffect, BathOp, BathOperation;$
$rdf : type, BathOperation, c : statevariable;$
$rdf : type, BathOperation, tb : BathOperation;$
$nt : CSP, BathOp, effect, BathOperation := true;$

Device 15: User Geographic Localizer

$nt : hasservice, deviceGPS, GetUserGeographicLocation;$
$nt : haseffect, GetUserGeographicLocation,$
$UgeoLocation;$
$nt : haseffect, GetUserGeographicLocation,$
$UgeoLocation_{known};$
$rdf : type, UgeoLocation, c : statevariable;$
$rdf : type, UgeoLocation_{known},$
$tb : statevariable;$
$rdf : type, UgeoLocation, c : UgeoLocation;$
$rdf : type, UgeoLocation_{known},$
$tb : UgeoLocation;$
$nt : CSP, GetUserGeographicLocation, effect,$
$UgeoLocation := res\_UgeoLocation,$
$UgeoLocation_{known} := true;$

Device 16: TV Stream

$nt : hasservice, DeviceTvStreaming, TV\_Stream;$
$nt : hasprecon, TV\_Stream, TV;$
$nt : hasprecon, TV\_Stream, TV_{changed};$
$nt : hasprecon, TV\_Stream, videoPort;$
$nt : hasprecon, TV\_Stream, videoPort_{known};$
$nt : hasprecon, TV\_Stream, videoCat;$
$nt : hasprecon, TV\_Stream, videoCat_{known};$
$nt : haseffect, TV\_Stream, TVCat;$
$nt : haseffect, TV\_Stream, TVCat_{known};$
$rdf : type, TV, c : statevariable;$
$rdf : type, TV_{changed}, c : statevariable;$
$rdf : type, videoPort, c : statevariable;$
$rdf : type, videoPort_{known}, c : statevariable;$
$rdf : type, videoCat, c : statevariable;$
$rdf : type, videoCat_{known}, c : statevariable;$
$rdf : type, TVCat, c : statevariable;$
$rdf : type, TVCat_{known}, c : statevariable;$
$rdf : type, TV, tb : tv;$
$rdf : type, TV_{changed}, tb : tv;$
$rdf : type, videoPort, tb : VideoPort;$
$rdf : type, videoPort_{known}, tb : VideoPort;$
$rdf : type, videoCat, tb : VideoCat;$
$rdf : type, videoCat_{known}, tb : VideoCat;$
$rdf : type, TVCat, tb : TVCat;$
$rdf : type, TVCat_{known}, tb : TVCat;$
$: query, TV, ?a, query, ?a, a, tb : tv;$
$: query, TV_{changed}, ?a, query, ?a, a, tb : tv;$
$: query, videoCat, ?a, videoPort, ?b, query,$
$?a, a, tb : VideoCat, ?b, a, tb : VideoPort,$
$?a, isVariableto, ?c, ?b, isVariableto, ?c;$
$: query, videoCat_{known}, ?a, videoPort_{known}, ?b, query,$
$?a, a, tb : VideoCat, ?b, a, tb : VideoPort,$
$?a, isVariableto, ?c, ?b, isVariableto, ?c;$
$nt : CSP, TV\_Stream, (TV \wedge TV_{changed} \wedge$
$videoPort_{known} \wedge videoCat_{known}), effect,$
$TVCat := videoCat, TVCat_{known} := true;$

Other Devices:
Other devices include light, doors, vents and windows, where each consists of on/off or open/close services.