Goal-Directed Planning and Plan Recognition for the Sustainable Control of Homes

ARCHIVES
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
AUG 01 2012
LIBRARIES

by

Wesley Graybill

S.B., Mathematics,
Massachusetts Institute of Technology (2010)
S.B., Electrical Engineering and Computer Science,
Massachusetts Institute of Technology (2011)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2012

© Massachusetts Institute of Technology 2012. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
February 29, 2012

Certified by .....
Brian C. Williams, Ph.D.
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Prof. Dennis M. Freeman
Chairman, Masters of Engineering Thesis Committee

Goal-Directed Planning and Plan Recognition for the Sustainable Control of Homes

by

Wesley Graybill

Submitted to the Department of Electrical Engineering and Computer Science
on February 29, 2012, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The goal of this thesis is to design an autonomous control system for the sustainable control of buildings. The control system focusses on satisfying three goals to encourage and facilitate a more sustainable lifestyle for the future: sustainability, comfort, and convenience. First, the system must be sustainable, meaning it controls the home to minimize the energy required to meet the living requirements of the resident. Second, the home must also place the resident's comfort as first priority, and not sacrifice comfort for energy savings. A central challenge facing the goal of comfort is uncertainty. Uncertain weather conditions can result in violations of the resident's comfort if the control system does not explicitly consider these factors. The home must probabilistically guarantee to meet resident comfort and functional requirements even under uncertain conditions. Finally, the system must be convenient and not place undue burden on the resident. To accomplish these goals, we provide three solutions: (1) goal-directed optimal planning, which supports efficiency, (2) risk-sensitive planning, which addresses comfort, and (2) intent recognition, which supports ease of use.

Goal-direction improves efficiency by specifying what energy consuming activities the users need and when, and enables peak demand to be reduced by specifying the flexibility that the user has with respect to when activities can be performed. Risk sensitive planning addresses user comfort by explicitly considering uncertain factors and planning to limit the risk of violating resident requirements. This solution uses a recently developed plan-executive called *probabilistic Sulu* (*p-Sulu*) that leverages a recent algorithm called *iterative risk allocation* (*IRA*) to robustly find an optimal control sequence for the home.

The second challenge, plan recognition, accomplishes our third goal of convenience. To facilitate widespread adoption, the control system should require minimal user interaction. Plan recognition solves this problem by predicting a resident's schedule based on observations of the resident. p-Sulu can then optimally control the home according to this schedule to minimize energy use, while ensuring the house is comfortable while the resident is home, and saving energy while the resident is away. We present the concept design of a novel solution to plan recognition over timed concur-

rent constraint automata (TCCA) that provides the initial capabilities necessary to achieve this goal.

Thesis Supervisor: Brian C. Williams, Ph.D.
Title: Professor of Aeronautics and Astronautics

Acknowledgments

I would like to express my gratitude to all of the people in my life who have helped me in the completion of this thesis.

First of all, I would like to express my gratitude towards my professor, Brian C. Williams, for all his support and encouragement. He has provided invaluable feedback and direction, without which this thesis would not have been possible.

I would like to thank all of my friends in the MERS lab for their insightful discussion and feedback throughout the process. I'm especially thankful of Masahiro Ono for all of his hard work on p-Sulu; it has been wonderful collaborating with him on the Connected Sustainable Home project. I would also like to thank David Wang for his hard work and technical support on tBurton, which served as the main motivation for a portion of this thesis. Steve Levine, Eric Timmons, Cheng Fang, and Peng Yu helped me tremendously through the review process.

I would also like to thank all of my collaborators at the Mobile Experience Lab at MIT for their guidance in the Connected Sustainable Home project. A big thanks goes to Federico Casalegno for providing me the opportunity to work on this project. Also, Bill Mitchell was enormously influential in the vision for the Connected Sustainable Home, working closely with Federico to bring the project to life. Unfortunately I didn't get the pleasure of working with him before he passed away, but I appreciate his vision and the work he put towards the project. I'd like to thank Orkan Telhan and Carl Yu for initiating the collaboration of MERS with MEL and laying the groundwork for this project. Also, I'd like to say thanks to Sotirios Kotsopoulos and Bob Hsiung for their valuable discussion about the home and all their hard work towards making the prototype a reality.

Most importantly, I'd like to thank my parents Cindy and Wesley V. Graybill for their endless support, encouragement, and love. I am eternally grateful to them for always encouraging me to pursue my goals in life, whatever they may be. Also, without the endless supplies of cookies and banana bread from my mother, I would not have had the fuel necessary to push through and complete this thesis.

This thesis was made possible thanks to funds from the Green Home Alliance, a research project between the Fondazione Bruno Kessler in Trento, Italy and the MIT Mobile Experience Lab, as well as Siemens AG under Addendum ID MIT CKI-2010-Seed_Fund-008. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the sponsoring agencies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Climate change has become one of society's most pressing issues, a challenge that does not concern one individual country, but the entire globe. Much recent research effort has been allocated towards developing new hardware, materials, and other physical systems to harness renewable energies in more and more efficient ways. These efforts in research have been matched by changes in policy encouraging sustainable living. For example, the EU has set a goal that by 2020, 20% of all electricity would come from renewables [8]. These research advances in wind turbines, solar cells, and related technologies are an important step towards guaranteeing a more sustainable future on this planet. However, much of this effort has been focussed on the supply side of the energy market, especially on hardware to improve the production of electricity. A complementary effort must be led on the *demand* side of the energy market. Currently, energy demand is treated as inelastic; suppliers must respond to demand, and adjust electricity generation to prevent blackouts. Greater emphasis must be placed on controlling energy demand and encouraging efficient use of energy. In particular, in this thesis we focus on the autonomous control of buildings for energy efficiency. The ultimate goal of our work is to develop a control system for smart buildings that minimizes the energy used to meet the daily requirements of a building's occupants.

In this thesis we make three main claims. First, we can save substantial energy by providing building services to occupants only when needed, and when service are needed, we do so in the most energy efficient manner possible and in a way that

minimizes peak load. This can be achieved by performing optimal model predictive control based on models of the building, the environment in which the building is placed, and guided by the occupant's goal activities.

Second, sources of uncertainty in the system must be explicitly taken into consideration for autonomous building control to achieve widespread adoption. Behavior of occupants, supply of energy, and the external environment are all uncertain. These uncertainties introduce a risk of failure to meet the occupant's comfort needs or operational constraints. For example, the pipes may freeze or the occupant may be sufficiently uncomfortable that he or she abandons the use of the technology. This can be achieved by performing risk-sensitive planning, which explicitly considers sources of uncertainty and plan to limit the risk of failing to meet constraints.

Third, users are unlikely to put in the time and effort to provide models of their activities and schedule to the system. Instead, we propose that they be acquired automatically.

The solution that this thesis delivers is in the form of an implemented risk-sensitive optimal planner that is guided by the user goal activities, and the concept design of a model-based recognizer for predicting user goal activites and schedules.

1.1 Motivation

In 2010, residential buildings consumed 21.52% of total energy usage in the U.S., or 21.54 quadrillion Btu of energy for that year [45]. Heating and cooling accounted for the largest portion of the residential energy consumption: 38.2% of the energy consumption in the residential sector. Thus, even small improvements in the intelligent use of energy within buildings will result in great strides towards our goal of sustainable living. The proposed approach will significantly reduce energy consumption in the residential sector. Combined with similar improvements in efficiency achieved elsewhere, this represents a substantial decrease in total energy consumption.

The vision of this thesis is to provide an autonomous control system for buildings that will provide *sustainable*, *comfortable*, and *convenient* living, saving users money

and contributing to the sustainability of our energy supply. Each of these three design goals is discussed in detail below.

1. **Sustainability**

   To achieve maximum energy savings, the autonomous control system of the building must minimize the energy use required to provide building services to the occupant.

2. **Comfort**

   Sustainable control of a building should not come at the cost of the occupant's comfort, but instead improves his or her standard of living, otherwise the technology will fall into disuse. The control system allows the occupant to specify comfortable living conditions, and controls the building so that these conditions are maintained whenever the building is occupied. For the control system to be adopted, the occupant's comfort must be first priority and should be guaranteed even under uncertain conditions, such as weather.

3. **Convenience**

   The control system of the building must require minimal effort on the part of the occupant. In order to optimally control the building, the system must have knowledge of the occupant's schedule. It must know when the occupant will be away, so it can turn off heating and cooling, and it must know when the building will be occupied, so it can guarantee the building is comfortable and ready for the occupant's arrival. Inputting all of this information by the user would become tedious and discourage use of the system. Thus, the system should automatically determine the schedule of the occupant.

1.1.1 Connected Sustainable Home

Throughout this thesis, discussion will center around the Connected Sustainable Home, a vision developed by William J. Mitchell and Federico Casalegno within

the School of Architecture at MIT [32] [33]. The vision of the Connected Sustainable Home is to combine passive and active design elements to provide sustainable, comfortable, and convenient living.

Figure 1-1: Artist's concept of the Connected Sustainable Home. A full-scale prototype will be built and completed in Rovereto, Italy in 2012.

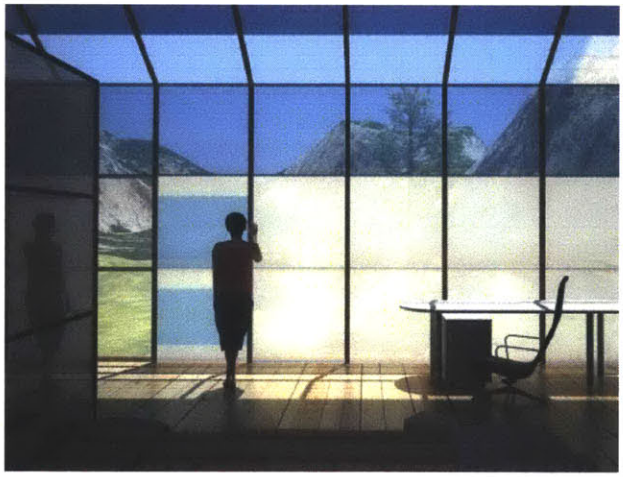


Figure 1-2: Artist's concept of the inside of the Connected Sustainable Home. The dynamic windows can be tinted to a spectrum of opacities.

The Connected Sustainable Home is designed with the goal of providing for a sustainable lifestyle that relies on as little energy from the grid as possible. Part of this goal is achieved through an array of solar panels on the roof to capture natural, renewable energy from the sun. However, commercially available solar cells only offer energy capture at an efficiency of 14-19% [44]. In the winter when heat is needed in the home, it may be more efficient to use the sun to heat the house directly through

a south-facing facade of windows. The Connected Sustainable Home has a facade of electrochromic windows, referred to as "dynamic windows", that may be tinted to a spectrum of opacities (Figures 1-1 and 1-2). Proper control of the dynamic windows can lead to significant reduction in heating and cooling (HVAC) use, especially in winter when heat from the sun can be captured during the day to reduce the need for the heater at night. Similarly, in the summer the windows can be fully tinted to block out heat to reduce the need for air conditioning.

Additionally, the Connected Sustainable Home is constructed with a large thermal mass, to effectively store heat energy in the same way a battery or capacitor might store electrical energy in a circuit. For example, in the winter day the dynamic windows can be changed to their maximum clearness to store heat in the structure, reducing the need for heating at night.

1.2 Challenges

In Section 1.1 we outlined three primary goals of the Connected Sustainable Home: sustainability, comfort, and convenience. In this section we present the approaches associated with each goal and the technical challenges of implementing each approach. The first two goals, sustainability and comfort, will be addressed by Risk-Sensitive, optimal, Goal-Directed Planning (Section 1.2.1), while the third will be addressed by Plan Recognition (Section 1.2.2).

1.2.1 Goal-Directed Planning that is Risk-Sensitive and Optimal

An autonomous controller for a residential building must not only know how to minimize energy use, but it must also be able to adapt to varying schedules of the resident. Current HVAC systems do not adapt to the wide variety of schedules a resident may keep. At the most primitive, heaters and air conditioners will provide the ability to set one setpoint and will reactively control to maintain that set-point. More sophisticated

systems allow residents to input a basic schedule and choose multiple setpoints. These static systems often don't provide the expressiveness to fully represent a resident's constraints, and they still control in a reactive manner. The Connected Sustainable Home must provide an expressive language to represent a resident's schedule and desired ranges of room temperature (i.e., state constraints in a continuous domain), and the ability to plan over this language. We will refer to this capability as **goal-directed planning with continuous effect**. The language we use to represent the resident's schedule and constraints is called the chance-constrained qualitative state plan (CCQSP)[3], and is described in more detail in Section 2.3.2.

Compared to current reactive controllers, it is predicted that an energy savings of up to 35% is possible by optimizing the operation of a building's HVAC system [14]. To capture the full benefits of the dynamic windows and large thermal mass of the Connected Sustainable Home, the controller must be capable of analyzing the dynamics of the home and calculating a control plan that minimizes use of the HVAC system. To provide for sustainable living within the home, we say the controller must achieve **optimal planning** of the HVAC system. That is to say, the home must consume as little non-renewable energy as possible.

Finally, one of the most difficult challenges facing building control is uncertainty. Weather patterns, including outside temperature and solar radiation, are inherently uncertain due to the stochastic and unpredictable nature of the atmosphere. Weather forecasts are notoriously error-prone. A control system that does not account for uncertainty and attempts to plan optimally will inevitably lead to constraint violations when operating in the stochastic environment. To provide a probabilistic guarantee that the resident's comfort constraints are not violated, the Connected Sustainable Home must explicitly consider sources of uncertainty and plan accordingly in a robust manner. We refer to this approach as **risk-sensitive planning**.

To achieve all three capabilities of risk-sensitive, optimal, goal-directed planning with continuous effect, we use a recently developed on-line risk-sensitive planner and executive called *Probabilistic Sulu* (*p-Sulu*) [36]. p-Sulu leverages a recent anytime algorithm, called Iterative Risk Allocation (IRA), to provide for robust planning in

the Connected Sustainable Home. It also expands upon previous work [37] to extend the off-line planning component of p-Sulu to include an executive component that performs receding horizon control in an on-line manner.

1.2.2 Plan Recognition

If we assume the schedule energy-consuming activities of the resident is known, then the control problem of the Connected Sustainable Home can be solved using goal-directed planning. However, in reality the schedule of the resident is the most significant source of uncertainty to the control system. For example, Figure 1-3 displays occupancy data taken from 10 different offices at Xerox PARC [12]. It is clear that not only is human behavior uncertain, but it varies widely from person to person, and even from day to day. Thus, the control system cannot simply plan over one fixed schedule, but must adapt to the changing schedule of the resident. One solution would be to require the resident to input his or her schedule into the system daily. However, this requirement is tedious and violates our goal of convenience. The challenge of the system is to predict the resident's activity schedule with little or no interaction required from the resident. It must observe the resident's behavior and predict his or her future actions and plans. The problem of inferring an agent's plans through observation is called **plan recognition**.

The plan recognizer must consider and satisfy the following requirements:

Consider temporal information The control system of the Connected Sustainable Home is operating in the physical world with real timing on events. Thus, the model behind the plan recognition approach must be capable of expressing temporal information. It should be able to accept a sequence of timed observations as input, and produce consistent plan explanations with timing on events.

Allow for missing and noisy observations Sensors are often noisy and unreliable, or some components of a plan may not even be observable. The plan recognition approach should reliably predict the plan of the resident, even in the face of missing

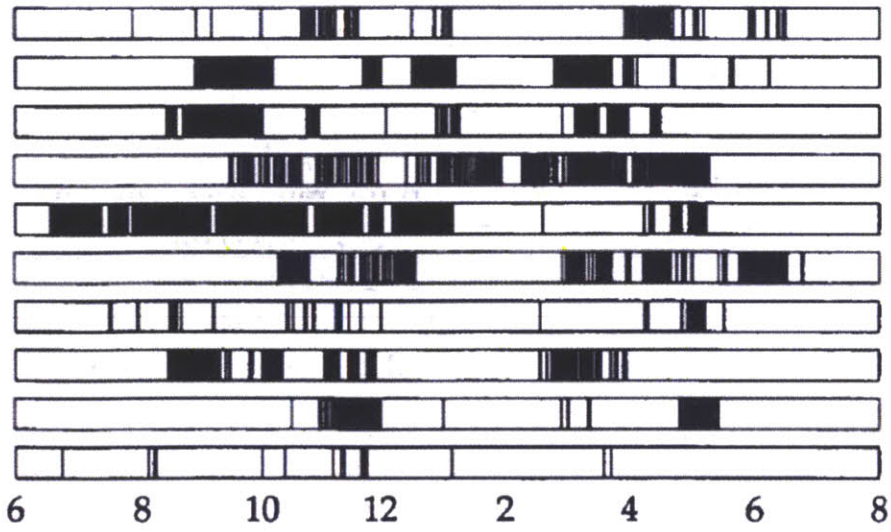


Figure 1-3: Occupancy patterns for 10 different offices taken over the period of a day at Xerox PARC. Grey shaded regions denote times that the office is occupied.

observations.

Represent human and machine activities Resident activities in a home are determined by the interaction with machines in the home. Plans are executed through a combination of human and machine activities operating concurrently. The plan recognizer should provide a representation for this interaction, and be able to reason over the states of the machines to predict resident activities.

Hybrid The behavior of machines can be described as a hybrid system with both discrete and continous states. The plan recognizer should integrate a hybrid estimation techniques within its framework.

Hidden state Human and device activities are often inferred indirectly through physical sensors, such as line current and voltage.

To solve the plan recognition problem, we note that plan recognition is closely related to planning. While in planning we are given a goal and must find a set of controls to reach the goal, in plan recognition we are given observations taken on control variables and must determine the plan pursued by the agent. Leveraging this

notion, we encode the plan recognition problem into a planning problem, and then use a planner to solve for the plan of the agent.

Specifically, we use a convenient representation for the model of our planning domain called a Timed Concurrent Constraint Automaton (TCCA). This model allows for a compact, intuitive encoding of a plan recognition problem.

1.3 Thesis Layout

This thesis presents two main innovations. First, we present an innovative application and simulation-based validation of risk-sensitive, optimal, goal-directed planning to the autonomous control of buildings. In Chapter 2, we describe the planning problem for the Connected Sustainable Home and describe the technology behind the risk-sensitive goal-directed planner, p-Sulu. Second, in Chapter 3 we present the concept design of a novel approach to plan recognition over timed concurrent constraint automata (TCCA), which encodes the plan recognition problem as a planning problem, and then uses a TCCA planner to find the solution. Then in Chapter 4 we summarize our work and discuss extensions for future work.

Chapter 2

Goal-Directed Planning and Control of Buildings

In this chapter we demonstrate a building control approach that achieves sustainability through the application of a risk-sensitive goal-directed model predictive controller called *probabilistic Sulu* (*p-Sulu*), referred to in [36] as p-Sulu on-line. Compared with traditional building control, p-Sulu is novel in that it explicitly considers the dynamics of the building to control optimally over the planning horizon. Specifically, it is innovative compared to recent model predictive control techniques for buildings in that it (1) provides an expressive and flexible representation for an occupant's requirements of the building, (2) prioritizes occupant comfort, explicitly considering uncertainty and providing a probabilistic guarantee that occupant requirements are satisfied, and (3) leverages flexibility in an occupant's schedule to produce further energy savings. We apply p-Sulu to the prototype house, Connected Sustainable Home, and compare its performance in simulation to both traditional building control and recent model predictive control techniques. We claim that p-Sulu achieves two of our three goals for autonomous building control, in particular, sustainability and comfort. We demonstrate that p-Sulu produces energy savings as high as 42% in the winter compared with traditional PID control, with significant savings across all other seasons as well. We also demonstrate that p-Sulu maintains resident constraints under uncertain conditions, a feature necessary for adoption of the technology. While

deterministic control yields comfort violations 30% of the time, the risk-sensitive p-Sulu produces almost zero constraint violations. In Chapter 3, we describe our work towards realizing the third goal of convenience.

2.1 Motivation and Goals for Autonomous Building Control

As we noted in Chapter 1, in 2010, residential buildings in the U.S. accounted for 21.52% of the total 100 quadrillion Btu of energy used in the country [45]. Heating and cooling alone accounted for the largest portion: 38.2% of total energy consumption in the residential sector. Compared to current reactive controllers, it is predicted that energy used for heating and cooling can be reduced by 35% by optimizing the operation of a building's HVAC system [14]. If this 35% savings is achieved across just residential buildings, such an optimizing control system could reduce the aggregate energy use in the U.S. by almost 3%, or almost 3 quadrillion Btu in annual energy savings. The goal of this thesis is to realize this energy savings through the application of p-Sulu to the control of homes. If the technology can achieve widespread use across all homes, we will make great strides towards creating a sustainable future. Our discussion in this section will focus on a single prototype house, the Connected Sustainable Home, but the techniques described are not specific to the prototype, and have the potential for wide-spread impact.

Recall that we outlined three goals of the Connected Sustainable Home: to provide sustainable, comfortable, and convenient living. We briefly recap what we mean by these goals in the context of building control:

1. **Sustainability**

   The control system must control the dynamic components of the home in such a way as to minimize the consumption of non-renewable energy and shift energy demand to off-peak hours. The dynamic components considered in this section include the dynamic windows and HVAC system, although one could also con-

sider any other building element that consumes energy. In this chapter we focus on minimizing total energy consumption, although the method described in this chapter could be used to minimize total cost of energy. This alternative formulation could have useful applications in shifting peak grid load at the demand level.

2. **Comfort**

   The control system should not sacrifice comfort of living for energy savings. It should allow the resident to specify comfortable indoor conditions, and reliably control the home to ensure that these conditions are maintained. Under uncertain conditions, it should provide a probabilistic guarantee that comfort will be maintained.

3. **Convenience**

   The control system should require minimal effort from the resident to accomplish the first two goals of sustainability and comfort. It should not place undue burden on the resident.

2.2 Related Work

Traditional HVAC controllers based on PID or "bang-bang" control take a simple reactive approach to heating and cooling based on a single set-point, turning the heat on when the temperature drops below a certain threshold, and turning the air conditioning on when the temperature rises above a threshold. This approach will violate the constraints of the resident, as it waits until the temperature falls out of a comfortable range to take action. Also, it wastes energy maintaining the constraints while the resident is away [14], or leads to constraint violations when the resident returns if the system is turned off while away. Many modern thermostats, such as the Hunter 44360 Set and Save programmable thermostat [13], permit users to input a schedule of temperature setpoints. These systems prevent some wasted energy. Early work in [2] showed that energy savings of 2-18% are possible using a programmable

thermostat. However, these thermostats do not utilize knowledge of the dynamics of the house to achieve further energy savings.

In [5][19][21][29], model-predictive control (MPC) approaches are applied to controlling a building, using a model of the thermal dynamics of the building in order to control indoor temperature for minimum energy use over the planning horizon. This approach does not explicitly consider the uncertainty of the system. It assumes deterministic weather and a fixed schedule of the resident as input. Yet weather is inherently uncertain, as is a resident's behavior. An optimal control must account for these uncertainties and plan to probabilistically guarantee that the constraints are satisfied.

Application of stochastic MPC (SMPC) methods to building control has recently become an active trend within the computational sustainability community. For example, [30] employs an SMPC approach using a stochastic occupancy model to attain a 4.3% energy reduction in a building HVAC. One of the most extensive efforts has been the OptiControl project at ETH Zurich [49]. In on publication from the OptiControl project, the authors employ SMPC to analyze the tradeoff between energy consumption and constraint violations [35]. The results of this work are complementary to ours, providing a necessary analysis of the cost of robust control. Although our work is similar to these efforts in that p-Sulu is also built upon SMPC, we employ a different problem formulation that is goal-directed with chance-constraints. A key innovation behind our work is that we are able to leverage flexibility in a resident's schedule to achieve further reduction in energy consumption.

2.3 Problem Statement

In this section we describe p-Sulu, and the problem it solves in terms of its inputs and outputs. Then in Section 2.4 we describe the approach that p-Sulu takes to solving this problem.

p-Sulu takes two inputs: a model of the dynamics of the home, and a plan encoding the temporal and comfort constraints of the resident. The problem is to generate an

on-line control sequence for the dynamic components of the home, given observations of past state, that uses minimal energy while maintaining the constraints of the resident.

We begin with a simple example of the requirements of the house a resident may have for a day. We describe the requirements in plain English as follows:

> "Maintain a comfortable sleeping temperature until I wake up. After waking up, maintain room temperature until I go to work. I can do some work at home, but I have to do 5 hours of work in the office sometime between 9am and 6pm. No temperature constraints while I am away, but when I get home, maintain room temperature until I go to sleep. The probability of failure of these episodes must be less than 1%. The entire time, make sure the house doesn't get so cold that the pipes freeze. Limit the probability of such a failure to 0.01%."

The plan representation p-Sulu takes as input is called a *chance-constrained qualitative state plan* (*CCQSP*), which encodes the resident's schedule as well as comfort preferences and the acceptable level of risk of a constraint violation. The schedule and comfort prefereneces are encoded as *time-evolved goals* and the level of risk is represented by *chance constraints* on the system. A chance constraint is a bound placed on the probability of a constraint violation. In our example, the 1% upperbound on the probability of discomfort and the 0.01% upperbound on the probability of pipes freezing are both chance constraints. Time-evolved goals are constraints placed on the state of the system, together with temporal information describing the period over which the constraints must be maintained. Time-evolved goals include "Maintain a comfortable sleep temperature until I wake up", and "Maintain room temperature until I go to sleep". Given a CCQSP, p-Sulu must find a control sequence that satisfies all time-evolved goals within the probability of failure specified by the chance constraints. We now describe the components of the problem more formally.

2.3.1 Stochastic Plant Model

The plan executive p-Sulu takes as input a linear plant model to describe the system being controlled. The model is of the form:

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}_t\boldsymbol{x}_t + \boldsymbol{B}_t\boldsymbol{u}_t + \boldsymbol{w}_t \tag{2.1}$$

where $\boldsymbol{x}$ is the state vector, $\boldsymbol{u}$ is the control vector, $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrices, and $\boldsymbol{w}$ is a disturbance. We assume that $\boldsymbol{w}_t$ follows a normal distribution with covariance matrix Σ_t. In Section 2.5.1, we describe how we model the Connected Sustainable Home in this form.

We also require an objective function J to be minimized,

$$\min_{\boldsymbol{u}} \mathrm{E}[J(\boldsymbol{x},\boldsymbol{u})]. \tag{2.2}$$

In our problem formulation, J is the total energy used by the home.

2.3.2 Chance Constrained Qualitative State Plan (CCQSP)

The Chance Constrained Qualitative State Plan (CCQSP) [3] is an extension of the Qualitative State Plan (QSP) described in [23] to include risk preferences, or chance constraints. It is used to represent a sequence of goals with temporal constraints, and may be depicted as an acyclic directed graph. A CCQSP for the example schedule from Section 2.3 is depicted in Figure 2-1.

An **event**, illustrated as a circle in Figure 2-1, represents a point of time, to which a discrete execution time is assigned. An **episode**, depicted as a rectangle, specifies the desired state of the system under control over a time interval. Each episode is a three-tuple $a = \langle e_a^S, e_a^E, R_a, c\rangle$, where e_a^S and e_a^E are the start and end events of the episode, respectively. Each episode a has a feasible state region R_a. For example, R_a for the "Maintain sleep temperature" is a closed interval on the indoor temperature, [18°C 22°C]. In our encoding below, R_a is represented by a set of linear state constraints. Each episode also has a chance constraint class c that will

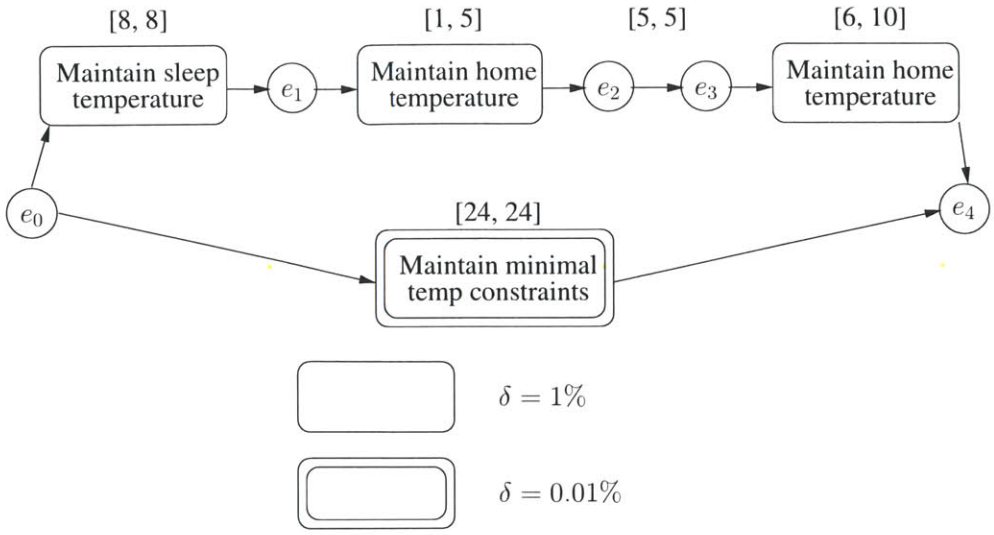


Figure 2-1: An example of a chance constrained qualitative state plan (CCQSP) for a resident's schedule in a planning problem for the Connected Sustainable Home.

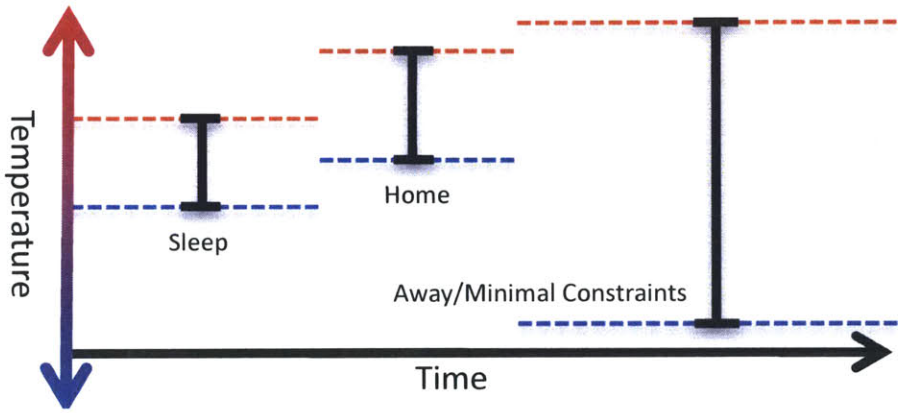


Figure 2-2: The state space of the CCQSP in Figure 2-1.

be described in more detail later. A **simple temporal constraint**, denoted by two numbers in a bracket as $[lb_e^{e'}\ ub_e^{e'}]$, specifies an upper bound $ub_e^{e'}$ and a lower bound $lb_e^{e'}$ on the temporal distance between two events $e, e' \in \mathcal{E}$. Note that $ub_e^{e'} = -lb_{e'}^{e}$. Finally, a **chance constraint** specifies an upper bound on the risk of failing to satisfy a set of episodes during execution. It is a two-tuple, $c = \langle \Delta_c, \Psi_c \rangle$, where Δ_c is the risk bound and Ψ_c is the set of episodes associated with a chance constraint c. For example, in the CCQSP shown in Figure 2-1, the first chance constraint is associated with three episodes: a "Maintain sleep temperature" episode and two "Maintain home

temperature" episodes.

We define the notion formally as follows:

Definition 1. *A **Chance Constrained Qualitative State Plan (CCQSP)** is a tuple $\langle \mathcal{E}, \mathcal{A}, \mathcal{T}, \mathcal{C} \rangle$ where:*

1. *$\mathcal{E}$ is a set of events. Events are plan elements that can be assigned to a specific point in time.*

2. *$\mathcal{A}$ is a set of episodes where each episode $a \in \mathcal{A}$ is itself a tuple $\langle e_S, e_E, h, g, c \rangle$. An episode is depicted graphically as a directed edge between events with constraints encoding the feasible state space. Episodes have a start event e_S, an end event e_E, and a set of linear state constraints represented by h and g as follows:*

$$\wedge_{k \in \mathcal{K}} \vee_{j \in \mathcal{J}} \boldsymbol{h}_{k,j}^T \boldsymbol{x}_t - \boldsymbol{g}_{k,j} \leq 0,$$

   where $\mathcal{K}$ is an index set over the conjunction of the disjunctive constraints indexed by $\mathcal{J}$. c is the constraint class of the episode.

3. *$\mathcal{T}$ is a set of temporal constraints represented as a simple temporal network (STN) [9], which specifies lower and upper bounds on the duration between pairs of events.*

4. *$\mathcal{C}$ is a set of chance constraint classes together with upper bounds Δ_c on the risk of failure of executing a set of episodes. That is, for a specific chance constraint class c, we require that the total probability of failure, over all episodes in that class, is less than Δ_c.*

2.3.3 Outputs

Optimal executable control sequence One of the two outputs of p-Sulu is an executable control sequence $\boldsymbol{u}_0 \cdots \boldsymbol{u}_T$ that minimizes a given cost function and satisfies all temporal and dynamical constraints specified by the input CCQSP according to the probability specified by the chance constraints. In the case of the Connected

Sustainable Home, the outputs are the opaqueness of the dynamic window, as well as the heat output of HVAC. Minimization only occurs over finite horizons, as the executive for building control must operate over an infinite time-horizon. We thus only guarantee optimality over the finite planning horizon.

Optimal schedule The other output of p-Sulu is the optimal schedule. Let $s_e \in \mathbb{R}_+$ be an execution time of the event e, where $\mathbb{R}_+$ is a set of non-negative real numbers. A schedule s is a set of execution times s_e of all events $e \in \mathcal{E}$. If any temporal constraints are flexible, then this corresponds to a choice of when to execute each episode. If all temporal constraints are fixed, then the schedule is already fixed.

2.4 Robust Plan Executive: p-Sulu

In this section, we describe the prior work on the technology and algorithms behind p-Sulu. p-Sulu consists of two nested loops. The outer loop performs continuous receding horizon planning, through repeated calls to a stochastic CCQSP planner. The CCQSP planning is implemented by the inner loop through the Iterative Risk Allocation-CCQSP (IRA-CCQSP) algorithm, which performs a series of fixed-risk CCQSP planning steps, finding an improved risk allocation at each step and replanning using this allocation.

A key concept used by p-Sulu is that of *risk allocation.* For each planning horizon, the CCQSP specifies a fixed risk tolerance Δ_c for each chance constraint class. Recall that by *risk*, we mean the probability of violating the constraints of any episode in the chance constraint class. The key idea behind IRA is that we can distribute this global risk of failure across individual risk bounds δ_t applied to each time step t. As long as the constraint $\sum \delta_t \leq \Delta_c$ holds, if we plan such that each individual risk bound is satisfied, Boole's inequality shows that the total risk bound Δ_c is satisfied.

In this section, we describe p-Sulu from the outside in, first describing the outer loop, and then moving into further detail with the inner loop. We begin in Section 2.4.1 by describing the p-Sulu executive that implements the receding horizon MPC

necessary for on-line control of the Connected Sustainable Home. In Section 2.4.2, we then describe how the planning problem for each planning horizon is encoded into a mixed integer linear program (MILP), given a fixed risk allocation. Then in Section 2.4.3, we describe the Iterative Risk Allocation algorithm (IRA) that solves the chance-constrained problem and outputs a risk allocation. For a more in-depth explanation, see [37].

2.4.1 The p-Sulu Executive as Model Predictive Control

The control system for the Connected Sustainable Home must be operated continuously over an infinite time horizon. It is impossible to plan and minimize non-renewable energy use over this entire horizon, so we must instead plan over a finite time horizon and repeat the planning step as time proceeds.

p-Sulu overcomes this challenge by employing a receding horizon control approach [31]. At each planning cycle, a planning problem is solved with a finite duration, which is called a *horizon*. In the next planning cycle, the planning problem is solved again over a planning horizon with the same duration starting from the current time (hence, the horizon is "receding"), by considering the latest observation of uncertain parameters. This replanning process is repeated with a fixed time interval.

In this subsection we describe the outer loop of the p-Sulu executive that performs receding horizon model predictive control. The loop repeatedly calls the inner loop, IRA-CCQSP, to determine the optimal risk allocation and control plan for a given planning horizon. IRA-CCQSP is described in Section 2.4.3.

Outer Loop: Receding Horizon Execution

The outer loop performs continuous receding horizon planning, through repeated calls to IRA-CCQSP. It exploits the anytime behavior of IRA-CCQSP [36] by running as many IRA-CCQSP iterations as possible within each planning horizon, in order to obtain the best available solution for a given replanning interval. The outer loop is outlined in Algorithm 1. We let t be the index of time steps, N_E be the number of

time steps in an execution horizon, $\mathcal{E}_u$ be a set of unexecuted events, and $\boldsymbol{u}$ be a control sequence.

Algorithm 1 p-Sulu

function pSulu(*ccqsp*)

1: $\mathcal{E}_u \leftarrow \mathcal{E}$
2: **while** $\mathcal{E}_u \neq \emptyset$ **do**
3: Wait until $t = (n-1)N_E + 1$
4: $\boldsymbol{u} \leftarrow$ IRA-CCQSP($ccqsp, \mathcal{E}_u$)
5: Execute the first N_E steps in $\boldsymbol{u}$
6: $\forall e \in \mathcal{E}_u$, remove e from $\mathcal{E}_u$ if e has been executed
7: $n \leftarrow n + 1$
8: **end while**

At the start of continuous planning and execution, $\mathcal{E}_u$ is initialized with the full set of events $\mathcal{E}$ in the given CCQSP *ccqsp* (Line 1). The executive replans every N_E time steps. It waits until the next scheduled replanning time in Line 3. A CCQSP planning problem is solved over N_P time steps at every planning cycle by IRA-CCQSP, in order to generate a sequence of optimal control inputs $\boldsymbol{u}$ (Line 4), of which the first N_E control inputs are executed in this horizon (Line 5). Finally, the events that are executed within this planning horizon are removed from $\mathcal{E}_u$ (Line 6). This iteration is repeated until all the events are executed and hence the set $\mathcal{E}_u$ becomes empty (Line 2). In the application for Connected Sustainable Home, new events are continuously added to the CCQSP so that the algorithm operates without termination.

2.4.2 The p-Sulu Planner

In this section we review how the planning problem for a CCQSP is encoded into a deterministic constrained optimization problem. We assume a fixed risk allocation and plan such that the risk bounds are maintained [37]. In Section 2.4.3, we describe how an optimal risk allocation is found.

Encoding the CCQSP

We take an approach to encode the CCQSP planning problem into a chance-constrained optimization, which is eventually reduced into a deterministic constrained optimization problem by the IRA-CCQSP algorithm. We begin by encoding the temporal constraints, and then encode the episodes, as well as chance constraints.

The simple temporal constraints in a CCQSP are encoded as follows:

$$\bigwedge_{e\in\mathcal{E}_u}\bigwedge_{e'\in\mathcal{E},e'\neq e}\left(lb_e^{e'}\leq s_{e'}-s_e\leq ub_e^{e'}\right) \tag{2.3}$$

$$\bigwedge_{e\in\mathcal{E}\backslash\mathcal{E}_u}(s_e=\bar{s}_e), \tag{2.4}$$

where $\mathcal{E}_u$ is a set of unexecuted events. (2.3) imposes upper and lower bounds on episodes that involve unexecuted events. The schedule of the events that have already been executed, $e\in\mathcal{E}\backslash\mathcal{E}_u$, are fixed to their execution time $\bar{s}_e$, as in (2.4).

Recall that we employ a receding horizon approach. We denote by $\mathbb{T}_n := \{(n-1)N_E+1\cdots(n-1)N_E+N_P\}$ the set of discrete time steps included in the nth planning horizon, where N_E and N_P are the number of time steps in an execution horizon and a planning horizon, respectively. Note that $N_E\leq N_P$. An episode a is satisfied if the state $\boldsymbol{x}_t$ is within the feasible region R_a whenever a is being executed (i.e., $s_{e_a^S}\leq t\leq s_{e_a^E}$). A chance constraint c is satisfied if all episodes in Ψ_c are satisfied with probability $1-\Delta_c$. Hence, for all chance constraints in $\mathcal{C}$, the following must be satisfied:

$$\bigwedge_{c\in\mathcal{C}}\left[\Pr\left[\bigwedge_{a\in\Psi_c}\bigwedge_{t\in\mathbb{T}}\left((s_{e_a^S}\leq t\leq s_{e_a^E})\Longrightarrow\boldsymbol{x}_t\in R_a\right)\right]\geq 1-\Delta_c\right]. \tag{2.5}$$

Here $\mathbb{T}$ denotes the set of all discrete time steps. Note that since the inner conjunction is taken over $\mathbb{T}$, rather than just $\mathbb{T}_n$, the constraint (2.5) allows the plan executive to postpone the execution of a by setting $s_{e_a^S}$ larger than all the time steps in $\mathbb{T}_n$. Postponed episodes are executed in later planning horizons. However, whenever an episode can be executed at the current horizon, the executive should not postpone its

execution since there is no guarantee that the episode is still feasible with regard to state and chance constraints at future time steps. Therefore, we penalize deferments of episode execution. Let p_a be the penalty, and M be a large positive constant. We require the following:

$$s_{e_a^S} > (n-1)N_E + N_P \Longrightarrow p_a = M. \tag{2.6}$$

Finally, we set the objective function. Let $(\boldsymbol{u}, s)$ be a cost function that is assumed to be a piecewise linear function of a control sequence $\boldsymbol{u}$ and a schedule s. The penalty p_a of all episodes must be added to the cost function. Hence, we minimize the following objective function:

$$\min_{\boldsymbol{u},s} J(\boldsymbol{u}, s) + \sum_{a \in \mathcal{A}} p_a. \tag{2.7}$$

For each planning horizon $\mathbb{T}_n$, p-Sulu solves a chance constrained program with the objective function (2.7) and constraints (2.1), (2.3), (2.4), (2.5), and (2.6).

Reformulation of Chance Constraints into a Risk Allocation

Next, we present a reformulation of chance constraints (2.5) to deterministic constraints. We first decompose the chance constraint (2.5) using the risk allocation approach [38]. It follows from Boole's inequality that the following is a sufficient condition for (2.5):

$$\bigwedge_{c \in \mathcal{C}} \bigwedge_{a \in \Psi_c} \bigwedge_{t \in \mathbb{T}_n} \Pr\left[(s_{e_a^S} \leq t \leq s_{e_a^E}) \Longrightarrow \boldsymbol{x}_t \in R_a\right] \geq 1 - \delta_{a,t} \tag{2.8}$$

$$\bigwedge_{c \in \mathcal{C}} \sum_{a \in \Psi_c, t \in \mathbb{T}_n} \delta_{a,t} \leq \Delta_c, \tag{2.9}$$

where $\delta_{a,t}$ is the risk allocated to episode a at time step t. The above two constraints can be transformed into a deterministic constraint as follows:

$$\bigwedge_{c\in\mathcal{C}}\bigwedge_{a\in\Psi_c}\bigwedge_{t\in\mathbb{T}_n}(s_{e_a^S}\leq t\leq s_{e_a^E})\Longrightarrow\bar{\boldsymbol{x}}_t\in R_a(\delta_{a,t,U},\delta_{a,t,L}) \tag{2.10}$$

$$\bigwedge_{c\in\mathcal{C}}\sum_{a\in\Psi_c,t\in\mathbb{T}_n}\delta_{a,t}^U+\delta_{a,t}^L\leq\Delta_c, \tag{2.11}$$

where $\delta_{a,t,U}$ and $\delta_{a,t,L}$ are the risk allocated to the upper and lower temperature bounds of episode a at time t. Note that $\delta_{a,t,U}+\delta_{a,t,L}=\delta_{a,t}$. $\bar{\boldsymbol{x}}_t$ is a *nominal* state, which is a deterministic variable defined as $\bar{\boldsymbol{x}}_t=E[\boldsymbol{x}_t]$. $R_a(\delta_{a,t})$ is a range of temperatures between what we call *safety margins*. A safety margin is a transformation of the stochastic constraints in (2.8) into deterministic constraints. Recall that we model the uncertainty $\boldsymbol{w}_t$ as Gaussian. Thus, for a given risk allocation $\delta_{a,t}$, we can calculate the equivalent deterministic constraints from the inverse Gauss error function, $\text{erf}^{-1}(\cdot)$.

For the Connected Sustainable Home, recall that each episode in a CCQSP specifies an acceptable range of the indoor temperature. Hence, the state constraint $\bar{\boldsymbol{x}}_t\in R_a(\delta_{a,t})$ in (2.10) is represented as follows:

$$T_a^L+m_t(\delta_{a,t}^L)\leq\overline{T}_t^{\text{in}}\leq T_a^U-m_t(\delta_{a,t}^U)$$

where $\overline{T}_t^{\text{in}}$ is the nominal (i.e., planned) indoor temperature, and T_a^L and T_a^U are the lower and upper bound of the indoor temperature of episode a. $\delta_{a,t}^L$ and $\delta_{a,t}^U$ are the risk allocations to the lower and upper bounds, which are assigned by the IRA algorithm, where $\delta_{a,t}=\delta_{a,t}^L+\delta_{a,t}^U$. The function $m_t(\delta)$ represents the width of the safety margin at time step t given the risk allocation δ, defined as follows:

$$m_{c,i}(\delta)=-\sigma_t^{\text{in}}\sqrt{2}\,\text{erf}^{-1}(2\delta-1),$$

where σ_t^{in} is the standard deviation of T_t^{in} and $\text{erf}^{-1}(\cdot)$ is the inverse of the Gauss error function. See [38] for the derivation.

Finally, since $\boldsymbol{w}_t$ is assumed to have a zero-mean disturbance, the following de-

terministic plant model is obtained from (2.1):

$$\bar{\boldsymbol{x}}_{t+1} = A_t \bar{\boldsymbol{x}}_t + B_t \boldsymbol{u}_t. \tag{2.12}$$

Fixed-risk CCQSP planning problem

We formulate the fixed-risk CCQSP planning problem below, that is solved in IRA-CCQSP to obtain nominal state trajectories with a given risk allocation. Let δ be a vector comprised of risk allocations $\delta_{a,t,j}$ for all $a \in \mathcal{A}$, $t \in \mathbb{T}_n$, and $j \in \{U, L\}$.

Definition 2. Fixed-risk CCQSP Planning Problem $\mathcal{P}_n(\delta)$ *is a constrained optimization problem with objective function (2.7) and constraints (2.3), (2.4), (2.6), (2.10), (2.11), and (2.12), given a fixed risk allocation* δ.

We argued above that $\bar{\boldsymbol{x}}_t \in R_a(\delta_{a,t,U}, \delta_{a,t,L})$ in (2.10) is equivalent to a set of two linear constraints. Furthermore, the implication in (2.6) and (2.10) is equivalent to a mixed-integer linear constraints [27]. Thus, the fixed-risk CCQSP planning problem is a mixed-integer linear program, which can be efficiently solved by a commercial solver, such as CPLEX.

2.4.3 Solving Chance-Constrained Problems through Iterative Risk Allocation

In this section we review the Iterative Risk Allocation (IRA) approach, applied to CCQSPs [37]. This algorithm is an iterative approach to finding the risk allocation for a planning horizon that minimizes the objective function J. Intuitively, the algorithm seeks to find the times at which taking risks provide the greatest benefit. In the application to the Connected Sustainable Home, it takes more risk of violating constraints at time points where the greatest energy can be saved.

Walk-Through of IRA-CCQSP

We walk through an example shown in Figure 2-3, a room temperature control problem with a 24 hour planning horizon in winter (hence the room must be heated). For

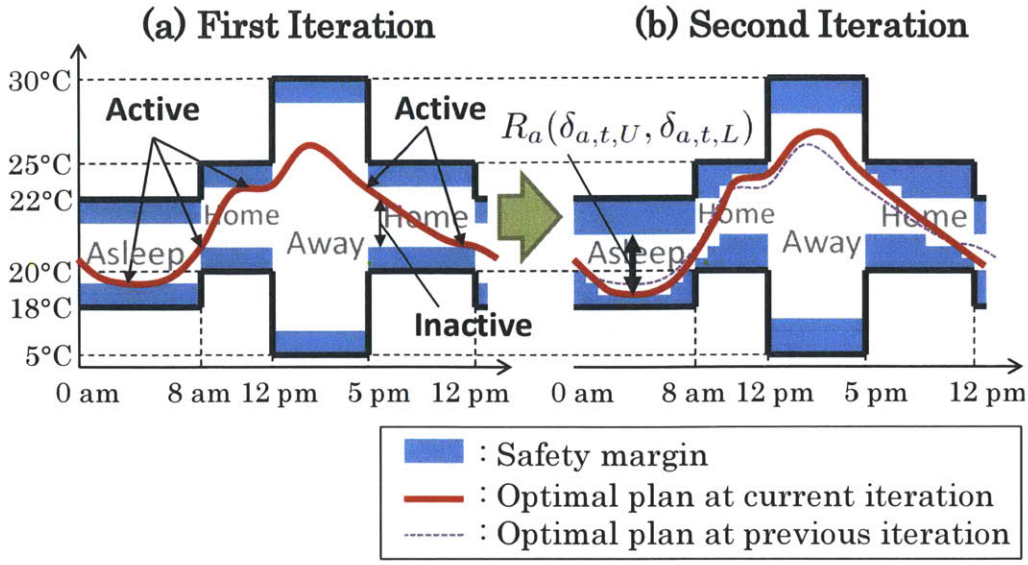


Figure 2-3: Intuitive explanation of the iterative risk allocation (IRA) algorithm.

the sake of simplicity of explanation, we assume a fixed schedule in this example, where the resident wakes up at 8 a.m., leaves home at 12 p.m., comes back home at 5 p.m., and goes to bed at 12 p.m. The room temperature is required to be within specified ranges according to the resident's state, as shown in Figure 2-3. For example, when the resident is awake and in the room, the temperature must be between 20 to 25 degrees Celsius.

The IRA-CCQSP algorithm starts from an arbitrary feasible risk allocation, such as the uniform one in Figure 2-3-(a), and improves the risk allocation through iteration. IRA-CCQSP guarantees satisfaction of chance constraints by setting a safety margin (shown as the shadowed areas in Figure 2-3) along the boundaries of the constraints, and planning a nominal state trajectory to remain outside of the margin. The width of the safety margin is determined so that the probability of constraint violation is below the risk allocated to each constraint. In Figure 2-3-(a) the safety margin is uniform for all the time because the initial risk allocation is uniform. The fixed-risk CCQSP planning problem, which is formally stated shortly, is solved to obtain the optimal plan that does not violate the safety margin, as shown in Figure 2-3-(a). The plan minimizes energy consumption by lowering the temperature during the night, while heating the room using sunlight during the day. It heats the room

to the maximum before the sunset at around 5 p.m. to store heat in the structure, so that the use of heater during the night can be minimized.

Note in Figure 2-3-(a) that, with this plan, constraints are active at a few points of time, while they are inactive at other points of time. To improve cost, IRA removes the risk that was allocated to the inactive constraints, and reallocates it to the active constraint. Note that reducing risk allocation results in a wider safety margin, while increasing risk allocation results in the opposite. Thus, the new risk allocation results in the safety margin shown in Figure 2-3-(b). The algorithm then solves the fixed-risk CCQSP planning problem again, in order to obtain the optimal plan that does not violate the new safety margin. In our example, the new plan is more energy efficient than the one in the previous iteration since the temperature can be lower during the night, while it higher in the evening to store more heat. In this way, the algorithm reallocates the risk again from inactive constraints to active constraints at every iteration. It terminates when all constraints become active or all constraints are inactive. The cost function value (i.e., energy consumption in this case) monotonically during successive iterations. The path generated at each iteration always satisfies the chance constraint since it does not violate the safety margin.

Inner Loop: CCQSP Planning using IRA-CCQSP

Next we present the IRA-CCQSP algorithm, which is used in every planning horizon of p-Sulu. Previous work [38] developed the IRA algorithm and applied it to simple path planning problems. This subsection presents IRA-CCQSP, which applies the IRA concept to planning with time evolved goals.

If the fixed-risk CCQSP planning problem $\mathcal{P}_n(\delta)$ is feasible with a risk allocation δ, and δ satisfies (2.11), we call such δ a *feasible risk allocation.* For the IRA-CCQSP algorithm, we assume that an initial feasible risk allocation is known. This assumption is reasonable since a fixed-risk CCQSP planning problem converges to the corresponding deterministic planning problem by increasing the risk bounds Δ_c with a uniform risk allocation, $\delta_{a,t,j} = \Delta_c/N_c, \forall a \in \Psi_c, t \in \mathbb{T}_n, j \in \{U, L\}$, where N_c is a number of constraints in c. Hence, for any feasible deterministic planning problem, we can find

a feasible risk allocation for the corresponding fixed-risk CCQSP planning problem by choosing appropriate risk bounds Δ_c.

Algorithm 2 IRA-CCQSP

1: Set initial risk allocation δ^1
2: $k = 1$
3: **repeat**
4: Solve $\mathcal{P}_n(\delta^k)$
5: **for all** $c \in \mathcal{C}$ **do**
6: $N_c \leftarrow$ number of active constraints in c; $\gamma_c \leftarrow 0$
7: **for all** $a \in \Psi_c$, $t \in \mathbb{T}_n$, $j \in \{U, L\}$ such that the constraint with index (a, t, j) is inactive **do**
8: $\delta^{k+1}_{a,t,j} \leftarrow \alpha \delta^k_{a,t,j} + (1 - \alpha) p_{a,t,j}(\bar{\boldsymbol{x}}_t)$
9: $\gamma_c \leftarrow \gamma_c + (\delta^k_{a,t,j} - \delta^{k+1}_{a,t,j})$
10: **end for**
11: **for all** $a \in \Psi_c$, $t \in \mathbb{T}_n$, $j \in \{U, L\}$ such that the constraint with index (a, t, j) is active **do**
12: $\delta^{k+1}_{a,t,j} \leftarrow \delta^k_{a,t,j} + \gamma_c / N_c$
13: **end for**
14: **end for**
15: $k \leftarrow k + 1$
16: **until** $\forall_{c \in \mathcal{C}}\ N_c = 0$ **or** $\forall_{c \in \mathcal{C}}\ \gamma_c = 0$

IRA-CCQSP is described in Algorithm 2. Here, k is the index of iteration. We denote by $\delta^k_{a,t,j}$ the risk allocated to the constraint (a, t, j) at iteration k, and by δ^k the vector comprised of all risk allocations at iteration k. The algorithm is initialized with a feasible risk allocation δ^1 is set in Line 1. Such a feasible risk allocation is found by Assumption 1. At each iteration, an optimal nominal state trajectory is obtained by solving the fixed risk CCQSP planning problem with a risk allocation δ^k (Line 4). In Lines 6-13 the algorithm reallocates risk from inactive constraints to active constraints. It reduces the risk allocated to inactive constraints (Line 8), and deposits the amount of risk removed from the inactive constraints in γ_c. In Line 8, $0 < \alpha < 1$ is an interpolation coefficient, and $p_{a,t,j}(\bar{\boldsymbol{x}}_t)$ is the probability of violating the constraint with index (a, t, j), given a nominal state $\bar{\boldsymbol{x}}_t$. In the case of the Connected Sustainable Home, $p_{a,t,j}$ is evaluated as follows:

$$p_{a,t,U}(\bar{T}^{in}_t) = 1 - F_t(T^U_a), \quad p_{a,t,L}(\bar{T}^{in}_t) = F_t(T^L_a),$$

where $\bar{T}_t^{in}$ is the nominal indoor temperature at time t, $F_t(\cdot)$ is the cumulative distribution function of the indoor temperature T_t^{in}, and T_a^U, T_a^L are the upper and lower bound of the comfortable temperature range of episode a. Intuitively, Line 8 obtains a new safety margin by interpolating the current safety margin and the current nominal state. Therefore, In Line 8 guarantees that the new safety margin is not violated by the current nominal state trajectory $\bar{\boldsymbol{x}}_t$. Then algorithm reallocates the amount of risk saved in γ_c to the active constraints. It splits the deposit of risk equally to the N_c active constraints in Line 12. By going through one iteration, the risk allocation is updated from δ^k to δ^{k+1}.

2.5 Application to the Connected Sustainable Home

In this section we describe our approach for applying p-Sulu to controlling the Connected Sustainable Home. We begin by describing the features of the home, and then in Section 2.5.1 we describe how the construction of the home is modeled using a stochastic plant model.

The Connected Sustainable Home is a one-story house with a single open room. It has a south-facing facade consisting of a 5×9 array of electrochromic window panes (dynamic windows). The 5×9 array of windows is split into two sections, one vertical 3×9 array and one slanted 2×9 array. Since the angle of incidence of solar radiation is different on each section, the opacities of the two sections are treated as separate control variables. It also has a heater and air conditioner, each of which can be set to output energy at a given power level. The external environment includes solar radiation, intermittent cloud cover, and varying outside temperature, all of which are uncertain. We assume that the home has a single occupant, so we do not need to concern ourselves with reconciling conflicting constraints of multiple occupants.

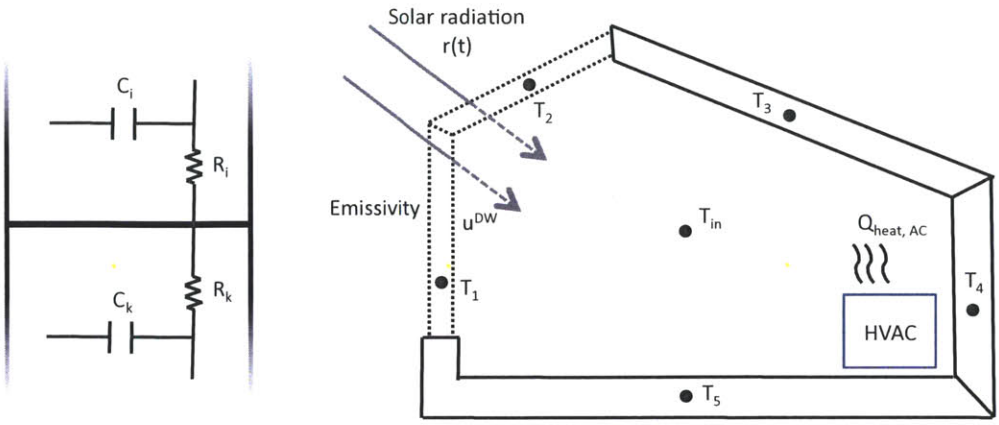


Figure 2-4: Model of temperature flow between lumped elements is analogous to an electric circuit (left). Depiction of state variables T_i and control variables $Q_{\text{Heat}}, Q_{\text{AC}}, u_t^{\text{DW}}$ (right).

2.5.1 Building Model

Recall that p-Sulu requires as input a stochastic plant model (2.1). We obtain a stochastic plant model of Connected Sustainable Home in the form of (2.1) by using a lumped parameter model for a thermal system [24]. The lumped parameter model is analogous to an electrical circuit. Separate components, or "lumped elements", of a building store heat according to its heat capacity C_i similar to how an electrical capacitor stores charge. Heat transfers between lumped elements subject to a thermal resistance R_i. Both C_i and R_i are physical properties of the materials. In our model of the home, we break the home down into a single lump for the indoor air mass, as well as a lump for each wall and window, as shown in Figure 2-4. In total the house is decomposed into 11 lumber elements: 4 for windows, 6 for walls, and 1 for the indoor air mass. For each component i, the following holds for a short time interval Δt:

$$C_i(T_{t+1}^i - T_t^i) = \Delta Q_{\text{Cond},t}^i + \Delta Q_{\text{Conv},t}^i + \Delta Q_{\text{Rad},t}^i, \tag{2.13}$$

here, T_t^i is the temperature of the ith component at the time step t. $\Delta Q_{\text{Cond},t}^i$, $\Delta Q_{\text{Conv},t}^i$, and $\Delta Q_{\text{Rad},t}^i$ are heat inputs to the ith component during the time interval through conduction, convection, and radiation, respectively. The outdoor environment is treated as a heat source, similar to a voltage source in the circuit analogy.

The conductive heat input $\Delta Q^i_{\text{Cond},t}$ accounts for all heat transfer through each wall. For example, the conduction heat input for the indoor air mass component is:

$$\Delta Q^{\text{in}}_{\text{Cond},t} = \sum_i \frac{A_i}{R^{\text{in}}_i}(T^i_t - T^{\text{in}}_t)\Delta t, \tag{2.14}$$

where T^{in}_t is the indoor temperature, T^i_t is the temperature of wall i, A_i is the area of contact between components i and j, and R^j_i is the heat resistance (R-value) between components i and j. The R-value of a material is a measure of how resistant the material is to thermal change. The convection term accounts for any heat transfer from the heater and air conditioner:

$$\Delta Q^{\text{in}}_{\text{Conv},t} = \Delta Q^{\text{Heat}}_t - \Delta Q^{\text{AC}}_t, \tag{2.15}$$

where ΔQ^{Heat}_t and ΔQ^{AC}_t are the outputs of the heater and air conditioner, respectively. Finally, the radiation term accounts for heat transfer from the sun through the glass facade:

$$\Delta Q^{\text{in}}_{\text{Rad},t} = A \cdot r(t) \cdot u^{\text{DW}}_t \Delta t, \tag{2.16}$$

where A is the area of the facade, r is the solar radiation, and u^{DW}_t is a control variable for the emissivity of the dynamic windows. By substituting (2.14)-(2.16) into (2.13), we obtain the following:

$$T^{\text{in}}_{t+1} = T^{\text{in}}_t + \frac{1}{C^{\text{in}}}\left(\sum_i \frac{A_i \Delta t(T_i - T_{\text{in}})}{R^{\text{in}}_i} + \Delta Q^{\text{Heat}}_t - \Delta Q^{\text{AC}}_t + Ar(t)u^{\text{DW}}_t \Delta t\right). \tag{2.17}$$

Similarly, the thermal model for the walls are obtained as follows:

$$T^i_{t+1} = T^i_t + \frac{1}{C^{\text{in}}}\left\{\frac{A_i \Delta t(T^{\text{in}}_t - T^i_t)}{R^{\text{in}}_i} + \frac{A_i \Delta t(T^{\text{out}}_t - T^i_t)}{R^{\text{out}}_i} + \Delta Q^i_{\text{Rad},t}\right\}. \tag{2.18}$$

We assume that future outdoor temperature T^{out}_t has uncertainty, which is represented by a Gaussian distribution. Hence,

$$T^{\text{out}}_t = \overline{T}^{\text{out}}_t + w_t, \tag{2.19}$$

where $\overline{T}_t^{\text{out}}$ is a constant representing the predicted temperature at time t, and w_t is a random variable that has a zero-mean Gaussian distribution with a known standard deviation σ_t.

Finally, the stochastic plant model (2.1) is obtained from (2.17)-(2.19) by defining the state vector and the control vector as follows:

$$\begin{aligned} \boldsymbol{x}_t &= [T_t^{\text{in}}, T_t^1, \ldots, T_t^{N-1}]^T, \\ \boldsymbol{u}_t &= [\Delta Q^{\text{Heat}_t}, \Delta Q_t^{\text{AC}}, u_t^{\text{DW}}]^T. \end{aligned}$$

Cost Function The cost function J in (2.7) is the total energy consumption over a planning horizon, give as follows:

$$J(\boldsymbol{u}, s) = \sum_{t=1}^{N_P} \frac{\Delta Q_t^{\text{AC}}}{\eta^{\text{AC}}} + \frac{\Delta Q_t^{\text{Heat}}}{\eta^{\text{Heat}}}$$

where η^{AC} and η^{Heat} are the thermal efficiencies of the air conditioner and heater, respectively, and N_P is the number of time steps in a planning horizon.

p-Sulu is also capable of controlling for other relevant parameters, such as illumination and humidity. These extensions are a focus of our future work to be included in the physical prototype.

2.6 Experimental Design

In our experimental design, we evaluate the performance of p-Sulu on the basis of two criteria. First, we evaluate the energy saved by p-Sulu. Specifically, we evaluate the energy saved from two features: model predictive control and occupant activities with flexible time bounds. Second, we evaluate the degree to which occupant comfort is improved by enforcing chance constraints. This section describes our experimental design and evaluation of success. In Section 2.7, we present and discuss the results of our experiments.

2.6.1 Experimental Setup

In this section we describe the setup of our experiment.

Schedules In our experiments, we assume the resident can specify one of 3 ranges: Home, Asleep, and Away, although in general we can select any number of temperature ranges. We assume that the temperature must be between 20 and 25 degrees celsius while the resident is home, between 18 and 22 degrees while sleeping, and between 4 and 35 degrees while away to ensure pipes do not freeze. With all Home and Asleep episodes, we associate a single chance constraint class with risk bound 10%, the risk the resident is willing to take that the temperature becomes uncomfortable. With all Away episodes, we associate a single chance constraint class with risk bound 0.01%, the risk the resident is willing to take that his or her pipes freeze.

In our experiments, we use two different schedules to illustrate the benefits of flexible control, one that takes advantage of flexible time bounds, and one that only uses fixed time bounds. Both schedules represent a typical work week of a resident living in the house; they consist of five work days followed by two weekends.

In the flexible schedule, each of the work days follows the schedule described in Section 2.3 and depicted in Figure 2-1. On these days, the resident sleeps until 8am, and then is at home for at least one hour. He must be gone from the house for 5 hours between 9am and 6pm to work at the office. The rest of the time he is at home. Each weekend is similar except that the resident does not go into work. That is, he sleeps until 8am and is at home all day.

The fixed schedule is similar to the flexible one, except that it does not capture the flexibility of when the resident goes into the office. On the work days, the resident sleeps until 8am, and then is at home for one hour. He leaves for work at 9am and arrives back home at 2pm. The rest of the time he is at home. The weekends of the fixed schedule are identical to those of the flexible schedule.

In all simulations, we let $\Delta t = 1$ hour, so there are 168 total time steps over the week-long schedules.

Baseline Controllers In our experiments we use two baseline controllers for comparison with p-Sulu: (a) a PID controller, and (b) Sulu, the deterministic predecessor to p-Sulu.

A PID controller (proportional-integral-derivative) is a controller that sets the control variable $u(t)$ as a function of the error of the state variable $x(t)$ from a given setpoint [1]. Let $e(t)$ denote this difference. The function is a linear combination of the error, the integral of previous errors, and the derivative of the error. In our PID controller, we let $u(t)$ be the heat Q added to the indoor air mass from both solar radiation and HVAC. The error $e(t)$ is calculated as the difference of the indoor temperature T^{in} from the setpoint T^*. Q is calculated by the PID linear combination of the error, scaled by gains for each term. The windows are set so that the maximum amount possible of the heat input Q needed to add to the indoor air mass comes from solar radiation. The HVAC is set to provide the remainder of Q to the house. The setpoint of the PID controller was chosen to be 21°C, a point that is feasible in every state. All gains were hand-tuned so that the temperature converges to the setpoint rapidly with minimal oscillation.

Sulu is an executive that is only different from p-Sulu in that it does not incorporate uncertainty into its planning algorithm. The original executive from [26] was not used; instead, it was replicated by using p-Sulu and setting all chance constraints Δ_c to 1.

Weather Data Nominal weather values for the simulation come from historical weather data from 2005 in Trento, Italy, a city close to Rovereto, the site of the prototype. The weather data consists of hourly measurements of outside temperature, global irradiation, diffuse horizontal irradiation, and direct normal irradiation.

Uncertainty For each weather parameter, the standard deviation of $\boldsymbol{w}_t$ for each hour is calculated based on the standard deviation of that parameter on each day within one week before and after the given day. For example, the standard deviation of the outside temperature at 9am on June 21st is calculated as the standard deviation

of all of the outside temperatures at 9am from June 14th to June 28th.

In simulation, all controllers use the weather data as the nominal "forecast" for the weather. The controllers plan according to the forecast, and then the output control parameters $\boldsymbol{u}_t$ are executed on the same model of the home, but with a disturbance introduced to the weather parameters drawn from a Gaussian distribution with the calculated standard deviation. At the end of each execution horizon, the controllers are given the actual, nominal value of $\boldsymbol{x}$ to use as the initial state in the next planning horizon.

Planning Parameters For all simulations, we use a planning horizon of 24 hours and an execution horizon of 12 hours.

2.6.2 Evaluation of Energy Savings

We separate our analysis of energy saved by p-Sulu to focus on two aspects: (1) robust model predictive control, and (2) flexible time bounds. For each aspect, we ask how much less energy p-Sulu uses compared to a baseline controller.

To evaluate (1), we compare the energy used by p-Sulu with the energy used by the traditional PID controller. To evaluate (2), we use p-Sulu to plan and execute the control over both the flexible and the fixed schedules, comparing the energy use for each. Results are given in Section 2.7.1.

2.6.3 Evaluation of Occupant Comfort

To evaluate the benefit of p-Sulu to occupant comfort, we compare the number of constraint violations resulting from a plan execution from both Sulu and p-Sulu. This measure evaluates how effectively p-Sulu maintains resident comfort requirements. In many trials, Sulu failed to complete due to infeasibility. These trials are not included in our measure of comfort as they do not provide an accurate measure of constraint violations. The number of failures is, however, included in our discussion to further support the argument for planning with chance constraints. Results are given in

Section 2.7.2.

When we say the planner failed due to infeasiblity, we mean that the planner failed to generate a feasible control sequence because of a constraint violation or overly restrictive constraints. In all of the failed trials, this was due to the final temperature of some execution horizon falling out of the feasible region. Then the initial temperature of the next planning horizon violates the temperature constraints, leading to an infeasible planning problem. This is a deficiency in general of MPC approaches that do not consider uncertainty[1]. Because MPC is formulated as an optimization program, the optimization will choose states $\bar{\boldsymbol{x}}_t$ that are at the edge of constraints. A small deviation in the system from nominal can cause $\boldsymbol{x}_t$ to violate the constraint.

2.7 Simulation Results

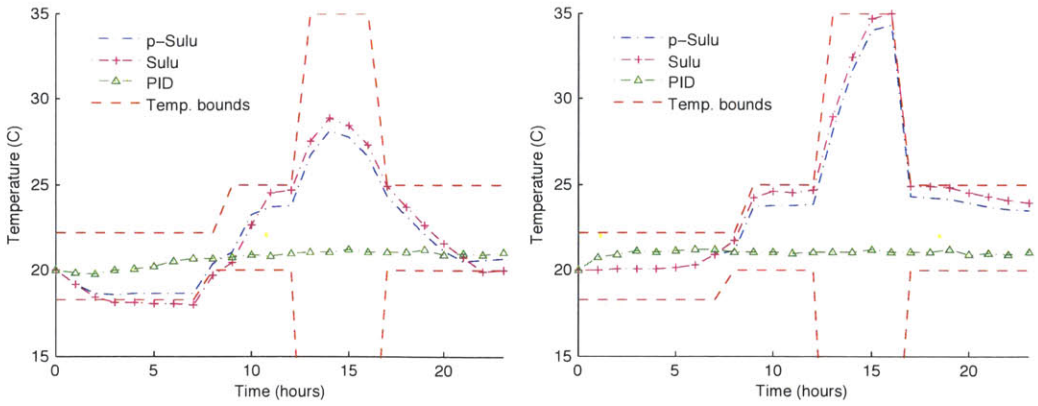
In this section we present and analyze the results of our experiments. In each subsection, we begin by taking a qualitative look at the results of the planning and execution of the single-day schedule depicted in Figure 2-1. We then analyze the quantitative results and discuss our conclusions. All values presented are averaged over 100 trials.

2.7.1 Analysis of Energy Savings

In this section we present and discuss the results of our experiments analyzing the energy that is saved by using p-Sulu to control the Connected Sustainable Home.

Figures 2-5a and 2-5b illustrate the results of a stochastic simulation over two different days in the year. Notice how both Sulu and p-Sulu take advantage of the schedule of the resident, planning so that energy can be saved during the day by allowing the temperature to rise while the occupant is away. Also, Sulu and p-Sulu are capable of reasoning over the model of the home, taking advantage of the large thermal mass of the home by capturing solar heat during the day to offset energy costs

[1]This issue is typically addressed by introducing slack variables, which allow constraints to be violated, but strive to minimize these violations.



(a) January 1 (b) July 1

Figure 2-5: Results of execution of the PID, Sulu, and p-Sulu controllers on January 1 (left) and July 1 (right).

at night. The PID controller is naive, only planning to meet a particular setpoint and not taking advantage of the variation in the resident's schedule and the dynamics of the home.

Analysis of Savings from Robust Model Predictive Control In Table 2.1 we present the results of the stochastic simulation on the flexible week-long scenario, averaged over 100 Monte Carlo trials each. We see that in the winter, p-Sulu yielded energy savings of 42.8% over the PID controller; in the spring, summer, and autumn, we saw 15.3%, 16.8%, and 4.4% savings respectively. We conjecture that the savings in the winter are much higher because of the advantages afforded by the south-facing facade of dynamic windows. In the winter, the difference between indoor and outdoor temperature is largest compared to other seasons. Intuitively, this difference leads to greater benefit from harnessing heat from the sun to offset heat that is lost more rapidly due to the larger temperature differential. Although not as substantial, the savings in the other three seasons show significant strides towards achieving the 35% possible energy reduction predicted by [14] by optimizing HVAC control. The savings in these seasons provide a more reasonable estimate of how p-Sulu would perform on the average home without the facade of electrochromic windows, although future work in applying p-Sulu to a variety of homes is needed to verify this statement. This data

Table 2.1: Comparison of energy use and failure rate for a simulation of the PID, Sulu, and p-Sulu controllers over a week-long schedule in all four seasons. Failure rate is measured as the percentage of time steps with constraint violations.

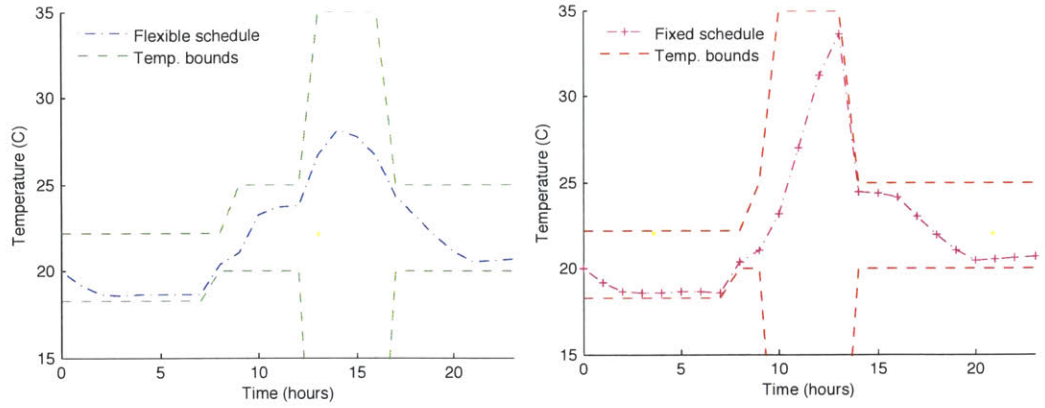
	Winter		Summer	
	Energy	Violation Rate	Energy	Violation Rate
p-Sulu	1.9379×10^4	0.000	3.4729×10^4	0
Sulu	1.6506×10^4	0.297	–	–
PID	3.9783×10^4	0	4.1731×10^4	0
	Spring		Autumn	
	Energy	Violation Rate	Energy	Violation Rate
p-Sulu	3.3707×10^4	0	3.8181×10^4	0
Sulu	3.0954×10^4	0.308	3.6780×10^4	0.334
PID	3.9816×10^4	0	3.9955×10^4	0

Table 2.2: Comparison of energy use for a simulation on the Connected Sustainable Home using both fixed and flexible time constraints.

	Winter	Summer
Flexible	1.9379×10^4	3.4729×10^4
Fixed	2.1625×10^4	3.5295×10^4
	Spring	Autumn
Flexible	3.3707×10^4	3.8181×10^4
Fixed	3.4253×10^4	3.8445×10^4

illustrates the large energy savings to be had by leveraging knowledge of the home in model predictive control.

Analysis of Savings from Flexible Temporal Constraints In Table 2.2 we present a comparison of the results of the stochastic simulation on both the flexible and fixed week-long scenarios. Compared to the fixed schedule, the temporally flexible schedule shows energy savings of 10.4%, 1.6%, 1.6%, and 0.7% in the winter, spring, summer, and autumn respectively. Again, we see the most energy savings in the winter at 10.4%. Refer to Figure 2-6 for a possible explanation of why the savings are so large. If the occupant leaves for work at 9am as he does in the fixed schedule, he is back by 2pm when the sun is still out. We postulate that instead of the control



(a) Flexible (b) Fixed

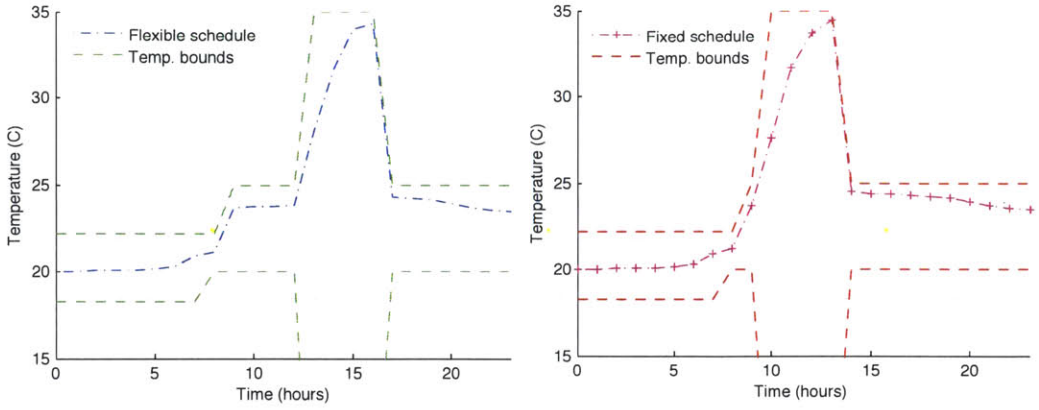
Figure 2-6: Results of execution of p-Sulu control on January 1st with both a flexible schedule (left) and a fixed schedule (right)

system being able to store the solar energy in the house generated from 2pm until the sun sets, it must allow the house to cool off by 2pm, effectively wasting any heat the system could have captured in the thermal mass of the home. If we control with flexible temporal bounds, then p-Sulu can choose the optimal schedule, and find the control sequence that minimizes energy use according to this schedule. This data only illustrates the energy savings from exploiting one aspect of flexibility in a resident's schedule. If all flexibility in a resident's activities is leveraged, we expect to see larger energy savings under the adoption of this technology.

2.7.2 Analysis of Occupant Comfort

Look again at Figures 2-5a and 2-5b. Notice that Sulu plans right up to the edge of the constraints, often violating constraints when a disturbance is introduced, while p-Sulu leaves a margin. The deterministic planning of Sulu leads to frequent uncomfortable conditions for the resident when constraints are violated.

This fact is reflected in the quantitative results of our simulations. Out of the 100 total trials of Sulu in the winter, 78 failed to complete due to infeasibility. In the spring 22 of 100 trials completed and in the autumn 11 of 100 completed. All trials of Sulu in the summer failed due to infeasibility. As a measure of comfort, we look at the trials that completed and consider the fraction of time steps on which a constraint



(a) Flexible (b) Fixed

Figure 2-7: Results of execution of p-Sulu control on July 1st with both a flexible schedule (left) and a fixed schedule (right)

is violated. From Table 2.1, we see that out of the 168 time steps, Sulu violated constraints on 29.7% of time steps in the winter, 30.8% in the spring, and 33.4% in the autumn. On the other hand, only one single trial of p-Sulu violated a single constraint in winter. For all other seasons, all other trials satisfied all temperature constraints. Using constraint violations as a measure of success, the robust MPC approach of p-Sulu outperforms the deterministic MPC approach of Sulu on every trial. Averaged across all trials, p-Sulu exhibits a difference of 30.88% in improvement in comfort over Sulu. These results illustrate how critical risk-sensitive control is in guaranteeing resident comfort and encouraging the adoption of the technology. A control system that produces uncomfortable conditions 30% of the time would quickly be abandoned by any users.

There is, naturally, a tradeoff to the improvement in comfort afforded by risk-sensitive control. p-Sulu resulted in increased energy consumption of 17.4%, 8.9%, and 3.8% in the winter, spring, and autumn respectively. Future work is needed to properly analyze the tradeoffs between comfort and energy savings.

Chapter 3

Plan Recognition using Timed Concurrent Constraint Automata

In order to use the building control system described in Chapter 2, the system must know the plan of the occupant, that is, the activities that the occupant would like to perform and constraints on when they should be performed (i.e. the occupant's qualitative state plan). We could require that the occupant input his or her schedule into the system. However, this would place undue burden on the occupant and violate our goal of convenience set forth in Chapter 1. Instead, we propose to *automatically* detect the plan of the resident based on home sensors, such as power sensors on outlets monitoring electricity usage, touch sensors on objects in the home to detect when they're used, and motion sensors in the home to detect when the resident enters a room. The problem of determining the plan of the resident based on observation is referred to in the literature as **plan recognition** [43][6][15].

In this chapter we begin in Section 3.1 by describing our vision for the interaction of the home control system with the resident. We then present preliminary work towards achieving this vision, an approach to plan recognition based on Timed Concurrent Constraint Automata (TCCA). We begin in Section 3.2 by presenting the motivation for this work and describing a motivating example that will be used throughout the chapter. In Section 3.3 we sketch our approach to solving the problem. Then in Section 3.4 we describe the representation we use for the problem. Next, in

Section 3.5, we formulate the plan recognition problem and in Section 3.6 describe the requirements we have of a solution. In Section 3.7 we present a discussion of related works. Finally in Section 3.8 we describe the technical details of our approach.

3.1 Vision for an Integrated Autonomous Control System

We envision an integrated home environment in which an autonomous control system in the home can work with the resident to sustainably accomplish home tasks. This includes not only providing a comfortable living environment, but also facilitating in performing other energy-consuming tasks, such as running a washing machine or charging an electric vehicle. For example, the house could be responsible for scheduling the charging of the vehicle during off-peak hours to save the resident money on energy during costly peak hours, and contribute to off-loading this peak grid-wide. To sustainably assist the resident in any of these tasks, the autonomous control system must have extensive knowledge of not only the plan of the resident, but also the infrastructure of the home, including appliances and the resident's interaction with these appliances. That is, the control system must have a model that describes the entire infrastructure, humans, appliances, and sensors.

Recognition is not just about the human, but the collective system. For example, the activity of making breakfast can be described by the interaction of the human with a coffee pot, a toaster, and a refridgerator, perhaps. Human behavior can be well defined in terms of interaction with the entire infrastructure, devices and sensors. In [39], the authors describe the "invisible human hypothesis", which states this notion: activities are well-defined by the objects manipulated during their performance. There must be a *unified representation for recognition* that integrates devices with human plans. We now describe our vision for this representation. In our discussion, we break the system down into three components: humans, devices, and sensors.

An issue that arises when defining a unifying representation is that a model suit-

able for describing humans is not suitable for describing devices, and vice versa. PDDL has become the standard for modeling tasks and plans that a human (or robot) performs for good reason, as it is a natural representation for the domain. When a human performs a task, often there is a set of preconditions that must be true to execute the task. For example, to make coffee, one has to first put a coffee filter in the machine and then fill the filter with coffee grounds. Once the task has been executed, there is a set of effects, or postconditions, that result from the task. When coffee is done brewing, one now has a cup of coffee that he or she may drink. However, from this author's experience, modeling *devices* in PDDL is awkward and unnatural.

Devices are better expressed as automata. They have states that they operate in, they transition between these states subject to some constraints, or after a time period has elapsed. The device may be affected by an external input, which serves as a requirement for transitioning. Pressing the *on* button on a coffee machine transitions the machine from off to brewing coffee. In this chapter we use TCCAs to represent devices.

To unify these two disparate representations, we propose to reformulate human actions into a unifying representation. In [11], the authors describe a general algorithm for reducing the state encoding size of a PDDL domain, and compactly encoding the domain in an alternative representation. Many modern PDDL planners utilize this technique to generate an alternative internal representation more efficient and amenable to the planning task. Within the MERS group at MIT, recent research has been focussed on translating a PDDL domain into a TCCA. To unify the representations for human actions and devices, we propose representing each in their natural language, PDDL for humans and TCCAs for devices. We then translate PDDL into a TCCA and perform plan recognition over the combined human-device representation. In this chapter we describe human plans as TCCAs; however, in our vision these TCCAs would be compiled from PDDL.

On top of this representation, a sensing framework is necessary to take observations on the human to enable recognition. Since we describe human behavior in terms

of devices, we require sensing to determine the states of devices. For appliances in the home that consume energy, we propose leveraging research on energy disaggregation. Energy disaggregation is the problem of analyzing an aggregate home line voltage and separating out the states of component appliances in the home. This topic is discussed further in Section 4.1.4.

Once a representation framework has been defined, another set of problems must be solved to enable seamless interaction of the home and resident. First, the models must be learned. Much previous research has been devoted to the task of model learning [16][10]. Once models for device and human behavior have been learned, the system must recognize plans of the human over this framework to enable the house to react to the resident. In this chapter we describe one piece of our vision: using TCCAs to perform plan recognition over devices.

3.2 Motivation for Plan Recognition to Predict Resident Behavior

Recall from Chapter 2 that our autonomous building controller p-Sulu can sustainably control the Connected Sustainable Home while satisfying all occupant requirements *if* p-Sulu knows the plan of the occupant. For the system to be completely autonomous, it must be capable of automatically detecting this plan.

We now describe a motivating example that will guide our exposition through the chapter. Suppose the occupant follows a similar routine everyday. He wakes up, goes into the kitchen, and turns on the coffee pot to start coffee brewing. Once the coffee is ready, he drinks the coffee and then exits the kitchen. On work days he is usually running late, so he only has 5-10 minutes to sit down and drink his coffee. However, on days that he spends at home, he takes his time drinking the coffee, often taking 20-30 minutes to leisurely sip at his cup. The entire time he leaves the coffee pot warming, but makes sure to turn it off before leaving the kitchen.

Assume the coffee pot operates in one of three states: off, brewing, or warming.

Suppose the house is outfitted with two sensors: one power sensor on the coffee pot outlet capable of detecting commands to the appliance, and one motion sensor to detect if the kitchen is occupied or vacant. The sensor on the outlet might leverage techniques of energy disaggregation [20][25] to determine the state of the coffee pot.

Since the building control system can save energy by turning off the HVAC system while the occupant is away, it must be able to determine, based on observations from the sensors, which plan the occupant is following, the rushed morning or the leisurely morning. Suppose we observed that the occupant entered the kitchen at 8am and switched the coffee machine off at 8:12am. The sensors missed him turning on the coffee machine and him later leaving the kitchen. We want to infer these missing observations and recognize that he must be on his way to work since he drank his coffee so quickly. This is the problem we solve in this chapter using plan recognition. In general, the plan recognition problem is to take a set of plans as input and a sequence of observations, and to output the plans that are consistent with the observations.

3.3 Overview of Approach

In this section we present an overview of our approach before going into the technical details in the following sections. In our motivating example, we separate the various components discussed into two categories: state variables and control variables. State variables include the state of the coffee machine, which can be off, brewing, or warming, and the state of the kitchen, which can be occupied or vacant. They also include the state of the occupant in pursuing his plan. For example, his state might be "in kitchen waiting on coffee", or "done with coffee". The control variables are the inputs to the system. The button on the coffee machine is a control variable. It sends an "on" or "off" control signal to the machine when flipped on or off. All observations are taken on control variables by sensors.

In plan recognition, we want to take the observations on control variables and determine what goal or plan the occupant is pursuing. That is, we're given control variables and ask what evolution of state variables is consistent with these controls.

In the related field of planning, we are given a goal state, and must determine the control sequence that is needed to reach that state. Essentially, plan recognition takes controls as input and outputs states, while planning takes states as input and outputs controls. This observation leads to the conclusion that plan recognition is essentially planning in reverse. This notion serves as the motivation for our approach.

To perform plan recognition we leverage the extensive body of work from the planning community. We propose using a planner to do plan recognition. The essential idea is to *encode* the control variables of observations as state variables in a planning domain. Specifically, we use timed concurrent constraint automata (TCCA) [7]. We encode the observations in such a way that enforces the control values and timing of the observations to be included in the control sequence output by the planner. That is, given an observation sequence, we encode the sequence as an automaton that permits the sequence. The planner will output a valid control sequence if and only if the observations are consistent with the plan of the occupant.

3.4 Plan Representation and Background

In this subsection we describe our representation for the plans of the occupant. We begin by explaining the requirements we have of the representation, and then we formally describe our representation that meets those requirements, *timed concurrent constraint automata* (*TCCA*).

As an occupant living in a house, we assume that most of one's plans are determined by interaction with various components of the house, be they devices, or perhaps rooms. In our example scenario, the resident's plan to make coffee is determined by his interaction with the coffee machine, and his entrance and exit to the kitchen. We assume the occupant's plans are fully described in terms of these interactions, and we choose to represent the actions he or she takes towards completing the plan in terms of states and transitions of these components.

We structure our representation of the world the occupant is acting in into two layers, an action layer and a plan layer. The action layer consists of the building

devices, their states, and transitions between states. The plan layer describes the occupant's plan in terms of the states of these devices.

The representation of the occupant's plan within the house must be capable of capturing both the various operating states of appliances and devices in the house, and it must be capable of representing how these lower-level states relate to higher-level states of the occupant. Since we must differentiate between similar plans with different temporal relations, the representation must express the notion of time and durative actions. Also, since the house contains many devices, it must be capable of representing many concurrently operating devices.

The representation we will use that has all of these properties is the timed concurrent constraint automaton. A TCCA is a way of representing multiple concurrently operating automata, called timed constraint automata (TCA), and the interaction between them. Each individual automaton has a state that it is operating in, as well as input variables that control the automaton, and output variables that allow the automaton to affect other automata. The input variables will serve as the observables in the devices on which the sensors act. The automaton transitions between two states dependent on a given condition, or *guard*, that must be true to take the transition. The guard is represented as a propositional formula that is a function of the input variables to the automaton. The transition function also places lower and upper bounds on the duration for which the guard must remain true for the transition to occur.

The TCA will serve as our representation of both the building devices in the action representation, and the plan representation of the resident. The TCCA will connect the action representation to the plan representation, capturing the role each device takes in the occupant's plan.

3.4.1 Action Representation

In this section we introduce our representation of the behavior of each component of the house, the timed constraint automaton (TCA).

We first introduce the concept of a TCA by way of an example, a coffee machine.

Consider the automaton representation of a coffee machine depicted in Figure 3-1. The coffee machine has three states: off, brewing, and warming. To transition from off to brewing, the automaton must receive the *on* command. To transition from brewing to warming, that is, for the coffee to finish brewing, the machine must not be turned off for 5 minutes. Once it is done brewing, the automaton outputs the *coffee warming* signal for the duration that it stays in the warming state. As we will see, this output will serve to signal a transition in another automaton representing the resident's plan for making and eating breakfast.

Formally, we describe a single constraint automaton $\mathcal{C}$ in the following definition inspired by [47] and [7].

Definition 3. *A **Timed Constraint Automaton (TCA)** is defined as a tuple* $\langle X, \delta, \mathcal{Q} \rangle$ *where:*

- X *is the set of variables* $x \in X$ *with finite domain* $D(x)$ *describing the state of the automaton. An assignment to the variable* x *is denoted as a pair* (x, v) *for some* $v \in D(x)$*. These variables are further partitioned into a single state variable* x^m*, a set of input, or control, variables* X^u*, and a set of output variables* X^y*. We refer to an assignment* (x^m, v) *to the state variable as the **state** of the automaton.*

- δ *is the transition function that associates with each pair of states a guard over* X^u *and a lower and upper time bound. A guard is a propositional formula that must evaluate to true for the automaton to transition between the two states. The lower and upper time bounds are bounds on the length of time for which the guard must remain true in order to make the transition. Note that the automaton does not enter the new state until this time has elapsed.*

- $\mathcal{Q}$ *is a state-constraint function that associates with each state, an input-output relation expressed as a propositional formula over the variables in* $X^u \cup X^y$*. In the given state, the state-constraint must evaluate to true. The state-constraint function provides a simple mechanism for describing the relationship between inputs and outputs.*

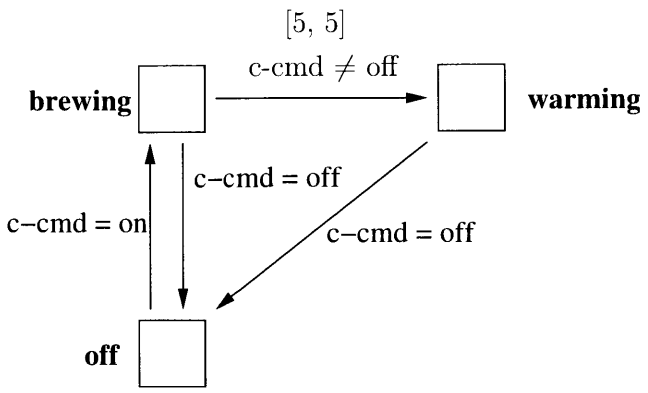


Figure 3-1: A TCA model of a coffee machine.

In Figure 3-1, states are represented by squares and transitions by directed edges. Guards are expressed as propositional formula on the edges, together with time bounds of the form $[lb, ub]$. State-constraint functions describing outputs are expressed as propositional formula on the states.

Throughout this thesis, we will sometimes refer to a TCA as simply an "automaton". When a TCA represents a device or component of the house, we refer to it as a *device TCA*, or device automaton. From this point forward we will let $\mathcal{A}_D$ denote the set of device automata in the house.

3.4.2 Plan Representation

Our representation of a resident's plan also takes the form of a TCA, which we will call a *plan TCA*, or simply a plan automaton. The plan automaton has the same representation as the device automata, except that we require its input to come from the output of the device automata. Suppose X_D^y is the set of output variables of the device automata. We require that a plan TCA is a TCA who's input variables come from the set of output variables X_D^y of $\mathcal{A}_D$. In the next section we describe how the plan representation and action representation interact, that is, how the output variables of the device automata are connected to the input of the plan automaton.

3.4.3 Connecting the Action and Plan Representations: Timed Concurrent Constraint Automata (TCCA)

The home contains more than one device, all of which play a role in the plan of the resident. Now that we have defined a single timed constraint automaton, we can describe how to connect them together to relate how the devices of the home interact with the plan representation of the resident. The timed concurrent constraint automaton describes how multiple timed constraint automata act on one another.

Before we define the notion formally, recall the example from Section 3.2. We can represent this scenario using three separate TCAs: one representing the coffee machine, one representing the occupancy of the kitchen, and another plan TCA tracking the progress of the occupant towards finishing breakfast. The coffee machine is represented by the automaton already discussed and illustrated in Figure 3-1. The automaton for kitchen occupancy has two states, occupied and vacant, where a transition occurs when the occupant walks through the kitchen door. The progress of the occupant is represented as a sequence of states. He starts off in the state *in bedroom.* When the kitchen occupancy automaton signals that he entered the kitchen, we transition to the state *in kitchen.* When the coffee machine is in the warming state, it means that the coffee is ready to be drunken. It stays in the warming state for 5-10 minutes while the occupant drinks the coffee before we transition to the state *coffee finished.* Finally, he leaves the kitchen and we transtion to the *gone* state.

This TCCA describing the resident's plan is represented in Figure 3-2.

Definition 4. *A **Timed Concurrent Constraint Automaton** is described by the tuple* $\mathcal{M} = \langle \mathcal{X}, \mathcal{X}^c, \mathcal{A}, \mathcal{I} \rangle$*, where:*

- $\mathcal{A}$ *is a set of timed constraint automata,*
- $\mathcal{X} = \bigcup_i X_i$ *is a set of variables.* X_i *denotes the set of variables of automaton* $A_i \in \mathcal{A}$*,*
- $\mathcal{X}^c \subset \mathcal{X}$ *is the set of control variables of the automaton. By control variable, we mean a variable whose value can be assigned by the planner, and*

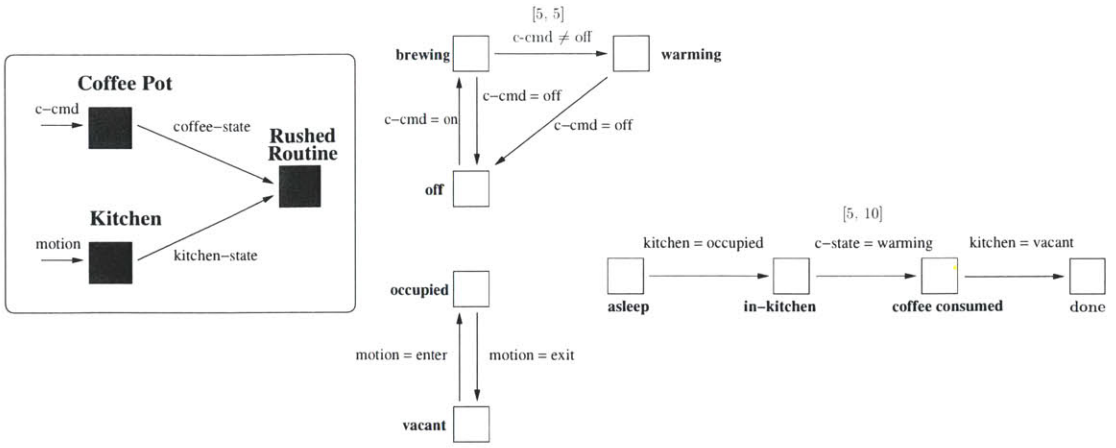


Figure 3-2: A TCCA model of making coffee on a rushed morning. All transitions without time bounds have bound $[0, \infty]$.

- *$\mathcal{I}$ is a set of interconnection constraints composed of equalities between variables of different automaton. These equalities provide a mechanism to allow the variables of one TCA to affect the behavior of another TCA.*

For a given plan automaton and a set of device automata, we connect them together into a TCCA, which we will call the **planning automaton**. Recall that all inputs of the plan automaton are taken from the outputs of the device automata. The planning automaton is constructed by defining interconnection constraints that reflect this, namely, they set the inputs of the plan automaton equal to the corresponding outputs of the device automata. We denote a planning automaton for the plan $P \in \mathcal{A}_P$ as the tuple $\langle P, \mathcal{A}_D \rangle$.

3.4.4 Observables

Recall our example of the resident making coffee. In this example, we had two components in $\mathcal{A}_D$ that were observable, the coffee machine and the occupancy of the kitchen. On the coffee machine, we could observe the input of the switch, on or off. On the kitchen, we could observe when someone enters or exits. In this section we describe our assumptions about observability and define what we mean by an observation.

First, we assume that the devices are partially observable. Let $\mathcal{A}_D = \{A_1, \ldots, A_n\}$

denote the set of device automata. We specify a subset $\mathcal{A}_O \subset \mathcal{A}_D$ of the device automata that are observable. By observable, we mean that all observations are taken on the input variables of the automata in $\mathcal{A}_O$. It may not be the case that every input to these automata is observed.

Next, we assume that the observations are only partially observable. That is, not every input to the observable automata $\mathcal{A}_O$ is observed.

Also, we assume that observations are not noisy. That is, every input that is observed actually occurred. We assume any error correction is handled by the lower-level sensing framework.

An **observation** o_i is an assignment to a single control variable of $\mathcal{A}_O$ together with a start time and an end time to the observation. Note that we allow for instantaneous controls by setting the start time and end time to be equal. Formally, we describe an observation as a tuple $o_i = \langle (x^c, v), t_S, t_E \rangle$ where

- (x^c, v) is an assignment of the control variable x^c to the value $v \in D(x^c)$,
- $t_S, t_E \in \mathbb{R}_+$ with $t_S \leq t_E$ are start and end times to the observation. We let $\Delta t_o = t_E - t_S$ denote the duration of the observation.

3.4.5 Space of Candidate Plans

Each individual plan a resident may be following is represented as a set of plan automata. We say that the space of candidate plans is the set of all plan automata that the resident might be following. We denote this set by $\mathcal{A}_P$.

3.5 Problem Formulation

This section describes the problem of plan recognition more formally.

Let us turn back to our example of making coffee. This example had two device automata in $\mathcal{A}_D$, the coffee machine and the occupancy of the kitchen, both of which are observable. There were also two plans of the resident we were trying to distinguish between, the leisurely coffee or the rushed coffee. These two plans comprise our set of

candidate plans $\mathcal{A}_P$. Recall our example sequence of observations saying the occupant entered the kitchen at 8am and switched the coffee machine off at 8:12am. We require our problem to take as input these observations $\mathbf{O}$, the device automata $\mathcal{A}_D$, and the candidate plans $\mathcal{A}_P$, and determine which plans are consistent with the observations. We present this problem formally in the following definition.

Definition 5. *A **Plan Recognition Problem** over TCCAs is described by the tuple* $\langle \mathcal{A}_D, \mathcal{A}_P, I, \mathbf{O}, G \rangle$*, where:*

- $\mathcal{A}_D$ *is a set of device automata representing the devices of the house.*
- $\mathcal{A}_P$ *is a set of plan automata encoding the candidate plans of the occupant.*
- I *is the initial state. I is a full assignment of states to the state variables of each TCA in $\mathcal{A}_D$ and $\mathcal{A}_P$.*
- $\mathbf{O} = (o_1, \ldots, o_n)$ *is a sequence of observations.*
- G *specifies the end goals of each candidate plan in $\mathcal{A}_P$. For each plan, G specifies an end state in the automaton, together with time bounds lb and ub specifying bounds on the time by which the end state must be achieved.*

3.6 Solution Definition

The solution to the plan recognition problem is the subset of plans $P \in \mathcal{A}_P$ for which there exists a satisfying control plan that reaches the goal $G(P)$ within the specified time bound. By control plan, we mean an assignment to the input variables over time. By "satisfying", we mean that the control plan contains all observations in $\mathbf{O}$, and the goal is reached when the control plan is applied to the planning automaton created from P and the device automata $\mathcal{A}_D$. We define these notions formally as follows.

In order to describe the solution to the plan recognition problem, we introduce the **Qualitative Control Plan (QCP)**. The QCP is an expressive representation of a control plan that encodes assignments to input variables over time.

Definition 6. *A **Qualitative Control Plan (QCP)** is a tuple $\langle \mathcal{E}, \mathcal{A}, \mathcal{T} \rangle$ where:*

- *$\mathcal{E}$ is a set of events. Events are plan elements that can be assigned to a specific point in time.*
- *$\mathcal{A}$ is a set of episodes where each episode $a \in \mathcal{A}$ is itself a tuple $\langle e_S, e_E, (x_i^c = v) \rangle$. An episode is represented as a directed edge between events with constraints on control variables. Episodes have a start event e_S, an end event e_E, and a state constraint that must be true over the duration of the episode, given as an assignment of v to the control variable x_i^c.*
- *$\mathcal{T}$ is a set of temporal constraints represented as a simple temporal network (STN), which specifies lower and upper bounds on the duration between events.*

We say that a QCP C **satisfies** a given plan $P \in \mathcal{A}_P$ of the plan recognition problem $\langle \mathcal{A}_D, \mathcal{A}_P, I, \mathbf{O}, G \rangle$ if the following conditions hold:

- For every observation $o_i \in \mathbf{O}$ with $o_i = \langle (x_i^c, v), t_S, t_E \rangle$, there is an episode $a = \langle e_S, e_E, (x_i^c = v) \rangle$ in C such that all legal assignments of times t_S^e, t_E^e to the events e_S, e_E respectively have the property that $t_S^e \leq t_S$ and $t_E^e \geq t_E$. That is, there exists an episode in the QCP that holds the observed control variable to the given value over at least the same time period as the original observation. We say that p *explains* the observation o_i. Note that we allow the control variable to be held at the value for longer than the observed time period.
- The state and temporal bounds of the goal $G(P)$ are satisfied under legal runs of the control plan C on the planning TCCA $\langle P, \mathcal{A}_D \rangle$.

We now formally define what we mean by a solution to the plan recognition problem, namely, the set of plans for which there exists a satisfying QCP.

Definition 7. *A **solution** to the plan recognition problem $\langle \mathcal{A}_D, \mathcal{A}_P, I, \mathbf{O}, G \rangle$ is the set of plans $\{P_i\} \subset \mathcal{A}_P$ such that for each P_i, there exists a QCP that satisfies P_i.*

3.7 Related Work

In 1978, Schmidt et. al. first described the plan recognition problem in very general terms [43]. They said "the problem of plan recognition is to take as input a sequence of actions performed by an actor and to infer the goal pursued by the actor and also to organize the action sequence in terms of a plan structure." This statement succinctly describes the problem and the work serves as a source of motivation for this thesis.

Existing plan recognition algorithms use a variety of techniques, from context free grammars and parsing algorithms, to Bayesian network inference, to specialized procedures. In [15], the authors use string parsing to probabilistically recognize plans based on specialized grammars. In [6], the authors create a variant of Hidden Markov Models to represent a plan domain and use a Rao-Blackwellised Particle Filter to perform probabilistic inference on the domain. All of these works use specialized domain representations or algorithms that are specialized or unstandardized within the planning community. In [28], the authors do use a standardized extension of STRIPS called UWL as their planning domain; however, they use a specialized graph representation and algorithms to check observations for consistency with plans.

Some more recent effort in the community has been put towards the idea of plan recognition as planning. In [40], Ramirez and Geffner argue that plan recognition is essentially planning in reverse. Whereas a planning problem seeks to find actions to achieve a specified goal, in plan recognition we seek to find the goal that best explains a sequence of actions. In later work, the authors extended their previous work to utilize off-the-shelf classical PDDL planners for plan recognition [41]. This work is a notable breakthrough in that it enables plan recognition to leverage decades of work from the planning community. Great effort has been put towards standardizing and optimizing the PDDL language and PDDL planners, and to the knowledge of this author, no such standardization exists for plan recognition.

While the work of Ramirez and Geffner exhibited the viability of plan recognition as planning, their recognition encoding does not leverage the structure of the problem, specifically when considering a human's interaction with devices. TCCAs

allow the recognition subset of the encoded domain to be compactly represented in a single Timed Constraint Automaton. Also, the authors did not provide a method for handling time. The work in this chapter allows for plan recognition with time.

3.8 Approach to Plan Recognition

In this section we present our approach to plan recognition using TCCAs. As we described in Section 3.3, we leverage the idea that plan recognition is planning in reverse. We will encode the plan recognition problem as a planning problem, and then solve it with a black-box TCCA planner. The problem is encoded by mapping control variables that appear in the observations into state variables in a new automaton, referred to as a *recognizing automaton.* The start time and end times on observations are encoded in our new automaton to force the planner to hold each control variable to the same value and same duration as in each observation.

We begin this section by describing the planning problem and defining the black-box TCCA planner. We then walk through encoding the example of the occupant making coffee. Then in Sections 3.8.2-3.8.6 we formally describe the algorithms for performing the encoding. For ease of exposition, in these sections we make the assumption that observations are non-overlapping. That is to say, for pairs of adjacent observations o_i and o_{i+1}, we require that $t_{E,i} \leq t_{S,i+1}$. In Section 3.8.7 we relax this assumption and give an overview of how to extend our method to allow for concurrent observations.

3.8.1 TCCA Planner as a Black Box

In order to describe how to perform plan recognition using planning, we must define the notion of a planning problem. We define the problem formally as follows:

Definition 8. *The **planning problem** over TCCAs is given by the tuple* $\langle \mathcal{M}, I, g \rangle$ *where:*

- $\mathcal{M}$ *is a TCCA,*

- *I is the initial state of the TCCA, given as a full assignment to the mode variables of $\mathcal{M}$, and*

- *g is the goal of the planning problem, encoded as a QSP.*

A solution to the planning problem is a QCP that satisfies the input goal QSP g. That is, all legal executions of the QCP from the initial state results in all episodes in g being observed.

A QSP is the same as the QCP presented in Section 3.6, except that constraints on episodes are over state variables of the the TCCA, not control variables. Formally, we define it as:

Definition 9. *A **Qualitative State Plan (QSP)** is a tuple $\langle \mathcal{E}, \mathcal{A}, \mathcal{T} \rangle$ where:*

- *$\mathcal{E}$ is a set of events. Events are plan elements that can be assigned to a specific point in time.*

- *$\mathcal{A}$ is a set of episodes where each episode $a \in \mathcal{A}$ is itself a tuple $\langle e_S, e_E, (x_i^c = v) \rangle$. An episode is represented as a directed edge between events with constraints on state. Episodes have a start event e_S, an end event e_E, and a state constraint that must be true over the duration of the episode, given as an assignment of v to the state variable x_i^m.*

- *$\mathcal{T}$ is a set of temporal constraints represented as a simple temporal network (STN), which specifies lower and upper bounds on the duration between events.*

Let TCCA-PLAN denote a TCCA planner that solves the planning problem. It takes as input a TCCA, the initial state, and a goal, expressed as a QSP. It outputs True if a satisfying control plan, or False otherwise. We define the following black box that will be used when checking plans for consistency.

Algorithm 3 TCCA-PLAN

function TCCA-PLAN($\mathcal{M}, I, g$)

1: Solve planning problem on inputs $\langle \mathcal{M}, I, g \rangle$
2: **if** Satisfying QCP C exists **then**
3: **return** TRUE
4: **else**
5: **return** FALSE
6: **end if**

As our candidate planner inspiring this work, we use a TCCA planner called tBurton developed within the Model-based Embedded Robotic Systems (MERS) group at MIT. tBurton is based off of the work described in [7] on CCAs, extended to plan over time-evolved goals. It is a temporal generative planner that leverages techniques of causal-graph decomposition to quickly reason over concurrently operating timed automaton. The key to its efficiency and scalability is a divide-and-conquer approach that combines constraint decomposition and causal order decomposition to generate a least-commitment plan. Constraint decomposition verifies the constraints encoded in each automaton, and if necessary, reorganizes them into other automaton more amenable to the planning task. Causal order decomposition identifies natural hierarchy within the automata and groups cyclically dependent automata.

3.8.2 Outer Loop: Checking Each Plan for Consistency

We begin by describing the outer loop to the algorithm. Recall that the plan recognition problem takes as input a set of all possible candidate plans $\mathcal{A}_P$ that the occupant could be following. In this loop, we iterate over all plans, checking if each is consistent with the observations.

Algorithm 4 Check each plan for consistency

function CHECK-PLANS($\mathcal{A}_D, \mathcal{A}_P, I, \mathbf{O}, G$)

1: *consistent-plans* $\leftarrow \{\}$
2: **for all** $P \in \mathcal{A}_P$ **do**
3: $A^R \leftarrow$ GET-RECOGNIZING-TCA($P, \mathcal{A}_D, \mathbf{O}$)
4: $\mathcal{M} \leftarrow$ MAKE-TCCA($P, \mathcal{A}_D, A^R$)
5: $g \leftarrow$ GET-RECOGNIZING-QSP($G(P), \mathbf{O}$)
6: *p-consistent* $\leftarrow$ TCCA-PLAN($\mathcal{M}, I, g$)
7: **if** *p-consistent* **then**
8: Add P to *consistent-plans*
9: **end if**
10: **end for**
11: **return** *consistent-plans*

The outer loop is described in Algorithm 4. For each plan $P \in \mathcal{A}_P$, the algorithm start by constructing the recognizing automaton (Line 3). This step is described in Section 3.8.4. It then constructs a TCCA by connecting together the plan automaton P, the device automatons $\mathcal{A}_D$, and the recognizing automaton A^R that was just constructed (Line 4). This step is described in Section 3.8.5. The algorithm finishes the encoding in Line 5 by encoding the temporal information of the observations into a QSP and combining this with the final goal of the plan P. This step is described in Section 3.8.6. The encoded TCCA $\mathcal{M}$ and QSP are then input into the black-box planner on Line 6 to check if P is consistent. If so, it is added to a list of consistent plans (Line 8).

In the next section we walk through the execution of this algorithm on the example problem, before describing the technical details in the following sections.

3.8.3 Walk-Through of Plan Recognition

In this section we walk through the process of encoding our example plan recognition problem of drinking coffee into a planning problem. Throughout the section we will

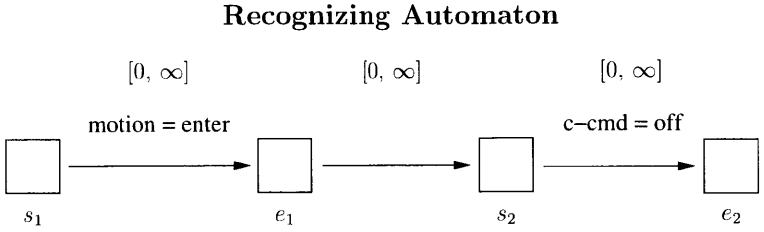


Figure 3-3: A recognizing automaton encoding two observations on an example scenario of making coffee.

reference lines in Algorithm 4. We suppose our two components in $\mathcal{A}_D$ are the coffee machine and the occupancy of the kitchen. In our library of plans $\mathcal{A}_P$ we have two plans, one for leisurely coffee and one for rushed coffee, each with the goal ($G(P)$) of finishing the coffee. Suppose we observe the kitchen being entered at 8am, and the coffee machine being turned off at 8:12am. We will walk through the encoding for the rushed coffee plan, but the other plan is similar.

In Line 3 we construct an automaton that encodes the durations and values of each observation. Each observation has a start state and an end state corresponding to the start and end time of the observation (Figure 3-3). In our case, the observations are instantaneous, so we place a lower time bound of 0 on the transition of each observation. We place no state or temporal constraints on transitions between the end of one observation and the start of another, so as to allow any missing observations to be filled in by the planner.

In Line 4 we connect the automaton in Figure 3-2 with the recognizing automaton in Figure 3-3 to construct the TCCA that is input into the planner (Figure 3-4). The ouput variables (*coffee-state* and *kitchen-state*) of the device automata are connected to the corresponding input variables of the plan automaton as in Figure 3-2. Also, the control variables (*c-cmd* and *motion*) of the device automata are connected to the input variables of the recognizing automaton so that the automaton ensures that all observations are explained in the output control plan of the planner.

In Line 5 we construct the QSP that will be given to the planner. This QSP encodes the absolute start and end times for each observation. As can be seen in Figure 3-5, we include one sequence of episodes for each observation. The first event

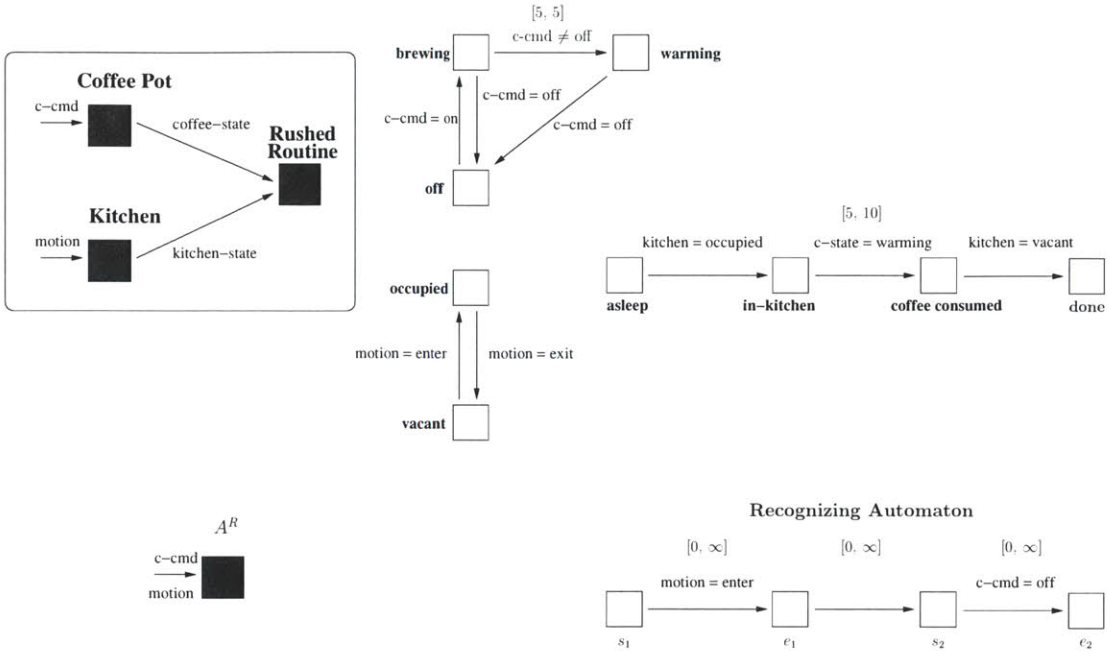


Figure 3-4: The TCCA created from combining the plan, device, and recognizing automatons for an example scenario of making coffee.

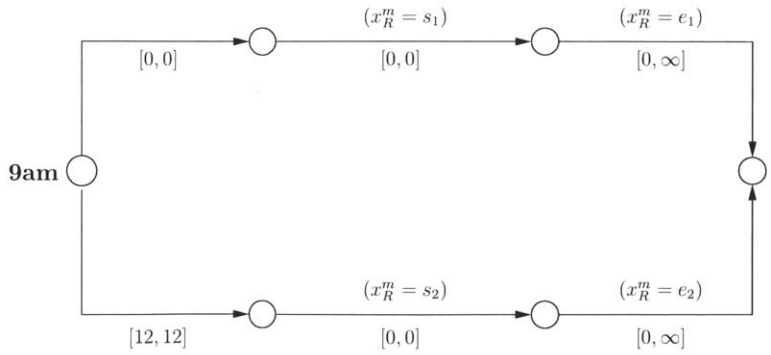


Figure 3-5: A recognizing QSP encoding two observations in an example scenario of making coffee.

E_s in each sequence represents the start time of the observation, and the second event E_e represents the end time. On each episode, we place state constraints that ensure that the recognizing automaton follows a state trajectory through each state s_i and e_i.

3.8.4 Encoding the Observations into a Recognizing Automaton

Next we present the construction of the recognizing automaton. The recognizing automaton is our tool for transforming the control variables in the observations into state variables that can be planned over by the planner TCCA-PLAN. For each observation, the automaton will include one state that represents the start to the observation, and another that represents the end of the observation. Transitions between the start and the end states of an observation are given temporal constraints that require the value of the control variable observed to be held for the duration of time the observation was observed.

Algorithm 5 Create recognizing automaton

function GET-RECOGNIZING-TCA($P, \mathcal{A}_D, \mathbf{O}$)

1: Create state variable x_R^m of the automaton
2: $N \leftarrow$ length($\mathbf{O}$)
3: Define domain variable $D(x_R^m) = \{s_1, \ldots, s_N, e_1, \ldots, e_N\}$
4: $X^u \leftarrow$ Set of observable controls variables of $\mathcal{A}_D$
5: $X^R \leftarrow x_R^m \cup X^u$
6: $\mathcal{Q}^R \leftarrow \{\}$
7: Initialize transition function δ^R
8: **for all** $\langle (x_{k_i}, v), t_{S,i}, t_{E,i} \rangle \in \mathbf{O}$ **do**
9: $\quad \delta^R \leftarrow$ ADD-TRANSITION(s_i, e_i, $(x_{k_i} = v)$, Δt_{o_i}, ∞)
10: $\quad \delta^R \leftarrow$ ADD-TRANSITION(e_i, s_{i+1}, TRUE, 0, ∞)
11: **end for**
12: **return** $\langle X^R, \delta^R, \mathcal{Q}^R \rangle$

The construction of the recognizing automaton is presented in Algorithm 5. We start by letting x_R^m be the state variable of the automaton. The domain of x_R^m consists of two states for every observation, a start s_i and an end e_i. In Line 4, we collect all of the variables of the control variables that are observable. The variables of the recognizing automaton are this set X^u together with the state variable x_R^m (Line 5). We then iterate over the observations $o_i = \langle (x_{k_i}, v), t_{S,i}, t_{E,i} \rangle$. For each observation we add two transitions. In Line 9 we add a transition from the start of the observation s_i to the end of the observation e_i, with a guard requiring that the control variable x_{k_i} be held to the value v observed for at least the duration of the observation $t_{E,i} - t_{S,i}$. In Line 10 we add a transition from the end of the observation e_i to the start of the next observation s_{i+1}. We place no guards on this transition to allow for any value on control variables in between observations. After all transitions are added, the construction of the recognizing automaton is complete.

3.8.5 Connecting the Automata into a TCCA

In this section we describe how the TCCA is constructed by connecting together the plan automaton P, the device automatons $\mathcal{A}_D$, and the recognizing automaton A^R.

Algorithm 6 Combine automata together to create the encoded TCCA

function MAKE-TCCA($p, \mathcal{A}_D, A^R$)

1: $x_R^m \leftarrow$ State variable of A^R

2: $x_p^m \leftarrow$ State variable of p

3: $\mathcal{X}_D^m \leftarrow$ Set of state variables in $\mathcal{A}_D$

4: $\mathcal{X}^R \leftarrow \mathcal{X}_{A_D}^m \cup \{x_R^m, x_p^m\}$

5: $\mathcal{X}^{R,c} \leftarrow$ Set of control variables of $\mathcal{A}_D$

6: $\mathcal{A}^R \leftarrow \mathcal{A}_D \cup \{p, A^R\}$

7: Initialize set of interconnection constraints $\mathcal{I}^R$

8: **for all** Control variables x^c **do**

9: $\quad \mathcal{I}^R \leftarrow$ Add connection between x^c in $\mathcal{A}_D$ and x^c in A^R

10: **end for**

11: **for all** Output variables x^y in $\mathcal{A}_D$ **do**

12: $\quad \mathcal{I}^R \leftarrow$ Add connection between x^y in $\mathcal{A}_D$ and corresponding input x^c in p

13: **end for**

14: **return** $\langle \mathcal{X}^R, \mathcal{X}^{R,c}, \mathcal{A}^R, \mathcal{I}^R \rangle$

Algorithm 6 describes this process of how the automaton in our encoding are connected together into a TCCA. In Lines 1-4, we collect together all of the state variables of the component automata. In Line 5, we specify that the control variables of the TCCA are the set of all control variables of the device automata in $\mathcal{A}_D$. Then we collect together all of the component automata of the TCCA (Line 6. Then in Lines 8-13 we define how variables of the component automata are connected together. Every control variable in the device automata must be connected to the corresponding control variable in the recognizing automaton (Line 8). Also, every output variable of the device automata must be connected to the corresponding input in the occupant's plan automata p (Line 11).

3.8.6 Encoding the Goal and Observations into a Recognizing QSP

The recognizing automaton constructed in Section 3.8.4 is only enough to ensure that the planner explains the duration of each observation, but it is not capable of enforcing constraints on specific points in time. That is, it can ensure that a control variable will be held for 5 seconds, but it cannot ensure that it is held for 5 seconds starting at 8am. The recognizing QSP presented in this section provides these complementary constraints to the recognizing automaton.

Algorithm 7 Create recognizing QSP

function GET-RECOGNIZING-QSP(g, $\mathbf{O}$)

1: $\mathcal{E} \leftarrow \{E_0, E_N\}$
2: $\mathcal{A} \leftarrow \{\}$
3: $\mathcal{T} \leftarrow \{\}$
4: **for all** $\langle (x_{k_i}, v), t_{S,i}, t_{E,i} \rangle \in \mathbf{O}$ **do**
5: $\quad \mathcal{E} \leftarrow \mathcal{E} \cup \{E_{s,i}, E_{e,i}\}$
6: $\quad \mathcal{A}, \mathcal{T} \leftarrow$ ADD-EPISODE(E_0, $E_{s,i}$, TRUE, $[t_{S,i}, t_{S,i}]$)
7: $\quad \mathcal{A}, \mathcal{T} \leftarrow$ ADD-EPISODE($E_{s,i}$, $E_{e,i}$, $(x_R^m = s_i)$, $[\Delta t_{o_i}, \Delta t_{o_i}]$)
8: $\quad \mathcal{A}, \mathcal{T} \leftarrow$ ADD-EPISODE($E_{e,i}$, E_N, $(x_R^m = e_i)$, $[0, \infty]$)
9: **end for**
10: With $g = \langle (x^m, v), t_{S,g}, t_{E,g} \rangle$:
11: $\mathcal{E} \leftarrow \mathcal{E} \cup \{E_{s,g}, E_{e,g}\}$
12: $\mathcal{A}, \mathcal{T} \leftarrow$ ADD-EPISODE(E_0, $E_{s,g}$, TRUE, $[t_{S,g}, t_{S,g}]$)
13: $\mathcal{A}, \mathcal{T} \leftarrow$ ADD-EPISODE($E_{s,g}$, $E_{e,g}$, $(x^m = v)$, $[\Delta t_g, \Delta t_g]$)
14: **return** $\langle \mathcal{E}, \mathcal{A}, \mathcal{T} \rangle$

Algorithm 7 describes the construction of the recognizing QSP. It is constructed by first adding episodes for each observation that enforce the absolute times on the observation (Lines 4-8). First, we create two events E_0 and E_N that will serve as the start and end events of the QSP (Line 1). E_0 corresponds to time $t = 0$. For each

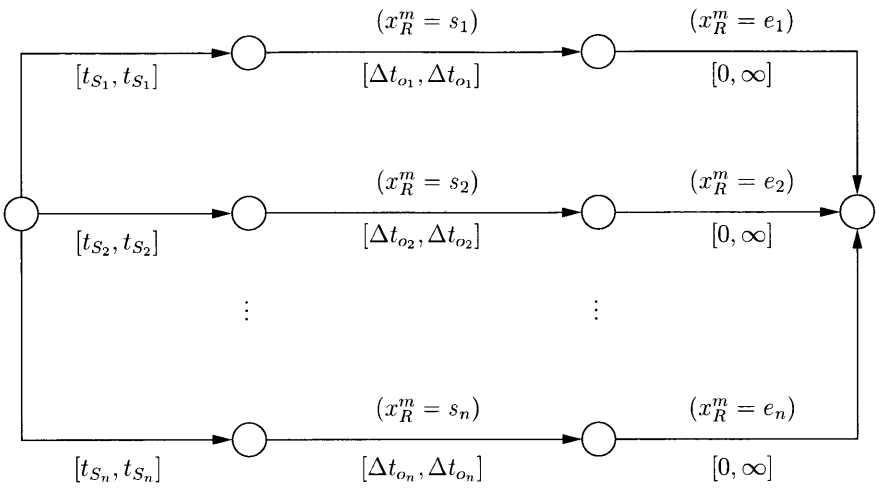


Figure 3-6: An encoding of observations into a recognizing QSP.

observation $o_i = \langle (x_{k_i}, v), t_{S,i}, t_{E,i} \rangle$, we first add an episode from E_0 to $E_{s,i}$ (Line 6). This episode is given a temporal constraint of $[t_{S,i},\ t_{S,i}]$ to ensure that the control value v of observation o_i will start at time $t_{S,i}$ in the QCP output by the planner. Then an episode is added in Line 7 that ensures that v will be held for the proper duration. By construction of the recognizing automaton, A^R can only transition to state e_i if the value v is held for the correct duration. The last episode added in Line 8 ensures that the state e_i is reached. In Figure 3-6 we depict this portion of the recognizing QSP encoding the observations. Then in Lines 10-13 we add the end goal of the occupant's plan so that the TCCA planner will generate a control plan that reaches this goal.

3.8.7 Extending to include concurrent observations

For clarity of exposition, in the previous section we assume that observations are non-overlapping. However, we can easily extend this technique to concurrent observations by including a recognizing automaton for each input variable $x \in \mathcal{X}^c$. In this section we present the intuition behind including concurrent observations.

In the previous section we were limited by the use of a single automaton for recognizing observations. The state trajectory of the recognizing automaton followed a linear path; each observation must be recognized before recognizing the next obser-

vation.

This limitation can be overcome by using a separate recognizing automaton for each control variable $x^c \in \mathcal{X}^c$. The recognizing automaton for the variable x^c is responsible for ensuring that the planner generates a plan that recognizes all observations of x^c in $\mathcal{O}$. The ability to handle concurrent observations so seamlessly is one of the most useful features of our choice of TCCAs as our planning domain representation.

3.9 Summary of Insights into Autonomous Control

In this chapter we presented the concept for a fully integrated autonomous control system for the sustainable control of homes. In our discussion we described a vision for a unifying representation for human interaction with a house. A key insight is that the behavior of each component of the system should be modeled in a language natural to that component. The languages should then be compiled into one unifying representation, over which plan recognition could be performed.

We also provided the concept design for plan recognition over TCCAs in which we reformulate a plan recognition problem as a planning problem to be solved by a planner. This design provides the foundation for plan recognition over a system with devices interacting with humans.

Chapter 4

Conclusions

This chapter begins by presenting several ideas for future extensions of the research presented in this thesis necessary for the adoption of the technology. We then conclude our discussion and discuss our contributions in Section 4.2.

4.1 Future Work

4.1.1 Handling Postponed Episodes

When performing receding horizon control over flexible temporal constraints in a CCQSP, often a planning horizon will end between the temporal bounds, and we must decide whether the episode should be executed in the current planning horizon, or postponed to the next. In Section 2.4.2, we described our method for handling this problem by adding a penalty to the objective function for postponed episodes. We argued that this penalty was reasonable because there is no guarantee the episode will be still be feasible if postponed. However, this may not be the optimal solution. Future work is needed to determine the best approach for handling temporally flexible episodes across planning horizons.

4.1.2 Modeling Uncertainty in Occupant Behavior

In Chapter 2, we assumed that the start and end times of episodes in an occupant's schedule are controllable. However, this does not hold in general. We might not know when the occupant arrives or leaves, and must explicitly consider this uncertainty in our model. In future work, we could model the start and end times as distributions. The p-Sulu algorithms must be modified to plan according to these distributions. In particular, equation (2.3) must be reformulated to include chance constraints. In [30], the authors use stochastic occupancy models in robust model predictive control of homes. In particular, the authors assume the occupancy of a building is determined by Markov chains, and formulate the robust MPC problem around this assumption. One could draw ideas from this work when approaching the occupancy problem under our problem formulation.

4.1.3 Expand Analysis of Energy Savings

In Chapter 2, we described p-Sulu and its application to the Connected Sustainable Home. Preliminary results showed the great impact such a controller could have on efficient use of energy in the home, producing energy savings of 42.8% in the winter, 15.3% in the spring, 16.8% in the summer, and 4.4% in the autumn. However, these results were taken over only a single scenario. To illustrate the true savings, we must analyze the results over a wide variety of realistic scenarios for typical residents of a house.

Also, we only analyzed the performance on the single prototype house. To encourage widespread use of the technology, we must demonstrate its benefit on a wide variety of homes, especially older buildings that aren't outfitted with the building technologies present in the Connected Sustainable Home.

4.1.4 Energy Disaggregation using Hybrid Mode Estimation

In Section 3.1, we set forth a vision for seamlessly integrating an autonomous home control system into a resident's daily life. In this discussion, we described a sensing

framework in which states, or *modes*, of an appliance are sensed through *energy disaggregation*. Energy disaggregation is the problem of taking a whole home energy signal and decomposing it into its component appliances. Previous authors have approached this problem using a variety of techniques. Early work simply looked at jumps in the energy signal to classify transitions in appliance state [20]. More recent work leverages additive factorial HMMs to perform energy disaggregation in an unsupervised setting [25]. See [48] for a recent survey of energy disaggregation techniques.

Through the course of this author's investigations, he experimented with applying a technique called *hybrid mode estimation* to the problem of energy disaggregation. In this approach, an appliance is modeled as a probabilistic hybrid automaton (PHA)[22]. A PHA is a representation of a device who's state can be represented by a combination of discrete states, or modes, and continuous variables. When in a particular state, the dynamics of the continuous variables are described by a set of linear dynamical systems. Hybrid mode estimation is the technique of estimating the mode, or state, of a PHA. Multiple PHAs that are interacting and are jointly observable can be represented as concurrent PHAs (cPHA). Multiple appliances on a circuit can be modeled as a cPHA where the aggregate power signal is the only observable. Previous research has already explored techniques for estimation of the states of concurrently operating automata (i.e. the operating state of an appliance)[22][4]. Preliminary experiments on a small subset of 6 appliances showed that each appliance could be well modeled as a probabilistic hybrid automaton. Applying hybrid mode estimation predicted the operating modes of single appliances with near perfect accuracy, although the findings were not pursued with enough rigor to present in this thesis. Although these experiments were not rigorous, they showed promise that concurrent hybrid mode estimation is a promising technique for energy disaggregation, especially when estimating the operating states of only a small number of appliances. This should be a topic of future research.

Although this problem is often framed as looking at the aggregate energy signal of the home, the level of fidelity required in plan recognition may require a more robust

sensing framework. For greater sensing granularity, one could consider applying the techniques at the circuit level, or even the outlet level. Individual energy monitors on each outlet are becoming more common in the growing web-of-things movement [17][34], so this level of sensing is not unreasonable. In [18], the authors introduce an API for interfacing with wireless smart outlets that would be useful for implementing this level of sensing granularity.

4.1.5 Probabilistic Plan Recognition

Currently our plan recognition algorithm only checks if the observations are consistent with a given plan, and if so, outputs a satisfying plan. Our algorithm allows for missing observations, but it is not forgiving of noisy observations. If an incorrect input is observed, a given plan may be declared inconsistent, even if it is in fact the correct goal.

Also, it may be the case that the observations are consistent with multiple plans. We need a mechanism to differentiate between the multiple consistent plans to find the best candidate.

One solution to these problems is to introduce probability into the model and rank the plans $P \in \mathcal{A}_P$ based on their probability $\Pr(P \mid \mathcal{O})$ conditioned on the observations. Many prior estimation techniques on HMMs [42], PHCAs [46], or POMDPs [42] could offer valuable insight into this problem of probabilistic plan recognition.

4.1.6 Model Learning

One of the biggest hinderances to performing plan recognition in the home is creating the model that describes the resident's behavior. As discussed in Chapter 1, humans can behave drastically different from one another. So the problem isn't to create one model that governs the entire system. Rather, models must be created on an individual basis. Requiring each individual user to describe his or her behavior is completely infeasible, especially in a modeling language completely foreign to the user. Thus, we need an algorithm that can autonomously learn models describing the

resident's behavior, with little or no interaction required from the user. Much prior work, such as [10], has focussed on the problem of learning and could offer useful insight into this problem.

4.2 Conclusions

In this thesis we presented an approach to autonomous control of buildings, and applied it to a prototype called the Connected Sustainable Home. Our goals were threefold: to provide sustainable, comfortable, and convenient living to the occupant of a building.

The first two goals were accomplished in Chapter 2 by our application of the risk-sensitive goal-directed executive *probabilistic Sulu* (*p-Sulu*) to the domain of building control. The first goal of sustainability is achieved through goal-directed optimal planning. We showed the contribution of p-Sulu to sustainable living in two ways. First, p-Sulu's explicitly considers the stochastic plant model of the home and the goal-directed representation of a resident's schedule (CCQSP), and is able to plan and control over the model and schedule in a near-optimal manner. We saw energy savings as high as 42.8% in the winter compared to a traditional PID controller, with savings of 15.3%, 16.3%, and 4.4% in the other three seasons. The second contribution comes from the ability to leverage flexibility in a resident's schedule. Comparing fixed and flexible schedules on an example scenario, flexible scheduling yielded energy savings of 10.4% in the winter. Savings in the other three seasons were less significant: 1.6%, 1.6%, and 0.7% in spring, summer, and autumn respectively.

Our executive p-Sulu contributes to the comfortable living of the resident through it's risk-sensitive control approach using chance constraints. Since the outdoor environment surrounding the home is uncertain, we assume a risk that the resident's comfort requirements may be violated if the weather deviates from the forecast. p-Sulu allows users to specify a bound on the risk they are willing to take that their comfort requirements are violated, and it executes a control sequence that achieves this level of risk. To evaluate its performance, we compared p-Sulu to its deterministic

counterpart Sulu, analyzing the fraction of time that the two controllers violate the resident's temperature constraints. We saw that Sulu violated constraints 30.88% of the time, while p-Sulu violated constraints on only one trial. This feature is critical to the adoption of the technology. If an autonomous building controller were to produce an uncomfortable living environment 30% of the time, the user would likely abandon the technology.

p-Sulu requires knowledge of the resident's activity schedule to maximize energy savings. Forcing a user to input his or her schedule would place undue burden on the resident and would likely be a barrier to adoption of the technology. To make the control system convenient for the resident, we must automatically predict their behavior. In Chapter 3 we presented a concept design that contributes towards achieving the third goal of convenience. We presented a method for plan recognition that would be able to predict the plan of a resident by observing him or her. The technique uses timed concurrent constraint automata (TCCA) to represent components in a house the resident may interact with, and predicts the resident's behavior based on these interactions. We leverage a novel encoding of the plan recognition problem as a planning problem, to be solved by a TCCA planner.

Bibliography

[1] K.H. Ang, G. Chong, and Y. Li. Pid control system analysis, design, and technology. *Control Systems Technology, IEEE Transactions on*, 13(4):559–576, 2005.

[2] R. Benton. The impact of thermostat performance on energy consumption and occupant comfort in residential electric heating systems. *ASHRAE Transactions*, 88, 1982.

[3] Lars Blackmore. *Robust Execution for Stochastic Hybrid Systems.* PhD thesis, Massachusetts Institute of Technology, 2007.

[4] Lars Blackmore, Stanislav Funiak, and Brian C. Williams. A combined stochastic and greedy hybrid estimation capability for concurrent hybrid models with autonomous mode transitions. *Robotics and Autonomous Systems*, 56(2):105 – 129, 2008.

[5] J.E. Braun. Reducing energy costs and peak electrical demand through optimal control of building thermal storage. *ASHRAE Transactions*, 96(2):876–888, 1990.

[6] H.H. Bui. A general model for online probabilistic plan recognition. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 1309–1318. IJCAI, 2003.

[7] S.H. Chung. *Model-based planning through constraint and causal order decomposition.* PhD thesis, Massachusetts Institute of Technology, 2008.

[8] REN21 Steering Committee et al. Renewable 2010, global status report. *World watch Institute Washington, DC*, 2010.

[9] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1-3):61–95, 1991.

[10] S. Dong. Unsupervised learning and recognition of physical activity plans. Master's thesis, Massachusetts Institute of Technology, 2007.

[11] S. Edelkamp and M. Helmert. Exhibiting knowledge in planning problems to minimize state encoding length. *Recent Advances in AI Planning*, pages 135–147, 2000.

[12] S. Elrod, G. Hall, R. Costanza, M. Dixon, and J. Des Rivieres. The responsive environment: Using ubiquitous computing for office comfort and energy management. Technical report, Technical Report CSL-93-5, Xerox Palo Alto Research Center, Tech. Rep, 1993.

[13] Hunter Fan. Hunter 44360 set and save programmable thermostat. `http://www.hunterfan.com/Products/Thermostats/Set-and-Save-7-Day-Programmable-44360/`, February 2012.

[14] M. Feldmeier and J.A. Paradiso. Personalized hvac control system. In *Internet of Things (IOT), 2010*, pages 1–8. IEEE, 2010.

[15] C.W. Geib and R.P. Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173(11):1101–1132, 2009.

[16] S. Gil. Robust learning of probabilistic hybrid models. Master's thesis, Massachusetts Institute of Technology, 2008.

[17] Dominique Guinard, Vlad Trifa, and Erik Wilde. A resource oriented architecture for the web of things. In *Proceedings of Internet of Things 2010 International Conference (IoT 2010)*, Tokyo, Japan, November 2010.

[18] Dominique Guinard, Markus Weiss, and Vlad Trifa. Are you energy-efficient? sense it on the web! In *Adjunct Proceedings of Pervasive 2009 (International Conference on Pervasive Computing)*, Nara, Japan, May 2009.

[19] M. Gwerder and J. Toedtli. Predictive control for integrated room automation. In *8th REHVA World Congress Clima*, 2005.

[20] G.W. Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, 1992.

[21] G.P. Henze, D.E. Kalz, S. Liu, and C. Felsmann. Experimental analysis of model-based predictive optimal control for active and passive building thermal storage inventory. *HVAC&R Research*, 11(2):189–213, 2005.

[22] Michael Hofbaur and Brian Williams. Mode estimation of probabilistic hybrid systems. In Claire Tomlin and Mark Greenstreet, editors, *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 81–91. Springer Berlin / Heidelberg, 2002.

[23] A. Hofmann and B. Williams. Robust execution of temporally flexible plans for bipedal walking devices. In *Proceedings of ICAPS*, 2006.

[24] F.P. Incropera, T.L. Bergman, A.S. Lavine, and D.P. DeWitt. *Fundamentals of Heat and Mass Transfer*. John Wiley & Sons, 2011.

[25] J. Zico Kolter and Tommi Jaakkola. Approximation inference in additive factorial hmms with application to energy disaggregation. Preprint, To appear in Proceedings of the American Control Conference, 2012.

[26] Thomas Léauté. Coordinating agile systems through the model-based execution of temporal plans. Master's thesis, Massachusetts Institute of Technology, 2005.

[27] Thomas Léauté and Brian. C. Williams. Coordinating agile systems through the model-based execution of temporal plans. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*, 2005.

[28] N. Lesh and O. Etzioni. A sound and fast goal recognizer. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1704–1710, 1995.

[29] Y. Ma, F. Borrelli, B. Hencey, A. Packard, and S. Bortoff. Model predictive control of thermal energy storage in building cooling systems. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 392–397. IEEE, 2009.

[30] A.E.D. Mady, G.M. Provan, C. Ryan, and K.N. Brown. Stochastic model predictive controller for the integration of building use and temperature regulation. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[31] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 2000.

[32] MIT Mobile Experience Lab. Sustainable connected home brochure. Available on-line at `http://mobile.mit.edu/fbk_wp-uploads/2010/01/connected_home_brochure.pdf`. Retrieved on February 20, 2012.

[33] William J. Mitchell and Federico Casalegno. *Connected Sustainable Cities.* MIT Mobile Experience Lab Publishing, 2008.

[34] Web of Things. Web of things: architecting a web of things for tinkers and hackers. `http://www.webofthings.org/`, February 2012.

[35] F. Oldewurtel, A. Parisio, C.N. Jones, M. Morari, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and K. Wirth. Energy efficient building climate control using stochastic model predictive control and weather predictions. In *American Control Conference (ACC), 2010*, pages 5100–5105. IEEE, 2010.

[36] Masahiro Ono. Energy-efficient control of a smart grid with sustainable homes based on distributing risk. Master's thesis, Massachusetts Institute of Technology, 2012.

[37] Masahiro Ono. *Robust, Goal-directed Plan Execution with Bounded Risk.* PhD thesis, Massachusetts Institute of Technology, 2012.

[38] Masahiro Ono and Brian C. Williams. An efficient motion planning algorithm for stochastic dynamic systems with constraints on probability of failure. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-08)*, 2008.

[39] D.J. Patterson, D. Fox, H. Kautz, and M. Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 44–51. IEEE, 2005.

[40] M. Ramırez and H. Geffner. Plan recognition as planning. In *Proc. Int. Joint Conf. on AI*, pages 1778–1783, 2009.

[41] M. Ramırez and H. Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *National Conference on Artificial Intelligence (AAAI2010)*, 2010.

[42] S.J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice hall, third edition, 2010.

[43] CF Schmidt, NS Sridharan, and JL Goodson. The plan recognition problem. *Artificial Intelligence*, 11(2), 1978.

[44] O. Schultz, A. Mette, R. Preu, and SW Glunz. Silicon solar cells with screen-printed front side metallization exceeding 19% efficiency. In *22nd European Photovoltaic Solar Energy Conference and Exhibition*, 2007.

[45] U.S. Energy Information Administration. Annual energy outlook 2010, 2010.

[46] B.C. Williams, S. Chung, and V. Gupta. Mode estimation of model-based programs: monitoring systems with complex behavior. In *IJCAI*, volume 17, pages 579–590, 2001.

[47] B.C. Williams, M.D. Ingham, S.H. Chung, and P.H. Elliott. Model-based programming of intelligent embedded systems and robotic space explorers. *Proceedings of the IEEE*, 91(1):212–237, 2003.

[48] M. Zeifman and K. Roth. Nonintrusive appliance load monitoring: Review and outlook. *Consumer Electronics, IEEE Transactions on*, 57(1):76–84, 2011.

[49] ETH Zurich. OptiControl project. `http://www.opticontrol.ethz.ch/`, February 2012.