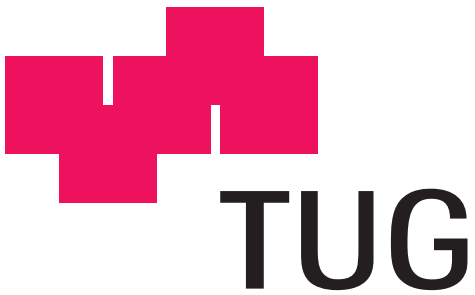
Dissertation

Control of Multi Component Hybrid Systems through Qualitative Pre-Selection

Wolfgang Kleißl



Institut für Regelungs- und Automatisierungstechnik
Technische Universität Graz

Fakultät für Elektrotechnik und Informationstechnik
Technische Universität Graz

Betreuer: O. Univ.-Prof. Dipl.-Ing. Dr. techn. Nicolaos Dourdoumas

Graz, im November 2006

Abstract

Many technical systems cannot be adequately captured by a single continuous model, but one uses several such models to capture the system's behavior at various modes of operation. A discrete automaton is used to model the transitions among these modes. Control of such hybrid systems is difficult as continuously valued and discretely valued variables have to be considered simultaneously.

This work deals with on-line control of a class of multi-component hybrid models. It proposes to split-up the hybrid control task into two simpler tasks: First a discretely valued abstraction of the hybrid model is utilized to efficiently pre-select a few promising sequences of operational modes by discrete search methods. Second, with the pre-selected mode-sequences specified the hybrid model can be regarded as time-variant continuous model. With this, the inputs that are applied to the system can be determined numerically.

To perform the qualitative pre-selection efficiently, a novel type of qualitative model is introduced. This is an abstraction of each of the hybrid model's components that can automatically be pre-compiled off-line. The special structure of the qualitative model allows straightforward simultaneous operation on the individual component models. Additionally, this structure can be related to the structure of the hybrid model to increase pre-selection efficiency.

However, qualitative models generally have the drawback that they also model qualitative behaviors which are not abstractions of any behavior of the hybrid model. If such a behavior is pre-selected, subsequently no valid numerical solution can be found. So a careful interplay between qualitative pre-selection and numerical deduction of the system inputs is introduced that helps to quickly focus onto a valid (sub optimal) solution to the hybrid control task.

Kurzfassung

Viele technische Systeme lassen sich nicht ausreichend gut durch ein einzelnes kontinuierliches Modell beschreiben, sondern man verwendet mehrere solcher Modelle um das Verhalten des Systems in seinen verschiedenen Betriebszuständen zu beschreiben. Ein diskreter Automat modelliert dabei die Übergänge zwischen diesen Betriebszuständen. Die Regelung solcher hybrider Systeme gestaltet sich schwierig, da kontinuierliche und diskretwertige Variablen simultan beachtet werden müssen.

Diese Arbeit behandelt die online-Regelung einer Klasse solcher Systeme die jede Komponente eines aus mehreren Komponenten aufgebauten Systems als hybriden Automaten modelliert. Es wird vorgeschlagen, die hybride Regelungsaufgabe in zwei einfachere Teile aufzuspalten: Zuerst wird eine rein diskretwertige Abstraktion des hybriden Modells herangezogen, um effizient durch diskrete Suchmethoden wenige günstige Abfolgen von Betriebszuständen vorzuselektieren. Durch Festlegung dieser Betriebszustände kann das hybride Modell als zeitvariables kontinuierliches Modell betrachtet werden. Damit können die Eingangsgrößen, die auf das System aufgeschaltet werden sollen, rein numerisch ermittelt werden.

Um diese Vorselektion effizient durchführen zu können, wird eine neue Art eines qualitativen Modells vorgestellt. Dies ist eine komponentenweise Abstraktion des hybriden Modells, die automatisch vorab kompiliert werden kann. Die spezielle Struktur des Modells erlaubt bei der qualitativen Vorselektion, die einzelnen Komponentenmodelle einfach simultan zu betrachten. Zusätzlich kann diese Struktur an die Struktur des hybriden Modells angelehnt werden, um noch effizienter vorselektieren zu können.

Qualitative Modelle haben im Allgemeinen aber den Nachteil, dass sie auch qualitative Verhalten modellieren, die keine Abstraktion eines Verhaltens des hybriden Modells sind. Wenn die qualitative Vorselektion auf so einem Verhalten basiert, kann nachfolgend numerisch keine gültige Lösung gefunden werden. Deshalb wird eine gezielte Interaktion zwischen qualitativer Vorselektion und numerischer Ermittlung der Eingangsgrößen vorgesehen, die dabei hilft, rasch auf eine gültige (suboptimale) Lösung der hybriden Regelungsaufgabe zuzusteuern.

Vorwort

Die vorliegende Arbeit entstand am Institut für Regelungs- und Automatisierungstechnik der Technischen Universität Graz. Ich möchte mich bei allen Mitarbeitern dieses Instituts herzlich bedanken, die mir immer wieder mit Rat und Kritik zur Seite gestanden sind und so zum Gelingen der Arbeit beigetragen haben.

Meinem Betreuer Prof. N. Dourdoumas danke ich besonders für seine – aufgrund meiner beruflichen Tätigkeit manchmal strapazierte – Geduld und seine Beharrlichkeit, die mir auch in schwierigen Zeiten stets ein Anreiz war, meine Arbeit mit Engagement voranzutreiben.

Besonders bedanken möchte ich mich vor allem auch bei Prof. M. Hofbaur, der mir mit unermüdlichem Einsatz stets für anregende fachliche Diskussionen zur Verfügung gestanden ist. Seine fundierte Kritik und seine konstruktiven Anregungen waren mir stets wichtig und haben wesentlich zum Gelingen der Arbeit beigetragen. Danke vor allem auch für das angenehme und kollegiale Arbeitsklima.

Während der Durchführung meiner Arbeit war ich zeitweise auch als wissenschaftlicher Mitarbeiter am Institut für Elektrische Meßtechnik und Meßsignalverarbeitung tätig. Herrn Prof. G. Brasseur danke ich diesbezüglich für die eingeräumten Freiräume, die das Arbeiten an meiner Dissertation erleichtert haben.

Contents

List of Figures

List of Symbols and Abbreviations

$\mathbf{x}$	Continuous state vector
$\mathbf{x}^{(i)}$	Continuous state vector of i'th component
$\mathbf{x}_k^{(i)}$	Vector of continuous states of i'th component at time t_k
$x_{j,k}^{(i)}$	The j'th element of vector $\mathbf{x}_k^{(i)}$
X	Qualitative abstraction of state vector $\mathbf{x}$
ξ	Qualitative values of the abstracted continuous state X
$\mathcal{X}$	Domain of qualitative values of X
x_d	Discrete state (operational mode)
$\mathcal{X}_d$	Domain of discrete state x_d
x_h	Hybrid state
$\mathbf{u}$	Vector of continuous inputs
$\mathbf{u}_k$	Vector of continuous inputs at time k
$u_{j,k}$	The j'th element of vector $\mathbf{u}_k$
U	Qualitative abstraction of continuous inputs $\mathbf{u}$
υ	Qualitative values of the abstracted continuous input U
$\mathcal{U}$	Domain of qualitative values of U
u_d	Discrete (command) inputs
$\mathcal{U}_d$	Domain of discrete inputs u_d
$\mathbf{y}$	Vector of continuous outputs
$\mathbf{y}_k$	Vector of continuous outputs at time k
$y_{j,k}$	The j'th element of vector $\mathbf{y}_k$
Y	Qualitative abstraction of continuous outputs $\mathbf{y}$
μ	Qualitative values of the abstracted continuous output Y
$\mathcal{Y}$	Domain of qualitative values of Y
y_d	Discrete outputs
$\mathcal{Y}_d$	Domain of discrete outputs y_d
$\mathbf{w}$	Vector of continuous intermediate variables
$\mathbf{w}_k$	Vector of continuous intermediate variables at time k
$w_{j,k}$	The j'th element of vector $\mathbf{w}_k$
W	Qualitative abstraction of continuous intermediate variables $\mathbf{w}$
ω	Qualitative values of the abstracted continuous intermediate variable W
$\mathcal{W}$	Domain of qualitative values of W

T	Transition variable
τ	A specific mode transition
$\mathcal{T}$	Domain of transitions
$\mathcal{A}$	Hybrid automaton
$\mathcal{A}^{(i)}$	Hybrid automaton representing the i'th component
$\mathcal{CA}$	Concurrent hybrid automaton
$\mathbf{Q}$	Qualitative trajectory
$\mathcal{Q}$	Domain of possible qualitative trajectories
$\mathbb{P}$	Polytopic region in a multidimensional value space
$\mathbb{R}(\alpha), \mathbb{S}(\alpha)$	Set of variables depending on index α
a, A	Scalar value
$\mathbf{a}$	Vector
$\mathbf{A}$	Matrix

BDD	Binary Decision Diagram
CSP	Constraint Satisfaction Problem
MLD	Mixed Logical Dynamical Systems
MPC	Model Predictive Control
PWA	Piecewise Affine Model
tDAG	Trajectory Directed Acyclic Graph

Chapter 1

Introduction

1.1 Motivation

Modern technology demands an automated on-line hybrid control scheme that is suitable for controlling multi-component systems which can exhibit a large variety of different modes of operation.

Modern technology, for example in the chemical, automotive or aeronautic industry, increasingly demands automatic control of complex systems. These systems do not only exhibit *continuous dynamics*, but this continuous evolution is also interleaved with *discrete changes* of the *mode of operation*. Such *hybrid systems* generally are difficult to control, as continuous dynamics and discrete evolution among modes of operation are dependent upon one another. Accordingly, *hybrid control* requires simultaneous deduction of the continuous actuation applied to the system together with the discrete commands that drive the system through a suitable sequence of modes of operation.

What makes control of hybrid systems additionally difficult is the usually very large number of possible mode-sequences. This can be especially large, if the hybrid model not only describes the nominal behavior of the system, but also includes modes of operation that capture the system's behavior during the occurrence of certain faults. If there are built-in *redundancies* in the hybrid system, including such *fault-modes* enables the control system to possibly counteract the fault's effects by utilizing the still fully operational parts of the system (*reconfiguration*).

Especially when multi-component systems are considered, even including only few such fault modes for each component results in a huge number of different (faulty) modes of operation for the overall system. Because of this large number, establishing an off-line control scheme (i.e. a control scheme that pre-defines a control strategy before the system starts operating for each of these faulty situations as well) would be practically impossible for larger systems. Furthermore, a major part of the work would be devoted to accounting for very unlikely combinations of faults in various components. For larger systems, therefore, an *on-line* control scheme that does not rely on pre-determined control strategies, but reacts on a given situation at run-time of the system seems advantageous.

The requirement of an on-line reaction to a given situation naturally constrains the time available for deducing appropriate control. So a feasible sequence of future operational modes and continuous actuation have to be determined very efficiently. For larger systems it is certainly impossible to take the intuitive hybrid control approach that determines continuous actuation for each mode sequence separately and then selects the sequence and actuation

that provides best results.

If, however, a few *good* mode-sequences can be pre-selected efficiently, this approach's advantage can still be exploited: A hybrid model with specified mode-sequence can be regarded as *time-variant continuous model*. This allows us to utilize well established methods from control theory to determine the continuous actuation for the hybrid system.

A natural approach to perform this efficient pre-selection is to take a look at the system's behavior at an abstracted level. If both, continuous evolution and discrete mode changes, are captured by a *qualitative model* among discretely-valued variables only, the hybrid control problem can be reformulated as discrete search problem. A task efficiently solved by some well established tools from the field of artificial intelligence (AI).

However, as this qualitative pre-selection only operates on an approximate model and, therefore, only provides approximate solutions, an additional *coordinated interplay* with the numerical control generation has to be established to validate results and assure convergence of the overall hybrid control scheme.

1.2 This Thesis

> *The proposed hybrid control scheme for complex multi-component systems builds upon a specialized formulation of an automatically compiled qualitative model to efficiently solve the hybrid control problem on-line by a two-phase symbolic/numeric control deduction and a coordinated interplay between the two phases.*

The task of the proposed control scheme is to provide on-line hybrid control for a class of multi-component hybrid systems. Each component can show various modes of operation, where each mode itself may exhibit linear or affine continuous dynamics. The control scheme utilizes a two-phase symbolic/numeric control deduction approach. First, discrete search based on an abstracted description of the system's behavior and an abstraction of the control goal provides an efficient pre-selection of a few 'good' and feasible mode-sequences.

Constrained on-line computation time and the large amount of possible mode-sequences require to perform this search very efficiently. This demands the introduction of a novel qualitative modeling scheme. To be able to consider continuous and discrete dynamics simultaneously, this qualitative model captures the hybrid model's mode changes exactly and provides the same type of encoding for an abstraction of the continuous dynamics.

Not to overburden computational resources by computing this abstraction on-line, the qualitative model is pre-compiled off-line. This, however, requires that the model encompasses every possible mode-change and every possible abstracted continuous behavior of the hybrid system. To still keep the size of this model manageable, the abstraction of the hybrid system's behavior is performed for each of the system's components separately. Furthermore, each of these component models is encoded by a *compact graphical representation*. This graphical representation yet serves another purpose which is at least as important as the compact encoding: It provides the possibility to explicitly represent the underlying *structure* of the hybrid system in the qualitative model. This helps in searching the model for promising mode sequences more efficiently.

Still, a well known and unavoidable problem of such qualitatively abstracted models is, that they generally not only provide abstractions of behaviors of the original numerical model, but also model additional qualitative behaviors that do not correspond to any valid behavior of the numerical model. To be able to bias qualitative search towards valid behaviors,

'cost values' are compiled into the qualitative model, which roughly discriminate qualitative behaviors that rather likely represent valid behaviors of the hybrid model from invalid ones.

The result of the qualitative pre-selection procedure is a small number of promising mode-sequences, their associated discrete input commands that drive the hybrid system through these mode sequences and a qualitatively abstracted version of the continuous behavior together with the associated continuous dynamics. These latter approximations require further numerical refinement.

Although the mentioned 'cost values' help to focus pre-selection onto valid abstracted behaviors, it still *cannot be guaranteed* that a pre-selected mode sequence allows a valid solution to the hybrid control task. Therefore, the numerical control-refinement step also has to include a validation of the qualitatively pre-selected mode-sequences. With this numerical validation, invalid sequences are discarded and qualitative pre-selection is resumed to provide alternative mode-sequences until a valid solution is found. With a valid mode-sequence pre-selected, the numerical refinement of continuous actuation can be reformulated and solved as a mathematical (quadratic) program.

A careful interplay between qualitative pre-selection and numerical control refinement helps in pre-selecting only mode sequences that pass the validation with high probability. So computationally demanding numerical control generation doesn't need to be performed too often.

This altogether allows the application of the proposed control scheme to on-line control hybrid systems that are composed of several components.

1.2.1 Related Work

Methodologies used for modeling, control and diagnosis of hybrid systems emerge from both fields, control theory and computer science. As these communities speak largely different languages, the resulting frameworks for hybrid systems are different as well. The proceedings of the annual conference on hybrid systems (Hybrid Systems: Computation and Control, HSCC [2, 42] and earlier volumes) provide a good overview of recent developments in hybrid systems research.

Hybrid Models

Modeling paradigms range from rather computer science oriented approaches to approaches more biased towards control theory with high emphasis on continuous dynamics [13]. An example of the former kind is [1], where a timed automata approach is used to model real-time systems. More control theory linked approaches include the following paradigms: Piecewise affine (PWA) systems [48] split up the state- and input- space into distinct regions, where the system can show different continuous dynamics in each of these regions. Mixed logic dynamic (MLD) systems [9] are models of linear dynamic equations that are subject to linear inequalities involving real and integer variables. [20] utilizes max-min-plus-scaling systems to model a class of discrete event systems. However, it has been shown that all these classes of hybrid systems are equivalent [28].

A component based hybrid modeling approach emphasizing complex continuous dynamics is found in [30, 31]. Here the individual components of a complex system are modeled by individual hybrid automata. The overall model then combines these individual automata to represent what is called a concurrent hybrid automaton.

Qualitative Models

Qualitative modeling approaches are a widespread field as well. As some representative examples for the qualitative abstraction of continuous dynamics, qualitative simulation [36] and the concept of non-deterministic automata [39, 40] are mentioned. This non-deterministic automaton concept is also applied to discrete event systems [24].

Qualitative simulation directly abstracts the underlying physics of the modeled dynamics to qualitative differential equations (qde), which represent an abstracted version of the differential equations used in a numerical model of the dynamics. On the other hand, non-deterministic automata separate the state-space of the dynamics into distinct regions and abstract the dynamic behavior as non-deterministic transitions among these qualitative states.

An issue in all these approaches is the generation of appropriate *landmarks* to distinguish qualitative regions. For example, [49, 47] provide a framework for the automatic generation of such landmarks for a given task. And the paper [18] starts with a very coarse separation and then refines landmarks until a specific question can be answered.

Hybrid Control and Reconfiguration

Literature presents several approaches to hybrid control that either deal with restricted classes of hybrid systems [55, 17] or are restricted in the sense that they work best with systems showing only few modes to keep the computational effort manageable.

Model predictive control (MPC) [41] and similar concepts are a popular tool for on-line control of hybrid systems. In [9, 6] the hybrid control problem is reformulated with MLD systems in order to solve it through powerful linear- or quadratic programming solvers. A recent extension [8] to this utilizes an additional (symbolic) CSP solver on the same problem formulation to focus the numerical solver. MPC is also applied to max-min-plus-scaling systems [21, 52]. [32] utilizes dynamic programming for optimal control of constrained PWA systems with disturbances. On a more global scope than calculating low-level control, [34] addresses the supervisory control of discrete-event hybrid systems.

In addition to the standard hybrid control problem, [51] explicitly addresses hybrid reconfiguration issues without utilizing qualitative methods by applying MPC to mixed-logic-dynamic (MLD) systems. It utilizes the well known benchmark example of a 3-tank system to demonstrate the approach. In comparison, [4] presents a similar example but address the hybrid reconfiguration problem by utilizing qualitative abstractions. Another symbolic way to address the reconfiguration problem emerging from the field of artificial intelligence (AI) is called reactive planning. An example for such an automatic planning engine is presented in [54].

The complementary problem to hybrid control, hybrid estimation and diagnosis is treated in the books [31, 26], for example.

Qualitative Reasoning

Qualitative reasoning methods like constraint-satisfaction are a very widespread topic in literature. The books [46, 22] provide an overview on this topic. The hybrid control scheme that is subject of this thesis concentrates on discrete search and especially search for paths in graphs. An overview of search algorithms that operate on graphs is presented in [23] and an especially efficient type of so called *best first* search algorithms known as A^* is presented in [27] and a link to graph theory is presented in [25].

On-line Numerical Control Design

A popular tool for on-line numerical controller design for systems subject to constraints is *model predictive control* (MPC), a receding horizon control strategy where the modeled future behavior of a system over a finite prediction horizon is corrected by the system inputs to optimally approximate a given control goal. A good overview over various formulations of MPC is found in [41].

For rather simple systems, the solution to this receding horizon problem can be explicitly calculated off-line and expressed as piecewise linear state-feedback [7, 10]. For more complex systems, the solution is calculated on-line.

Operating with a finite receding prediction horizon requires some additional measures to ensure stability of the closed loop system. Some formulations guaranteeing stability of the closed-loop system are presented in the survey-paper [43]. Stability issues for PWA systems are discussed in [38].

1.2.2 Contributions

The main contributions of this thesis are along the following lines of research:

Automated Generation of a Qualitative Model

This thesis presents a method that automatically generates a qualitative model from a hybrid model. The qualitative modeling framework is especially tailored to the task of run-time hybrid control. So, the qualitative model shows some special features that are advantageous in searching the model for a sequence of modes of operation that is suitable to meet a desired control goal. These features are the very compact graphical representation of the model and an explicit representation of structural properties of the underlying hybrid system in the model.

Run-time Qualitative Control Deduction

Control deduction at system run-time may be advised for complex systems, since an overwhelming number of modes of operation for these systems prevents calculation of a separate controller for each situation. To meet time-constraints related to the on-line control deduction, an efficient pre-selection of feasible mode-sequences has to be performed.

Utilization of the specialized qualitative modeling scheme allows to reformulate this pre-selection as shortest path search, what makes well established efficient solvers like A^* applicable. Formulation of the search problem according to the underlying structure of the hybrid system itself, further, increases efficiency.

1.3 Thesis Outline

The following chapter provides a first motivating example for the proposed two-phase qualitative/numerical approach to hybrid control. It also informally outlines the control scheme in a way that omits most low-level detail. Further, the chapter briefly presents some of the most important tools and theory that are utilized throughout the control approach.

Chapter 3 formalizes the class of complex multi-component hybrid systems that is tackled with the presented hybrid control scheme.

Chapter 4, then, formalizes the qualitative modeling framework. It also deals with the automated abstraction of the hybrid model in detail.

With this qualitative model at hand, Chapter 5 first formalizes the two-phase hybrid control scheme in Section 5.1. Section 5.2 illustrates how the the qualitative search can be performed efficiently by exploiting the special formulation of the qualitative model. Finally, Section 5.3 deals with the subsequent numerical refinement of the continuous actuation and Section 5.4 presents how a careful interplay between the two solvers further helps in increasing efficiency.

Chapter 6 illustrates the proposed hybrid control scheme by some examples. Chapter 7 summarizes the thesis, Appendix A provides some theoretical background and Appendix B outlines some algorithms used in the thesis.

Chapter 2

Motivating Example

In this chapter, a hybrid control task for a simple academic single-component system will be solved intuitively. This shall motivate the proposed hybrid control scheme and shall give a first glance on the hybrid control framework and the underlying qualitative modeling. Not to loose focus on the overall story by discussing low level details in this introductory chapter, formalisms and through explanations are left out whenever possible.

Nevertheless, this chapter will at least provide a glimpse onto all necessary modeling and control deduction steps and the tools and theory utilized for them. This shall provide a good intuition of the overall control scheme which will then be formalized for more complex multi-component systems in the subsequent chapters 3-5.

Section 2.1 first introduces the example system that guides through the chapter. Additionally, the control objective is defined and some key difficulties in achieving it are pointed out.

Section 2.2 provides an intuitive solution to the hybrid control problem obtained by merely *'looking at'* the system and trying to determine a good solution by some *'educated guessing'*. This approach of a human looking at the system's dynamics and quickly coming up with a good solution by reasoning will motivate the intention to introduce qualitative pre-selection as a first step in solving the hybrid control problem.

Consequently, Section 2.3 provides a first outline of a qualitative modeling scheme that enables to focus and automatize the previous sections *'guessing'*. Further, the considerations that motivate the utilization of a very specialized type of qualitative model are discussed.

Section 2.4 finally outlines the overall hybrid control procedure composed of qualitative pre-selection (of promising sequences of modes of operation), numerical refinement (of continuous actuation) and a careful interplay between the two.

2.1 Example Hybrid System

This section introduces a simple hybrid control example where some intuitive understanding of the system's dynamics and some human reasoning quickly can provide a *good idea* of how to solve a hybrid control task. The underlying procedure followed by the *human controller* shall motivate utilization of automated qualitative reasoning as a tool to quickly and moreover automatically provide these *good ideas* for more complex systems.

2.1.1 Hybrid Model

The utilized example is a simple single-component hybrid model with three modes of operation $\{m_1, m_2, m_3\}$. Continuous dynamics for each of these modes m_i are specified by linear, time invariant state-space models

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}_i\mathbf{x} + \mathbf{b}_i u,\ ^1$$

more specifically for the example:

$$\text{mode } m_1: \ \frac{d}{dt}\mathbf{x} = \begin{bmatrix} -0.5 & -1 \\ 1 & -0.5 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0.2 \\ 0 \end{bmatrix} u \tag{2.1a}$$

$$\text{mode } m_2: \ \frac{d}{dt}\mathbf{x} = \begin{bmatrix} -0.2 & 0.17 \\ 0.15 & -0.2 \end{bmatrix}\mathbf{x} + \begin{bmatrix} -0.06 \\ -0.1 \end{bmatrix} u \tag{2.1b}$$

$$\text{mode } m_3: \ \frac{d}{dt}\mathbf{x} = \begin{bmatrix} 0.85 & 1 \\ -1 & 0.85 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0.2 \\ 0 \end{bmatrix} u, \tag{2.1c}$$

where $\mathbf{x} = [x_1, x_2]^T$ represents the system's *continuous state* and the continuous input u stands for the *continuous actuation* applied to the system. Figure 2.1 shows the vector fields of the three modes of operation for $u = 0$ and $u = 1$, respectively.

In addition to applying continuous actuation u to the system, its behavior can be influenced by changing the mode of operation (denoted the system's *discrete state* x_d) utilizing the so called *command input* u_d. This command input can take one out of three discrete values, i.e.

$$u_d = \{\texttt{switch-to-m}_1, \texttt{switch-to-m}_2, \texttt{switch-to-m}_3\}. \tag{2.2}$$

However, to make control of the system not too easy, it is assumed that the mode m_2 may only be activated when the system's state $\mathbf{x} = [x_1, x_2]^T$ is within a certain region, i.e.

$$x_1 \leq -0.75, \qquad x_2 \leq -0.75. \tag{2.3}$$

This is illustrated in Figure 2.2, where the system's modes of operation are represented by the nodes. The labels of the edges connecting them represent the conditions under which the corresponding mode-change takes place.

What, as in many real systems, further complicates control are constraints on the continuous state and input of the system. The actuation u is limited to

$$0 \leq u \leq 1 \tag{2.4}$$

and the state $\mathbf{x}$ of the system has to be kept inside the region displayed in Figure 2.1, i.e.

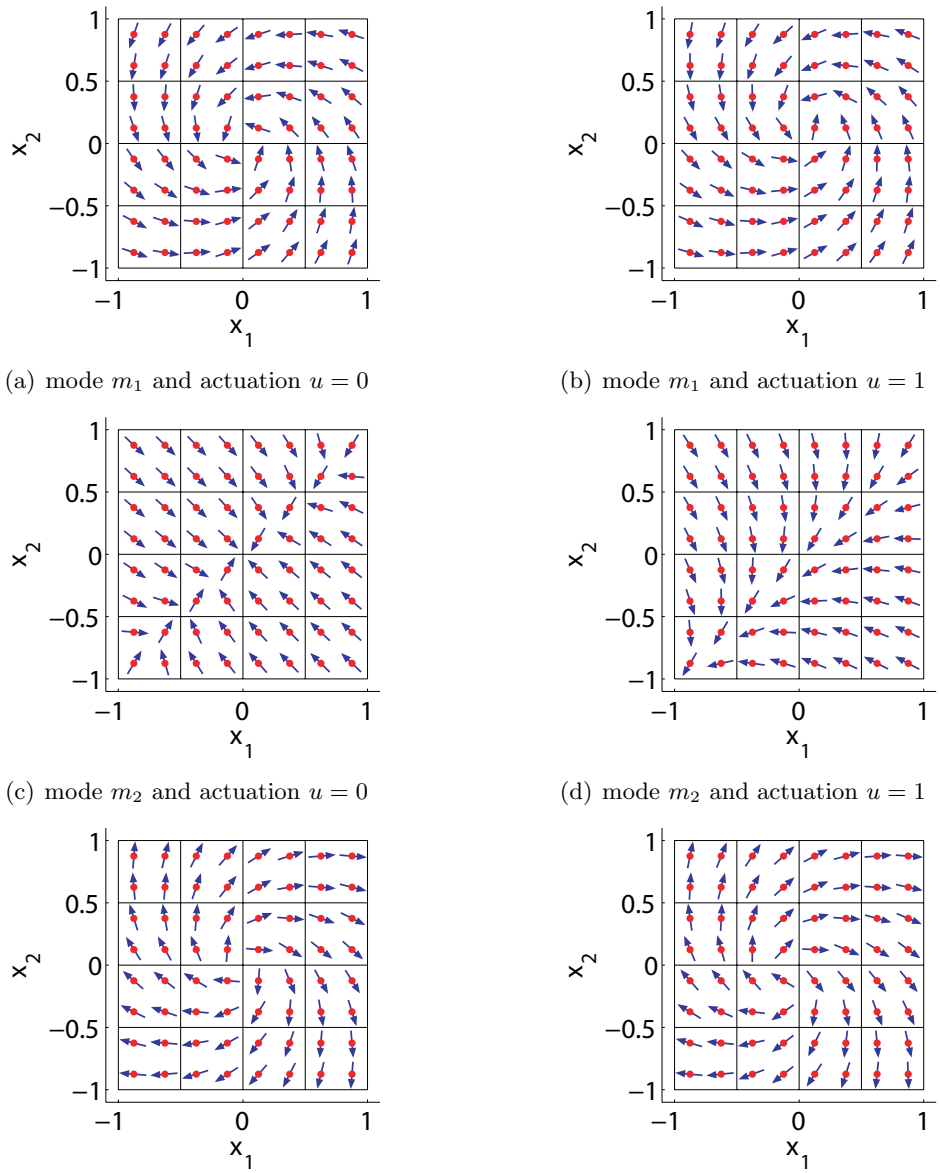
$$-1 \leq x_1 \leq 1 \tag{2.5a}$$

$$-1 \leq x_2 \leq 1 \quad . \tag{2.5b}$$

In spirit of digital control, changes in the continuous actuation $u(t)$ and mode-change commands u_d can only be issued at the sampling times

$$t_k = t_0 + kT_s, \qquad k = 0, 1, \ldots$$

[1] For a more compact display, the time-dependence of variables (e.g. $\mathbf{x}(t)$, $u(t)$) is not explicitely mentioned whenever possible.



(a) mode m_1 and actuation $u = 0$

(b) mode m_1 and actuation $u = 1$

(c) mode m_2 and actuation $u = 0$

(d) mode m_2 and actuation $u = 1$

(e) mode m_3 and actuation $u = 0$

(f) mode m_3 and actuation $u = 1$

Figure 2.1: Vector fields for the example system

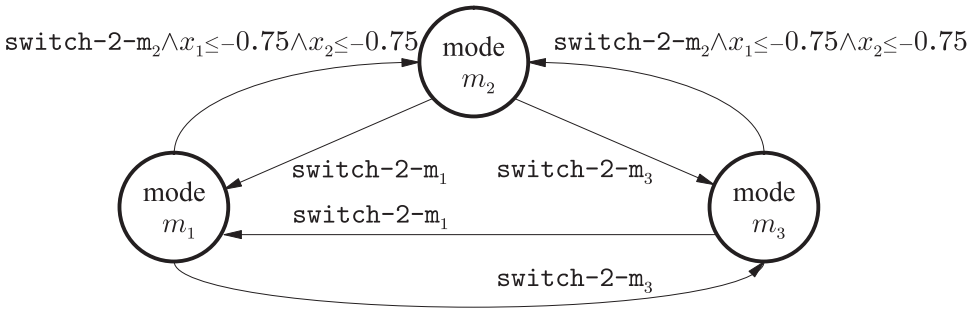


Figure 2.2: Mode transition graph for the example system

where t_0 and $T_s = 1.5$ denote the initial time and sampling period (1.5 seconds) of the digital control system, respectively. The continuous actuation u is kept constant within the sampling period at

$$u(t) = u_k, \qquad t_k \leq t < t_{k+1}$$

In the same sense, the sampled continuous state $\mathbf{x}_k$ of the system is given by

$$\mathbf{x}_k = \begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix} = \mathbf{x}(t_k)$$

and the discrete state is kept at

$$x_d(t) = x_{d,k}, \qquad t_{k-1} < t \leq t_k.$$

Since for digital control only the system quantities at sample times t_k are decisive, the models (2.1) describing the system's continuous dynamics are rewritten in discrete-time form

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_i \mathbf{x}_k + \mathbf{b}_{di} u_k.$$

With the above standard conventions in digital control, the transition matrices $\mathbf{\Phi}_i$ and the input vectors $\mathbf{b}_{di}$ can be obtained by

$$\mathbf{\Phi}_i = e^{\mathbf{A}_i T_s} \qquad \text{and} \qquad \mathbf{b}_{di} = \int_0^{T_s} e^{\mathbf{A}_i \tau} \mathbf{b}_i \, d\tau$$

The example system's continuous dynamics for each mode of operation m_1, m_2, m_3, hence, are described by the models

$$m_1 : \mathbf{x}_{k+1} = \begin{bmatrix} 0.03 & -0.47 \\ 0.47 & 0.03 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} 0.15 \\ 0.12 \end{bmatrix} u_k \tag{2.6a}$$

$$m_2 : \mathbf{x}_{k+1} = \begin{bmatrix} 0.76 & 0.19 \\ 0.17 & 0.76 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} -0.09 \\ -0.14 \end{bmatrix} u_k \tag{2.6b}$$

$$m_3 : \mathbf{x}_{k+1} = \begin{bmatrix} 0.25 & 3.57 \\ -3.57 & 0.25 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} 0.34 \\ -0.44 \end{bmatrix} u_k \tag{2.6c}$$

2.1.2 Hybrid Control Task

The task of automatic control is to actuate the hybrid system such that its hybrid state follows a particular trajectory, reaches a specific state in the state-space or remains within the vicinity of a desired operational point. As particular control objective for the example, the system's continuous state state shall be moved towards the goal state

$$\mathbf{x}_g = \begin{bmatrix} -0.75 \\ -0.75 \end{bmatrix} \tag{2.7}$$

without violating any of the constraints (2.4-2.5). The limited continuous actuation and the relatively large sampling time make the control task difficult to achieve, but allow to demonstrate many aspects of the proposed control scheme.

At time t_0, the system is at the hybrid initial state

$$x_{d,0} = m_1 \tag{2.8a}$$

$$\mathbf{x}_0 = \begin{bmatrix} -0.75 \\ 0.75 \end{bmatrix}. \tag{2.8b}$$

2.2 An Intuitive Approach to Hybrid Control

To motivate the further developments, this control task shall now be solved intuitively by a human. For this, we first do not deal with the dynamic equations (2.6) in detail, but just take a quick look at the corresponding vector fields graphically displayed in Figure 2.1 on page 9.

As a brief description of these vector fields, we can qualitatively classify the vector fields for mode m_1 as counter-clockwise stable spirals for all continuous acuation within the allowed range of $0 \leq u_k \leq 1$ and the behavior of the system's state at mode m_3 can be classified as unstable clockwise spirals. The discrete state m_2 is the only mode of operation where the limited actuation $0 \leq u_k \leq 1$ results in a significant change of the vector field around the goal state $\mathbf{x}_g = [-0.75, -0.75]^T$. One can qualitatively classify stable trajectories towards the center of the state space for zero actuation and stable trajectories towards points in the lower left of the state space for larger actuation. With this, one can reason that only mode m_2 is capable of keeping the continuous state near the goal.

Utilizing this qualitative understanding of the system's dynamics to solve the hybrid control task, one immediately gets the intuition that the stable counter-clockwise trajectories of mode m_1 quickly traverse the system's state from its initial value in the upper left region ($\mathbf{x}_0 = [-0.75, 0.75]^T$) towards the goal state in the lower left region ($\mathbf{x}_g = [-0.75, -0.75]^T$), where (according to Figure 2.2 on page 9) the system can be switched to the stabilizing mode m_2.

However, after this first qualitative analysis of the control problem and quick reasoning about a possible solution one needs to validate ones intuition by taking a closer look at dynamic equation (2.6a) for the pre-selected mode m_1. This detailed numerical treatment tells that with the limited actuation (2.4) the system's state cannot directly reach the state-space region (2.3) what would be necessary for switching to mode m_2. Figure 2.3 illustrates this fact by showing the reachable state-space regions for the mode-sequence $\{m_1, m_1, m_1\}$.

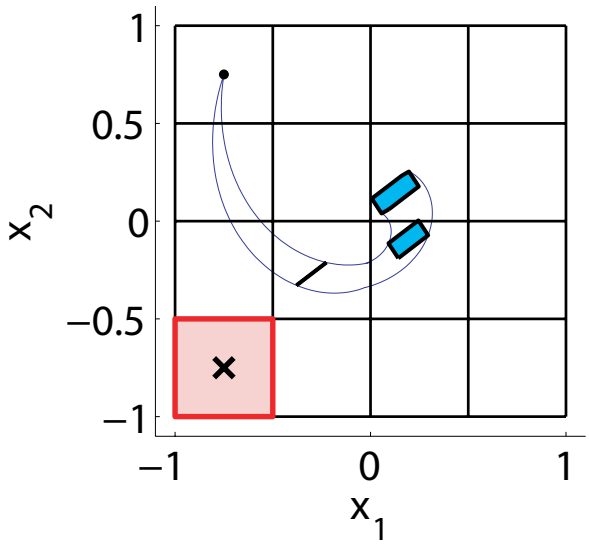


Figure 2.3: Reachable states for the mode sequence $\{m_1, m_1, m_1\}$

This figure together with the qualitative classification of the behaviors for mode m_3 as unstable clockwise spirals provides a further idea: If the system behaves according to mode

m_1 until the continuous state *passes by* the goal region, one could use the unstable mode m_3 to reverse the direction of the state movement and let it proceed right towards the goal-area (2.3).

Trying to validate the results of qualitative reasoning numerically again, one observes that this new pre-selected mode-sequence $\{m_1, m_1, m_3\}$ *allows* trajectories that move the continuous state to the goal region. This is illustrated by displaying the reachable states in Figure 2.4.

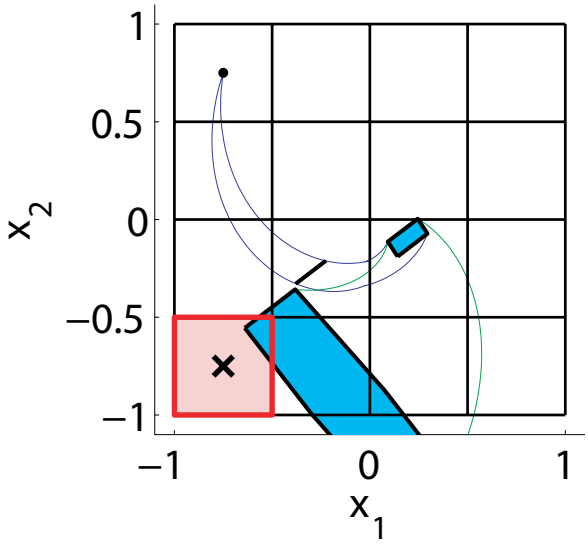


Figure 2.4: Reachable states for the mode sequence $\{m_1, m_1, m_3\}$

Here one is allowed to switch to mode m_2 which is qualitatively considered as being able to keep the system's state near the goal. So one can pre-select a mode-sequence $\{m_1, m_1, m_3, m_2, m_2, \ldots\}$.

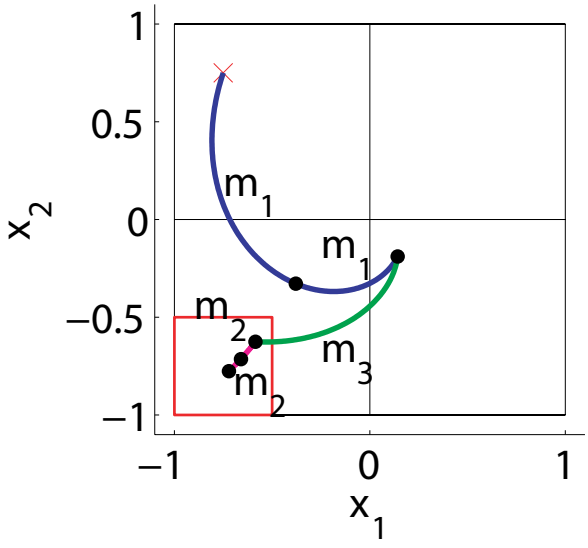


Figure 2.5: Trajectory for mode sequence $\{m_1, m_1, m_3, m_2, m_2\}$ and actuation (2.10)

With the mode-sequence specified, the hybrid model simply can be regarded as time-variant continuous model, specifically

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{\Phi}_1\mathbf{x}_0 &&+\mathbf{b}_{d1}u_0 \\ \mathbf{x}_2 &= \mathbf{\Phi}_1\mathbf{x}_1 &&+\mathbf{b}_{d1}u_1 \\ \mathbf{x}_3 &= \mathbf{\Phi}_3\mathbf{x}_2 &&+\mathbf{b}_{d3}u_2 \\ \mathbf{x}_4 &= \mathbf{\Phi}_2\mathbf{x}_3 &&+\mathbf{b}_{d2}u_3 \\ \mathbf{x}_5 &= \mathbf{\Phi}_2\mathbf{x}_4 &&+\mathbf{b}_{d2}u_4 \\ &\vdots \end{aligned} \tag{2.9}$$

and the continuous actuation can be determined by well established methods from the field of control theory, e.g. Model Predictive Control [41]. An exemplary result of such a control design determines continuous actuation $u_0, \dots u_4$ as

$$u_0 = 0 \qquad u_1 = 0 \qquad u_2 = 0.16 \qquad u_3 = 1 \qquad u_4 = 0.87. \tag{2.10}$$

The resulting trajectory for the continuous state of the hybrid system is depicted in Figure 2.5, where the dots indicate sampled states.

2.3 A First Outline of the Qualitative Model

In this section, the intuitive reasoning previously used to solve the example shall be put onto a more firm basis in order to integrate it into an automatic hybrid control scheme. For this, it is first defined how the hybrid system's behavior is specified qualitatively.

To obtain such a qualitative model, the continuous state- and input-space of the hybrid model (2.6) are separated into a *finite* number of qualitatively distinct regions. This gives the opportunity to approximately capture the continuous dynamics of the hybrid model in terms of a *finite* set of relations among discretely valued variables.

This will be illustrated right after some more details on this qualitative separation have been discussed: The separation should be reasonably coarse to keep the qualitative model compact. However, it is advisable that the qualitative separation at least captures all the qualitative distinctions and constraints provided by the original hybrid automaton. For the example, this means that one needs to qualitatively discriminate the state space region defined by (2.3) – where a transition to mode m_2 is possible – against all other regions of the state space. Making this distinction will later on allow to *exactly* capture the discrete part of the hybrid model in our qualitative model.

In this introductory chapter, we choose a very coarse separation that primarily distinguishes the sign of the individual states x_1 and x_2. This may not be very well suited for solving general control tasks on the example, but it is well suited for the presentation here, because it helps to point out some details on the control scheme by compact illustrations. However, to be able to exactly capture the discrete part of the hybrid model, specifically the distinction between state-space regions where mode transitions to m_2 are possible and those where they are not, an additional qualitative region is specified: The lower left region of the constrained state space given by the mode-transition-requirement (2.3). Figure 2.6 displays the resulting qualitative separation of the constrained state space (2.5) into 5 different qualitative regions $\xi_1, \dots, \xi_5$. For the same illustrative reasons a very coarse separation of the state space was selected, only one single qualitative region is utilized as abstraction of the constrained input space (2.4).

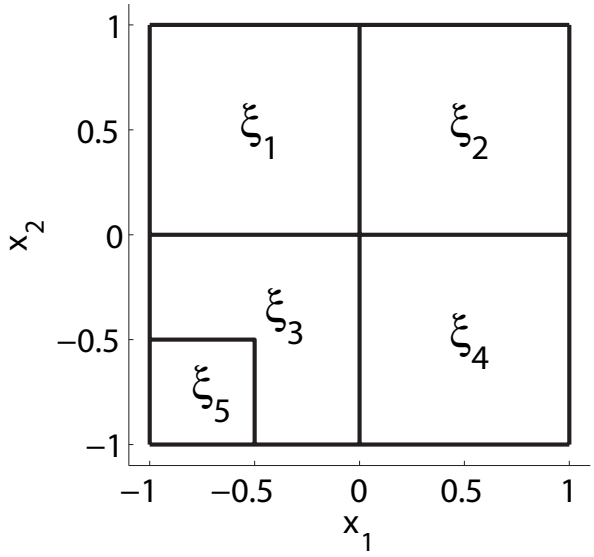


Figure 2.6: Separation of the state space into 5 qualitative regions

With this qualitative separation, continuous trajectories of the hybrid model can be encoded as finite set of relations among the qualitatively abstracted variables. To keep the qualitative model compact, only trajectories for a small time-horizon, (e.g. $t_k \rightarrow t_{k+1}$) are encoded. This single time-step qualitative model can later on be used consecutively for reasoning about longer trajectories.

To provide an example, it is shown how continuous trajectories emerging from the state space region covered by qualitative state ξ_1 according to mode m_1 are encoded as 'qualitative trajectories'. The hybrid model's equation for describing these trajectories is (2.6a):

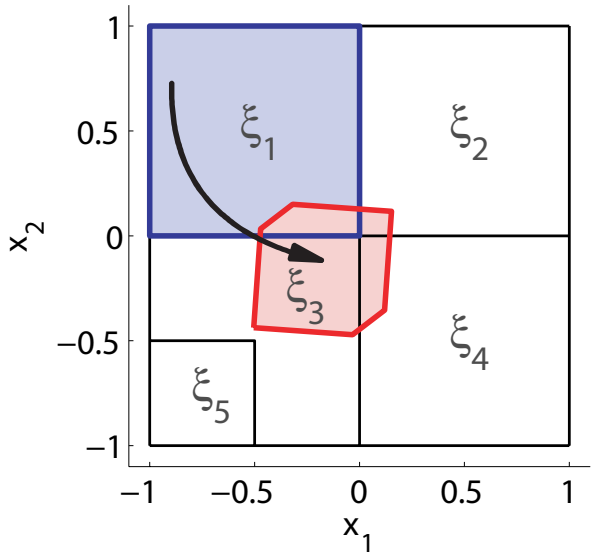


Figure 2.7: Trajectories emerging from qualitative region ξ_1

$$m_1 : \mathbf{x}_{k+1} = \begin{bmatrix} 0.03 & -0.47 \\ 0.47 & 0.03 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} 0.15 \\ 0.12 \end{bmatrix} u_k.$$

Due to this equation and limited actuation $0 \leq u_k \leq 1$ (2.4) all states $\mathbf{x}_{k+1}$ lie within the region depicted in Figure 2.7.

This covers parts of the state space regions that belong to qualitative states ξ_1 to ξ_4. So, to capture all these trajectories qualitatively we have to distinguish four different abstracted trajectories: trajectories leading from state ξ_1 with mode m_1 to states ξ_1, ξ_2, ξ_3 or ξ_4, respectively.

With qualitative variables X_k and X_{k+1} to denote the qualitatively abstracted state at times t_k and t_{k+1}, respectively, and the notational convention that the operational mode valid from time t_k to t_{k+1} is denoted $x_{d,k+1}$, these trajectories can be encoded qualitatively as[2]

$$\begin{array}{lllllllll} (& X_k = \xi_1 & \wedge & x_{d,k+1} = m_1 & \wedge & X_{k+1} = \xi_1 &) & \vee \\ (& X_k = \xi_1 & \wedge & x_{d,k+1} = m_1 & \wedge & X_{k+1} = \xi_2 &) & \vee \\ (& X_k = \xi_1 & \wedge & x_{d,k+1} = m_1 & \wedge & X_{k+1} = \xi_3 &) & \vee \\ (& X_k = \xi_1 & \wedge & x_{d,k+1} = m_1 & \wedge & X_{k+1} = \xi_4 &) & . \end{array}$$

This introduces ambiguity to the qualitative model, as for a single source state ($X_k = \xi_1$) and operational mode ($x_{d,k+1} = m_1$) there are multiple destination states X_{k+1}.

If one, however, looks at Figure 2.7, one observes that most of the destination states are inside the region covered by qualitative state ξ_3. This is now utilized to partially resolve the ambiguity by associating likelihood values to each of the ambiguous qualitative trajectories. A simple way to calculate such likelihoods (denoted L) is to draw a sufficiently large number of random samples for values $\mathbf{x}_k$ within a qualitative state and determine associated values $\mathbf{x}_{k+1}$ by the hybrid model. For the above example trajectories, these likelihoods are

X_k	$x_{d,k+1}$	X_{k+1}	L
ξ_1	m_1	ξ_1	0.13
ξ_1	m_1	ξ_2	0.03
ξ_1	m_1	ξ_3	0.74
ξ_1	m_1	ξ_4	0.10 .

To complete the qualitative model, qualitative trajectories emerging from all 5 qualitative states according to all 3 operational modes are recorded. Completing the qualitative model

[2] As in this presentation the whole range of allowed continuous inputs u is covered by only one single qualitative value, to keep the display compact, the input is omitted from this display of qualitative trajectories.

for all other states X_k and modes $x_{d,k+1}$ results in:

X_k	$x_{d,k+1}$	X_{k+1}	L	X_k	$x_{d,k+1}$	X_{k+1}	L
ξ_1	m_1	ξ_1	0.13	ξ_3	m_1	ξ_4	0.87
ξ_1	m_1	ξ_2	0.03	ξ_3	m_2	ξ_3	0.87
ξ_1	m_1	ξ_3	0.74	ξ_3	m_2	ξ_5	0.13
ξ_1	m_1	ξ_4	0.10	ξ_3	m_3	ξ_1	0.10
ξ_1	m_2	ξ_1	0.73	ξ_3	m_3	ξ_2	0.01
ξ_1	m_2	ξ_2	0.07	ξ_3	m_3	ξ_3	0.03
ξ_1	m_2	ξ_3	0.20	ξ_3	m_3	ξ_4	0.01
ξ_1	m_3	ξ_1	0.00	ξ_3	m_3	ξ_5	0.00
ξ_1	m_3	ξ_2	0.07	ξ_4	m_1	ξ_2	1.00
ξ_1	m_3	ξ_4	0.01	ξ_4	m_1	ξ_4	0.00
ξ_2	m_1	ξ_1	0.80	ξ_4	m_2	ξ_2	0.04
ξ_2	m_1	ξ_2	0.20	ξ_4	m_2	ξ_3	0.19
ξ_2	m_2	ξ_1	0.00	ξ_4	m_2	ξ_4	0.77
ξ_2	m_2	ξ_2	0.97	ξ_4	m_3	ξ_3	0.04
ξ_2	m_2	ξ_3	0.01	ξ_4	m_3	ξ_4	0.01
ξ_2	m_2	ξ_4	0.02	ξ_4	m_3	ξ_5	0.02
ξ_2	m_3	ξ_2	0.00	ξ_5	m_1	ξ_4	1.00
ξ_2	m_3	ξ_4	0.05	ξ_5	m_2	ξ_3	0.00
ξ_3	m_1	ξ_2	0.12	ξ_5	m_2	ξ_5	0.98
ξ_3	m_1	ξ_3	0.00				

(2.11)

Such a model could be captured in terms of a non-deterministic automaton [39, 40]. However, for larger examples or more fine grained qualitative abstractions such a model would be rather large in size. So a more compact representation that is very similar to so called Ordered Binary Decision Diagrams (OBDDs) [3] (also see appendix A.4) will be utilized. Similar to OBDDs, one represents the qualitative model in terms of a directed, acyclic graph (DAG), where each qualitative trajectory (each 'line' in the above table) is represented by a path from the root node of the graph to one of its leaves. The various leaves of that so-called trajectory-DAG (tDAG) represent the different likelihood values. The successive layers of the graph represent the successive columns of (2.11) and the graph's edges leaving a node in a particular layer represent a particular qualitative value of the corresponding qualitative variable. To define this succession for our example, we specify the ordering:

$$X_k \prec x_{d,k+1} \prec X_{k+1}.$$

A graph that, based on this ordering, illustrates the first ten lines of (2.11) is shown in Figure 2.8.

This graphical representation, however, isn't very compact yet, because it uses a separate leaf node for each distinct likelihood value. And this, moreover, is in contradiction to our qualitative modeling idea: We want to obtain an abstracted (thus only approximative) model of a hybrid system that gives a quick intuition of the system's possible behaviors. To give this quick intuition, it is not necessary to consider likelihood values that are calculated to a few digits behind the comma, but a rough distinction of e.g. 'likely', 'rather unlikely' and 'very unlikely' qualitative trajectories seems appropriate. In the example, we define this qualitative separation as:

$$\begin{array}{rcccccl} & & L & \geq & 0.33 & \rightarrow & \text{likely} \\ 0.33 & > & L & \geq & 0.033 & \rightarrow & \text{rather unlikely} \\ 0.033 & > & L & & & \rightarrow & \text{very unlikely.} \end{array}$$

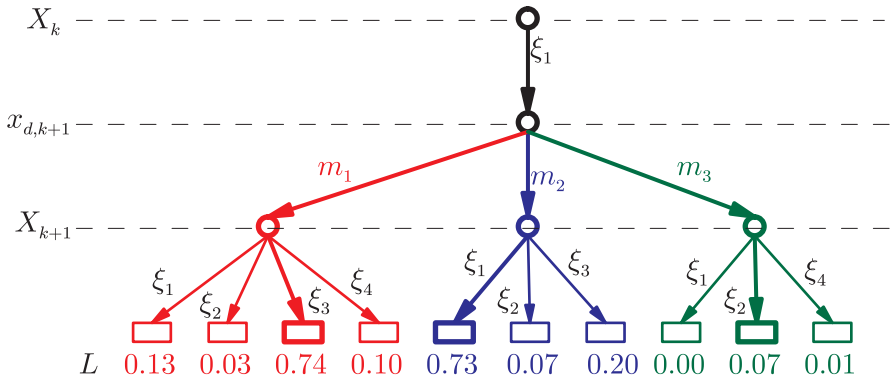


Figure 2.8: Graph representing qualitative trajectories starting from state ξ_1

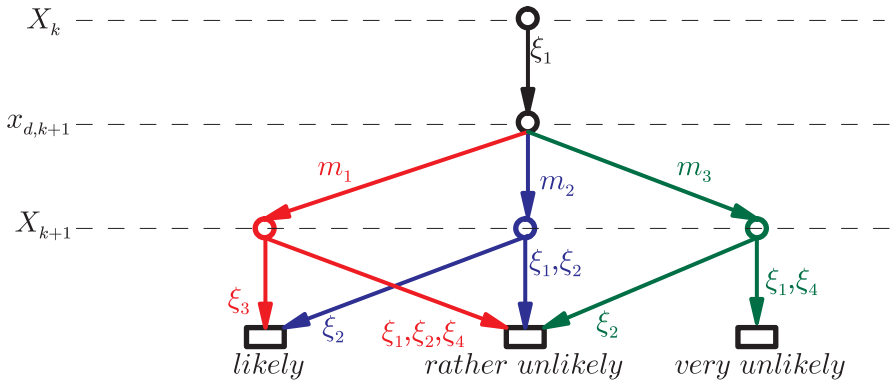


Figure 2.9: Qualitative trajectories starting from state ξ_1

For, again, the first ten lines of (2.11) the graphical representation obtained according to this qualitative separation of trajectory-likelihoods is shown in Figure 2.9.

We later on want to utilize such a graphical representation of the qualitative model to formulate qualitative pre-selection as shortest path search. Therefore, we want to represent qualitative trajectory-likelihoods by path-lengths in the graph. For this, the qualitative trajectory-likelihoods are represented by numeric cost values C_L such that less likelihood corresponds to a higher cost value. For the example, we choose:

$$\begin{aligned} \text{likely} &\rightarrow C_L = 0 \\ \text{rather unlikely} &\rightarrow C_L = 0.8 \\ \text{very unlikely} &\rightarrow C_L = 3.7. \end{aligned}$$

Treating these cost values as path-lengths in the graph allows to map likelihoods to edge lengths in the graph.

Additionally, as the tDAG is based on *Binary* Decision Diagrams, the values of all variables in the qualitative model have to be represented as expressions among binary variables:

X	$\rightarrow$	X^{B1}	X^{B2}	X^{B3}
ξ_1		0	0	0
ξ_2		0	0	1
ξ_3		0	1	0
ξ_4		0	1	1
ξ_5		1	0	0

x_d	$\rightarrow$	x_d^{B1}	x_d^{B2}
m_1		0	0
m_2		1	0
m_3		0	1

With this binary representation of qualitative values, the model with approximated transition likelihood costs for abstracting continuous evolution of the hybrid model for time t_k to t_{k+1} displays as:

X_k	$x_{d,k+1}$	X_{k+1}	C_L	X_k	$x_{d,k+1}$	X_{k+1}	C_L
000	00	000	0.8	010	00	011	0
000	00	001	3.7	010	10	010	0
000	00	010	0	010	10	100	0.8
000	00	011	0.8	010	01	000	0.8
000	10	000	0	010	01	001	3.7
000	10	001	0.8	010	01	010	3.7
000	10	010	0.8	010	01	011	3.7
000	01	000	3.7	010	01	100	3.7
000	01	001	0.8	011	00	001	0
000	01	011	3.7	011	00	011	3.7
001	00	000	0	011	10	001	0.8
001	00	001	0.8	011	10	010	0.8
001	10	000	3.7	011	10	011	0
001	10	001	0	011	01	010	0.8
001	10	010	3.7	011	01	011	3.7
001	10	011	3.7	011	01	100	3.7
001	01	001	3.7	100	00	011	0
001	01	011	0.8	100	10	010	3.7
010	00	001	0.8	100	10	100	0.
010	00	010	3.7				

(2.12)

For, again, the first ten lines of this qualitative model, we present the according graphical representation as trajectory-DAG in Figure 2.10. This graph encodes all qualitative trajectories starting from $X_k = \xi_1 \hat{=} (X_k^{B1} = 0 \wedge X_k^{B2} = 0 \wedge X_k^{B3} = 0)$. In this graph, all 10 trajectories are represented by directed paths from the root node to one of the leaves. Additionally, the approximated likelihood costs C_L of the trajectories are mapped to the graphs edges such that the sum of all edge values of a path equals the likelihood cost C_L of the corresponding trajectory.

Discrete mode transitions are represented by a similar tDAG to obtain the same type of qualitative model for both, qualitatively abstracted continuous dynamics and discrete mode transitions. This simplifies reasoning because then both types of evolution can be treated simultaneously without utilizing different tools to handle both of them.

The compactness of the representation of the qualitative model as tDAGs may not be obvious for the small example in Figure 2.10. However, things get more impressive when bigger models or larger time-horizons are considered. For example a qualitative model for representing continuous evolution and discrete mode transitions, both for a 3-step time-horizon $t_k \rightarrow t_{k+3}$, needs to encompass

7266 qualitative trajectories, each represented by 27 binary variables.

However, it's corresponding tDAG, based on a variable-ordering

$$x_{d,k} \prec X_k \prec u_{d,k} \prec x_{d,k+1} \prec X_{k+1} \prec u_{d,k+1} \prec x_{d,k+2} \prec X_{k+2} \prec u_{d,k+2} \prec x_{d,k+3} \prec X_{k+3}$$

only consists of

952 nodes and 1287 directed edges. (2.13)

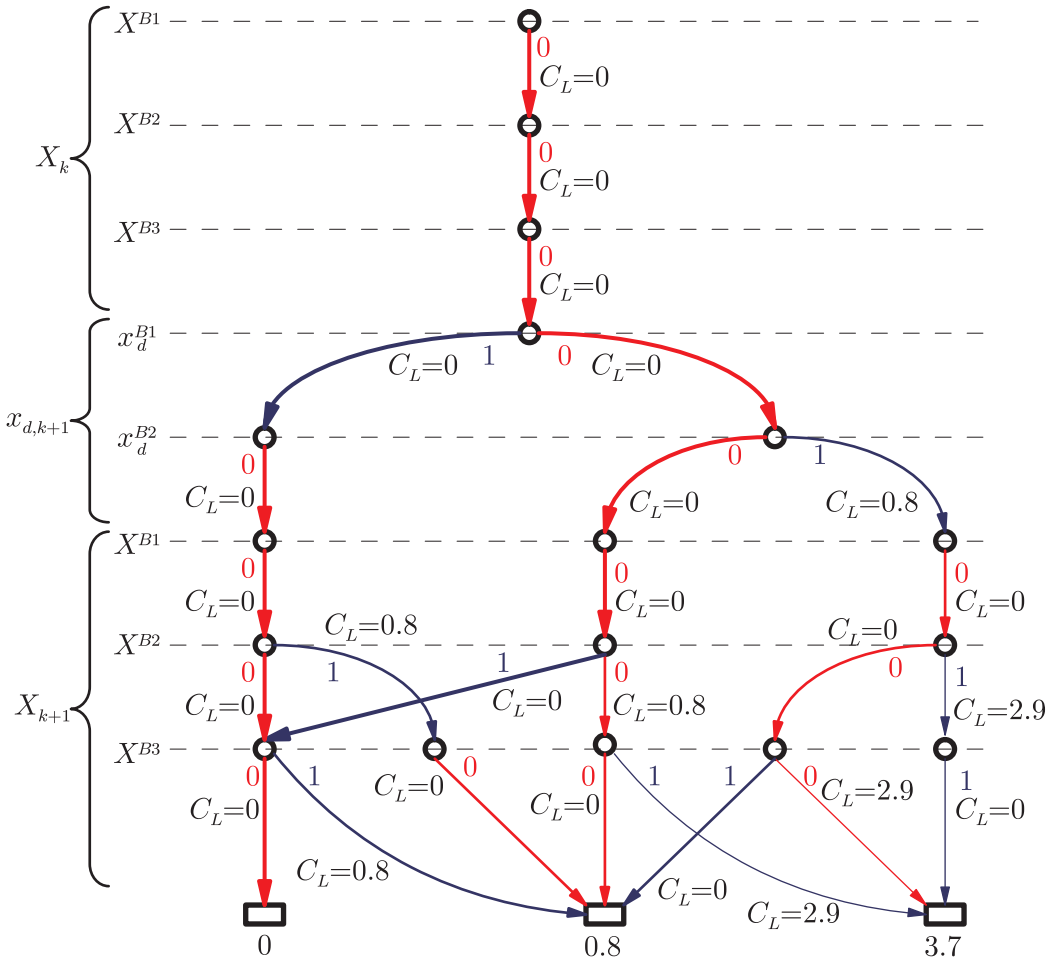


Figure 2.10: tDAG representing qualitative trajectories starting from state $\xi_1(\hat{=}000)$

We have to mention that, similar to Binary Decision Diagrams, the size of the tDAG can strongly depend on the specified ordering of variables. However, experience shows that defining the ordering among qualitative variables based on the causal interdependencies among the hybrid model's equations is generally a good choice for obtaining small graphs.

Summarizing, as result of automated qualitative modeling for the introductory example, we obtain a tDAG that qualitatively captures all continuous trajectories as well as the possible discrete mode transitions for a time horizon t_k to t_{k+3}. This model is generated at 'compile time', i.e. before the hybrid control task is performed. So compactness of representation is more important than time-efficiency of its compilation.

Further, it has to be noticed that – despite of its compact representation – a qualitative model which results from a separation of the state-, input- and output space generally grows exponentially with the dimensionality of the system. However, with our hybrid modeling framework (Section 3) it is assumed that a hybrid system can be described by rather low-dimensional hybrid models for the various components of the system. The system's overall complex behavior results from interaction among these components.

2.4 An Outline of the Hybrid Control Framework

We will now demonstrate how the compiled qualitative model can be utilized for an automated on-line pre-selection of promising mode sequences. To introduce this pre-selection procedure and its integration into an automated hybrid control framework, a very simple formulation will be presented first. Then some drawbacks of this basic formulation are pointed out and we present how qualitative pre-selection can be performed more efficiently.

The basic reason why we propose to utilize *qualitative* pre-selection of promising mode-sequences as first stage in hybrid control is that the hybrid control task is solved more easily when treated at a level of abstraction that only utilizes discretely-valued variables. This enables to reformulate investigation of different qualitatively abstracted hybrid trajectories as finite *constraint satisfaction problem* (CSP) or to look for such trajectories by discrete search among a *finite* number of possibilities. Efficient solvers from the field of artificial intelligence (AI) can be applied to such formulations.

To demonstrate this on our example, we define the control task to quickly traverse the continuous state of the hybrid system from its initial state $\mathbf{x}_0 = [-0.75, 0.75]^T$ towards the goal state $\mathbf{x}_g = [-0.75, -0.75]^T$ without violating the constraints on state (2.5) and input (2.4). This task shall be solved by a receding horizon control strategy that, at each time t_k, evaluates finite $N = 3$ time-step trajectories ($t_k \rightarrow t_{k+3}$) and determines actuation with respect to these.

For qualitative pre-selection, not only the hybrid model but also the hybrid control task has to be abstracted to the qualitative domain. The bounding constraints on state and input are already captured by the qualitative model as only trajectories satisfying the constraints are included in the model. The abstracted control goal can be formulated qualitatively as: Quickly traverse the hybrid system from its qualitatively abstracted initial state $X_0 = \xi_1$ near the abstracted goal state $X_g = \xi_5$.

To evaluate the 'distance' of a qualitative state X_k from its reference value, we introduce reference-cost variables $C_{R,k}$. These reference-costs $C_{R,k}$ are then – in a weighted sense – utilized together with the previously introduced likelihood costs $C_{L,k}$ to determine the quality of qualitative trajectories. To specify the values for the reference-costs, each qualitative region $X_k = \xi_i$ receives a cost value $C_{R,k} = c_{Ri}$ associated to each qualitative region $X_k = \xi_i$ that corresponds to the minimum distance from any continuous state $\mathbf{x}_k$ qualitatively abstracted by $X_k = \xi_i$ to the goal state $\mathbf{x}_g$. This is shown in Figure 2.11.

At time t_0, qualitative pre-selection of a 'good' trajectory can now be formulated as discrete search problem for an assignment to qualitative variables

$$x_{d,0},\ X_0,\ x_{d,1},\ X_1,\ x_{d,2},\ X_2,\ x_{d,3},\ X_3.$$

Consecutively analyzing the qualitative model ((2.11) on page 16) for times $t_0 \rightarrow t_1$, $t_1 \rightarrow t_2$ and $t_2 \rightarrow t_3$ allows to determine possible abstractions $X_0 \ldots X_3$. A similar model that was not explicitly displayed would be utilized to model allowed mode changes, such that for example a mode change to $x_{d,k+1} = m_2$ is only possible, if $X_k = \xi_5$.

The qualitative trajectory 'nearest' to the goal is

$x_{d,0} = m_1$	$X_0 = \xi_1$		
$x_{d,1} = m_1$	$X_1 = \xi_3$	$C_{R,1} = 0.25$	(line 3 in (2.11))
$x_{d,2} = m_3$	$X_2 = \xi_5$	$C_{R,2} = 0$	(line 28 in (2.11))
$x_{d,3} = m_2$	$X_3 = \xi_5$	$C_{R,3} = 0$	(last line in (2.11))

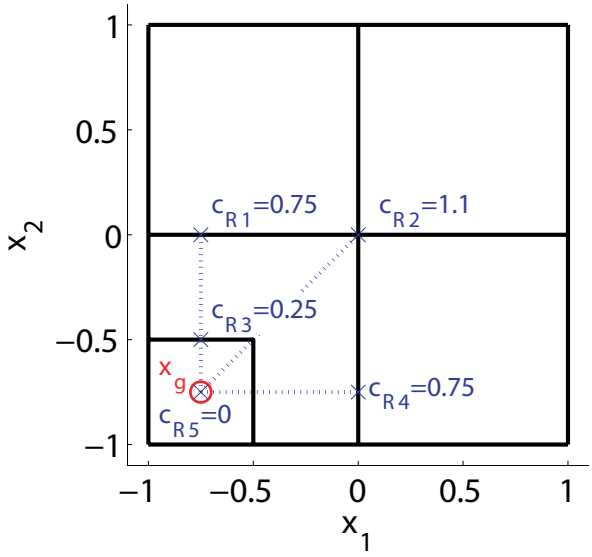


Figure 2.11: Cost values for deviations from the reference state

We choose to not consider likelihood costs here (weight them by 0), so this 'nearest' trajectory is the 'qualitatively best' one. It specifies a mode-sequence

$$x_{d,1} = m_1,\ x_{d,2} = m_3\ x_{d_3} = m_2$$

for the hybrid system. To solve the hybrid control task, however, continuous actuation has to be determined (numerically), yet.

To solve this task, for example *Model Predictive Control* (MPC) can be used. Determination of the continuous actuation u of the hybrid model (2.6, page 10) by MPC is rather straightforward, because with a specified mode sequence the hybrid model can be considered as standard *time-variant continuous model*, specifically:

$$\begin{aligned}
\mathbf{x}_1 &= \begin{bmatrix} 0.03 & -0.47 \\ 0.47 & 0.03 \end{bmatrix} \mathbf{x}_0 + \begin{bmatrix} 0.15 \\ 0.12 \end{bmatrix} u_0 \\
\mathbf{x}_2 &= \begin{bmatrix} 0.25 & 3.57 \\ -3.57 & 0.25 \end{bmatrix} \mathbf{x}_1 + \begin{bmatrix} 0.34 \\ -0.44 \end{bmatrix} u_1 \\
\mathbf{x}_3 &= \begin{bmatrix} 0.76 & 0.19 \\ 0.17 & 0.76 \end{bmatrix} \mathbf{x}_2 + \begin{bmatrix} -0.09 \\ -0.14 \end{bmatrix} u_2 \ .
\end{aligned}$$

However, although the above *qualitative* trajectory (and, thus, the mode sequence) is a valid trajectory according to the *qualitative* model, MPC fails. With this time-variant continuous model, no continuous actuation u_0, u_1, u_2 can be determined such that no constraint is violated. This is due to a well known and unavoidable problem related to qualitative modeling in general: A model that is obtained by abstracting all possible trajectories of a continuously valued model to qualitative trajectories may generally also allow *additional* qualitative trajectories which are *not* an abstraction of any trajectory of the original model. In literature, these additional qualitative behaviors are called *spurious behaviors* [36]. We will present how we are trying to avoid such spurious behaviors with high likelihood in the next section. However, since it isn't possible to generally avoid them with certainty, will will show

now how the hybrid modeling framework is able to handle situations when a qualitatively determined mode sequence does not allow a valid numerical solution.

If a pre-selected mode-sequence is based on such a spurious behavior, it won't be possible to numerically determine continuous actuation for the pre-selected time-variant model without violating any constraint. If such a violation of constraints is detected, qualitative pre-selection (i.e. search for the qualitatively next-best trajectory) is resumed. This new trajectory, specifies an alternative mode-sequence and, hence, another time-variant continuous model. This new model, again, is utilized to try to refine continuous actuation. If this model still is based on a pre-selected spurious behavior, successively new mode sequences are requested until one that allows a valid numerical solution is found.

In the example, the first mode-sequence that allows to a valid numerical solution is

$$x_{d,1} = m_1,\ x_{d,2} = m_1,\ x_{d,3} = m_3.$$

A valid continuous actuation that keeps the respective time-variant continuous model within its constraints is:

$$u_0 = 0,\ u_1 = 0,\ u_2 = 0.$$

After evolving one time step according to pre-selected mode $x_{d,1} = m_1$ and numerically determined actuation $u_0 = 0$, state

$$\mathbf{x}_1 = \begin{bmatrix} -0.38 \\ -0.33 \end{bmatrix}$$

is reached and qualitative pre-selection restarts form the (abstracted) hybrid state

$$x_{d,1} = m_1,\ X_1 = \xi_3.$$

The resulting trajectory that is obtained by the described receding horizon control strategy for the next few time steps is the same as obtained by the 'intuitive' solution from Section 2.2 and is displayed in Figure 2.5 on page 12.

The outlined approach to hybrid control can be summarized as an interplay between a qualitative pre-selection engine and subsequent numerical control deduction (MPC for a time-variant continuous model):

1. Qualitative abstraction of the current hybrid state and control goal
2. Qualitative pre-selection
   - Utilize a pre-compiled qualitative model to determine a hybrid trajectory close to the goal
   - The mode sequence specified by the qualitative trajectory allows to regard the hybrid model as time-variant continuous model.
3. Numerical refinement of the continuous actuation
   - Try to determine continuous actuation for the time-variant continuous model such that no constraints are violated
   - If a necessary violation of constraints is detected, go back to 2 and request pre-selection of the qualitatively next best trajectory
4. Hybrid Evolution

- Apply command inputs (determined during qualitative pre-selection) and continuous actuation (determined by numerical refinement) to the hybrid system
- At next sample time: restart at 1.

This is illustrated graphically in Figure 2.12.

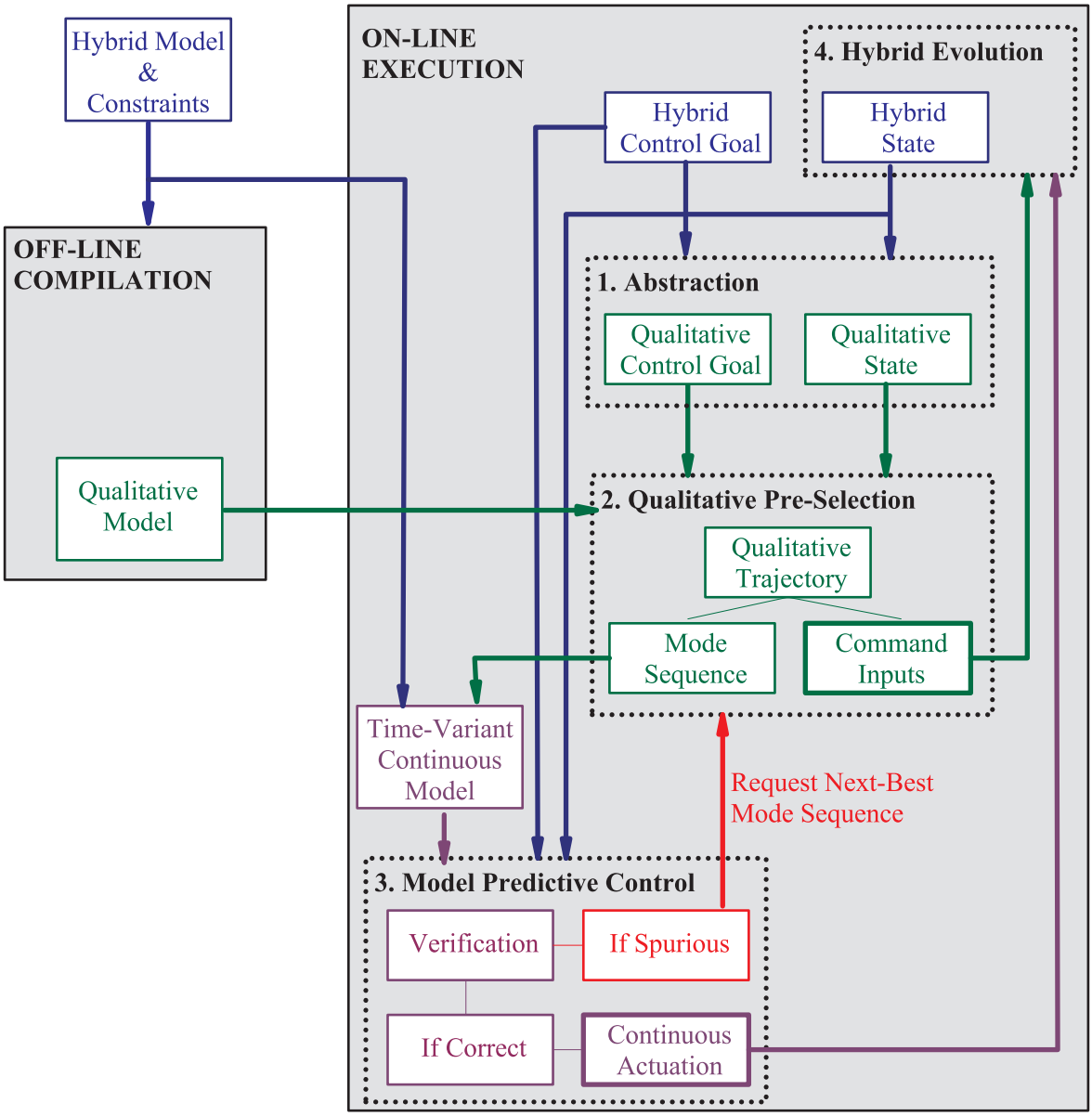


Figure 2.12: Hybrid control scheme

2.5 Qualitative Pre-Selection

The benefit of performing hybrid control through qualitative pre-selection is that an abstracted, thus simplified, description of the hybrid system's behavior can be analyzed more

easily than the detailed numerical model. Qualitative pre-selection only has to select a good solution among a *finite* number of possible discrete trajectories.

However, the number of these trajectories can be quite large for complex systems. So sophisticated search strategies have to be utilized and a well suited problem formulation has to be determined.

Additionally, there is always the risk of encountering spurious behaviors that provide mode-sequences which do not allow valid solutions to subsequent numerical refinement of continuous actuation. In principle, the outlined control scheme can cope with such a situation as shown above. However, numerical verification of a pre-selected qualitative trajectory needs some computational effort. So one should not have to try too many different mode sequences until a satisfactory solution is found. This requires to avoid spurious behaviors with high likelihood.

Like before, this introductory chapter won't go into detail and formalism here, but just highlights the most important concepts of the qualitative pre-selection scheme. First, we discuss how we can modify the qualitative pre-selection as outlined above such that *spurious behaviors* are avoided with high likelihood. In this hybrid control scheme, possible spurious behaviors emerge from the fact that only single-time step transitions between qualitative states are included in the qualitative model, but qualitative pre-selection needs to reason about longer trajectories. Additionally, the trajectories of the hybrid model do start from a specific numerical initial state. For example, the qualitative model 2.11 allows a trajectory

$$x_{d,0} = m_1, X_0 = \xi_1, x_{d,1} = m_1, X_1 = \xi_3, x_{d,2} = m_3, X_2 = \xi_5. \tag{2.14}$$

However, with the specific initial state $\mathbf{x}_0 = [-0.75; 0.75]^T$ such a trajectory is not possible. This is illustrated in Figure 2.13. Trajectories that lead from ξ_3 to ξ_5 with mode m_3 may only start from a very small region inside ξ_3 and this region cannot directly be reached from the initial state. When one considers the very small size of the region in ξ_3 that allows the state-transition to ξ_5, it seems rather likely that a qualitative trajectory containing this qualitative transition might be spurious.

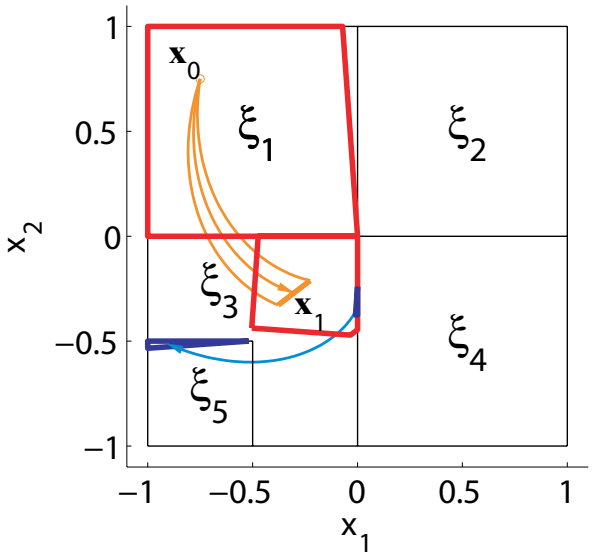


Figure 2.13: Possible ranges for state-transitions $\xi_1 \rightarrow \xi_3$ (m_1) and $\xi_3 \rightarrow \xi_5$ (m_3).

In the presented qualitative model (2.12), this is captured by the likelihood-cost value C_L associated to each trajectory. Considering these likelihood costs C_L together with the reference costs C_R for evaluating the 'quality' of a qualitative trajectories can largely help in avoiding spurious behaviors.

Still, avoiding spurious behaviors alone is not enough, as the (though finite) number of possible qualitative trajectories may be quite large. It is necessary to utilize an efficient search strategy to quickly select among all possible qualitative trajectories. As a result of this pre-selection, in principle, one is only interested in the qualitatively *best* (i.e. near the goal and most probably non spurious) trajectory. Thus, so-called *best-first search* is utilized. These search-strategies explore the search space by starting with an empty assignment of qualitative variables and then consecutively assigning values to the different variables until a solution is found. As search intends to find the best solution first, always only the best of the (partial) assignments is explored for one more variable. This basic idea is illustrated by the expansion of a search tree in Figure 2.15a-e, where the graph shown in Figure 2.14 is explored for the shortest path from node A to D.

Still, search can be performed more efficiently by A^*-search, a sophisticated variant of best-first search that utilizes ideas from dynamic programming (DP) [11] to stop further exploring assignments that won't ever turn out to be better than others. To give an example, in Figure 2.16 both assignments

$$a = 0, b = 1 \text{ with } C = 3$$
$$a = 1, b = 1 \text{ with } C = 4$$

lead to the same node in the graph. Therefore, both will only have the same possible extension ($c = 1$). So it doesn't make sense to investigate the worse of the two ($a = 1$, $b = 1$) any further. This is presented in Figure 2.17a-d.

2.6 Chapter Summary

Numerical control deduction for hybrid systems is a demanding task. Intuitively solving a hybrid control task by taking a qualitative view on the dynamics of a simple example hybrid model motivated the idea to utilize qualitative pre-selection of mode-sequences as an approach to hybrid control.

This approach splits up the task of on-line hybrid control deduction into two separate sub-tasks, each of which can be solved more easily than the hybrid control task itself, because each of the sub tasks operates only in one domain – discrete or continuous. Specifically, these sub-tasks are qualitative pre-selection of a sequence of operational modes and numerical refinement of continuous actuation.

With a pre-selected mode sequence, the hybrid control task can be regarded as control of a standard time-variant continuous model. This makes numerical deduction of continuous actuation much easier, as no discretely valued variables have to be considered.

Further, qualitative pre selection only operates on a simplified, symbolic representation of the hybrid system's behavior. This allows efficient tools for solving discrete-search problems to be applied.

However, taking a qualitative (abstracted) view of the hybrid system's operation introduces the problem of spurious behaviors. These are trajectories predicted by the simplified qualitative model which are not possible for the detailed numerical model. Such wrong predictions lead to an impossibility of numerical deduction of continuous actuation.

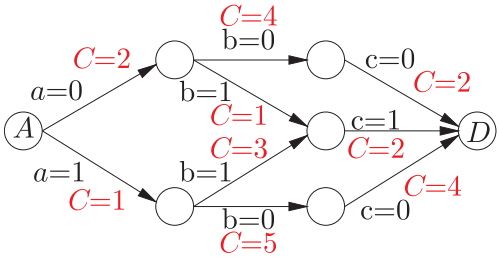
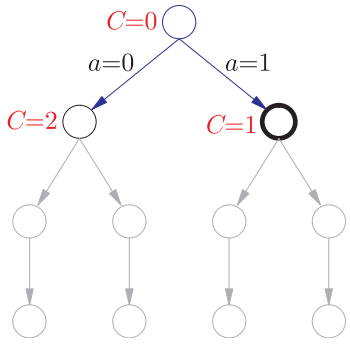
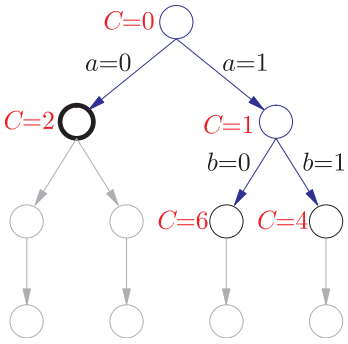


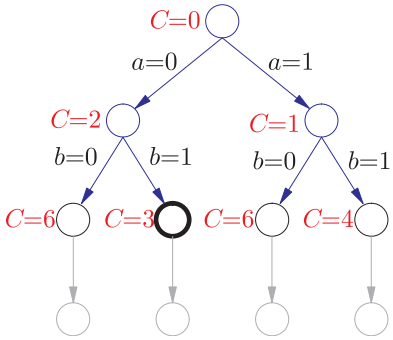
Figure 2.14: Shortest path from A to D?



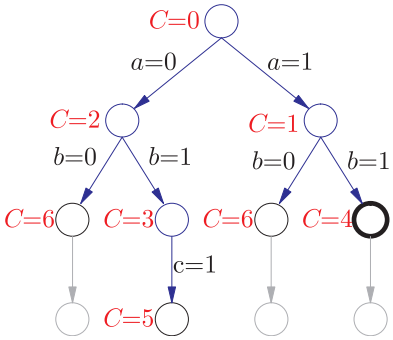
(a) Start from empty assignment and explore a



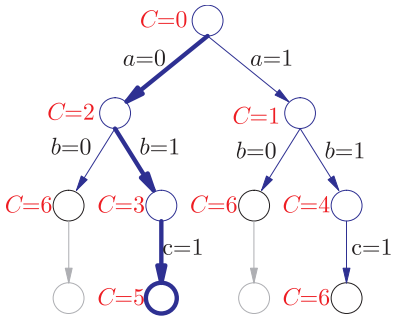
(b) Explore best unexpanded node further



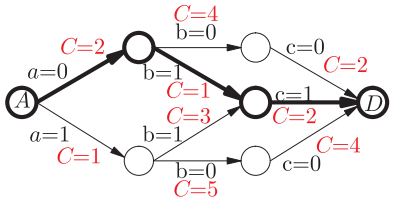
(c) Continue exploration from best node



(d) A path $A \rightarrow D$ is found, but there are better nodes



(e) The best node specifies a path path $A \rightarrow D$



(f) Best path

Figure 2.15: Best first search for shortest path

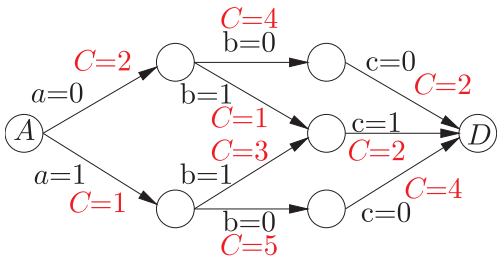
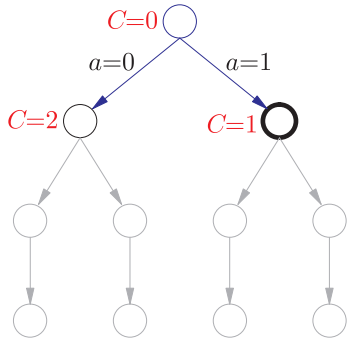
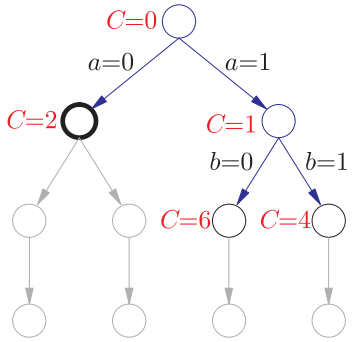


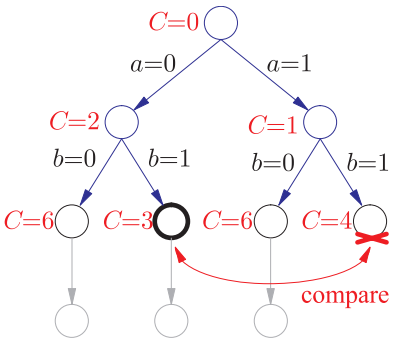
Figure 2.16: Again: shortest path from A to D?



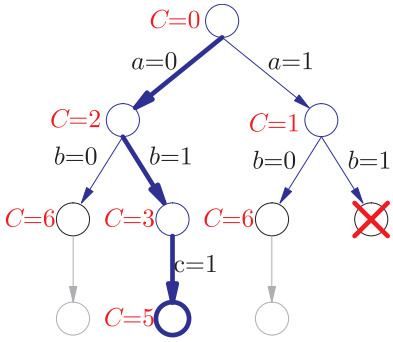
(a) Start from empty assignment and explore a



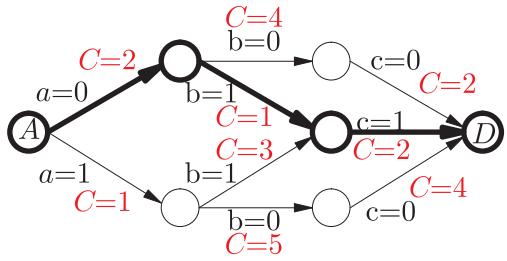
(b) Explore best unexpanded node further



(c) Two explored paths lead to the same node in the graph



(d) Best path $A \rightarrow D$ is found



(e) Best path

Figure 2.17: Best first search with dynamic programming

If such a spurious behavior is pre-selected, this impossibility is detected and qualitative pre-selection is requested to provide another mode-sequence. However, frequently predicting such spurious behaviors is prohibitive, as the numerical treatment of the time-variant continuous model – though much easier than hybrid control itself – still is computationally intensive.

So in this chapter a qualitative model was introduced that utilizes cost values to indicate whether a qualitative behavior is rather likely to be spurious or not. With this additional information available, qualitative pre-selection can be biased towards mode-sequences that allow trajectories that are both: close to a specified goal and non-spurious with high likelihood.

Additionally, the qualitative model is set up in a way that allows off-line compilation and compact storing. This saves computational resources for on-line qualitative pre-selection and numerical control refinement.

In this introductory chapter, simple examples were utilized to informally demonstrate the basic concepts of the proposed approach to automated on-line hybrid control. The subsequent chapters now provide formalization of these concepts and put the statements made in this introduction on a firm basis.

Chapter 3

Multi Component Hybrid Models

The proposed hybrid control scheme aims at control of multi-component systems that evolve in a hybrid continuous/discrete manner (hybrid systems). The state of these systems evolves according to different continuous dynamics for distinct modes of operation and this continuous evolution is interleaved with discrete changes of the operational mode.

This very general description of hybrid systems, however, allows a wide variety of different systems to be captured. Since these systems may be very different, developing a single framework that fits to all of them well isn't easy. Therefore, depending on the type of hybrid systems considered, in literature there exists a wide range of different hybrid modeling schemes, some putting more emphasis on continuous dynamics (e.g. [13, 31]), some emphasizing discrete dynamics (e.g. [1]).

With respect to continuous dynamics, in this thesis it is assumed that the underlying hybrid systems at each operational mode only exhibit continuous dynamics that can adequately be captured in terms of linear or affine difference and algebraic equations. Additionally, it is assumed that complexity of the behavior of the overall system mainly results from tight interaction among the various parts (components) of the system, whereas each component itself can be captured by a rather simple hybrid model.

Section 3.1 starts with a presentation of the hybrid automaton model that we use to model each of the components of a multi-component hybrid system. This hybrid model is a restricted variant of the hybrid automaton presented in [30, 31].

Section 3.2 shows how the overall model for a hybrid system is obtained by stringing together the various component automata.

Section 3.3 then provides formalization for both, the individual component automata as well as their interconnection. This overall model is captured as a concurrent hybrid automaton [30, 31].

3.1 Single Component Hybrid Automata

Systems composed of several components generally are difficult to describe by a single model since the number of different operational modes for the overall system can be very large. A system composed of 10 components each of which evolves according to 5 different modes of operation can show $5^{10} \approx 10.000.000$ different overall operational modes. However, a component based modeling approach only needs to record models for $5 \cdot 10 = 50$ different operational modes. The complexity of the behavior of the overall system is not explicitly represented in the model (making it unreasonably large) but is just an implicit consequence

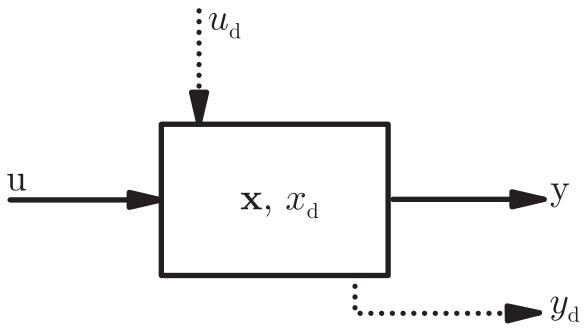


Figure 3.1: Inputs and outputs of a hybrid component model

of interactions among the much simpler component models.

Each of the components of the system is represented by a model that describes transitions of the operational mode (*discrete state*) x_d of the component as well as the dynamic evolution of its *continuous* internal *state* $\mathbf{x}$. These evolutions can be influenced by a discretely valued *command input* u_d and continuously valued actuation $\mathbf{u}$. The operation of the system can be observed by a discretely valued output y_d and continuously valued outputs $\mathbf{y}$ (Figure 3.1). This thesis does not deal with hybrid estimation [31], so full measurement of the discrete mode x_d and continuous state $\mathbf{x}$ is assumed throughout, such that

$$\begin{aligned} y_d &= x_d \\ \mathbf{y} &= \mathbf{C}_i \cdot \mathbf{x} \qquad \mathbf{C}_i = \left[\begin{array}{c} \mathbf{I} \\ \hline \overline{\mathbf{C}}_i \end{array} \right]. \end{aligned} \tag{3.1}$$

Here, $\mathbf{I}$ is the identity matrix and $\overline{\mathbf{C}}_i$ determines additional outputs depending on the operational mode $x_d = m_i$. This is a restriction with respect to the hybrid modeling framework in [31], but it shall be relaxed in the future.

3.1.1 Discrete Evolution

To build a model for a component of a hybrid system, first its operational modes m_i have to be identified. For example, these could represent the system being operating in standby-mode, operating normally or in some specific faulty manner.

Next, the conditions under which the operational mode of the system changes have to be determined. It is assumed that these mode-changes only can take place at the sample times

$$t_k = t_0 + k \cdot T_s \tag{3.2}$$

of the digital control, where t_0 is the initial time and T_s specifies the sampling period. These mode transitions can either be so-called *commanded transitions* that only depend on the discrete command input u_d or *autonomous transitions* that additionally depend on the current continuous inputs $\mathbf{u}$ and the system's dynamic state $\mathbf{x}$. The expressions that determine when a transition is taken are called the *transition guards*. Not to introduce ambiguity, these transition guards have to be formulated such that in any case only one of the expressions guarding all transitions leading 'away' from a particular mode may evaluate to 'true'.

As example, three different values of the command input $u_d = \{\texttt{turn on}, \texttt{turn off}, \texttt{reset}\}$ are used to switch a system on and off and to recover from faults, respectively. Additionally, the continuous input $\mathbf{u}$ dropping below a bound $\mathbf{u} < \mathbf{0}$ autonomously leads to the specified

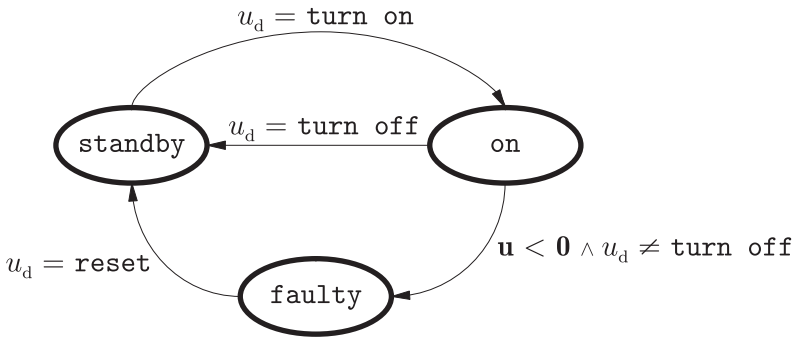


Figure 3.2: Commanded mode transitions and an autonomous one

fault in the system. These mode-transitions can be expressed in terms of a graph like Figure 3.2. We observe that the transition leading to the fault mode must additionally be guarded by an expression $u_d \neq \texttt{turn off}$ because otherwise two guards would evaluate to true if $u_d = \texttt{turn off}$ and $\mathbf{u} < \mathbf{0}$. Whenever the guards of all transition edges leading away from the current mode evaluate to false, the so-called *no-transition* is taken and the operational mode remains unchanged.

As far as mentioned so far, the model only includes transitions that are certain. That means, given the current discrete state x_d, continuous state $\mathbf{x}$, command input u_d and continuous actuation $\mathbf{u}$ the model exactly specifies the subsequent operational mode. To allow more general models, ambiguous transitions that lead to multiple follow-up states are included in the modeling framework. This can be utilized to specify probabilistic distributions among several possible target modes of a transition.

Again for the example, turning the system in Figure 3.2 on could be successful only with a probability of 95%. 5% of the times when one attempts to turn on the system, it just stays at standby mode. Such probabilistic ambiguities can graphically be expressed by splitting up a transition edge after the associated guard into multiple arcs that lead to different target modes. These arcs are labeled according to their probability such that the sum of probabilities over all arcs sums up to 1. For the example system, such a non-deterministic transition is displayed in Figure 3.3.

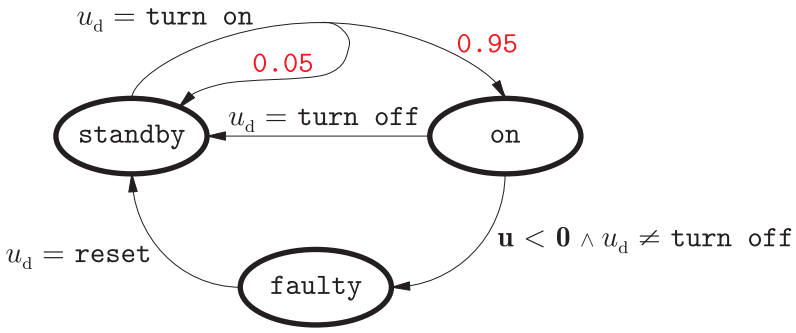


Figure 3.3: Non-deterministic transition targets

3.1.2 Continuous Evolution

The dynamic system at each discrete state $x_d = m_i$ is captured in terms of its continuously valued state

$$\mathbf{x} = [x_1, \ldots, x_{n_x}]^T$$

and receives continuously valued inputs

$$\mathbf{u} = [u_1, \ldots, u_{n_u}]^T .$$

Its operation can be observed through the outputs

$$\mathbf{y} = \left[y_1, \ldots, y_{n_y}\right]^T$$

and is described in terms of an affine model

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}_{ci}\,\mathbf{x} + \mathbf{B}_{ci}\,\mathbf{u} + \mathbf{e}_{ci} \tag{3.3a}$$

$$\mathbf{y} = \mathbf{C}_{ci}\,\mathbf{x} + \mathbf{D}_{ci}\,\mathbf{u} + \mathbf{f}_{ci}, \tag{3.3b}$$

where the index i of the constant matrices $\mathbf{A}_{ci}$, $\mathbf{B}_{ci}$, $\mathbf{C}_{ci}$, $\mathbf{D}_{ci}$, and the constant vectors $\mathbf{e}_{ci}$, and $\mathbf{f}_{ci}$ indicates the dependence upon a particular operational mode m_i.

Why Affine models

The choice to model the dynamic system by a set of affine models is a restriction compared to [30], where more general non-linear models are allowed. However with the intended task of run-time hybrid control in mind, this restriction in introduced to be able to rely on simpler methods for numerical controller design (linear Model Predictive Control for time-variant systems) than would be necessary in the more general non-linear case.

It has to be noticed, however, that such nonlinear models

$$\frac{d}{dt}\mathbf{x} = f_x(\mathbf{x}, \mathbf{u}) \tag{3.4a}$$

$$\mathbf{y} = f_y(\mathbf{x}, \mathbf{u}), \tag{3.4b}$$

where $f(\cdot)$ stands for a differentiable function, can (arbitrarily close) be approximated by a set of affine models that are obtained by a Taylor-Series expansion (Appendix A.1) of the function around a number of (continuous) operational points

$$\langle \mathbf{x}_{OP}, \mathbf{u}_{OP} \rangle .$$

If that operational point was a *steady state*

$$\langle \mathbf{x}_{OP}, \mathbf{u}_{OP} \rangle = \langle \mathbf{x}_{SS}, \mathbf{u}_{SS} \rangle$$

of the system, that means that

$$f_x(\mathbf{x}_{SS}, \mathbf{u}_{SS}) \stackrel{!}{=} 0,$$

Taylor series-expansion of f_x and f_y would lead to constant vectors $\mathbf{e}_{cSS} = \mathbf{0}$ and $\mathbf{f}_{cSS} = \mathbf{0}$ equal to zero. So, as a result of linearization one would obtain a linear model

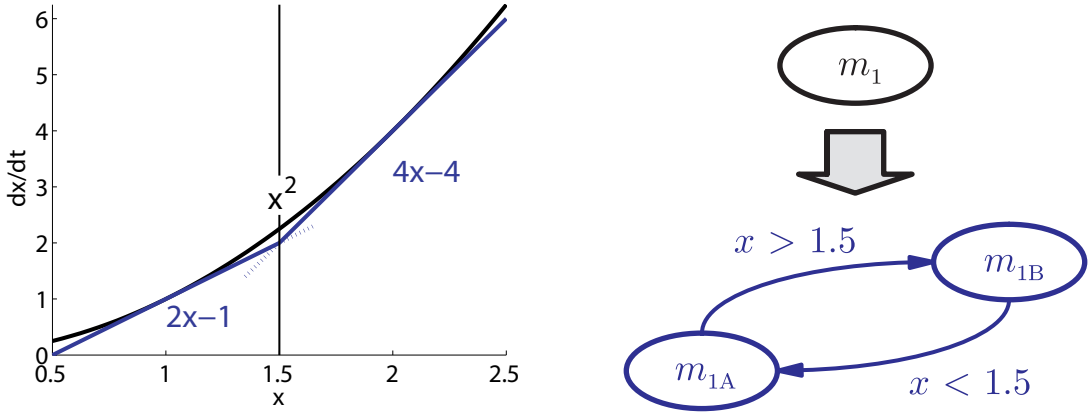
$$\frac{d}{dt}\mathbf{x} = \mathbf{A}_{cSS}\,\mathbf{x} + \mathbf{B}_{cSS}\,\mathbf{u}$$

$$\mathbf{y} = \mathbf{C}_{cSS}\,\mathbf{x} + \mathbf{D}_{cSS}\,\mathbf{u}.$$

The limitation to linearization around steady states, however, is over-restrictive. Control tasks may require operation of the systems far away from these particular steady state points. In this case, the linear models cannot adequately describe the system behavior and would thus lead to degraded controller performance.

If, however, linearizations around several non-steady states are available as well, the controller can always rely on an affine model that closely approximates the non-linear continuous dynamics of the system.

In a hybrid modeling sense the non-linear dynamics (3.4) of a system at a particular mode m_i are represented as set of affine models, each of which is valid in a certain region of the state-space (*Piecewise affine model* [48]). For this, the operational mode m_i of the non-linear system is 'split up' into several new operational modes m_j, each of which is associated with one of the affine linearizations. Transitions among these new discrete states are guarded by the borders of the corresponding regions of the state space.



(a) Linearized approximations

(b) Separation into new operational modes

Figure 3.4: Approximation of non-linear dynamics by linearization

This is illustrated in Figure 3.4, where the model

$$x_d = m_1 : \frac{d}{dt}x = x^2$$

is linearized around operational points $x_{OP1} = 1$ and $x_{OP2} = 2$ to obtain affine models

$$x_d = m_{1A} : \frac{d}{dt}x_c = 2x - 1$$
$$x_d = m_{1B} : \frac{d}{dt}x_c = 4x - 4.$$

The former one better approximates the non-linear model as long as $x < 1.5$, whereas the latter is the better approximation for continuous states $x > 1.5$.

The linearization around the steady-state $x_{SS} = 0$, i.e.

$$dx(t)/dt = 0$$

certainly does not suffice to approximate the non-linear behavior in the region depicted in Figure 3.4a.

However, although a set of affine linearizations can closely approximate the function $f_x = x^2$, clearly no linearization can capture the finite-escape time property

$$\frac{d}{dt}x(t) = x(t)^2 \;\rightarrow\; x(1/x_0) = \infty$$

of this non-linear model.

Approximating non-linear continuous dynamics by a set of affine models certainly makes the hybrid model's discrete dynamics more complex by introducing new discrete states and new autonomous transitions, i.e. transitions that are not commanded by particular discrete inputs but that are imposed by the continuous dynamics of the system itself. So the choice of the number of operational points x_{OPi} for obtaining linearizations has to be a compromise between quality of approximation and increase in complexity of the discrete dynamics.

Discrete time operation

The model (3.3) is defined in a continuous-time manner. The intended on-line hybrid controller, however, operates in discrete time and calculates new controls only at times

$$t_k = k \cdot T_s, \tag{3.5}$$

where T_s denotes the sampling period of the hybrid controller. The continuous inputs $\mathbf{u}$ are kept constant within the sampling period at

$$\mathbf{u}(t) = \mathbf{u}_k \quad t_k \le t < t_{k+1}. \tag{3.6}$$

Similarly,

$$\begin{aligned} \mathbf{x}_k &= \mathbf{x}(t_k) && \text{(3.7a)} \\ x_{d,k} &= x_d(t_k) && \text{(3.7b)} \\ \mathbf{y}_k &= \mathbf{y}(t_k) && \text{(3.7c)} \end{aligned}$$

denote sampled continuous state, discrete state and output respectively.

With the transition matrix

$$\Phi_i(T_s) = e^{\mathbf{A}_{ci}\,T_s} \tag{3.8}$$

the general solution to the dynamic model (3.3) at time $t = T_s$ is

$$\mathbf{x}(T_s) = \Phi_i(T_s) \cdot \mathbf{x}(0) + \int_0^{T_s} \Phi_i(T_s - \tau) \cdot \mathbf{B}_{ci} \cdot \mathbf{u}(\tau)\, d\tau + \int_0^{T_s} \Phi_i(T_s - \tau) \cdot \mathbf{e}_{ci}. \tag{3.9}$$

Since with the above conventions $\mathbf{u}(\tau) = \mathbf{u}_0$ is constant during the interval $0 \le t < T_s$ we can rewrite this equation in the form

$$\mathbf{x}_1 = \mathbf{A}_i\,\mathbf{x}_0 + \mathbf{B}_i\,\mathbf{u}_0 + \mathbf{e}_i \tag{3.10}$$

where, after substitution of variables, the matrices $\mathbf{A}_i$, $\mathbf{B}_i$ and vector $\mathbf{e}_i$ are calculated as

$$\begin{aligned} \mathbf{A}_i &= \Phi(T_s) && \text{(3.11a)} \\ \mathbf{B}_i &= \int_0^{T_s} \Phi_i(\tau)\mathbf{B}_{ci}\, d\tau && \text{(3.11b)} \\ \mathbf{e}_i &= \int_0^{T_s} \Phi_i(\tau)\mathbf{e}_{ci}\, d\tau. && \text{(3.11c)} \end{aligned}$$

In the same way, $\mathbf{x}_{k+1}$ can be calculated from $\mathbf{x}_k$ and $\mathbf{u}_k$ and we get the discrete-time model

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}_i\,\mathbf{x}_k + \mathbf{B}_i\,\mathbf{u}_k + \mathbf{e}_i && \text{(3.12a)} \\ \mathbf{y}_k &= \mathbf{C}_i\,\mathbf{x}_k + \mathbf{D}_i\,\mathbf{u}_k + \mathbf{f}_i && \text{(3.12b)} \end{aligned}$$

Sampling does not change the output-equation of (3.3), so we have

$$\begin{aligned} \mathbf{C}_i &= \mathbf{C}_{ci} && \text{(3.13a)} \\ \mathbf{D}_i &= \mathbf{D}_{ci} && \text{(3.13b)} \\ \mathbf{f}_i &= \mathbf{f}_{ci}. && \text{(3.13c)} \end{aligned}$$

Due to the discrete-time operation of the hybrid controller, the modeling framework only allows discrete time models (3.12) to capture the hybrid system's continuous dynamics at each operational mode. These discrete time models do not necessarily have to result from sampling of continuous-time system behavior (3.11), but, of course, can also describe more general systems which, themselves, operate in discrete-time.

3.1.3 Hybrid execution

With the utilized modeling framework, changes in continuous actuation $\mathbf{u}$ as well as changes in the operational mode x_d can only take place at times t_k. For the discrete state x_d, standard hybrid systems notation

$$x_{d,k+1} = x_d(t) \quad t_k < t \leq t_{k+1} \tag{3.14}$$

is used. Note here the difference in index numbering and inequality signs to the input

$$\mathbf{u}_k = \mathbf{u}(t) \quad t_k \leq t < t_{k+1}. \tag{3.15}$$

Following these conventions

- the continuous state $\mathbf{x}_k$, the discrete state $x_{d,k}$ and the output $\mathbf{y}_k$ are sampled at times t_k,
- instantly, new controls $\mathbf{u}_k$ and $\mathbf{u}_{d,k}$ are calculated right at t_k as well,
- continuous actuation is kept constant at $\mathbf{u}_k$ for the sampling period.
- Further, at t_k the guards of possible mode transitions are evaluated and possibly enforce a mode change so that $x_d(t) = x_{d,k+1}$ *immediately after* the sampling time t_k.

This mode change takes place in an infinitely short time ϵ and we denote

$$t_k' = t_k + \epsilon.$$

The hybrid system, then, evolves according to this new discrete state $x_{d,k+1}$ until the next sampling time t_{k+1}.

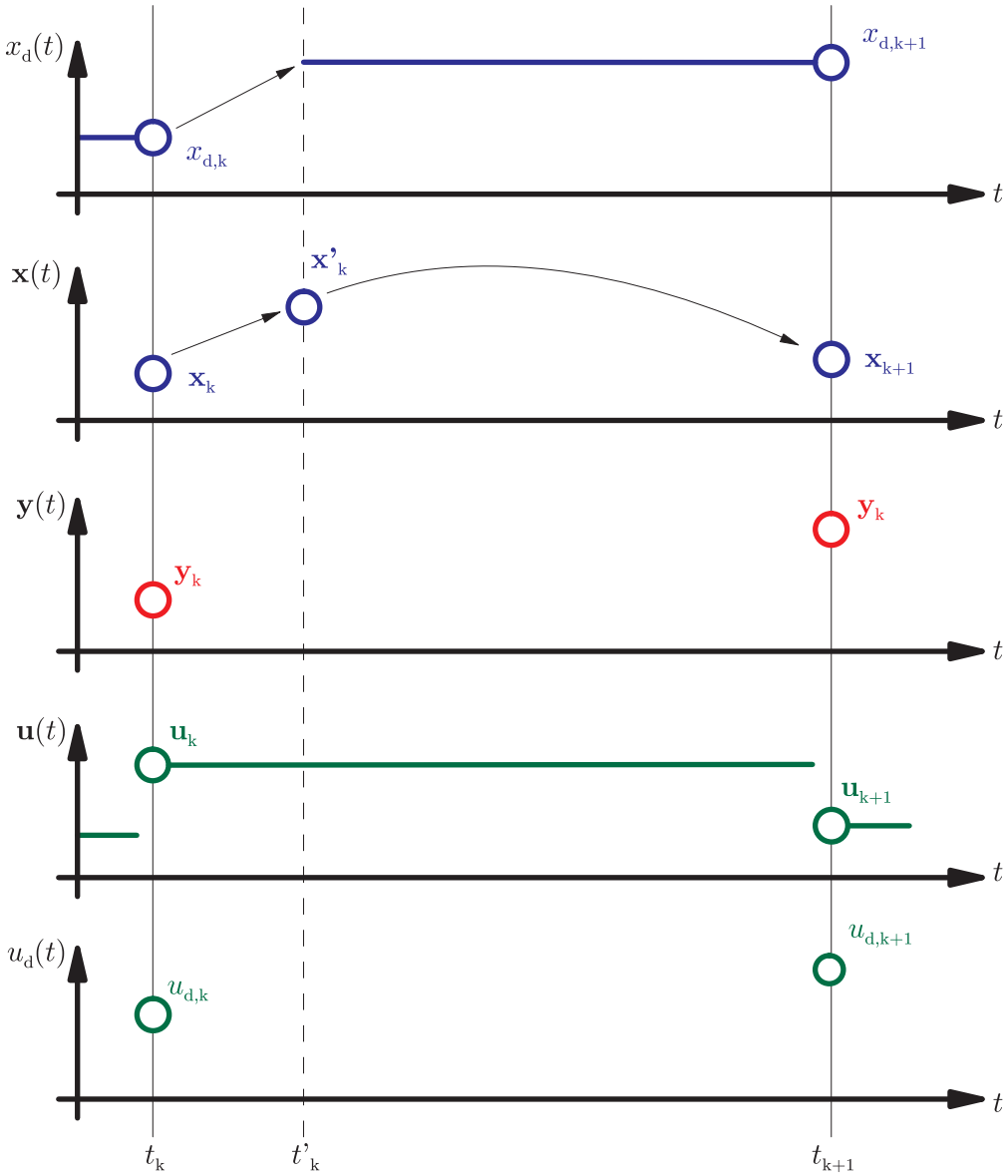


Figure 3.5: Hybrid execution

Discontinuities in the continuous state x

Apart from continuous evolution of the continuous state $\mathbf{x}$ *during* a sampling period (i.e. from time t_k to t_{k+1}), additionally instant *resets* of the continuous state $\mathbf{x}$ are allowed, whenever a transition between operational modes takes place. Like the operational mode, the continuous state $\mathbf{x}$ changes from time t_k to time t'_k. This is modeled as

$$\mathbf{x}'_k = \mathbf{R}_{ji}\,\mathbf{x}_k + \mathbf{r}_{ji},$$

where indices j and i indicate the transition from discrete states $x_{d,k} = x_{d,j}$ to $x_{d,k+1} = x_{d,i}$. During the time

$$t'_k \leq t \leq t_{k+1}$$

the continuous state $\mathbf{x}$ evolves from $\mathbf{x}'_k$ to $\mathbf{x}_{k+1}$ according to the dynamics imposed by discrete state $x_{d,k+1}$ and the continuous input $\mathbf{u}_k$.

Introduction of this new intermediate continuous state $\mathbf{x}'_k$ forces a modification to the model (3.12). With the possible state-reset included, continuous evolution of the hybrid system during a sampling period can be modeled as:

$$\mathbf{x}'_k = \mathbf{R}_{ji}\,\mathbf{x}_{c,k} + \mathbf{r}_{ji} \tag{3.16a}$$
$$\mathbf{x}_{k+1} = \mathbf{A}_i\,\mathbf{x}'_k + \mathbf{B}_i\,\mathbf{u}_k + \mathbf{e}_i \tag{3.16b}$$
$$\mathbf{y}_{k+1} = \mathbf{C}_i\,\mathbf{x}_{k+1} + \mathbf{D}_i\,\mathbf{u}_{k+1} + \mathbf{f}_i \tag{3.16c}$$

A time-line illustration of the evolution of that hybrid model is presented in Figure 3.5.

3.2 Concurrent Hybrid Automaton

To be able to capture more complex systems composed of several components, the hybrid models for each of these components have to be combined to form an overall model. To keep the modeling-complexity manageable even for larger compositions, no single hybrid automaton for the overall system shall be developed. Instead, the overall model basically sticks to the component models and only defines their interconnection and their connection to the outside world. An illustration of an interconnection of two hybrid components is shown in Figure 3.6.

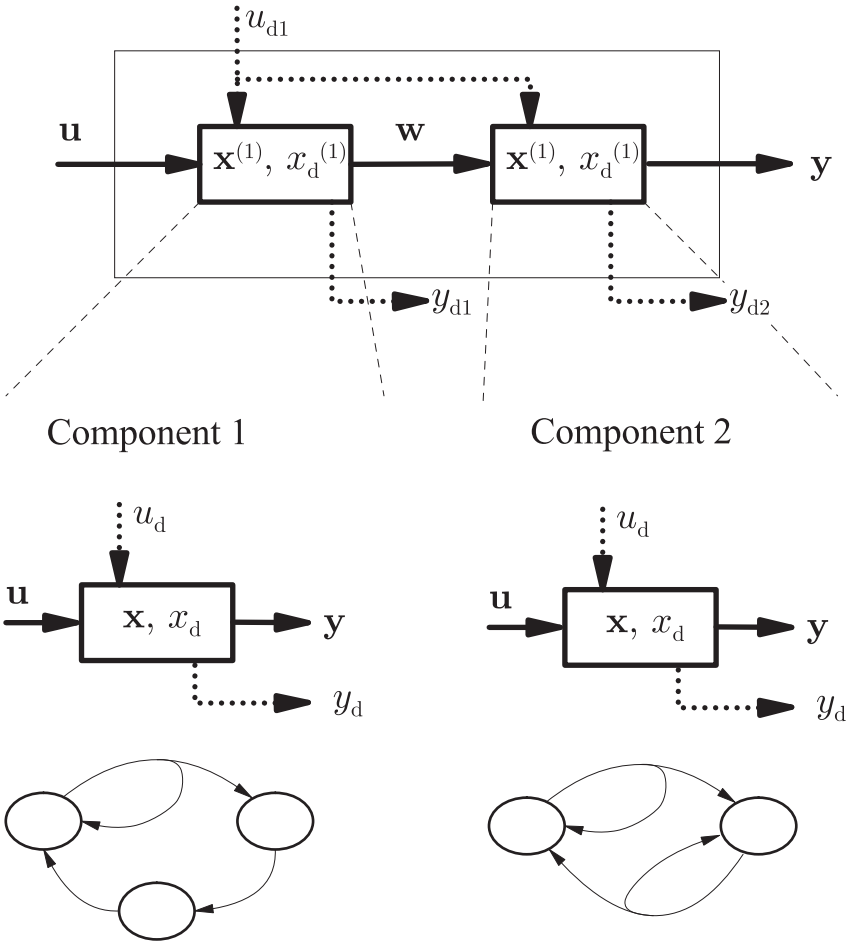


Figure 3.6: Interconnection of two hybrid components

In terms of notation, $\mathbf{u}$ is used to denote the continuous actuation that can be applied to the overall system from the outside world. Likewise, $\mathbf{y}$ provides the the continuous outputs to the outside world and $\mathbf{w}$ denotes intermediate variables that connect various components of the overall model but do not have a connection to the outside. Similarly, $\mathbf{u}_d$ and $\mathbf{y}_d$ represent the command inputs and discrete outputs of the overall model.

For notational distinction between inputs and outputs of the overall model and inputs and outputs of the specific components, a superscript (i) will be used to denote the variables used in the model for component i. For example $\mathbf{x}^{(i)}$ denotes the continuous state of the i'th component of the overall model.

The interconnections of the overall model are modeled by specifying which variables of each component model are represented by which variable in the overall model. For example for the system depicted in Figure 3.6 this interconnection specification looks like:

	determined by...	provided to...
$\mathbf{u}$	ext.	$\mathbf{u}^{(1)}$
$\mathbf{w}$	$\mathbf{y}^{(1)}$	$\mathbf{u}^{(2)}$
$\mathbf{y}$	$\mathbf{y}^{(2)}$	ext.
u_d	ext.	$u_d^{(1)}, u_d^{(2)}$
y_{d1}	$y_d^{(1)}$	ext.
y_{d2}	$y_d^{(2)}$	ext.

3.3 Formalization of the Hybrid Modeling Framework

3.3.1 Hybrid Automaton

The hybrid model describing a component of the overall system can formally be expressed very similarly to a discrete-time *probabilistic hybrid automaton* (PHA) [30, 31].

Each component is modeled by a *hybrid automaton* (HA) $\mathcal{A}$. This automaton is formally expressed as tuple

$$\mathcal{A} = \langle \mathbf{x}_h, \mathbf{u}_h, \mathbf{y}_h, F, T, \mathbf{X}, \mathbf{U}, \mathcal{X}_d, \mathcal{U}_d, T_s \rangle : \tag{3.17}$$

- the *hybrid state*

  $$\mathbf{x}_h = \langle x_d, \mathbf{x} \rangle$$

  is composed of the discrete mode x_d with finite domain

  $$x_d \in \mathcal{X}_d = \{m_1, \ldots, m_l\}$$

  and the continuous state $\mathbf{x}$ with constrained state-space

  $$\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^{n_x}.$$

  The constrained state space is defined as polytope in $\mathbb{R}^{n_x}$:

  $$\mathbb{X} = \{\mathbf{x} \mid \mathbf{M}_x \mathbf{x} < \mathbf{m}_x\}$$

- the *input*

  $$\mathbf{u}_h = \langle u_d, \mathbf{u} \rangle$$

  is composed of a finite set of discrete commands

  $$u_d \in \mathcal{U}_d$$

and limited continuous actuation

$$\mathbf{u} \in \mathbb{U} \subset \mathbb{R}^{n_u},$$

where

$$\mathbb{U} = \{\mathbf{u} \mid \mathbf{M}_u\, \mathbf{u} < \mathbf{m}_u\}$$

- the *output*

$$\mathbf{y}_h = \langle y_d, \mathbf{y} \rangle$$

provides direct observation of the hybrid state $\mathbf{x}_h$ plus possible additional continuous outputs. (These can, later, serve as input in other components).

- *continuous evolution* of the hybrid automaton is specified by a function F that relates an affine model

$$\begin{aligned} \mathbf{x}_{k+1} = &\quad \mathbf{A}_i \mathbf{x}'_k + \mathbf{B}_i \mathbf{u}_k + \mathbf{e}_i \\ \mathbf{y}_k = &\quad \mathbf{C}_i \mathbf{x}_k + \mathbf{D}_i \mathbf{u}_k + \mathbf{f}_i \end{aligned} \tag{3.18}$$

with

$$\mathbf{C}_i = \left[\begin{array}{c} \mathbf{I} \\ \hline \mathbf{C}_i \end{array} \right]$$

to each discrete state $x_d = m_i \in \mathcal{X}_d$ of the automaton. If $\mathcal{F}$ denotes the set of all these models, F takes the form

$$F : \mathcal{X}_d \to \mathcal{F}$$

- *discrete evolution* among operational modes is specified by the definition of transitions τ_i. Each of these transitions is represented by a triple

$$\tau_i = \langle c_i, p_i, r_i \rangle$$

  - c_i denotes the guard of the transition. The guard is a boolean function in terms of boolean expressions $b(u_d)$ on the discrete command input u_d and polytopic constraints on the continuous state- and input-space

$$c_i = b(u_d) \wedge (\mathbf{P}_u\, \mathbf{u} < \mathbf{p}_u) \wedge (\mathbf{P}_x\, \mathbf{x} < \mathbf{p}_x)\,.$$

    A transition is taken if and only if the associated guard is true.
  - When a transition is taken, p_i specifies possible target modes as probability distribution among the set of discrete states $\mathcal{X}$.
  - Additionally, there is a function r_i associated to each of the target modes m_j of a transition that resets the continuous state to

$$\mathbf{x}'_k = \mathbf{R}_{ij}\, \mathbf{x}_k + \mathbf{r}_{ij}.$$

The function T associates a subset of these transitions τ_i to each operational mode. If $\mathcal{T}$ denotes the set of all transitions and $2^{\mathcal{T}}$ is the set of all subsets thereof, the function T can takes the form

$$T : \mathcal{X}_d \to 2^{\mathcal{T}}$$

- T_s denotes the sampling time of the automaton. This is only important, as in this hybrid modeling framework all automata that are connected have to have synchronous sampling.

3.3.2 Concurrent Automaton

Two hybrid automata $\mathcal{A}_1$ and $\mathcal{A}_2$ only can be composed within the modeling framework, if they operate synchronously with common sampling time T_s. The automata can only be connected via their continuous inputs and outputs. They do not share common states.

The hybrid automata presented above can only model single components. The composition of several such automata, however, does not yet specify the interconnections among them, nor does it specify the systems connection to the outside world (Figure 3.7). The in-

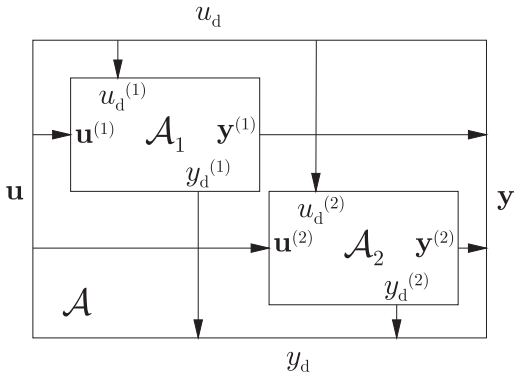


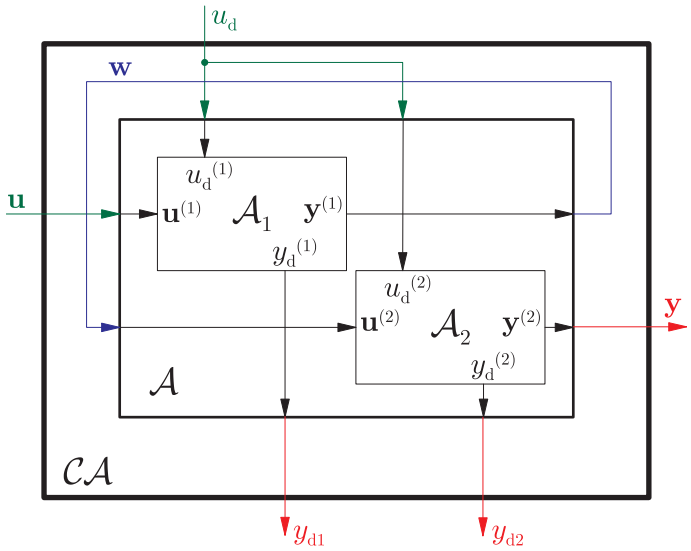
Figure 3.7: Composition of two hybrid automata

terconnection of the hybrid system's components and the system's connection to the outside world is specified by a concurrent hybrid automaton (cHA). Formally, this automaton $\mathcal{CA}$ is expressed as tuple

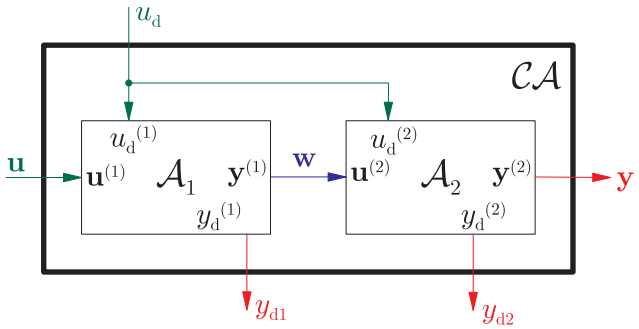
$$\mathcal{CA} = \langle \mathbf{u}_h, \mathbf{y}_h, \mathbf{w}_h, \mathcal{C}, \mathcal{A} \rangle : \tag{3.19}$$

- $\mathbf{u}_h = \langle \mathbf{u}_d, \mathbf{u} \rangle$ specifies the exogenous command inputs $\mathbf{u}_d$ and inputs $\mathbf{u}$ for continuous actuation provided to the automaton from the outside world.
- $\mathbf{y}_h = \langle \mathbf{y}_d, \mathbf{y} \rangle$ defines the outputs that can be observed from the outside world.
- $\mathbf{w}_h = \langle \mathbf{w}_d, \mathbf{w} \rangle$ specifies intermediate variables that do interconnect individual component automata $\mathcal{A}_i$ but have no connection to the outside world.
- Using these variables, the connection specification $\mathcal{C}$ determines the interconnection of the components by relating each individual input and output of each component to an individual input, output or intermediate variable of the concurrent automaton.
- $\mathcal{A}$ specifies the composition of the component-automata $\mathcal{A}_i$.
- Full measurement of the hybrid state $\mathbf{x}_h$ has to be assured for operation within the hybrid control framework.

An example specification of a concurrent automaton around composed component automata as in Figure 3.7 is shown in Figure 3.8a. A more intuitive display is given in Figure 3.8b.



(a) Composition of automata with specified interconnection



(b) Simpler display for the same concurrent automaton

Figure 3.8: Example of a concurrent automaton

3.4 Chapter Summary

This chapter presented the formalism that defines the class of hybrid models that can be controlled by the hybrid control scheme that will be formalized in the following chapters. This class of models captures complex system behavior as a set of interconnected hybrid automata. This representation of complex systems as several simpler components will be beneficial for qualitative modeling, because it allows to keep the qualitative models compact and allows to handle model complexity by efficient qualitative reasoning among the components.

Chapter 4

Automated Qualitative Abstraction

This chapter specifies automatic compilation of a compact qualitative model that is especially suited for efficient pre-selection of promising sequences of operational modes

The concurrent probabilistic hybrid automaton defined in Chapter 3 is a powerful tool to model complex systems that exhibit both, continuous and discrete dynamics. It gives a compact representation of systems composed of several components by relying on rather simple models for each component and putting those under a shell that defines their interconnection in a natural sense. However, this modeling framework utilizes different types of variables to model continuous and discrete dynamics of hybrid systems. While continuous dynamics are modeled as equations among continuously-valued variables, discrete changes of the operational mode of the system are captured in terms of an automaton that is described by discretely valued variables.

Chapter 2 gave a first motivation, how an abstract, *qualitative*, view on a hybrid system's behavior can help in solving control tasks. A (beforehand compiled) qualitative model is utilized to efficiently solve the control task on an abstracted level. The result of this qualitative control generation then specifies the discrete part of the hybrid control, leaving only the continuous part left to numerical control generation. This way, the complex *hybrid* control task is split into two much less complex ones: An abstracted control problem posed on a *qualitative* model and a standard numerical control problem posed on a purely (though time variant) *continuous* model.

Efficiency and accuracy of the qualitative pre-selection highly depend on form and accuracy of the qualitative model representing the hybrid system. This chapter introduces formalization of a qualitative modeling framework that is especially designed to be efficiently utilized for the intended pre-selection task. The chapter outlines the automated compilation of that qualitative model from the hybrid model.

Section 4.1 first details the requirements on the qualitative model that are needed to be met to utilize it in hybrid control. The following sections, then, present the details of automated qualitative modeling: Selecting qualitative variables and domains (Section 4.2) and compilation of a first qualitative model (Section 4.3). Section 4.4 finally introduces a specialized graphical representation of that model that fulfills all requirements relevant for utilization in hybrid control.

4.1 Model Outline

The qualitative model is intended to serve in on-line control of hybrid systems. I.e. the actuation that is provided to the system is generated from a model of the system at run-time. Model Predictive Control (MPC) [41] is an example of such an on-line control scheme for continuous systems.

For our multi-component hybrid systems, an on-line control strategy provides the advantage that it allows to react on the current 'situation' the system is in (e.g. system state, occurred faults, present constraints) without pre-specifying a separate controller for each operational mode of the overall system. Interactions among the various system components can result in an overwhelming number of overall modes for the system. Accounting for each of these modes in advance and specifying a distinct controller for each of them often is hardly manageable.

However, on the other hand, determination of the system's actuation at run-time has to be performed in 'real-time'. This means for the qualitative model, that it has to be compiled off-line. This, of course, demands that the qualitative model has to encompass *all* possible behaviors of the original hybrid model. This may seem as troublesome as compiling a separate controller for each operational mode, and it is certainly somewhat true if one only looks at the overall hybrid system as a whole. If, however, *each component* is abstracted *separately*, all abstracted behaviors of the hybrid system can be captured by a set of (in comparison) rather small qualitative models.

This compromise, however, adds new demands to the on-line pre-selection performed on the qualitative model. It is not possible to deduce the required discrete control on behalf of a single overall model, but the controller has to reason among several interacting component models. To make this reasoning more efficient, the qualitative component models will be represented in a structured sense that is founded on the structure (i.e. the interdependencies) of the system's components. This structured representation allows to simultaneously reason among the various component models almost as if they were a single overall model.

So far, requirements on the qualitative model can be summarized as:

- Constrained time for on-line control deduction requires off-line compilation of the qualitative model.
- Complexity of the overall system behaviors requires component based qualitative modeling.
- Required efficiency of on-line qualitative reasoning demands a well structured representation of the component models.

However, one major problem immanent to qualitative models in general has not been considered yet: If a detailed numerical model is abstracted to a qualitative one, it is generally unavoidable that the qualitative model will additionally model some behaviors that are *not* an abstraction of *any* behavior of the detailed model. These additional behaviors of the qualitative model are called spurious behaviors [36].

So if hybrid control only works with a qualitative model in first place, one has to be aware that the result of qualitative pre-selection could be based on such a spurious behavior and, thus, could be invalid. Checking the qualitatively determined results with the numerical model detects such a case and qualitative reasoning about another behavior can be continued. However, with respect to the constrained on-line computation time it is advantageous to take measures that help to pre-select a valid solution right in first place.

So an additional requirement will be included to qualitative modeling:

- It is required that the model contains indicators that tell how likely a particular qualitative behavior is *spurious*.

4.2 Qualitative Abstraction

The intention followed by qualitative modeling is to capture the complex hybrid behavior of a system by an abstracted model among discretely valued variables only. With such a model one can reason about behaviors of the hybrid systems by utilizing efficient *discrete* search techniques. Simultaneous treatment of the discretely valued variables and abstractions of the continuously valued ones doesn't require different tools to handle both types of variables.

4.2.1 Basic Automaton Concept

As basis for the qualitative model for each component, an abstraction of each of the system's components that is similar to a non-deterministic automaton [39, 40] is utilized. This type of qualitative model abstracts the continuously valued variables of a model to qualitative variables by separating the value-space of each such variable into distinct regions. Each of these regions represents a different value of the qualitative variable.

In this work, continuously valued variables are denoted by lowercase letters a for scalars or lowercase boldface letters $\mathbf{a}$ for vectors. The corresponding qualitative variables are denoted by uppercase letters A and their valuations (qualitative values) by lowercase Greek letters α.

	continuously valued variable	qualitative variable	qualitative values
states	$x, \mathbf{x}$	X	$\xi_1, \xi_2, \ldots$
inputs	$u, \mathbf{u}$	U	$\upsilon_1, \upsilon_2, \ldots$
outputs	$y, \mathbf{y}$	Y	$\mu_1, \mu_2, \ldots$
other	$w, \mathbf{w}$	W	$\omega_1, \omega_2, \ldots$

Regardless whether the corresponding continuously valued variable is a scalar or a vector, qualitative variables are always scalars. The dimensionality of the continuous variable only contributes to the dimensionality of the regions represented by the qualitative values. An example is given in Figure 4.1 where a variable $\mathbf{x} = [x_1, x_2]^T$ is abstracted.

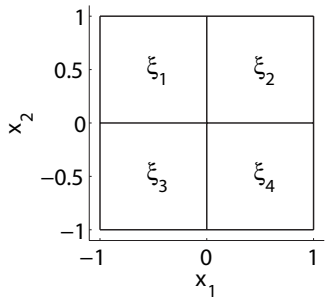


Figure 4.1: Qualitative abstraction of a vector

$$
\begin{aligned}
\mathbf{x} \in \{x_1, x_2 \mid\ & -1 \leq x_1 < 0, & 0 \leq x_2 < 1\} &\longleftrightarrow X = \xi_1 \\
\mathbf{x} \in \{x_1, x_2 \mid\ & 0 \leq x_1 < 1, & 0 \leq x_2 < 1\} &\longleftrightarrow X = \xi_2 \\
\mathbf{x} \in \{x_1, x_2 \mid\ & -1 \leq x_1 < 0, & -1 \leq x_2 < 0\} &\longleftrightarrow X = \xi_3 \\
\mathbf{x} \in \{x_1, x_2 \mid\ & 0 \leq x_1 < 1, & -1 \leq x_2 < 0\} &\longleftrightarrow X = \xi_4
\end{aligned}
$$

In this example, the 2 dimensional variable $\mathbf{x}$ is separated into 4 equally-sized rectangular qualitative regions. Yet, the proposed qualitative modeling scheme allows more general separations into polytopes

$$\mathbf{H}_i\mathbf{x} \leq \mathbf{k}_i \longleftrightarrow X = \xi_i.$$

With such a separation (for the state $\mathbf{x}$ as well as for input $\mathbf{u}$ and output $\mathbf{y}$), a continuous model

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k \end{aligned}$$

can be abstracted to an automaton with states ξ_i in the following way: The automaton allows a transition labeled υ_l and μ_m from one state $X_k = \xi_i$ to another state $X_{k+1} = \xi_j$ if and only if the continuous model allows trajectories from $\mathbf{x}_k$ inside the region represented by ξ_i to $\mathbf{x}_{k+1}$ inside the region represented by ξ_j while observing an output $\mathbf{y}_k$ inside region μ_m if there is applied an input $\mathbf{u}_k$ inside region υ_l.

Generally, trajectories starting from different points in a region represented by $X_k = \xi_i$ will not always lead to the same qualitative region X_{k+1}. So the automaton will show ambiguous transition targets. To partially resolve this ambiguity, each ambiguous transition to a specific target is additionally labeled with a probability value p that indicates the likelihood to encounter a trajectory to a corresponding state $\mathbf{x}_{k+1}$ while observing output an $\mathbf{y}_k$ in region μ_m if state $\mathbf{x}_k$ and input $\mathbf{u}_k$ are uniformly sampled from their qualitative regions ξ_i and υ_l.

An example for this is shown in Figure 4.2 for a state space separation as in Figure 4.1 and a continuous model

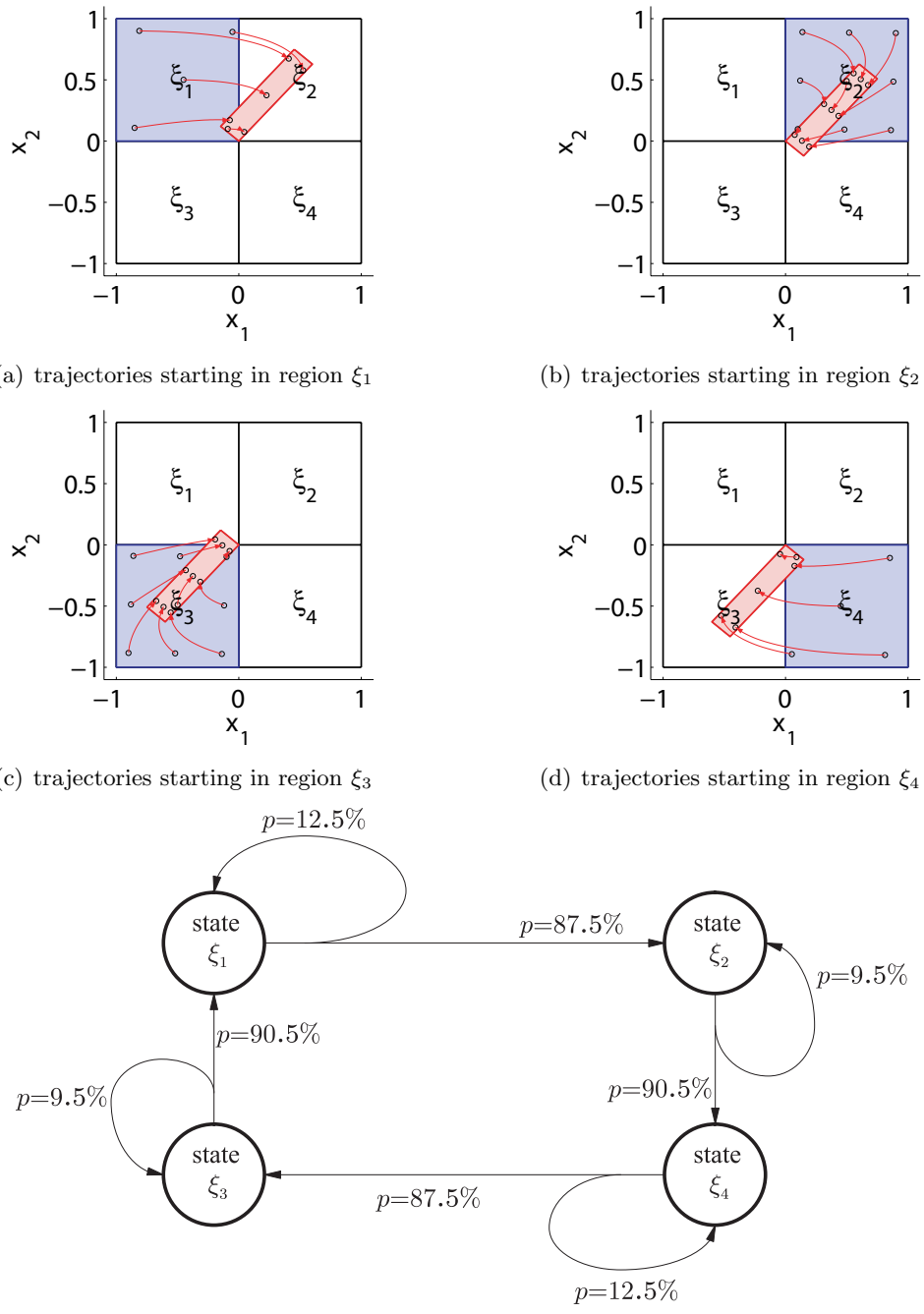
$$\mathbf{x}_{k+1} = \begin{bmatrix} 0.15 & 0.60 \\ -0.12 & 0.63 \end{bmatrix} \mathbf{x}_k. \tag{4.1}$$

The major drawback of a value-space separation as used in non-deterministic automata and the reason why such concepts were only moderately successful in the past is that the complexity increases exponentially with the dimensionality of the space. However, this drawback is not too striking with our hybrid models that are utilized here. This class of hybrid models aims at systems composed of several components, each of which can be described by a low-dimensional model. So the complexity of the individual abstractions of the individual low-dimensional component models will be manageable.

4.2.2 Choice of Qualitative Variables

The non-deterministic automaton model outlined above operates on a continuous model of a single component only. This is well in accordance to the intended qualitative abstraction of multi-component systems, as this abstraction is going to be performed component-wise anyway. However, it is necessary to additionally demand that the individual abstractions of the components are represented in a way that makes simultaneous reasoning among all of them easy.

As a first step towards this, we want to use the same qualitative representation for a variable that interconnects several components in all those component models. The fact that generally not all outputs of one component will serve as inputs in one other component provides constraints on the choice of the qualitative separation of these outputs. To be more illustrative, an example will aid in the discussion of this topic: Three components are



(a) trajectories starting in region ξ_1

(b) trajectories starting in region ξ_2

(c) trajectories starting in region ξ_3

(d) trajectories starting in region ξ_4

(e) non-deterministic automaton

Figure 4.2: Abstraction of a continuous system to an automaton

interconnected by the connection specification

	determined by...	provided to...
w_1	$y_1^{(1)}$	$u_1^{(2)}, u_2^{(3)}$
w_2	$y_2^{(1)}$	$u_2^{(2)}, u_1^{(3)}$
w_3	$y_3^{(1)}$	$u_3^{(2)}$

The first column represents the variable name in the overall model, the second specifies which variable of which component specifies its value and the third column specifies where the variable serves as input. Superscripts in brackets indicate the component while subscripts denote the 'position' in the vector, e.g. $\mathbf{u}^{(3)} = [u_1^{(3)}, u_2^{(3)}]^T$ denotes the input to component number 3. An outline of this interconnection is depicted in Figure 4.3.

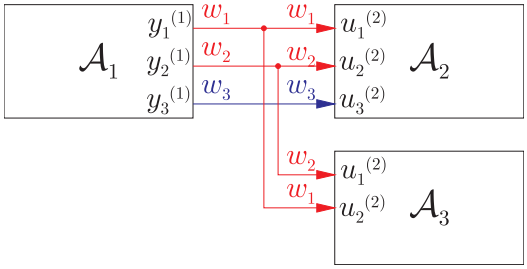


Figure 4.3: Example of 3 interconnected components

Using a single qualitative variable W to represent the whole vector $\mathbf{w} = [w_1, w_2, w_3]^T$ would allow a partitioning of the value-space as depicted in Figure 4.4a[1]. However, with such a partitioning it is not directly possible to compile a separate non-deterministic automaton for each component, as the partitioning cannot directly be used for the input of component 3 ($\mathbf{u}^{(3)} = [w_2, w_3]^T$). This would require an additional nontrivial mapping function that projects the depicted partitioning onto the input $\mathbf{u}^{(3)}$.

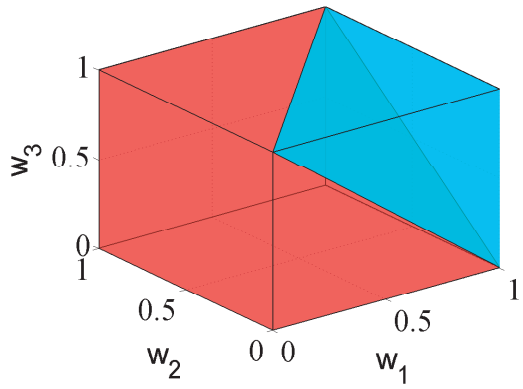
This problem certainly is avoided, if each scalar variable w_1, w_2 and w_3 is represented by a separate qualitative variable W_1, W_2 and W_3. So the partitioning of the value-space of the inputs ($\mathbf{u}^{(2)}$, $\mathbf{u}^{(3)}$) and outputs ($\mathbf{y}^{(1)}$) of the three components is necessarily rectangular. Figure 4.4b).

However, looking at Figure 4.3, it can be observed that w_1 and w_2 are, both, determined by component 1 and are, both, used in component 2 as well as component 3. Only w_3 is different in that (although as well being determined by component 1) it is only used in component 2. So a more general partitioning can be allowed, if a single qualitative variable (W_1) is selected to represent all variables ($[w_1, w_2]^T$) that share their 'origin' (component 1) and 'destinations', (components 2 and 3). But another qualitative variable(W_2) has to be selected to represent w_3. This guarantees that qualitative variables can be used for compiling the separate non-deterministic automata without non-trivial mappings from the qualitative variables of one component to those of another, but it allows more general than rectangular partitions (Figure 4.4c).

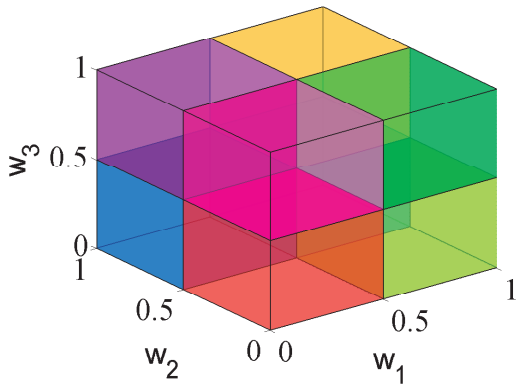
Following this argument, a rule for selecting qualitative variables can be given:

- Represent all continuously valued variables of a multi-component system that are determined by the same component (or are all determined by external inputs) and that influence the same set of components together by a single qualitative variable.

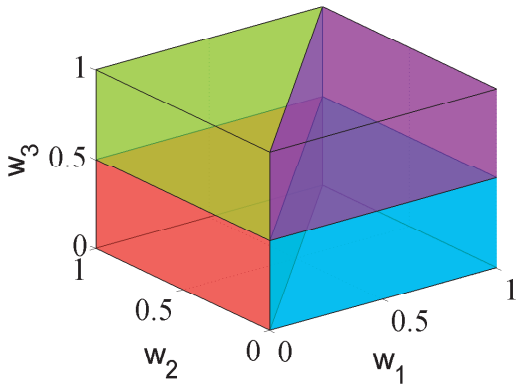
[1]The 3D plots of the qualitative separations are generated using the multi-parametric toolbox [37] for MATLAB



(a) common partitioning of w_1, w_2 and w_3



(b) independent partitioning of w_1, w_2 and w_3



(c) common partitioning only of w_1 and w_2 but independent partitioning of w_3

Figure 4.4: Some variants for separating the value-space of a multi-dimensional variable

This rule naturally implies that the continuous state vector $\mathbf{x}_c^{(i)}$ of each component is represented by a single qualitative variable X_i as well.

4.3 Compilation of Hybrid Automata to Qualitative Models

Non-deterministic automata usually are used as qualitative approximations of continuous models. Nevertheless, the discrete mode transitions of hybrid automata can be incorporated into such an automaton as well. This provides the advantage, that the continuous and the discrete part of a hybrid model can uniformly be represented by the same qualitative model.

The previous section outlined the basics of qualitative modeling by non-deterministic automata and how such non-deterministic automata models can approximate the continuous dynamics of the individual components of a multi-component continuous model. Further, it discussed how the interconnection of the various components influences the choice of qualitative variables.

In this section we more formally show how to approximate each hybrid component model as such an automaton. As basis for this, the hybrid model for each of the components is needed together with the determined set of qualitative variables and according finite partitionings of the continuous variable spaces. Since the main intention of this work is control of physical

systems, it can be assumed that the value space for each continuously valued variable is bounded. These bounds are integrated into the qualitative model by associating qualitative values only to polytopic regions within these bounds.

To compile the automaton, in principle, one first enumerates all possible 'initial conditions'. That is, all combinations of

- discrete states $x_{d,k} = m_i \in \mathcal{X}_d$
- qualitatively distinguished continuous states $X_k = \xi_i \in \mathcal{X}$
- discrete input commands $u_{d,k} \in \mathcal{U}_d$
- qualitatively abstracted continuous inputs, which themselves are an enumeration of all possible combinations of valuations of the corresponding qualitative variables $U_{j,k} = \upsilon_i \in \mathcal{U}_j$ and $W_{j,k} = \omega_i \in \mathcal{W}_j$.

Next, for each of these 'initial conditions', one evaluates all possible discrete mode-transitions $x_{d_k} \rightarrow x_{d,k+1}$ – including the possibility of the so-called 'no-transition', i.e. the mode staying the same – and the associated discrete outputs $y_{d,k+1}$. Here it is necessary to notice, that these transitions may not only be guarded on boolean functions of the discrete command inputs but as well on the continuous state $\mathbf{x}_k$ and input $\mathbf{u}_k$ being in certain (polytopical) regions of the value-space. This has implications on the partitioning into qualitative values, as later discussed in Section 4.3.1.

For the resulting enumeration of 'initial conditions' and follow-up-modes, one can now determine all possible continuous outputs and follow-up states by utilizing the continuous dynamic models associated to the respective operational modes.

$$\begin{aligned} x_{d,k} &= m_a \\ x_{d,k+1} &= m_b \\ \mathbf{x}_{k+1} &= \mathbf{A}_{mb}\mathbf{x}_k + \mathbf{B}_{mb}\mathbf{u}_k + \mathbf{e}_{mb} \\ \mathbf{y}_k &= \mathbf{C}_{ma}\mathbf{x}_k + \mathbf{D}_{ma}\mathbf{u}_k + \mathbf{f}_{ma} \end{aligned}$$

How to evaluate qualitative abstractions and likelihoods for these continuous outputs and trajectories will be shown in Section 4.3.2.

So the basic qualitative model finally lists all possible combinations of

- 'initial conditions' as given above
- follow-up discrete states $x_{d,k+1} = m_i \in \mathcal{X}_d$
- discrete outputs $y_{d,k+1}$
- abstracted follow up continuous states $X_{k+1} = \xi_i \in \mathcal{X}$
- and the associated valuations for the qualitative output variables $Y_{j,k} = \mu_i \in \mathcal{Y}_j$ and $W_{j,k} = \omega_i \in \mathcal{W}_j$.

Each of these combinations represents a transition in a non-deterministic automaton from a particular state $< x_{d,k}, X_k >$ to a state $< x_{d,k+1}, X_{k+1} >$. The transition is guarded by $< y_{d,k+1}, U_k, W_k, Y_k >$. The transition additionally has an associated likelihood value that is calculated as shown in section 4.3.2.

4.3.1 Choice of Qualitative Values

Before calculation of likelihood values is discussed, we first take a closer look at the partitioning of the value-space of continuously valued variables to specify qualitative values.

Since the modeled hybrid systems are assumed to be physical systems, it can be assumed that the value space for each continuously valued variable is bounded. These bounds are integrated into the qualitative model by associating qualitative values only to polytopic regions within these bounds. If qualitative variables are chosen properly (section 4.2.2), the partitioning of the value-space of the associated continuously valued variables into polytopic regions can in principle be chosen quite arbitrarily. However, this partitioning may have some influence on the performance of the qualitative pre-selection.

It further has to be noticed that discrete mode transitions of hybrid automata may be guarded by conditions that evaluate, whether the continuous state $\mathbf{x}_k$ or input $\mathbf{u}_k$ is inside or outside of certain polytopes in the value-space of these variables.

As it should be avoided do introduce unnecessary ambiguity into the part of the qualitative model that represents the discrete part of the hybrid model, it is required that the qualitative model allows to uniquely check these conditions.

So qualitative values for each variable have to be chosen such that the corresponding regions either are completely inside or outside of these polytopes. If the qualitative variable is used in more than one component, of course all transition guards of all these component's hybrid models have to be taken into account to determine the necessary qualitative distinctions.

Throughout this work it is assumed that such a separation is possible for a given interconnection of components. A counterexample, where this would not be possible, is a guard condition

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \leq 0$$

if the interconnection of components forces to abstract u_1 and u_2 to separate qualitative variables U_1 and U_2 what results in a necessarily rectangular partitioning of the value space of $[u_1, u_2]^T$.

Despite these necessary qualitative distinctions, the partitioning of value spaces into qualitatively distinct regions can be chosen quite arbitrarily. Making efficient use of this freedom, however, remains subject to further research beyond the scope of this work. In diagnosis, there exist approaches to introduce qualitative distinctions on basis of the intended observation-task ([47]). These, however, cannot be directly applied to the case here, as no assumptions on the subsequent control tasks are made at compile-time of the qualitative model. Our examples on multi-component systems (with low-dimensional components) indicate, however, that usually a partitioning into a reasonably small number of rectangular regions is sufficient.

4.3.2 Spurious Behaviors and Likelihood Values

The introduction to this section provided a discussion on how a hybrid automaton can be transformed into a non-deterministic automaton by enumerating all possible transitions

$$< x_{d,k}, X_k > \rightarrow < x_{d,k+1}, X_{k+1} >$$

of its state. In addition, the associated guard-tuples

$$< u_{d,k}, y_{d,k+1}, U_k, W_k, Y_k >$$

were evaluated.

This section will now deal with the problem to associate likelihood values to the transitions which are used to partially resolve ambiguity among the transitions. Here, two different likelihoods are distinguished. The first one is imposed by the probabilistic nature of the hybrid model itself. Recall that our hybrid modeling framework (Section 3.3) allows to specify a probability-distribution among several transition target modes. The second likelihood is occurring only due to the qualitative abstraction itself.

As already indicated by Figure 4.2 on page 47, generally not even with a specific mode all trajectories that satisfy $X_k = \xi_i$ ($U_k = \upsilon_i$) will lead to follow up states that have the same qualitative abstraction $X_{k+1} = \xi_j$. However, as the figure indicated, some valuations for X_k and X_{k+1} are satisfied for trajectories emerging from a large region while others are only satisfied by trajectories starting in 'distant corners' of the considered qualitative region.

As a heuristic, qualitative search for good mode sequences and command inputs that satisfy a given control goal, should not rely too much on these very unlikely qualitative state transitions, because they might turn out to be *spurious behaviors*.

Spurious behaviors are a well known and unfortunately unavoidable problem of general abstracted models. According to Kuipers' original definition [36], spurious behaviors are sequences of qualitative values that – although valid for the qualitative model – are *not* abstractions of any possible behavior of the original numerical model.

In this work, the term *spurious behavior* is used in a somewhat wider sense that, however, still captures the idea. In addition to Kuipers' definition, here the term also is used for sequences of qualitative values that do not represent an abstraction of any trajectory of the numerical model that starts *from a specific initial state* inside the qualitative region represented by the *qualitative initial state*.

The following figures will provide illustration for that. Once more the single-mode model (4.1)

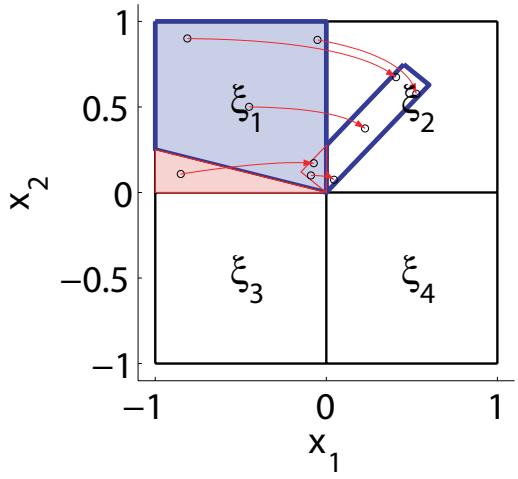
$$\mathbf{x}_{k+1} = \begin{bmatrix} 0.15 & 0.60 \\ -0.12 & 0.63 \end{bmatrix} \mathbf{x}_k.$$

that has already been illustratively abstracted to a non-deterministic automaton in Figure 4.2 is utilized.

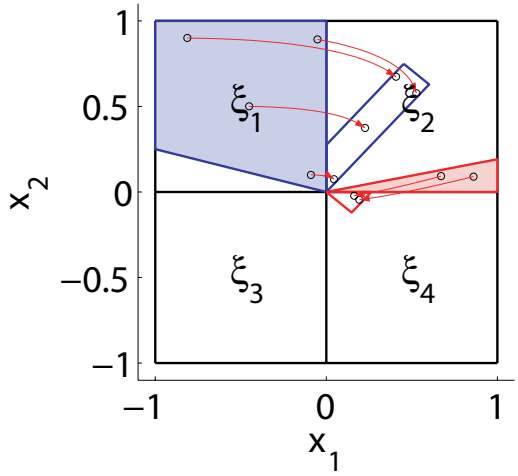
As illustration for the reasons of spurious behaviors resulting from the widened definition including specific initial states, trajectories starting from qualitative state $X_k = \xi_1$ are considered. In Figure 4.5a it can be observed, that trajectories starting from states $\mathbf{x}_k$ in most of the region related to ξ_1 (87.5% of the area covered by ξ_1) reach states $\mathbf{x}_{k+1}$ that lie in the region related to qualitative value ξ_2. Only a small area in the lower left of the region related to qualitative state ξ_1 (12.5%) provides starting points for trajectories that reach states $\mathbf{x}_{k+1}$ that again lie inside the region covered by ξ_1. The qualitative model allows trajectories $\xi_1 \rightarrow \xi_2$ as well as trajectories $\xi_1 \rightarrow \xi_1$. However, for the specific initial state $\mathbf{x}_0 = [-0.5\ 0.5]^T$ the next state $\mathbf{x}_1 = [-0.225\ 0.375]^T$ lies inside region ξ_2.

So, for this specific initial state (and in fact for initial states that lie in 87.5% of the region covered by ξ_1), the qualitative trajectory $\xi_1 \rightarrow \xi_1$ is spurious. This discrimination of areas that provide starting points for trajectories that show specific qualitative abstractions usually is the basis for calculating transition-likelihoods in non-deterministic automata.

The other type of spurious behaviors (which is in accordance to Kuipers' definition) is illustrated in Figure 4.5b. According to Figure 4.2e, the qualitative model allows qualitative trajectories $\xi_1 \rightarrow \xi_2$ as well as $\xi_2 \rightarrow \xi_4$. So the model allows a qualitatively abstracted



(a) Different qualitative follow-up states from different initial states



(b) Spurious behaviors in Kuipers' sense

Figure 4.5: Different types of spurious behaviors

trajectory $X_k = \xi_1, X_{k+1} = \xi_2, X_{k+2} = \xi_4$. However, it is not possible to find any trajectory

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k \\ \mathbf{x}_{k+2} &= \mathbf{A}\mathbf{x}_{k+1} \end{aligned}$$

that correspond to that abstracted trajectory.

If the area in the state space that can be reached by trajectories that abstract to qualitative trajectories $\xi_1 \rightarrow \xi_2$ covers only a very small portion of the area that corresponds to qualitative state ξ_2 it is rather likely that this area does not provide starting points for trajectories that abstract to $\xi_2 \rightarrow \xi_4$. So the small area indicates that it is rather likely that extending the qualitative trajectory $\xi_1 \rightarrow \xi_2$ for one further time-step leads to a spurious behavior. To contribute to this, an additional heuristic likelihood indicator of a qualitative trajectory is included which represents the area of the region in $X_{k+1} = \xi_j$ that can be reached from states $X_k = \xi_i$ compared to the whole area of the region represented by qualitative value ξ_j.

Summarizing, the likelihood value associated to each transition specification in the non-

deterministic automaton, i.e.

$$< x_{d,k}, X_k, u_{d,k}, y_{d,k+1}, U_k, Y_k, x_{d,k+1}, X_{k+1} >$$

is calculated as:

- According to the specification of the hybrid model, with specified command input $u_{d,k}$ and qualitative values for continuous state and input (X_k and U_k), the guard condition for a certain transition from mode $x_{d,k}$ is satisfied. According to the model, this transition leads to the specified mode $x_{d,k+1}$ with probability p_D.

- With modes $x_{d,k}$ and $x_{d,k+1}$ specified, only trajectories starting inside a region (hyper-volume V_1) of the value-space of

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix}$$

  as specified by the valued of X_k and U_k reach states $\mathbf{x}_{k+1}$ inside the region specified by X_{k+1}. This hyper-volume V_1 divided by the hyper-volume V_2 of the whole region specified by the values of X_k and U_k is the likelihood value L_{c1}. A low value L_{c1} indicates that a specific qualitative transition is spurious for many numerical initial conditions that correspond to the qualitative initial conditions (Figure 4.5a).

- The hyper-volume V_3 of the region that can be reached by the considered trajectories inside the state and output-space represented by the values of X_{k+1} and Y_k divided by the hyper-volume V_4 of the whole region represented by the values of X_{k+1} and Y_k is the likelihood value L_{c2}. A low value L_{c2} indicates that many states $\mathbf{x}_{k+1}$ and associated outputs $\mathbf{y}_k$ represented by qualitative state X_{k+1} and output Y_k cannot be reached form states $\mathbf{x}_k$ that correspond to X_k. So if the model is utilized for estimating qualitative trajectories beyond $k+1$, it is rather likely that they will be spurious (Figure 4.5b).

- The likelihood-value associated to the transition in the non-deterministic automaton is

$$L = p_d \cdot L_{c1} \cdot L_{c2}.$$

More details on the calculation of likelihood values L_{c1} and L_{c2} including some approximations that can be evaluated more efficiently and further illustrations are given in Appendix A.2. It has to be noticed here, that qualitative trajectories that are an abstraction of particular numerical trajectories *only* if they start or end at a border of the respective qualitative regions, will receive a likelihood value of 0. Because of this 0 likelihood, such qualitative trajectories are not included in the qualitative model.

4.3.3 Summary of Non-Deterministic Automaton Compilation

With calculation of the likelihood values, compilation of the non-deterministic automaton abstraction of a component of the hybrid model is completed. The interconnection among these various component models is represented by using the same qualitative variable for abstracting an output of one component and all the inputs in other components this output is connected to. So if the model of one component specifies a specific qualitative value for this variable, this value is used in all the other components as well.

Enumeration of all possible 'initial conditions', all possible follow-up modes, discrete outputs, qualitatively abstracted continuous follow-up states and abstracted continuous outputs

guarantees that the qualitative model covers abstractions for *all possible hybrid trajectories* of the hybrid automaton.

However, unavoidably, the qualitative model generally also will include some spurious behaviors. To provide a rough indication whether a qualitative trajectory

$$< x_{d,k}, X_k > \rightarrow < x_{d,k+N}, X_{k+N} >$$

is likely to be spurious or not, likelihood values L_{c1} and L_{c2} are associated to each transition in the non-deterministic automaton which contribute to whether the specified transition is valid for states $\mathbf{x}$ and inputs $\mathbf{u}$ that cover a large portion of the value space represented by qualitative values of X_k and U_k and whether possible states $\mathbf{x}_{k+1}$ that can be reached from there by this transition cover a large portion of the value-space specified by X_{k+1}.

This second likelihood value is necessary, since the automaton only encodes transitions

$$< x_{d,k}, X_k > \rightarrow < x_{d,k+1}, X_{k+1} >$$

but it has to be utilized it for reasoning about longer qualitative trajectories

$$< x_{d,k}, X_k > \rightarrow < x_{d,k+N}, X_{k+N} > .$$

It has to be confessed that computation of a non-deterministic automaton abstraction from a hybrid automaton can be a computationally intensive task and that the resulting model can be of considerable size. However, as with component-based hybrid modeling generally only abstraction of low-dimensional hybrid models is required, this keeps the computational burden and model-size manageable. Further, compilation of the qualitative model is performed off-line in advance to the on-line operation of the qualitative hybrid controller. So the required computation time is not of major importance.

The non-deterministic automaton model already satisfies some of the requirements postulated for the qualitative model in order to be useful for qualitative hybrid control. However, subsequently a different graphical representation of the non-deterministic automaton model will be specified. This new representation additionally will be very compact and will be able to very well represent the underlying 'structure' of the composed hybrid system. This representation of structure helps in the simultaneous treatment of the various component models and proves most useful in focusing search towards 'good' qualitative trajectories.

Illustrative Example

To illustrate compilation of the non-deterministic automaton and our likelihood values we consider a simple 2-component hybrid system:

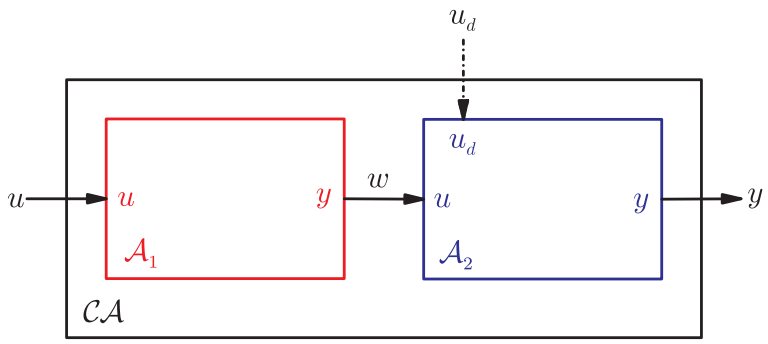
$$\begin{array}{lllrcl}
\text{component 1:} & \text{mode 1:} & x_{k+1} & = & -0.8 \cdot x_k - 0.2 \cdot u_k \\
 & & y_k & = & x_k \\
 & \text{mode 2:} & x_{k+1} & = & -0.9 \cdot x_k + 0.2 \cdot u_k \\
 & & y_k & = & x_k \\
\text{component 2:} & \text{mode 1:} & x_{k+1} & = & -0.1 \cdot x_k + u_k \\
 & & y_k & = & x_k \\
 & \text{mode 2 :} & x_{k+1} & = & -0.1 \cdot x_k - u_k \\
 & & y_k & = & x_k \, .
\end{array} \tag{4.2}$$

The following constraints apply:

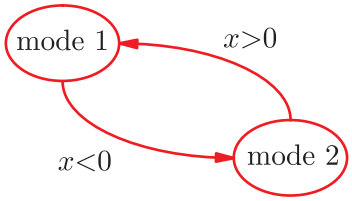
$$\begin{array}{lrcccl} \text{component 1:} & 0 & \leq & u & \leq & 1 \\ & -1 & \leq & x & \leq & 1 \\ & -1 & \leq & y & \leq & 1 \\ \text{component 2:} & -1 & \leq & u & \leq & 1 \\ & 0 & \leq & x & \leq & 1 \\ & 0 & \leq & y & \leq & 1\,. \end{array}$$

The discrete transition specifications and interconnection of the components are presented in Figure 4.6. If additionally the no-transitions are explicitly mentioned, transition guard conditions and target modes are:

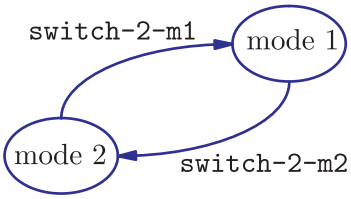
$$\begin{array}{llllll} \text{component 1:} & \text{mode 1:} & \mathrm{t}_1: & 0 \leq x \leq 1 & \rightarrow & m_1 \\ & & \mathrm{t}_2: & -1 \leq x < 0 & \rightarrow & m_2 \\ & \text{mode 2:} & \mathrm{t}_1: & 0 \leq x \leq 1 & \rightarrow & m_1 \\ & & \mathrm{t}_2: & -1 \leq x < 0 & \rightarrow & m_2 \\ \text{component 2:} & \text{mode 1:} & \mathrm{t}_1: & \texttt{switch-2-m1} & \rightarrow & m_1 \\ & & \mathrm{t}_2: & \texttt{switch-2-m2} & \rightarrow & m_2 \\ & \text{mode 2:} & \mathrm{t}_1: & \texttt{switch-2-m1} & \rightarrow & m_1 \\ & & \mathrm{t}_2: & \texttt{switch-2-m2} & \rightarrow & m_2 \end{array}$$



(a) Interconnection



(b) Transition specification for component 1; Transitions do not reset the continuous state



(c) Transition specification for component 2; Transitions do not reset the continuous state

Figure 4.6: Specification of the example system

According to our guidelines for the selection of qualitative variables, we abstract the continuous state $x^{(1)}$ of component 1 and the continuous state $x^{(2)}$ of component 2 to qualitative variables X_1 and X_2, respectively. Additionally, in the interconnection of the 2 components we have 1 variable (u) that is an external input provided to component 1, we have 1 variable (w) that is an output provided by component 1 to component 2 and we have 1 variable (y)

that is provided by component 2 and serves as external output. As each of these variables is *provided by* and *provided to* different components, according to our guidelines, we abstract each of them to a different qualitative variable U, W and Y, respectively.

To separate the constrained value spaces into qualitative regions, we choose to distinguish only the sign:

$$\begin{array}{llllll} X_1 = \xi_1 & \longleftrightarrow & -1 \leq x^{(1)} < 0 & X_1 = \xi_2 & \longleftrightarrow & 0 \leq x^{(1)} \leq 1 \\ X_2 = \xi_3 & \longleftrightarrow & 0 \leq x^{(2)} \leq 1 & & & \\ U = \upsilon_1 & \longleftrightarrow & 0 \leq u \leq 1 & & & \\ W = \omega_1 & \longleftrightarrow & -1 \leq w < 0 & W = \omega_2 & \longleftrightarrow & 0 \leq w \leq 1 \\ Y = \mu_1 & \longleftrightarrow & 0 \leq y \leq 1 & & & \end{array}$$

This captures the necessary distinction of regions $x < 0$ and $x > 0$ imposed by the mode-transition specification of component 1.

To compile the non-deterministic automaton for each component, we first have to identify the automaton's states, i.e. the qualitative abstractions of the component's hybrid state $< x^{(i)}, x_d^{(i)} >$:

$$\begin{array}{lllll} \text{component 1:} & m_1, \xi_1 & m_1, \xi_2 & m_2, \xi_1 & m_2, \xi_2 \\ \text{component 2:} & m_1, \xi_3 & & m_2, \xi_3 & \end{array}$$

For each of these qualitative states $< X_k, x_{d,k} >$ and all combinations of qualitatively abstracted inputs (possible values: υ_1 for component 1 and ω_1 and ω_2 and `switch-2-m1` and `switch-2-m2` for component 2 in our example), possible values of qualitative abstracted states $< X_{k+1}, x_{d,k+1} >$ and outputs (W_k for component 1 and Y_k for component 2) have to be determined by the hybrid model.

As example for this we demonstrate this determination of qualitative trajectories (together with the calculation of likelihood values) for component 1 and

$$X_k = \xi_1, \quad x_{d,k} = m_1, \quad U_k = \upsilon_1 \ .$$

The hybrid model for component 1, mode m_1 specifies

$$y_k = x_k,$$

the mode-transition-specification specifies that with $X_k = \xi_1$ a mode change

$$x_{d,k} = m_1 \rightarrow x_{d,k+1} = m_2 \qquad \text{with } p_d = 1$$

occurs and the hybrid model for this new mode m_2 specifies

$$x_{k+1} = -0.9 \cdot x_k + 0.2 \cdot u_k.$$

With this, the output w_k of component 1 can take only values that abstract to $W_k = \omega_1$ and only continuous states x_{k+1} can be reached that satisfy $X_{k+1} = \xi_2$[2].

However, not all allowed values for x_k and u_k that correspond to ξ_1 and υ_1 lead to values x_{k+1} within the constraints. Comparing 'possible' values to all that correspond to the qualitative regions leads to likelihood value

$$L_{c1} = 0.97,$$

[2]We do not take trajectories into account that are only possible if values at the borders of the respective qualitative regions are considered, because for these trajectories the calculated likelihoods would be 0.

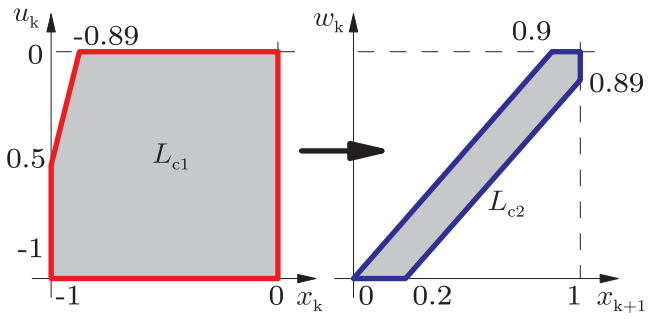


Figure 4.7: Calculation of Likelihood Values

as illustrated in Figure 4.7. Additionally, not all values for x_{k+1} and w_k that correspond to qualitative regions ξ_2 and ω_1 can be reached. This leads to a likelihood value

$$L_{c2} = 0.19.$$

So the qualitative trajectory's likelihood is

$$L = p_d \cdot L_{c1} \cdot L_{c2} = 0.19\ .$$

This is no likelihood in a probabilistic sense such that the values related to all possible trajectories starting from a specific qualitative state sum up to 1, but it is just an indication whether a particular qualitative trajectory bears a higher or a lower risk to be part of a spurious behavior. [3] But in trade-off for a heuristic that, as we believe, helps to better estimate whether a qualitative behavior is spurious, we accept to drop this probabilistic interpretation.

Proceeding like above for all qualitative trajectories leads to the model for component 1:

$x_{1d,k}$	$X_{1,k}$	U_k	W_k	$x_{1d,k+1}$	$X_{1,k+1}$	L
mode 1	ξ_1	υ_1	ω_1	mode 2	ξ_2	0.19
mode 1	ξ_2	υ_1	ω_2	mode 1	ξ_1	0.20
mode 2	ξ_1	υ_1	ω_1	mode 2	ξ_2	0.19
mode 2	ξ_2	υ_1	ω_2	mode 1	ξ_1	0.20

And the model for component 2 is

$x_{2d,k}$	$X_{2,k}$	W_k	Y_k	$u_{d,k}$	$x_{2d,k+1}$	$X_{2,k+1}$	L
mode 1	ξ_3	ω_1	μ_1	`switch-2-m2`	mode 2	ξ_1	0.90
mode 1	ξ_3	ω_2	μ_1	`switch-2-m1`	mode 1	ξ_1	0.90
mode 2	ξ_3	ω_1	μ_1	`switch-2-m2`	mode 2	ξ_1	0.90
mode 2	ξ_3	ω_2	μ_1	`switch-2-m1`	mode 1	ξ_1	0.90

The two component models are connected by their shared variable W_k, which has to have the same value in both models.

[3] To keep this probabilistic interpretation we would have set L_{c2} to 1 and include an additional qualitative state to the automaton that covers all regions where constraints are violated.

4.4 Compact Compilation as Trajectory Graphs

The non-deterministic automaton model already satisfies most of the requirements we introduced in Section 4.1, as it provides a component-based off-line qualitative abstraction of multi-component hybrid systems. And, additionally, the introduced likelihood values associated to the state-transitions in the non-deterministic automaton provide a heuristic to estimate whether a particular trajectory through the states of the automaton is likely to be spurious with respect to the hybrid model.

It is, however, furthermore required to find a representation of those qualitative component-models, that can more intuitively be treated simultaneously. This new, structured, representation will additionally support more efficient qualitative reasoning as it allows to exploit structural properties of the hybrid model for focusing onto promising qualitative trajectories.

Before we discuss these issues in more detail, we take a glimpse at the intended control task that has to be fulfilled with the qualitative model.

4.4.1 An Excurse to Search

The task of qualitative pre-selection is to search the qualitative component models for qualitative behaviors (i.e. assignments to all the qualitative variables of all component models) regarding a finite time-horizon

$$t_k \rightarrow t_{k+N}.$$

One problem at this is that the qualitative model only is *compiled* for hybrid trajectories involving a single component and a time-horizon of

$$t_k \rightarrow t_{k+1}.$$

However, this qualitative automaton model is utilized to reason among several components and for longer time horizons. So, as illustrated above, the qualitative trajectories predicted by the automaton model bear the risk to be spurious. To contribute to this, the modified likelihood values were introduced.

Another problem is implied by the necessary component-wise abstraction of the hybrid model: Qualitative pre-selection cannot simply operate on single qualitative model, but needs to reason among several 'overlapping' component models.

In principle, this could be done on behalf of the non-deterministic automata as follows:

- Look at all transitions leaving the automata's states $< x_{d,k}^{(i)}, X_k^{(i)} >$
- determine the sets of qualitative values that satisfy the guard conditions for at least one transition in all component automata simultaneously
- Follow these transitions to determine the automata's next states $< x_{d,k+1}^{(i)}, X_{k+1}^{(i)} >$

No further detail will be presented on this procedure here, but it has to be noticed that it is not straightforward how the required sets of qualitative values can be evaluated efficiently. Instead, it is temporarily postulated that one had been able to encode the qualitative models as 'well structured' (we go into detail about this later) directed acyclic graphs (DAGs).

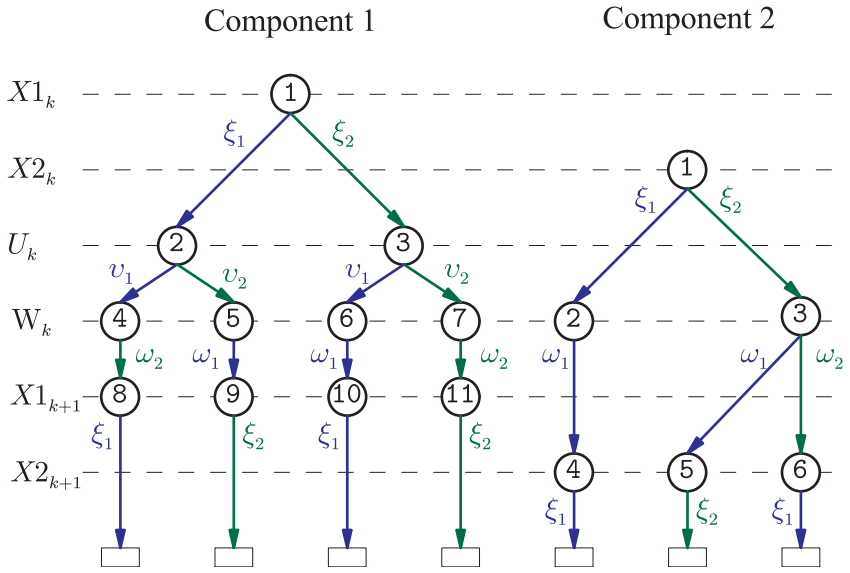


Figure 4.8: Qualitative models for 2 components as DAGs

A Motivation for Directed Acyclic Graphs

Figure 4.8 shows two directed acyclic graphs, each representing a component of a multi component model. The nodes of the graphs represent qualitative variables and the edges represent qualitative values.

The first graph encodes the qualitative state transitions

current state	guard-	-condition	next state
$X1_k$	U_k	W_k	$X1_{k+1}$
ξ_1	υ_1	ω_2	ξ_1
ξ_1	υ_2	ω_1	ξ_2
ξ_2	υ_1	ω_1	ξ_1
ξ_2	υ_2	ω_2	ξ_2

(4.3)

for a component with input U, state $X1$ and output W. The second component uses W as input to update state $X2$. Graph 2 encodes the associated qualitative state transitions

current state	guard condition	next state
$X2_k$	W_k	$X2_{k+1}$
ξ_1	ω_1	ξ_1
ξ_2	ω_1	ξ_2
ξ_2	ω_2	ξ_1

(4.4)

It can be observed, that both graphs were built upon the same pre-specified ordering of variables

$$X1_k \prec X2_k \prec U_k \prec W_k \prec X1_{k+1} \prec X2_{k+1} \tag{4.5}$$

This common ordering provides the advantage, that both graphs can straightforwardly be processed simultaneously. This is demonstrated by searching for an assignment to the variables U_k, W_k, $X1_{k+1}$ and $X2_{k+1}$ starting from the initial state

$$X1_k = \xi_1 \qquad X2_k = \xi_2.$$

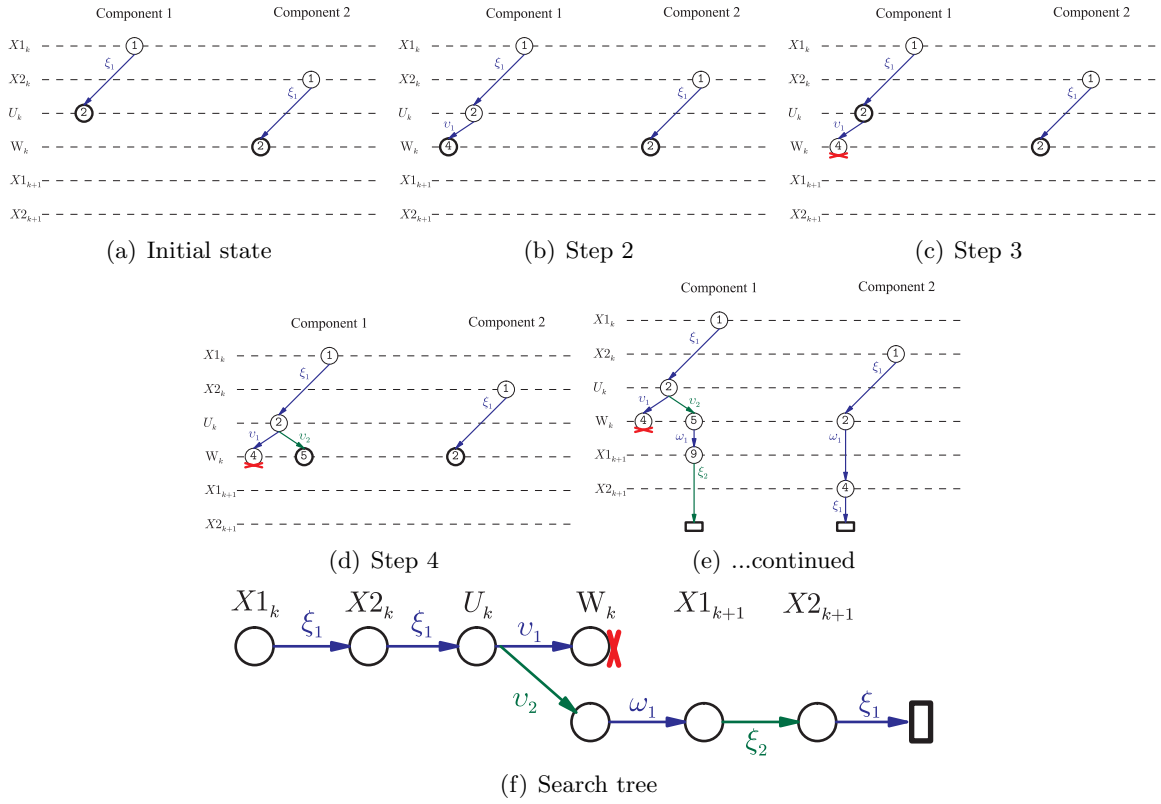


Figure 4.9: Searching several models (DAGs) simultaneously

Illustration for this is provided in Figure 4.9

Search is conducted following the same ordering of variables (4.5) as is used for building the graphs. Parallel to building up the search tree according to this ordering, one can think of traversing along the model-graphs to – informally speaking – evaluate the 'effects' of the selected variable assignments. Search starts at the root node (1) of both graphs.

According to the ordering of variables (4.5) exploration of the graphs starts with the initial state $X1_k = \xi_1$ which forces to traverse the arc from node (1) to node (2) in graph 1. $X2_k = \xi_1$ forces to traverse from node (1) to node (2) in graph 2.

Now search starts exploring possible assignments to U_k, e.g. $U_k = \upsilon_1$. This forces traversing from node (2) to node (4) in graph 1. Node (2) of graph 2, where one is assumed to 'be at' at the moment already represents a variable (W_k) that is ordered beyond the actual variable U_k. This means, that model 2 does not provide any constraint on U_k here and, thus one only has to look at graph 1 for the moment.

Next, W_k is processed, while at node (4) in graph 1 and at node(2) in graph 2. Both these nodes represent the same variable W_k, so both models provide constraints on W_k. Unfortunately, model 1 only allows $W_k = \omega_2$ (there is only this edge leaving node (4))while model 2 only permits $W_k = \omega_1$. So no assignment is found that satisfies both models and search has to back-track one step (of course, going back to node (2) in graph 1) and to investigate another assignment to variable U_k.

This way, search is continued until an assignment to all variables is found. The exploration of the model-graphs along this process is displayed in Figure 4.9a-e and the corresponding search tree is depicted in Figure 4.9f.

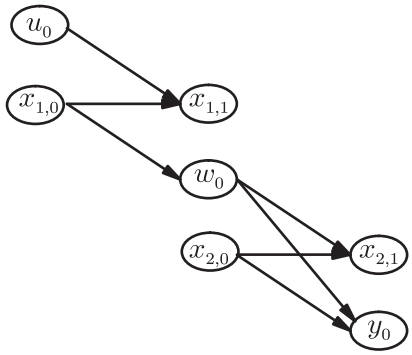


Figure 4.10: Interdependencies among the model's variables

This demonstrates, how searching for an assignment to qualitative variables by processing them sequentially and utilizing models represented as DAGs helps to search multiple overlapping component models simultaneously. The use of such DAGs as qualitative models will further be motivated by the observation that a clever ordering of variables can be utilized for exploiting structural properties of the underlying hybrid model's structure to make search more efficient, as is discussed in the following.

Further Motivation by Including Structure

So far, compilation of qualitative models as non-deterministic automata was presented and some motivation for representing them as DAGs has been obtained by a short excurse to qualitative pre-selection. Before it will be discussed how to actually build such DAGs, motivation for the graphical representation shall further be increased by taking yet another short excurse: We will investigate how a clever ordering of variables that is common to the construction of all the DAGs can help to make search more efficient.

For this, a simple example system consisting of two components is utilized:

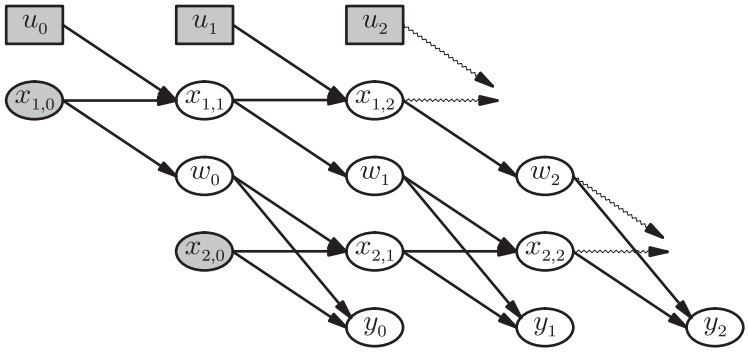
$$\begin{aligned}
\text{component 1:} \quad & x_{1,k+1} = x_{1,k} + u_k && \text{(4.6a)}\\
& w_k = x_{1,k} && \text{(4.6b)}\\
\text{component 2:} \quad & x_{2,k+1} = x_{2,k} + w_k && \text{(4.6c)}\\
& y_k = x_{2,k} + w_k && \text{(4.6d)}
\end{aligned}$$

To keep the presentation simple, purely continuous models with scalar inputs, outputs and states for both components are assumed as this provides more compact illustrations and as the same arguments can equivalently be applied to hybrid systems as well.

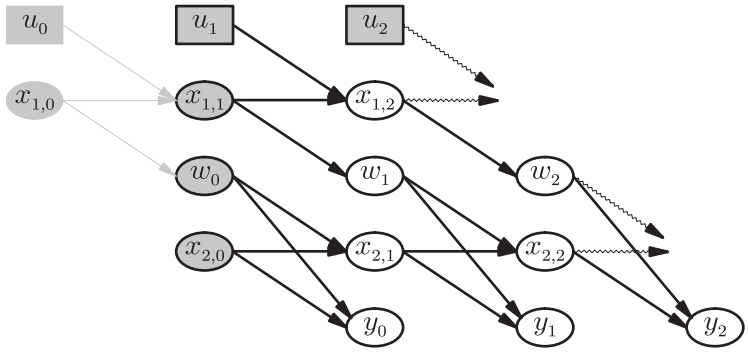
The specific numerical values in these equations aren't interesting here, but only the 'structure' of this model, that is, the interdependencies among the qualitative variables is relevant. These interdependencies are illustrated by Figure 4.10. For more complex hybrid systems, such a graph can be built upon a causal analysis [50] of the model (Appendix A.3).

Figure 4.11 shows this causal graph for time t_0 beyond t_2. These illustrations are used as an aid in the following discussion how the system's structure can be exploited to make qualitative search more efficient.

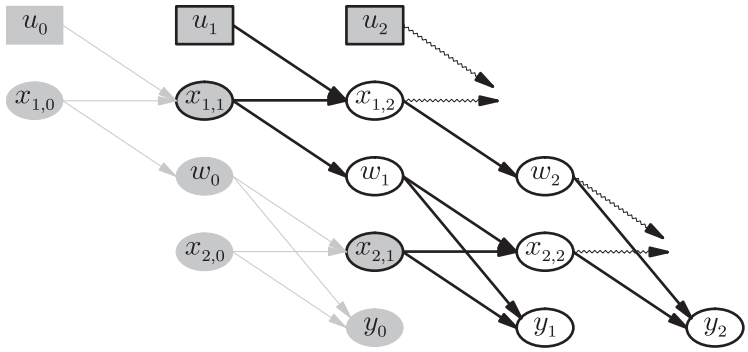
In Figure 4.11a it can be observed that the system's initial state $x_{1,0}, x_{2,0}$ (together with the inputs u_0, $u_1 \ldots$) is sufficient to determine all other variables. That means, all edges



(a) $x_{1,0}, x2, 0$ plus current and future inputs determine all other variables



(b) Out of $\{x_{1,0}, x_{2,0}, w_0, u_0, x_{1,1}\}$ the subset of variables $\{x_{2,0}, w_0, x_{1,1}\}$ (plus the current and future inputs) is enough to determine all other variables



(c) Out of $\{x_{1,0}, x_{2,0}, u_0, w_0, x_{1,1}, y_0, x_{2,1}\}$ the subset of variables $\{x_{1,1}, x_{2,1}\}$ (plus the current and future inputs) is enough to determine all other variables

Figure 4.11: Utilizing the structure of a model

of the graph that lead to the undetermined variables can be reached from $x_{1,0}, x_{2,0}$ or the inputs.

If a search algorithm now sequentially processes variables

$$w_0 \prec u_0 \prec x_{1,1} \prec \dots$$

the situation is like depicted in Figure 4.11b: Out of all the specified variables, i.e.

$$\{x_{1,0}, x_{2,0}, u_0, w_0, x_{1,1}\} \tag{4.7}$$

the subset

$$\{x_{2,0}, w_0, x_{1,1}\} \tag{4.8}$$

(together with the inputs u_1, $u_2 \dots$) is sufficient to determine all undetermined variables. The influence of

$$\{x_{1,0}, x_{2,0}, u_0\} \tag{4.9}$$

is already subsumed in these variables.

So, the search algorithm that determines assignments to qualitative variables according to the specified sequence, can already at this point (after processing $x_{1,1}$) compare some of the investigated assignments (i.e. assignments to the variables (4.7). Since the subset (4.8) of these variables is enough to specify all undetermined variables, only the *best* (quality issues will be discussed later) trajectory with a specific assignment to these variables (4.8) needs to be investigated further.

The idea to compare partial assignments during a search process and to only stick to the best ones for further investigation is one of the key ideas that makes Dynamic Programming [5, 11] an efficient tool for discrete optimization. What is utilized when dropping the worse partial assignments is Belman's principle of optimality which states that the tail of an optimal solution is, itself, the optimal solution from its starting point.

A similar situation to Figure 4.11b is depicted in Figure 4.11c, if the next variables are further processed according to ordering

$$y_0 \prec x_{2,1} \prec \dots .$$

Here, partial assignments can be compared with respect to variables

$$\{x_{1,1}, x_{2,1}\}.$$

Both illustrations, Figure 4.11b-c are provided to point out that with the proposed component-based modeling approach a clever ordering of variables usually not only allows comparison with respect to the overall model's state at particular times but with respect to other small sets of variables as well. This is important, because it allows a search algorithm that processes variables sequentially to drop a lot of (partial) assignments from further investigation. This largely helps in focusing search onto good solutions.

It has to be noticed here, however, that these ideas only were demonstrated on a continuous and not a qualitative model intentionally. Because of possible spurious behaviors, application of these principles to our case, where a qualitatively abstracted model is utilized for pre-selecting mode sequences of a hybrid model, unfortunately, is not strictly straightforward. However, the motivation still is given here in anticipation of Chapter 5 where it will be shown how to utilize a slightly modified variant of the mentioned comparisons that also is applicable for qualitative pre-selection.

4.4.2 Trajectory Graph Representation of Non-Deterministic Automata

The sections above gave motivation to represent the qualitative models as *directed acyclic graphs*, because this allows an intuitive simultaneous treatment of all component models and, moreover, allows to utilize dynamic programming for focusing search onto good solutions by representing structure in the model as pre-specified *ordering of variables*. Additionally, the component models are compiled off-line and, hence, have to be stored for on-line use. So a *compact* representation of the models is required.

These three requirements

- representation as directed acyclic graph
- compact compilation
- pre-specified ordering of variables that is represented in the graph,

in principle, makes *Binary Decision Diagrams* (BDDs) [14, 3] an ideal choice as representation of the qualitative models. A brief introduction to these diagrams as they are utilized in this work are given in Appendix A.4.

Such diagrams are used for compactly representing all solutions of boolean functions as directed acyclic graphs. The qualitative models – as described in Section 4.3 – however, are encoded as non-deterministic automata. So BDDs cannot directly be applied, but a very similar type of graph can be introduced. We call it *trajectory directed acyclic graph* (tDAG). Similarly to BDDs, this graph is *directed*, *acyclic*, *compact* and it is built upon a *pre-specified ordering of variables*. Further, it can be generated and be operated on by similar operations as are used for generating and operating on BDDs.

This section, now, presents all the necessary steps of generating a tDAG from a non-deterministic automaton model and outlines the link between the tDAG and a BDD.

Ordering of Variables

One motivation for selecting a directed-acyclic-graph-representation for the qualitative model is the possibility to incorporate structural information on the hybrid model as pre-specified ordering of variables into the graph.

The graph is built such that each node corresponds to a qualitative variable and each edge that leads away from a node corresponds to a particular qualitative value for that variable. According to the specified ordering of variables, these edges only lead to nodes representing variables that come later in the ordering. The edges that lead away from the nodes corresponding to the last variable in the ordering lead to the terminal nodes representing the likelihood values. This way, an acyclic graph is obtained.

To simultaneously search multiple graphs, as briefly outlined above in section 4.4.1, a common pre-specified ordering among variables is utilized in all graphs. Additionally, this ordering of variables shall represent the hybrid system's structure. For this, first a graph that displays the dependencies among the qualitative variables as given by the equations of the hybrid model is obtained (Appendix A.3). Based on this causal graph, then qualitative variables have to be ordered into a sequence. To give a formalization of this we introduce the notation of the child variable $C(V)$ and the parent variable $P(V)$ of a variable V. These relations are defined by the same relations among the corresponding nodes in the graph.

Assuming a pre-specified sequence

$$V_1 \prec V_2 \prec \ldots V_\alpha \prec \ldots V_N, \tag{4.10}$$

based on the index α we define:

- The set of specified variables

$$\mathbb{S}(\alpha) = \{V_1, \ldots, V_\alpha\} \tag{4.11}$$

- The set of unspecified variables

$$\overline{\mathbb{S}}(\alpha) = \{V_{\alpha+1} \ldots, V_N\} \tag{4.12}$$

- The set of variables required to be able to determine the the unspecified variables

$$\mathbb{R}(\alpha) = \left\{P(V) \mid V \in \overline{\mathbb{S}}(\alpha)\ P(V) \notin \overline{\mathbb{S}}(\alpha)\right\} \tag{4.13}$$

- And the set of determined variables, the influence of which is already subsumed by the specification of other determined variables

$$\overline{\mathbb{R}}(\alpha) = \{V_i \mid V_i \in \mathbb{S}(\alpha),\ V_i \notin \mathbb{R}(\alpha)\} \tag{4.14}$$

If a search algorithm proceeds by sequentially assigning values to variables according to the specified sequence (4.10), these partial assignments can be compared to each other at each index α, where the influence of a variable gets subsumed, i.e.

$$\overline{\mathbb{R}}(\alpha) \neq \overline{\mathbb{R}}(\alpha - 1). \tag{4.15}$$

With two such indices α_1 and α_2 that satisfy

$$\overline{\mathbb{R}}(\alpha_1) \quad \neq \overline{\mathbb{R}}(\alpha_1 - 1) \tag{4.16}$$

$$\overline{\mathbb{R}}(\alpha_2) \quad \neq \overline{\mathbb{R}}(\alpha_2 - 1) \tag{4.17}$$

$$\overline{\mathbb{R}}(i) \quad = \overline{\mathbb{R}}(i - 1) \quad \mid \alpha_1 < i < \alpha_2 \tag{4.18}$$

and $D(i)$ as the number of distinct values of variable V_i, the 'maximum search graph width' (MGW) can be defined as

$$MGW(\alpha_1, \alpha_2) = \prod_{i=\alpha_1+1}^{\alpha_2} D(i). \tag{4.19}$$

To obtain a good ordering of variables that allows many comparisons of partial assignments during search and keeps the search graph narrow the following heuristic can be used: Start with the variables representing the initial state. Further, given the beginning $V_1 \prec \ldots \prec V_i$ of a sequence of variables

- for each variable $V_\alpha \in \mathbb{R}(i)$ there is a minimum set of variables $\{V_{i+1} \ldots V_{i+\beta}\}$ such that

$$V_\alpha \cap \mathbb{R}(i + \beta) = \{\}$$

  if the variables are added to the sequence in any ordering

- for each such additional sequence, there is a minimum 'maximum search graph width'

$$MGW(i + 1, i + \beta)$$

- choose the next variable in the sequence $V_{i+1} = V$ out of the set $\{V_{i+1} \ldots V_{i+\beta}\}$ that leads to

$$\min_\alpha\ MGW(i + 1, i + \beta)$$

An algorithm that determines such an ordering is given in Appendix B.1.

Binary Graph-Representation of Non Deterministic Automaton

A non-deterministic automaton as compiled in section 4.3 is defined in terms of a set of qualitative transitions. This can equivalently be regarded as an automaton with states $< X, x_d >$, where each qualitative transition specifies a transition among these states. These transitions the would be labeled with the required qualitative inputs that have to be applied in order to take the transition and the qualitative outputs that would be observed. Additionally, each transition is labeled with a likelihood value. This value, however, has no strict probabilistic interpretation but is just a heuristic indicator that helps to determine whether a qualitative behavior bears a high risk to be spurious or not.

This qualitative model shall be utilized in on-line hybrid control for an efficient preselection of promising sequences of operational modes for the individual components. To perform this task efficiently we want to change the representation of the component models. We already outlined our motivation to represent them as directed acyclic graphs. This is advantageous because it helps to incorporate information on the structure of the hybrid model into the qualitative model. Additionally, this representation makes simultaneous treatment of the individual component models straightforward. We call this graphical representation of our qualitative model a trajectory directed acyclic graph (tDAG).

Since this graph is based on Ordered *Binary* Decision Diagrams (OBDDs), we first have to encode qualitative trajectories as binary expressions. For this, each qualitative value of a variable is uniquely encoded as a binary expression, i.e. in terms of sequences of values 0 and 1. That means, to represent a qualitative variable with N qualitative values, we need

$$n = \lceil \log_2(N) \rceil$$

binary qualitative variables, where $\lceil \cdot \rceil$ stands for rounding up to the next integer value. We use a superscript B to denote binary qualitative variables, e.g.

X	$\rightarrow$	X^{B1}	X^{B2}	X^{B3}
ξ_1		0	0	0
ξ_2		0	0	1
ξ_3		0	1	0
ξ_4		0	1	1
ξ_5		1	0	0

x_d	$\rightarrow$	x_d^{B1}	x_d^{B2}
m_1		0	0
m_2		1	0
m_3		0	1

The next important thing that is needed to generate an *Ordered* Binary Decision Diagram from our qualitative model is to specify an ordering among these binary variables. The structure and size of the OBDD strongly depend on this ordering so we have to be careful when defining it. By specifying this ordering, we want to represent the 'structure' of the hybrid model in the qualitative model such that this information can be exploited to keep search for promising mode sequences focused. Above, we outlined a way how such an ordering among the qualitative variables can be obtained based on a causal analysis among the hybrid models equations. Only the qualitative variables have to be replaced by their binary counterparts.

Based on this ordering, we can generate a graph (a tree) the nodes of which represent the binary variables. Each node can be left by two edges, one represents value 0 for this variable and the other represents value 1. To generate such a graph, each (binary encoded) qualitative trajectory can be represented as a path through this tree from the first variable in the ordering down to the last, such that the edges leaving a node point to that variable in the qualitative trajectory which comes next in ordering. The edges that leave the nodes

representing the last variable in ordering point toward special leaf nodes which represent the likelihood value of the respective qualitative trajectory.

For example, a qualitative model with binary variables $B1_k$, $B2_k$, $B1_{k+1}$ and $B2_{k+1}$ is given as

$B1_k$	$B2_k$	$B1_{k+1}$	$B2_{k+1}$	L
0	0	0	1	0.570
0	0	1	1	0.013
0	1	1	1	0.726
1	0	0	0	0.521
1	0	0	1	0.007
1	0	1	0	0.012
1	0	1	1	0.0002
1	1	1	0	0.663
1	1	1	1	0.009

With an ordering among these variables specified as

$$B1_k \prec B2_k \prec B1_{k+1} \prec B2_{k+1}$$

the resulting tree looks like shown in Figure 4.12. This graph is very similar to an OBDD

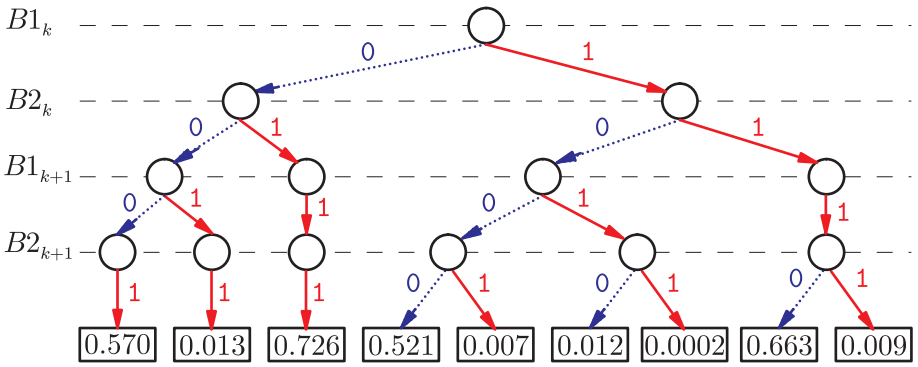


Figure 4.12: Tree representation of a binary qualitative model

with the exception that the leaf nodes represent likelihood values instead of boolean values `true` and `false`.

However, the same reduction techniques as for OBDDs can be applied to obtain a more compact representation. That is

- Duplicate leaf nodes – that is, ones that represent the same likelihood value – can be combined to a single node and the edges pointing towards one of the duplicates get redirected towards that single node.

- Duplicate non-terminals – that is, two nodes in the graph that both have one common target node their 1-edge is pointing to and one common target node their 0-edge is pointing to – can be combined likewise.

- Redundant tests – that is, a node the 0-edge and the 1-edge of which point to the same target node – can be eliminated. All edges that pointed to this node get redirected to that target.

A more detailed description to these reduction operations is found in Appendix A.4.

The problem with respect to these reduction operations in our context is, that with OBDDs the two latter ones are particularly successful because the first one combines all leaves of the tree to just two nodes representing the two boolean values `true` and `false`. However, terminal nodes in the tDAG represent a (usually very large) number of different likelihood numbers! So a large number of nodes will remain after the reduction.

However, our likelihood values do not have a strict probabilistic interpretation anyway. They just serve as heuristic indicators that help to determine whether a particular qualitative trajectory bears a high risk of being spurious or not. With this in mind it seems justified to approximate them in order to obtain more compact models.

Approximation for Reducing Graph Size

A Binary Decision Diagram generally is a very compact representation of a boolean function, if the underlying ordering of variables is chosen appropriately. However, this statement does not hold so far for our trajectory graphs, if they use terminal nodes that represent the likelihood values as calculated for each transition specification.

The qualitative models are utilized to *quickly* reason about a *good* (though not necessarily the best) sequence of operational modes that can be utilized to greatly simplify the subsequent numerical determination of continuous controls. This numerical refinement additionally provides a verification of the qualitative pre-selection and detects possible spurious behaviors.

Further, the calculated likelihood values generally are no exact probabilities imposed by a stochastic behavior of the hybrid system itself, but are just *heuristic indicators* that help to determine whether a particular qualitative trajectory might be spurious. So, in this context of our qualitative model, it seems justified to regard these likelihoods on a qualitative basis as well. Utilizing just a few different likelihood values to discriminate qualitative trajectories that bear a *very high* risk of contributing to spurious behaviors from others where this risk is *almost neglectable* even provides a more focussed view on the 'nature' of a qualitative trajectories than the original likelihood values.

Combining similar likelihood values to a 'likelihood class' and, hence using only a single leaf node in the graph as terminal node for the respective qualitative trajectories can help in largely reducing the model size. An open question remains how to cleverly group likelihood values into such likelihood classes. In our examples an evenly spaced separation of the range of likelihood values (0 to 1) into up to 4 regions usually works well. To get best approximation, each likelihood class then is assigned the mean likelihood value of the likelihoods of all qualitative trajectories that belong to the class. With M specifying the number of likelihood classes, N_j specifying the number of qualitative trajectories that belong to the j^{th} class and L_{ji} being the likelihood value of the i^{th} transition specification that belongs to the j^{th} class, the numerical likelihood value assigned to each class is

$$\overline{L_j} = \frac{1}{N_j} \sum_{i=1}^{N_j} L_{ji} \tag{4.20}$$

and the overall quality of the approximation of transition likelihoods by the M likelihood classes can be evaluated by the approximation inaccuracy

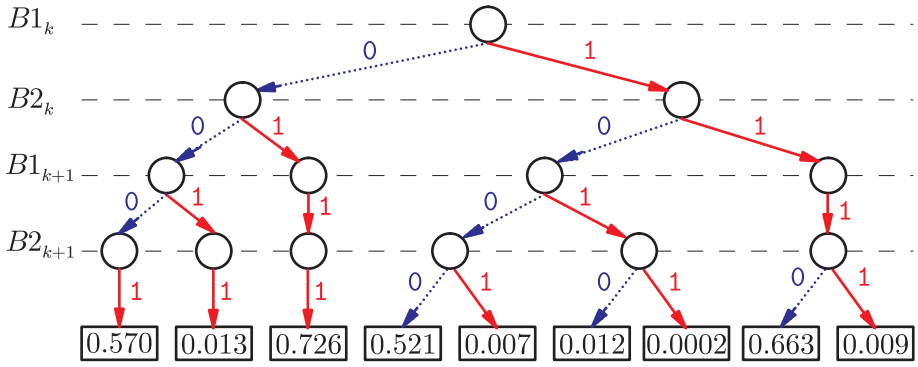
$$a = \frac{1}{\sum_{j=1}^{M} N_j} \sum_{j=1}^{M} \left(\sum_{i=1}^{N_j} \left(L_{ji} - \overline{L_j} \right)^2 \right) \tag{4.21}$$

This value together with the number of likelihood classes can be utilized to find a good compromise between the approximation of transition likelihood values and model size. Good approximations with respect to this measure can be obtained by an algorithm [45] that utilizes a histogram of likelihood values as input.

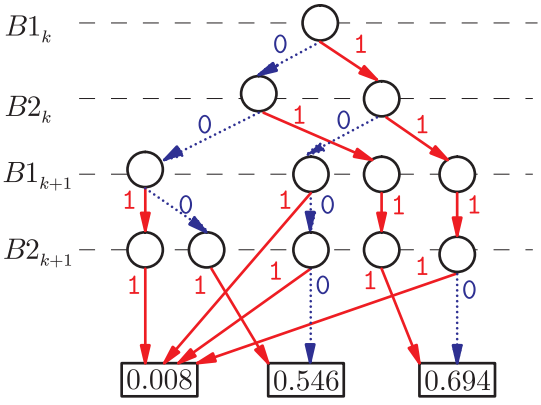
An example is provided in Figure 4.13, where trajectory graphs for different numbers of likelihood classes are shown. The calculation of likelihood values for this example are based on the model

$$\mathbf{x}_{k+1} = \begin{bmatrix} 0 & 0.9 \\ -0.96 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0.12 \\ 0.1 \end{bmatrix}$$

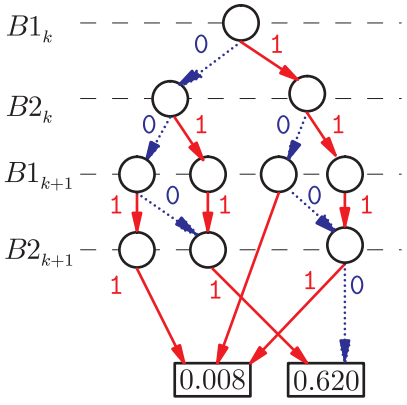
with the state vector $\mathbf{x} = [x_1, x_2]^T$ abstracted to a qualitative variable with 4 values, based on the sign of x_1 and x_2. This qualitative variable X has a binary representation with $B1 = 1$ and $B2 = 1$ indicating the positive sign of x_1 and x_2, respectively.



(a) no approximation



(b) 3 likelihood classes



(c) 2 likelihood classes

Figure 4.13: Compaction of a model by likelihood classes

The calculated transition specifications and likelihood values are

$B1_k$	$B2_k$	$B1_{k+1}$	$B2_{k+1}$	no approximation	3 classes	2 classes
0	0	0	1	0.570	0.546	0.620
0	0	1	1	0.013	0.009	0.009
0	1	1	1	0.726	0.694	0.620
1	0	0	0	0.521	0.546	0.620
1	0	0	1	0.007	0.009	0.009
1	0	1	0	0.012	0.009	0.009
1	0	1	1	0.0002	0.009	0.009
1	1	1	0	0.663	0.694	0.620
1	1	1	1	0.009	0.009	0.009

If no approximation is used, the resulting trajectory graph consists of 22 nodes and the approximation inaccuracy clearly is $a = 0$ (Figure 4.13a). If likelihood values are separated into 3 distinct classes, the resulting trajectory graph is reduced to 15 nodes and the approximation inaccuracy is $a = 3.7 \cdot 10^{-4}$ (Figure 4.13b). If likelihood values are separated into only 2 distinct classes, the resulting trajectory graph is further reduced to 12 nodes and the approximation inaccuracy increases to $a = 2.8 \cdot 10^{-3}$ (Figure 4.13c).

Final tDAG

The qualitative model as specified so far, already satisfies the requirements postulated in Section 4.1. The hierarchical structure of the trajectory graphs that is based on a common ordering of qualitative variables allows an intuitive simultaneous reasoning among all component models. The carefully selected ordering of variables allows efficient utilization of dynamic programming for searching the models. And, last but not least, component based modeling and the use of only a few likelihood classes allow to obtain reasonably small models, although they represent an explicit enumeration of all possible qualitative trajectories for time $t_k \rightarrow t_{k+1}$.

To motivate one last change in representation of the qualitative model, a further brief look ahead to on-line qualitative pre-selection (Chapter 5) is taken: In on-line qualitative hybrid control, the models will be used to reason about 'good' qualitative trajectories, where quality of a qualitative trajectory is defined in terms of a high likelihood and a good correspondence to actual reference values for certain variables. While the exact definition of this 'quality' of trajectories is not yet relevant here, we already notice two things:

- Evaluation of the correspondence to reference values will only be accomplished efficiently, if a qualitative variable is fully specified (i.e. if only values for some of the binary variables that contribute to a qualitative variable are specified, this measure usually cannot be evaluated efficiently.)

- Evaluation of transition likelihoods can only be done in the current representation of qualitative models, if a full path through the graph is specified

To enable a good interaction between the two criteria for evaluating trajectory-quality (likelihood and matching a reference) it would be helpful to be able to evaluate both at the same time based on partial assignments of variables.

Hence, representation of the graphs is changed such that transition likelihoods can be estimated on behalf of partial paths in the graph as well. This is accomplished by mapping

the likelihood values of the terminal nodes to the edges of the graph. This requires the edges likelihoods to be set such that multiplying all edge-likelihoods along a path from the root node to a leaf node leads to the likelihood value represented by that node.

With this, the search for a likely trajectory shall be formulated as a shortest path search on the graph. To be able to utilize standard algorithms for this, the likelihood values are represented as path lengths, where an optimal likelihood of 1 represents a path length of 0 and the impossible likelihood 0 represents an infinitely long path. To accomplish this, approximate likelihood-class values $\overline{L}$ are mapped to likelihood costs C_L by taking the standard approach

$$C_L = -\ln(\overline{L}). \tag{4.22}$$

This maps the multiplication of edge likelihoods to a summation of edge lengths. Additionally, path costs are normalized

$$\overline{C_{L,i}} = C_{L,i} - \min(C_L),$$

such that the most likely path always is normalized to a length of 0.

If, now, transition likelihoods are represented by edge-lengths in the graph, this allows a search algorithm to discriminate more likely partial variable assignments against rather unlikely ones as well. However, during qualitative pre-selection not only the likelihood-costs of a qualitative trajectory have to be considered, but also its correspondence to a given reference trajectory has to be evaluated. As already postulated above, this evaluation can only be accomplished efficiently, if a qualitative variable is fully specified (i.e. all corresponding binary variables are known). As long as only some of its corresponding binary variables are known, only the transition likelihoods compiled into the qualitative model can be utilized to focus search onto 'good' solutions.

So, since in-between points in the model graphs where qualitative variables are fully specified only transition likelihood will guide search for 'good' qualitative trajectories, it is useful only to include these qualitative variables as nodes into the final model graph. All paths in between these nodes are then explicitly represented as separate edges.

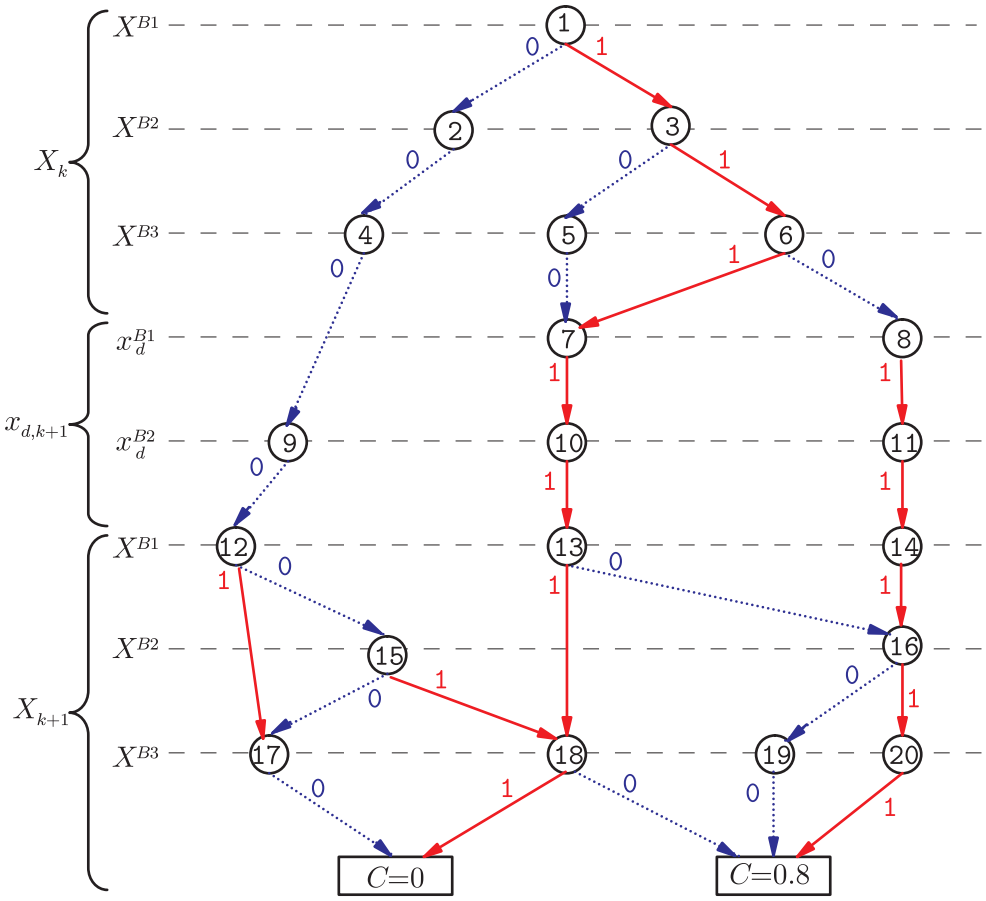
An algorithm that achieves this is given in Appendix B.2. In fact, this algorithm only removes the binary variables from the graph and, again, replaces them by the original qualitative variables. This way, the model graphs, at each node, will not only allow a binary decision among values 0 and 1, but will allow to decide among a possibly larger number of different qualitative values. These possible choices are sorted, such that the choice with the lowest associated edge-cost is ordered first.

An example is provided in Figure 4.14, where the binary version of a trajectory graph (Figure 4.14a) is transformed to its final representation as tDAG (Figure 4.14b).

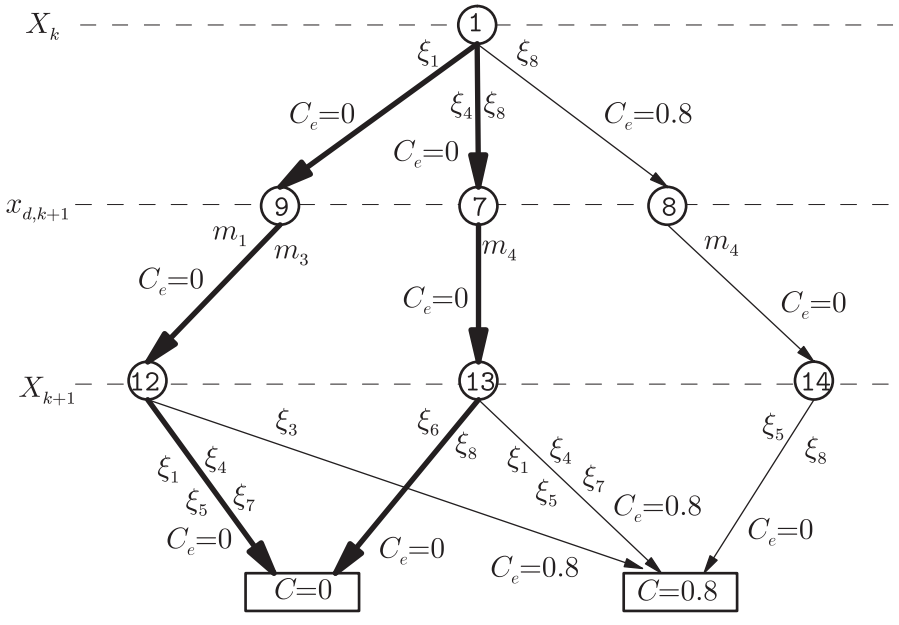
4.5 Summary

This chapter presented a type of qualitative model, that can be used to very compactly abstract the class of hybrid systems as defined in Chapter 3.

Compactness of the model is achieved by abstracting the hybrid model component-wise and by utilizing a compact graphical representation similar to reduced Ordered Binary Decision Diagrams. Further, the various component models all utilize the same pre-specified ordering of variables to define a hierarchic architecture of the graphs, so that they can simultaneously be processed intuitively, if a shortest path search algorithm proceeds according to the same ordering of variables. Moreover, the specification of this ordering based on the



(a) Binary trajectory graph



(b) Corresponding tDAG

Figure 4.14: Elimination of binary variables

structure of the hybrid model allows efficient utilization of dynamic programming ideas during search.

The presented type of qualitative model is based on a non-deterministic automaton and, likewise, encodes only transitions of the qualitatively abstracted hybrid state for times $t_k \rightarrow t_{k+1}$ and is re-used for reasoning about longer time-trajectories. However, it differs from a non-deterministic automaton in that it makes use of only a small number of heuristic likelihood classes, instead of specifying a separate likelihood for each state-transition.

The likelihood classes are utilized to overcome an unavoidable problem of qualitative modeling: spurious behaviors. These are qualitative trajectories that are not an abstraction of any possible trajectory of the hybrid model with a given initial state. However, the likelihood classes help to estimate whether there is a high risk that a particular qualitative trajectory leads to a spurious behavior. The likelihood classes do not provide estimates that lead exact results on the probability of spuriosity, but nevertheless they help to focus qualitative search towards trajectories that are non-spurious.

Compilation of the qualitative model – especially the basic non-deterministic automaton – can be a computationally intensive task for higher dimensional models as this requires an enumeration of all valuations of all qualitative variables. However, with the component-based hybrid modeling scheme of Chapter 3 and component-based qualitative abstraction, fairly low-dimensional models can be assumed that have to be abstracted. Moreover, compilation is performed off-line. So computation time is not a prime issue.

The qualitative model in its final representation is a set of directed acyclic graphs, each abstracting a component of the hybrid model. Each graph consists of a set of nodes that, each (except the terminal nodes), represent a qualitative variable. The nodes are connected by edges, where each of the edges represents a set of qualitative values. Additionally, each edge has a path length that is determined by the approximated likelihood values.

Chapter 5

Hybrid Control

This chapter specifies an on-line hybrid control scheme that makes efficient use of a pre-compiled qualitative model to quickly pre-select promising sequences of operational modes before numerically refining continuous actuation and verifying the results

The previous chapter presented *off-line compilation* of a special type of qualitative model. This model will now be used for qualitatively *pre-selecting* sequences of operational modes and discrete actuation in an *on-line hybrid control* scheme.

In Section 5.1, the overall hybrid control scheme is outlined. It is composed of two parts:

- a qualitative part, that solves a discrete control task posed on a symbolic model
- a numerical part, that solves a continuous control task posed on a continuous time-variant model.

Then, the qualitative pre-selection is thoroughly discussed in Section 5.2. This qualitative pre-selection is an important part in the hybrid control scheme because it needs to *quickly* evaluate a sequence of operational modes that allows to actuate the system such that it closely follows a specified reference behavior. To achieve this task, the specialized qualitative model developed in Chapter 4 is utilized and a qualitative abstraction of the hybrid control goal is determined to reformulate hybrid control as discrete shortest-path search on the qualitative model.

The qualitative model is especially designed, such that this discrete search can be performed by an class of search algorithms known as A^*-search [35] very efficiently. These algorithms work especially well with the presented qualitative model, because they can directly exploit the structural information on the hybrid model that has been compiled into the qualitative model.

Section 5.3 addresses the subsequent numerical refinement of continuous actuation and Section 5.4 finally discusses the issue of spurious behaviors and how the hybrid control scheme handles and avoids them by a careful interplay between the qualitative and the numerical solver.

5.1 Outline of the Control Scheme

Hybrid control is difficult to achieve for multi-component systems, because one has to (discretely) select among a huge number of discrete mode-sequences, while simultaneously evaluating continuous dynamics and determining continuous actuation.

This procedure is greatly simplified by splitting it into two distinct operations:

- discrete qualitative pre-selection of promising mode sequences
- numerical evaluation of continuous system inputs.

The idea behind this procedure is to separate the control task into the above two domains, qualitative control of a symbolic model and numerical control of a time-variant continuous model *and* – what is the key idea to the procedure – to encode the constraints imposed by the respective other control task in the language of the respective domain.

This means, qualitative control operates on the qualitative part of the hybrid control problem, while being supported with a qualitative encoding of the constraints imposed by the continuous dynamics. Likewise, subsequent numerical control operates on the continuous dynamics part of the hybrid control problem. However with the discrete mode sequence specified, (thus, the qualitative control already folded into the hybrid model), the remaining control task can be formulated solely in the continuous domain.

This allows to use the two solvers sequentially instead of the simultaneous operation required in other hybrid control formulations such as the control of mixed logic dynamic systems (MLD) [9]. A recent development in that line of research, also, uses a dedicated qualitative solver to support the numerical control design [8], however, the two solvers are still required to operate simultaneously.

5.1.1 Hybrid Control Task

To outline the two-step approach to hybrid control, first the continuous control task is defined. The continuous part of the hybrid control problem is specified by an N-step finite horizon optimization problem of trajectories involving continuous states

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(L)} \end{bmatrix}$$

(where $\mathbf{x}^{(i)}$ denotes the continuous state of the i^{th} component in a hybrid model with L components) and the external inputs

$$\mathbf{u} = [u_1, \ldots, u_M]^T.$$

For these variables, reference values at different times within the receding horizon of length N can be defined at system run-time, i.e.

$$\begin{array}{ll} \left\{\mathbf{r}_{x,1}^{(1)}, \ldots, \mathbf{r}_{x,1}^{(L)}\right\} & \{r_{u1,0}, \ldots, r_{uM,0}\} \\ \left\{\mathbf{r}_{x,2}^{(1)}, \ldots, \mathbf{r}_{x,2}^{(L)}\right\} & \{r_{u1,1}, \ldots, r_{uM,1}\} \\ \vdots & \vdots \\ \left\{\mathbf{r}_{x,N}^{(1)}, \ldots, \mathbf{r}_{x,N}^{(L)}\right\} & \{r_{u1,N-1}, \ldots, r_{uM,N-1}\} \end{array}$$

Deviations from these reference values, then, are defined as

$$\begin{aligned} \mathbf{e}_{x,k}^{(i)} &= \mathbf{x}_k^{(i)} - \mathbf{r}_{x,k}^{(i)} \\ e_{ui,k} &= u_{i,k} - r_{ui,k}. \end{aligned}$$

With respect to the receding horizon control strategy, for these deviations from the continuous reference-values, a cost value

$$c_{Rc} = \sum_{k=1}^{N} \left(\sum_{i=1}^{L} q_{x,k}^{(i)} \, \mathbf{e}_{x,k}^{(i)\,T} \, \mathbf{e}_{x,k}^{(i)} + \sum_{i=1}^{M} q_{ui,k-1} e_{ui,k-1}^{2} \right) \tag{5.1}$$

is defined, where we can use design parameters

$$q_{x,k}^{(i)} \quad \text{and} \quad q_{ui,k-1}$$

to put more emphasis on certain components or times.

Similarly, reference values for the operational mode for each component and time can be specified. However, these references cannot directly be used to derive penalty costs for deviations thereof, because no specification is made what makes a particular operational node being 'close' to another. So, besides run-time definition of reference modes

$$r_{d,k}^{(i)},$$

as design parameter, just specification of penalty costs

$$q_{d,k}^{(i)}$$

for any other mode than the reference is allowed

$$e_{d,k}^{(i)} = \begin{cases} 0 & \text{if } x_{d,k}^{(i)} = r_{d,k}^{(i)} \\ 1 & \text{if } x_{d,k}^{(i)} \neq r_{d,k}^{(i)} \end{cases} . \tag{5.2}$$

So, the cost value for deviations from reference modes is evaluated

$$c_{Rd} = \sum_{k=1}^{N} \sum_{i=1}^{L} q_{d,k}^{(i)} e_{d,k}^{(i)} \tag{5.3}$$

The hybrid control task is now, to actuate the hybrid system by its

$$\text{continuous inputs} \qquad \underline{\mathbf{u}}^T = [\mathbf{u}_0^T, \ldots, \mathbf{u}_{N-1}^T] \tag{5.4a}$$

$$\text{discrete inputs} \qquad \underline{\mathbf{u}}_d^T = [u_{d1,0}, \ldots, u_{dO,0}, \ldots, u_{d1,N-1}, \ldots, u_{dO,N-1}], \tag{5.4b}$$

such that the resulting cost values are minimal, i.e. the hybrid control task is to determine $\underline{u}$ and $\underline{u}_d$ such that

$$\min_{\underline{\mathbf{u}}, \underline{\mathbf{u}}_d} (c_{Rc} + c_{Rd}) \tag{5.5}$$

However, beforehand a qualitative reasoning step is utilized to quickly pre-select discrete inputs and a mode sequence. Although this control scheme, in principle, can determine this optimal solution (as will be pointed out in Section 5.4.3), it is by far more efficient to allow sub-optimal solutions. For more complex systems, this is necessary to keep the computational burden manageable.

So with c_{Rc}^* and c_{Rd}^* being the values at the optimal solution to (5.5), in fact, the modified hybrid control problem

$$\text{determine} \quad \underline{\mathbf{u}} \quad \text{and} \quad \underline{\mathbf{u}}_d \quad \text{such that} \qquad c_{Rc} + c_{Rd} \approx c_{Rc}^* + c_{Rd}^* \tag{5.6}$$

is solved.

5.1.2 Qualitative Control Problem Formulation

A major part in obtaining a problem formulation that allows separation of the hybrid control task into qualitative pre-selection and *subsequent* numerical control refinement has already been achieved by compiling the continuous dynamics of the hybrid model at each mode into the qualitative model. What is left, is to abstract formulation (5.6) to be used in the qualitative domain.

The discrete cost value c_{Rd} is already defined in terms of the discrete operational modes, so it can directly be used in the qualitative model. The definition of the continuous cost value c_{Rc}, however, has to be qualitatively abstracted to be usable with respect to qualitative values. Unlike the discrete modes, where there was the problem that no 'function' could be obtained for evaluating the distance between two qualitative values, for the qualitative abstractions of continuously valued variables such a distance can be evaluated on behalf of the definition of the polytopic regions that are represented by each qualitative value.

Along this argument the deviation $E^{(i)}_{x,k}$ of a qualitative value $X^{(i)}_k = \xi_\alpha$ from the reference value $\mathbf{r}^{(i)}_{x,k}$ can be defined similar to (5.2). For this, the definition of qualitative values by polytopic regions

$$\mathbf{H}_\alpha \mathbf{x} \leq \mathbf{k}_\alpha \longleftrightarrow X = \xi_\alpha$$

is utilized to define the set $\mathbb{P}$ of all values 'inside the qualitative value'.

$$\mathbb{P}(\xi_\alpha) = \{\mathbf{x} \mid \mathbf{H}_\alpha \mathbf{x} \leq \mathbf{K}_\alpha\}$$

With this, the deviation $E^{(i)}_{x,k}$ of a qualitative value $X^{(i)}_k = \xi_\alpha$ from the reference value $\mathbf{r}^{(1)}_{x,k}$ is defined as

$$E^{(i)}_{x,k} = \min_{\mathbf{x}^{(i)}_k \in \mathbb{P}(\xi_\alpha)} \left(\mathbf{x}^{(i)}_k - \mathbf{r}^{(i)}_{x,k}\right)^T \left(\mathbf{x}^{(i)}_k - \mathbf{r}^{(i)}_{x,k}\right)$$

and similarly

$$E_{ui,k} = \min_{u_{i,k} \in \mathbb{P}(\upsilon_\alpha)} \left(u_{i,k} - r_{ui,k}\right)^2 .$$

Determination of these values, however, requires the solution of a quadratic program. As reference values are determined at system run-time, values $E^{(i)}_{x,k}$ and $E_{ui,k}$ cannot be pre-compiled and solving a quadratic program for each such evaluation during qualitative pre-selection is prohibitive.

Therefore, once again an approximation will be used that under-estimates the distances $E^{(i)}_{x,k}$ and $E_{ui,k}$: The smallest bounding ball $B(\cdot)$ for each qualitative value's polytopic region is pre-compiled and its center $\mathbf{c}_B$ and radius r_B are stored.

With

$$\overline{\mathbb{P}}(\xi_\alpha) = \mathbb{P}(B(\xi_\alpha))$$

being the range of all states $\mathbf{x}$ inside the smallest bounding ball around the polytope related to qualitative value ξ_α, approximative values

$$\overline{E}^{(i)}_{x,k} = \min_{\mathbf{x}^{(i)}_k \in \overline{\mathbb{P}}(\xi_\alpha)} \left(\mathbf{x}^{(i)}_k - \mathbf{r}^{(i)}_{x,k}\right)^T \left(\mathbf{x}^{(1)}_k - \mathbf{r}^{(i)}_{x,k}\right) \tag{5.7}$$

$$\overline{E}_{ui,k} = \min_{u_{i,k} \in \overline{\mathbb{P}}(\upsilon_\alpha)} \left(u_{i,k} - r_{ui,k}\right)^2 \tag{5.8}$$

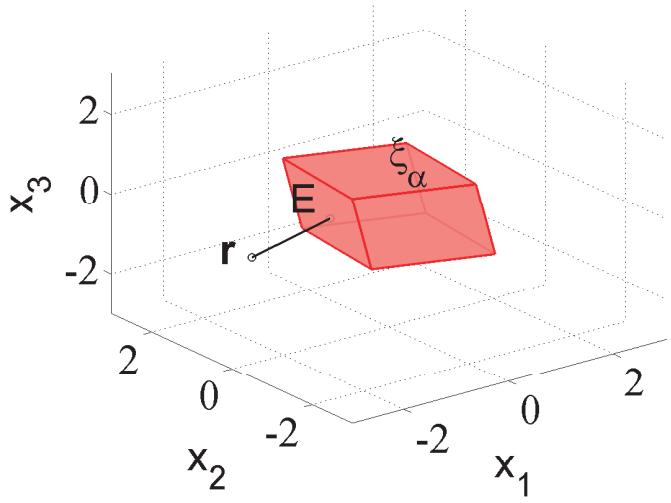
for the deviations from reference values can be defined.

These distances most easily can be evaluated at system run-time by

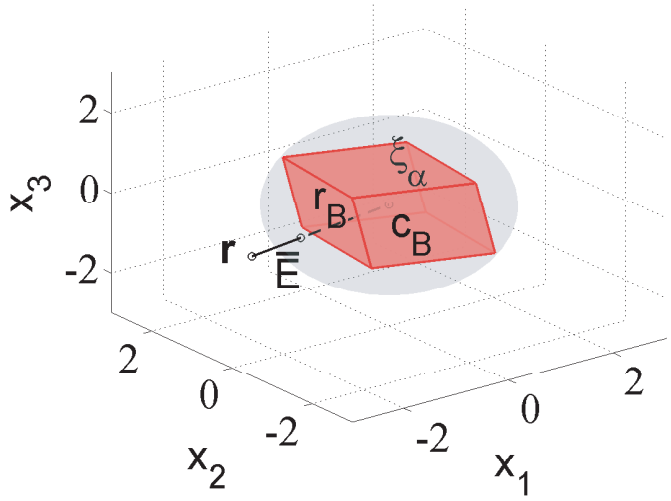
$$\overline{E}^{(i)}_{x,k} = \left(|\mathbf{c}_B(\xi_\alpha) - \mathbf{r}^{(i)}_{x,k}| - r_B(\xi_\alpha)\right)^T \left(|\mathbf{c}_B(\xi_\alpha) - \mathbf{r}^{(i)}_{x,k}| - r_B(\xi_\alpha)\right) \tag{5.9}$$

$$\overline{E}_{ui,k} = (c_B(\upsilon_\alpha) - r_{ui,k} - r_B(\upsilon_\alpha))^2 \tag{5.10}$$

with the only exception that they additionally are limited to zero from below. An illustration of these distance-values E is shown in Figure 5.1.



(a) Minimum distance from reference value to polytope



(b) Easy-to-compute-approximation of the distance from reference value to polytope

Figure 5.1: Deviations from reference values for qualitative values

With these approximations, the qualitative abstraction of the cost value c_{Rc} can be defined

as

$$C_{Rc} = \sum_{k=1}^{N} \left(\sum_{i=1}^{L} q_{x,k}^{(i)} \overline{E}_{x,k}^{(i)} + \sum_{i=1}^{M} q_{ui,k-1} \overline{E}_{ui,k-1} \right) \tag{5.11}$$

So, the qualitative control task can be formulated as a discrete search for the system's actuation, i.e.

$$\begin{aligned} \text{abstracted continuous inputs} \quad & \underline{\mathbf{U}} = [U_{1,0}, \ldots, U_{M,0}, \ldots, U_{1,N-1}, \ldots, U_{M,N-1}] \\ \text{discrete inputs} \quad & \underline{\mathbf{u}}_d = [u_{d1,0}, \ldots, u_{dO,0}, \ldots, u_{d1,N-1}, \ldots, u_{dO,N-1}], \end{aligned}$$

such that the qualitative approximations of the cost values are minimal

$$\min_{\underline{\mathbf{U}} \in \underline{\mathcal{U}}\,,\, \underline{\mathbf{u}}_d \in \underline{\mathcal{U}}_d} \left(C_{Rc} + c_{Rd} \right). \tag{5.12}$$

However, since the qualitative model and the control task formulation are only approximations of the hybrid model and the hybrid control task, respectively, it cannot be guaranteed that the optimal solution

$$\underline{\mathbf{U}}^*, \underline{\mathbf{u}}_d^*$$

to the qualitative control task (5.12) is an abstraction of the optimal solution to the hybrid control task (5.5).

However, if qualitative values are chosen with an appropriate granularity, this solution to the qualitative control task should at least be an abstraction of a sub-optimal solution according to (5.6). But even this may not hold since the solution to the qualitative control task can be based on a spurious behavior. That means, it is not an abstraction at all of any valid trajectory the hybrid model can exhibit from its given initial state. If this is the case, the subsequent the numerical control will detect such a situation and qualitative search is resumed to provide the next-best qualitative solution.

To reduce the risk of finding solutions to the qualitative control problem that are spurious behaviors, the likelihood values related to qualitative trajectories are included into the qualitative control problem formulation. These likelihood values are compiled into the graphical representation of the qualitative model as edge-lengths of a directed, acyclic graph. They provide an indication whether a particular qualitative trajectory-segment bears a higher or a lower risk to lead to a spurious solution.

To formalize this for qualitative control, a particular valuation of all qualitative variables within the N-step receding horizon is called a qualitative trajectory $\mathbf{Q}$. Based on this and the qualitative model definition, an additional cost value C_L can be specified that penalizes a high risk of spuriosity as the follows: C_L is the sum of all edge-costs along the paths (through all tDAGs) that are specified by the qualitative trajectory $\mathbf{Q}$.

Additionally, to bias qualitative search towards more 'likely' or more 'accurate' solutions, scaling factors s_L and s_R are allowed as design parameters. High values of s_L lead to concentration on most likely non-spurious qualitative trajectories on the one hand and high values of s_R emphasize qualitative trajectories that pretend to be (they might be spurious!) abstractions of very good hybrid trajectories on the other hand.

With these scaling factors (and $\mathcal{Q}$ denoting the set of all possible qualitative trajectories), the qualitative control problem is finally defined as search for a qualitative trajectory $\mathbf{Q}$ that achieves

$$\min_{\mathbf{Q} \in \mathcal{Q}} \left(s_L C_L + s_R \left(C_{Rc} + c_{Rd} \right) \right). \tag{5.13}$$

5.1.3 Numerical Control Problem Formulation

The solution $\mathbf{Q}$ to the qualitative control problem (5.13) specifies all discrete variables of the hybrid model for all times in the N-step receding horizon, but it specifies the continuous variables only to the extent of a polytope.

Therefore, an additional numerical control-deduction step is needed to specify these variables in more detail. This numerical control formulation, however, has not to be posed on the complex 'original' *hybrid* model, because the solution $\mathbf{Q}$ to the qualitative control task already specifies a sequence $\mathbf{M}^{(i)}$ of operational modes for each component of the hybrid model.

Continuous Model Representation of each Component

This allows to regard each component-hybrid-automaton

$$\mathcal{A}^{(i)} = \langle \mathbf{x}_h^{(i)}, \mathbf{u}_h^{(i)}, \mathbf{y}_h^{(i)}, F^{(i)}, T^{(i)}, \mathbf{X}^{(i)}, \mathbf{U}^{(i)}, \mathcal{X}_d^{(i)}, \mathcal{U}_d^{(i)}, T_s \rangle$$

as a time-variant continuous model. The constraints related to the determined mode-transitions can be expressed in terms of a polytope the model inputs have to be in.

Due to the specified mode sequence, the continuous part of the hybrid model can be expressed as time-variant continuous model

$$\begin{aligned} \mathbf{y}_k^{(i)} &= \mathbf{C}_k^{(i)} \mathbf{x}_k^{(i)} + \mathbf{D}_k^{(i)} \mathbf{u}_k^{(i)} + \mathbf{f}_k^{(i)} \\ \left(\mathbf{x}_k^{(i)}\right)' &= \mathbf{R}_k^{(i)}\, \mathbf{x}_k^{(i)} + \mathbf{r}_k^{(i)} \\ \mathbf{x}_{k+1}^{(i)} &= \mathbf{A}_{k+1}^{(i)} \left(\mathbf{x}_k^{(i)}\right)' + \mathbf{B}_{k+1}^{(i)} \mathbf{u}_k^{(i)} + \mathbf{e}_{k+1}^{(i)} \end{aligned} \tag{5.14}$$

The unusual part in these equations comes from the definition of a state-reset

$$\left(\mathbf{x}_k^{(i)}\right)' = \mathbf{R}_k^{(i)}\, \mathbf{x}_k^{(i)} + \mathbf{r}_k^{(i)}$$

specified for each mode transition. If the mode stays the same, $\left(\mathbf{x}_k^{(i)}\right)' = \mathbf{x}_k^{(i)}$.

Regarding the automaton-part of the hybrid model, the mode sequence $\mathbf{Q}$, of course specifies all discretely valued variables. However, the satisfaction of the guards of the involved mode transitions is assured with respect to the discrete actuation $u_{d,0}^{(i)}, \ldots, u_{d,N-1}^{(i)}$ only. The involved polytopic constraints on the continuous states and inputs have to be passed on to the continuous control problem as well as the limited allowed range of each continuous variable has to be considered. These constraints can, all together, be expressed as:

$$\begin{aligned} \mathbf{H}_{x,k}^{(i)} \mathbf{x}_k^{(i)} &\leq \mathbf{k}_{x,k}^{(i)} \\ \mathbf{H}_{u,k}^{(i)} \mathbf{u}_k^{(i)} &\leq \mathbf{k}_{u,k}^{(i)} \\ \mathbf{H}_{y,k}^{(i)} \mathbf{y}_k^{(i)} &\leq \mathbf{k}_{y,k}^{(i)}. \end{aligned} \tag{5.15}$$

If the initial states $\mathbf{x}_0^{(i)}$ are known, the component models (5.14) can be used to reformulate the constraints on states and outputs as constraints on the inputs as well. For example, dropping superscripts and assuming $\left(\mathbf{x}_k^{(i)}\right)' = \mathbf{x}_k^{(i)}$:

$$\begin{aligned} \mathbf{H}_{x,1} \mathbf{x}_1 &\leq \mathbf{k}_{x,1} \\ \mathbf{H}_{x,1} \left(\mathbf{A}_1 \mathbf{x}_0 + \mathbf{B}_1 \mathbf{u}_0 + \mathbf{e}_1\right) &\leq \mathbf{k}_{x,1} \\ \mathbf{H}_{x,1} \mathbf{B}_1 \mathbf{u}_0 &\leq \mathbf{k}_{x,1} - \mathbf{H}_{x,1} \mathbf{A}_1 \mathbf{x}_0 - \mathbf{H}_{x,1} \mathbf{e}_1 \end{aligned}$$

All terms on the right are known, so we can write

$$\overline{\mathbf{H}}\mathbf{u}_0 \leq \overline{\mathbf{k}}$$

If additionally notation

$$\underline{\mathbf{u}}^T = [\mathbf{u}_0^T, \ldots, \mathbf{u}_{N-1}^T]$$

is used, all constraints on states, inputs and outputs together can compactly be expressed as

$$\mathbf{H}_{\underline{u}}\underline{\mathbf{u}} \leq \mathbf{k}_{\underline{u}}. \tag{5.16}$$

With this and the specification of the cost value c_{Rc} (5.1), the remaining continuous control task is to determine inputs $\overline{\mathbf{u}}^*$ such that

$$\min_{\mathbf{H}_{\underline{u}}\overline{\mathbf{u}}^* \leq \mathbf{k}_{\underline{u}}} c_{Rc}. \tag{5.17}$$

Notation $\overline{\mathbf{u}}^*$ is used to express that this is the optimal input with respect to the hybrid control problem for a specified mode sequence only. This hasn't necessarily to be equal to the continuous input $\underline{\mathbf{u}}^*$ that leads to the optimal solution of the hybrid control problem (5.5).

5.1.4 Hybrid Control

Both simplified control problems (5.13) and (5.17) are solved. Then, in spirit of receding horizon control, the discrete commands $\mathbf{u}_{d,0}$ specified by the 'qualitative solution' $\mathbf{Q}$ and the continuous actuation $\mathbf{u}_0$ specified by the 'continuous solution' $\overline{\mathbf{u}}^*$ are applied to the hybrid system and the controller waits until at the next sample time a new N-step hybrid control problem is solved.

An illustrative view of this control scheme (already displayed as Figure 2.12) is repeated here in Figure 5.2.

5.2 Qualitative Pre-Selection

This section, details how to solve the qualitative control problem, i.e. to search the qualitative model for an assignment to qualitative variables $\mathbf{Q} \in \mathcal{Q}$ that leads to a minimal cost value with respect to

$$\min_{\mathbf{Q}\in\mathcal{Q}} \left(s_L C_L + s_R \left(C_{Rc} + c_{Rd}\right)\right)$$

5.2.1 Simultaneous Search in Multiple Graphs

The first problem encountered when searching the qualitative model for an assignment $\mathbf{Q}$ of qualitative values is, that the qualitative model is presented in form of multiple graphs, which each encode qualitative trajectories of a single component and a single time-step. However, the fact that all graphs are built with respect to the same ordering of variables

$$V_1 \prec V_2 \prec \ldots V_\alpha \prec V_{\alpha+1} \prec \ldots.$$

makes a simultaneous treatment of all graphs straightforward. To demonstrate this, we first recall how these tDAGs represent the overall qualitative model. Each node in such a tDAG (except the leaf nodes) represents a qualitative value. The nodes are left by edges that represent allowed choices of qualitative values for the variable. So assigning a qualitative

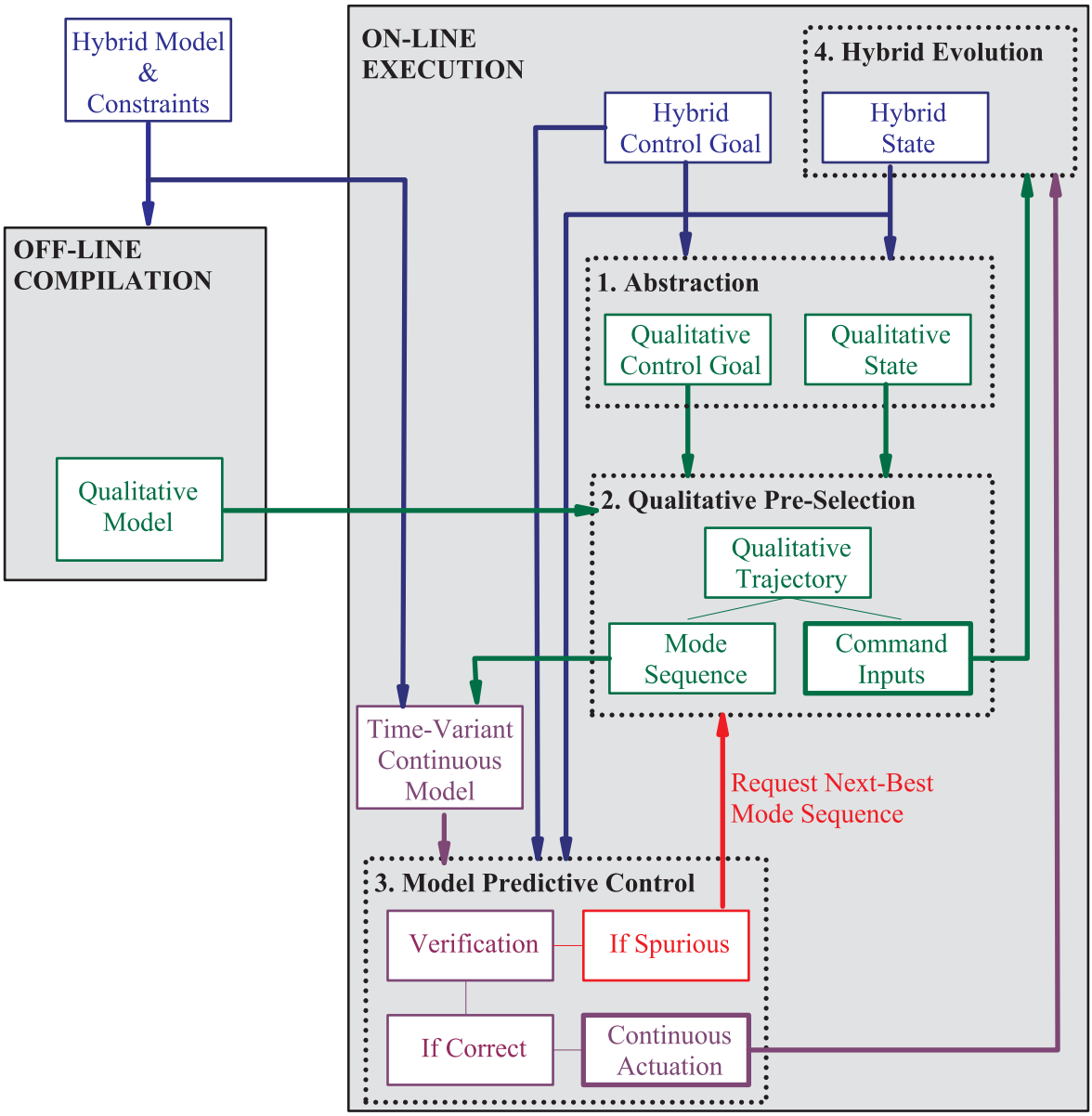


Figure 5.2: Hybrid control scheme

value to a qualitative variable corresponds to traversing along an edge in the tDAG. This way, valid qualitative trajectories, i.e. valid assignments to all qualitative variables, specify a path through each component tDAG. If a hybrid model consists of several interconnected component models, the respective tDAGs are 'connected' by shared variables, which means that nodes representing these variables occur in several tDAGs.

In Figure 5.3 two tDAGs representing such interconnected components are illustrated. Shared variables are V_1, V_3 and V_5, whereas V_2 is only constrained by the first component and V_4 is only constrained by the second. From these model graphs one could construct an overall model graph by investigating for each assignment of qualitative values to $V_1 \dots V_5$ whether it specifies a path through both graphs. For example the assignment

$$V_1 = \nu_1 \quad V_2 = \nu_2 \quad V_3 = \nu_2 \quad V_4 = \nu_1 \quad V_5 = \nu_1$$

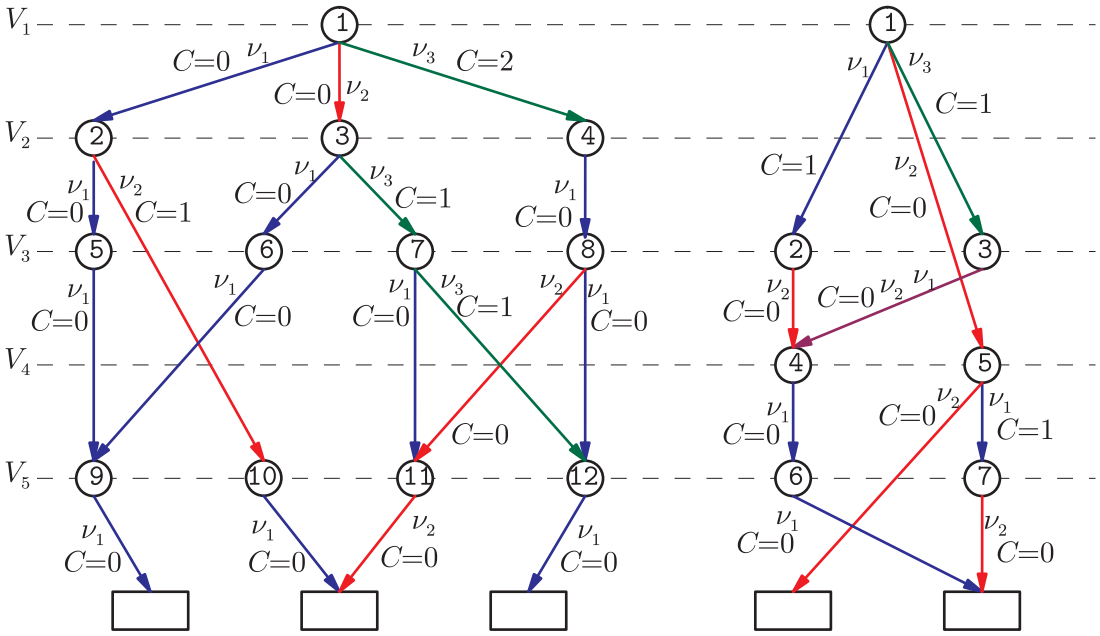


Figure 5.3: tDAGs for 2 components

specifies a path through both graphs (via nodes 1, 2 and 10 in tDAG 1 and via nods 1, 2, 4 and 6 in tDAG 2). So this assignment of qualitative values represents a valid qualitative trajectory of the overall model, whereas an assignment containing

$$V_1 = \nu_1 \qquad V_2 = \nu_1$$

is invalid, because this specifies a path to node 5 in tDAG1 (thus constraining V_3 to ν_1) and to node 2 in tDAG2 (thus constraining V_3 to ν_2). No qualitative value for V_3 can be found that allows to continue a path in both tDAGs.

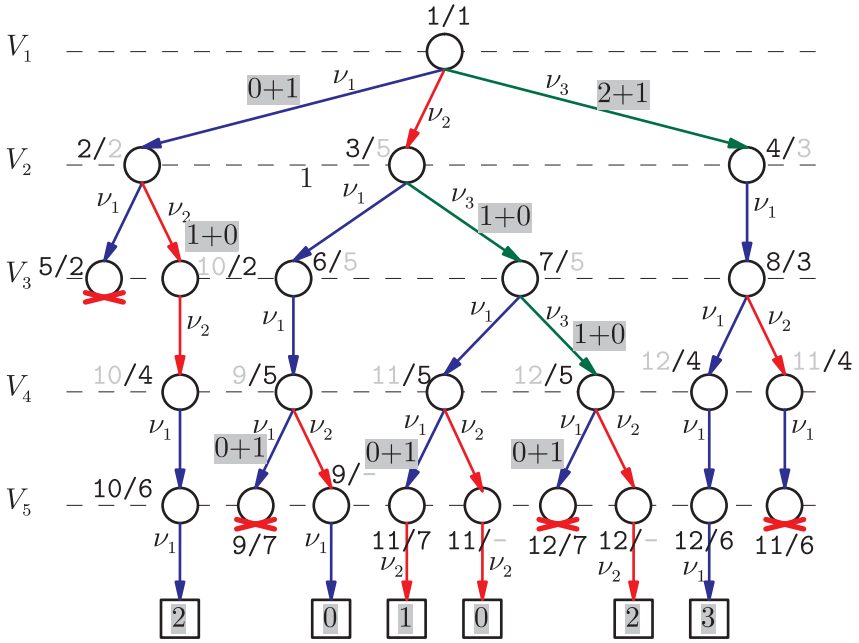
Compilation of such an overall model often is not manageable for more complex systems. Instead, we are going to investigate only a very focussed branch of it on-line by searching for qualitative trajectories according to the same ordering of variables that was used to build the component tDAGs. Each partial assignment $\overline{\mathbf{Q}}$ of variables

$$V_1 \prec V_2 \prec \ldots V_\alpha$$

specifies a path from the root node of each graph to a particular node. We call this the 'active node' $A^{(i)}$ of each of the i graphs.

If now possible assignments for the next variable $V_{\alpha+1}$ are investigated, just all the graphs have to be considered, the active nodes $A^{(i)}$ of which represents just that variable $V_{\alpha+1}$. All the other graphs do not pose any constraints on variable $V_{\alpha+1}$ in the respective branch of the graph.

All the qualitative model graphs' edges have an associated likelihood-cost value c_L. Additionally, the corresponding qualitative values determine reference cost values C_{Rc} and c_{Rd}. To focus search onto good qualitative trajectories, these cost values are used to further investigate the graphs along that edge that leads to minimal costs. Of course, a particular choice of a qualitative value is only allowed, if *all* active nodes that represent variable $V_{\alpha+1}$ are left by an edge that is labeled with that value. The target nodes of these edges, then, become the new active nodes of these graphs.



(a) Construction of the overall model graph with 'active node' labels at each node of the overall model graph. Grayed active nodes correspond to variables that come later in ordering

Figure 5.4: Simultaneous search in multiple graphs

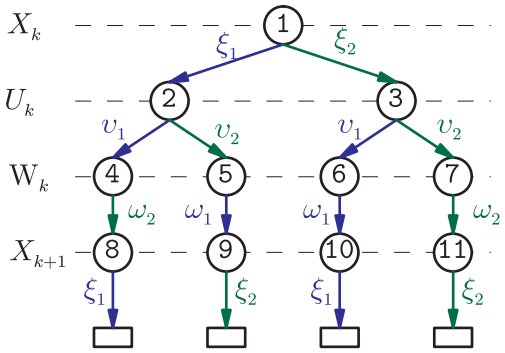
An example for the construction of such an overall model from the two separate model graphs displayed in Figure 5.3 is shown in Figure 5.4. This is the graph that has to be searched for the shortest path. We notice, however, that this overall search tree as a whole is built here for illustrative reasons only. During qualitative pre-selection, only those parts of the graph that lie along a very focussed branch towards good solutions will actually be built and investigated.

5.2.2 N-Step Receding Horizon control with Single-Time-Step Qualitative Model

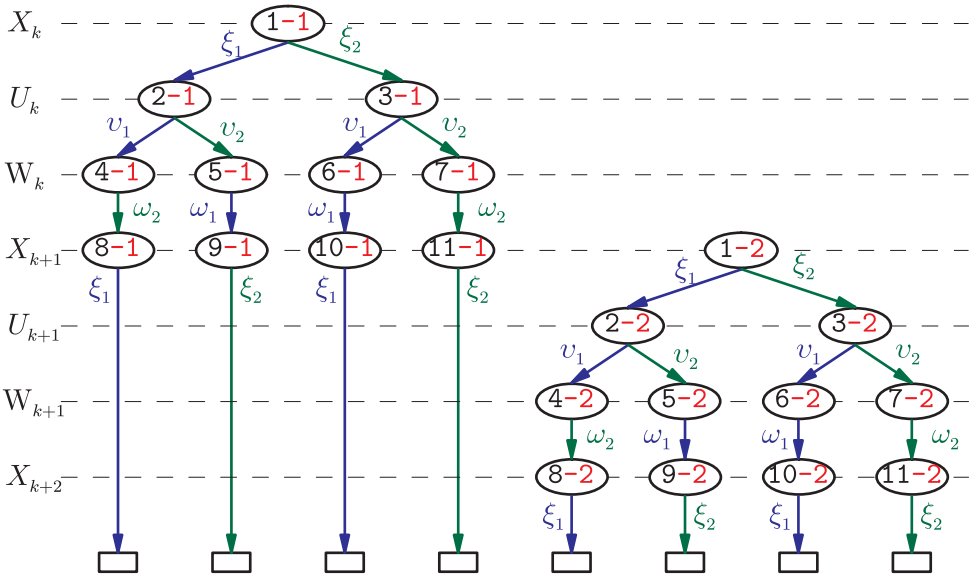
This procedure for searching multiple graphs simultaneously is not only used to evaluate qualitative variables in multiple components, but it is also needed to reason about longer time-trajectories.

The hybrid control problem is posed as an N-step receding horizon control problem and so qualitative trajectories $t_k \rightarrow t_{k+N}$ need to be evaluated. However, to keep the model size small, only qualitative trajectories for times $t_k \rightarrow t_{k+1}$ have been compiled. These models have to be utilized to reason about longer time trajectories. This is easily achieved by utilizing multiple instances of the compiled model graphs to represent qualitative trajectory segments at different times.

In Figure 5.5, two instances of a component-model graph are used to reason about trajectories $t_k \rightarrow t_{k+1} \rightarrow t_{k+2}$. Like various component models overlap with respect to the components' input and output variables, the graphs representing trajectories at different times overlap with respect to the state variables.



(a) Compiled model graph



(b) Two instances of this graph are used to reason about longer time trajectories

Figure 5.5: Multiple graph-instances for longer time trajectories

5.2.3 Qualitative Pre-Selection by A^*-search

Basis for qualitative search for a good trajectory, i.e. for the qualitative trajectory that satisfies (5.13)

$$\min_{\mathbf{Q}\in\mathcal{Q}} \left(s_L C_L + s_R \left(C_{Rc} + c_{Rd}\right)\right) \tag{5.18}$$

is an overall qualitative model graph that covers all components of the hybrid model and allows reasoning about time trajectories $t_k \rightarrow t_{k+N}$. However, this graph is not pre-compiled, but instead very focussed branches of that graph are built on-line by utilizing the pre-compiled component models.

Best First Shortest Path Search

Although no explicit representation of the overall qualitative model is available, one can intuitively treat all component tDAGs simultaneously to build the 'interesting' branches of that model graph, if a search algorithm proceeds assigning values to qualitative variables in a sequence according to the pre-specified ordering of variables that was used for the construction of the trajectory DAGs.

This way, qualitative pre-selection can be formulated as shortest path search through the overall model graph. The 'interesting' branches of this overall model graph are generated on-line. For this, the individual component models determine the constraints on allowed qualitative values as well as the likelihood costs C_L imposed by a particular qualitative value. Additionally, for the qualitative value also reference costs C_{Rc} and c_{Rd} can be determined by the qualitatively abstracted control problem formulation (e.g. bounding ball approximation for estimating the distance from a qualitative value to a reference value). These cost values determine the edge-length

$$s_L C_L + s_R \left(C_{Rc} + c_{Rd}\right)$$

of the corresponding edge in the overall model graph.

The fact that our primary interest is to find only the shortest path, i.e. the best qualitative trajectory **Q** as defined in 5.18, motivates to utilize best-first-search as a variant of shortest path search that will explore the search tree in a way, such that the first path through the graph that is found certainly is the shortest one.

This means, exploration starts at the root node and 'expands' the search graph (i.e. it investigates possible assignments to the corresponding variable and determines the associated edge-costs). Each of the newly explored nodes is assigned a 'utility value' equal to the edge-length[1] by which it is reached. Then best-first search selects the (yet unexpanded) node with lowest utility value and expands it further. The utility value of this node's child nodes is the original utility value increased by the length $s_L C_L + s_R(C_{Rc} + c_{Rd})$ of the edge that connects the two.

The algorithm proceeds always expanding the node with lowest utility value, until it detects a leaf node that has a lower utility than all other unexpanded nodes. The path from the root node to this leaf then is the shortest path from the root node to any leaf in the graph.

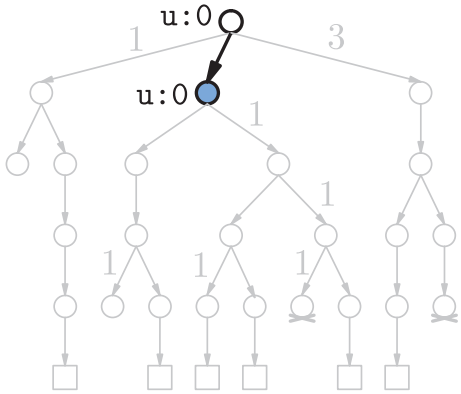
An efficient variant of this search algorithm always expands a node to its 'best' child. The subsequent children only get expanded sequentially as so-called 'follow up' expansions, if one of the (already explored) children, gets expanded itself. To illustrate this search variant, the graph depicted in Figure 5.4 is searched for the shortest path from the root to one of its leaves. Progress of search is illustrated in Figure 5.6.

Dynamic Programming

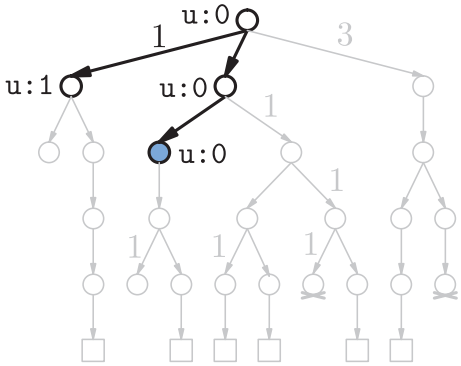
Another strategy for performing shortest-path search is provided by dynamic programming [11]. This type of search makes intensively use of Bellman's principle of optimality, which states, that the tail of an optimal solution is, itself, the optimal solution from its starting point. In terms of our qualitative pre-selection this can be expressed as: if two (partial) paths starting at the root node reach a common node in the overall model graph – and, hence, both have the same optimal tail – the longer one surely is not part of the optimal solution (Figure 5.7).

To utilize this principle in qualitative pre-selection, we have to notice that the overall model graph in fact is no 'tree' but a graph where several paths can lead to the same node. The on-line generation of the overall model graph from the individual component models as outlined so far does not illustrate this yet, but only leads to trees. To generate the overall model graph not as a tree but as connected graph, we need to utilize additional information.

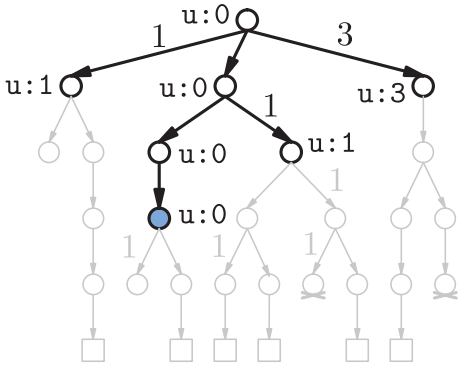
[1] That is the sum $s_L C_L + s_R(C_{Rc} + c_{Rd})$ of the edge-lengths associated to the respective qualitative value in all individual component models and the reference costs associated to the qualitative value.



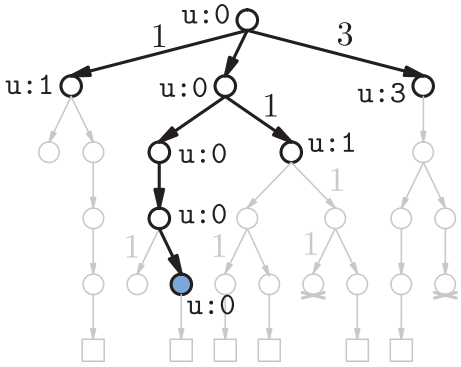
(a) Search starts at the root node and explores its best child



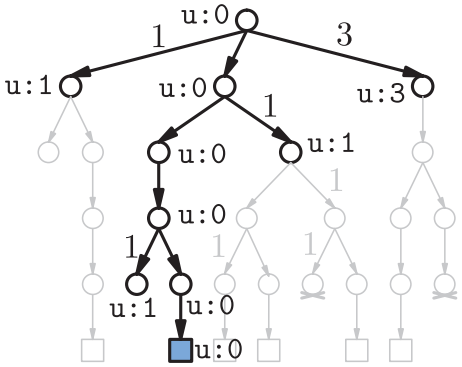
(b) The unexplored node with the best utility value is selected and its best child is explored. Additionally, another of the root nodes children is explored (follow-up expansion)



(c) The node with lowest utility value and its parents are expanded



(d) Again, only the best child is explored



(e) The node with best utility and its parent is expanded. After this expansion, the node with best utility is a leave node and so the shortest path is found

Figure 5.6: Best-first search

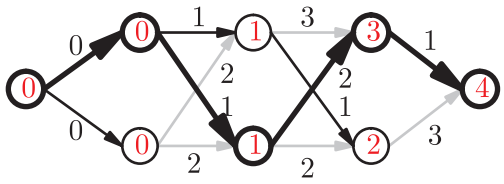


Figure 5.7: The dynamic programming idea

Here the 'structure' of the hybrid model, i.e the causal dependencies among the hybrid models equations come into play. These can be represented as causal graph (Section 4.4.1, Appendix A.3). Such a graph illustrates which variable in the model directly influences which other variables (i.e. the variable occcurs in an equation or transition specification that determines the other variable).

Based on a specified ordering of variables

$$V_1 \prec V_2 \prec \dots V_\alpha \prec V_{\alpha+1} \prec \dots,$$

by the causal graph one can evaluate which subset of variables

$$\mathbb{R}(\alpha) \subseteq \{V_1, \dots V_\alpha\},$$

together with the external inputs, completely specifies all other variables

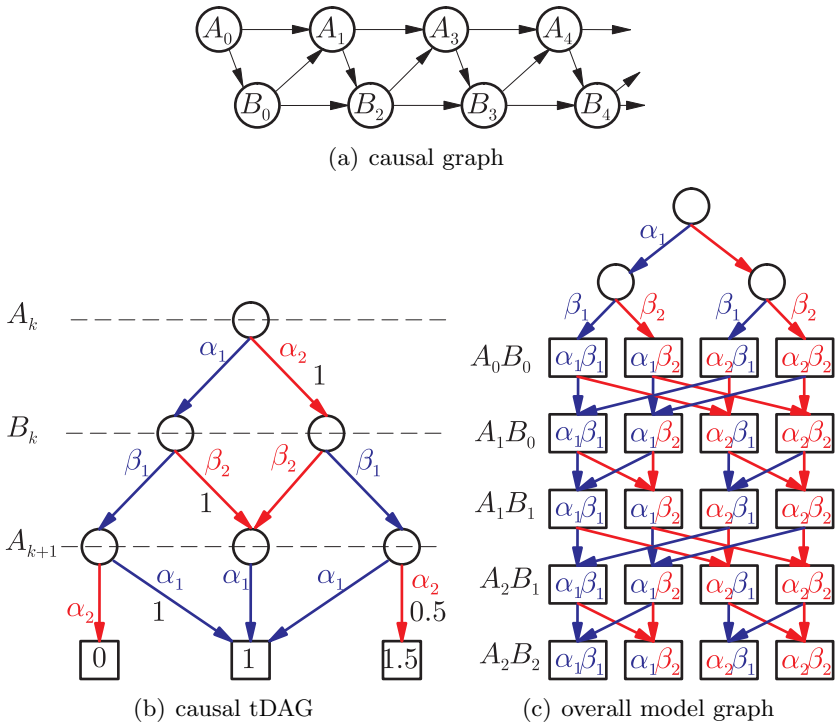
$$\overline{\mathbb{S}}(\alpha) = \{V_i \mid i > \alpha,\ V_i \text{ is not anexternal input}\}$$

(recall page 65).

The influence of all those variables from V_1 to V_α which are not included in $\mathbb{R}(\alpha)$ is already subsumed by the variables in $\mathbb{R}(\alpha)$ – just like in a standard continuous model all past states $\mathbf{x}_{k-i}, i = 1, 2, \dots$ are subsumed in the current state $\mathbf{x}_k$. This has an implication on qualitative pre-selection, because if two *different* assignments for variables $V_1, \dots V_\alpha$ *only differ* in values for those *'subsumed' variables*, they can only be extended by the *same assignments* for variables $V_i, i > \alpha$. Therefore, like in the graph 5.7 where only the shortest path to each node is investigated further, only the best assignment of variables $V_1, \dots, V_\alpha$ that contains a particular assignment to variables $\mathbb{R}(\alpha)$ needs to be investigated further.

In fact, (for each α) the qualitative assignments to variables $\mathbb{R}(\alpha)$ can be regarded as an 'identifier' for a particular node representing V_α in the overall model graph. If two qualitative assignments specify the same qualitative values for the variables $\mathbb{R}(\alpha)$, the corresponding paths in the overall model graph lead to the same node and we can use the principle of optimality to only investigate the one with the lower utility value further. The sets $\mathbb{R}(\alpha)$ can be determined at compile-time of the qualitative model from the causal graph of the hybrid model and the pre-specified ordering of variables.

However, it has to be noticed here that – because of possible spurious behaviors – the principle of optimality only holds with respect to the qualitative pre-selection and not with respect to the hybrid control problem. This means, we can stop to further investigate certain paths in the model as described above and we will still obtain the qualitatively best solution as shortest path through the overall model graph. However, this may be a spurious solution, while the paths that weren't investigated further could be non-spurious. A more detailed discussion on this topic and on the implications for qualitative pre-selection follows in section 5.2.4.



(a) causal graph

(b) causal tDAG

(c) overall model graph

Figure 5.8: Causal graph and ordering of variables determine search graph's connections

The presentation here will continue by providing an illustrative example for a connected overall model graph in Figure 5.8. There, Figure 5.8b shows a tDAG and Figure 5.8a shows the model's causal graph. Based on this, the resulting overall model graph is displayed in Figure 5.8c. By the causal graph and the ordering of variables $A_0 \prec B_0 \prec A_1 \prec \ldots$ we determine that once A_0, B_0 and A_1 are known, the influence of A_0 on the yet unspecified variables is subsumed by B_0 and A_1. So both paths that are given by qualitative values

$$A_0 = \alpha_1 \quad B_0 = \beta_1 \; A_1 = \alpha_1$$
$$A_0 = \alpha_2 \quad B_0 = \beta_1 \; A_1 = \alpha_1$$

lead to the same node in the overall model graph.

A^*-search

A search strategy that does both, searching a graph in best-first manner, while utilizing dynamic programming principles to compare partial paths through the graph is so-called A^*-search [27]. Usually, A^*-search additionally utilizes an 'admissible heuristic', that is a value which is added to the utility value of investigated nodes and is an 'optimistic estimate' (i.e. it always *underestimates*) of the path length from the node to a leaf of the graph. However, we haven't found a way how we can determine such an admissible heuristic from our qualitative component based models efficiently, so we always set this heuristic value to 0.

A^*-search is the ideal search strategy for qualitative pre-selection, because

- pre-selection only is interested in the best solution
- the search problem is posed in a well structured way that leads to a very 'connected'[2]

[2]Many nodes in the graph can be reached by multiple paths.

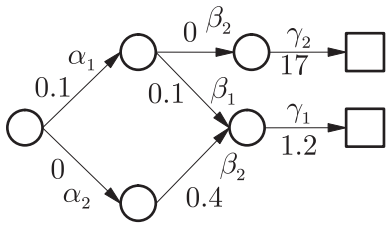


Figure 5.9: Spurious solution prevents non-spurious solution from being investigated

overall model graph. This allows intensive utilization of the principle of optimality to compare partial qualitative trajectories.

Summarizing, the qualitative control problem

$$\min_{\mathbf{Q}\in\mathcal{Q}} \left(s_L C_L + s_R \left(C_{Rc} + c_{Rd}\right)\right)$$

is posed as shortest path search in a search graph that is highly connected (i.e. there are lot of nodes shared by several paths through the graph so that intensive use of the principle of optimality can be made. The overall model graph's edges represent likelihood costs which are easily evaluated from the pre-compiled component models and approximative cost-values for deviations from reference values. These are evaluated at system run-time by easy-to-calculate 'optimistic' approximations.

The shortest path through the graph specifies a qualitative trajectory $\mathbf{Q}$, which itself represents values for the discretely values variables and qualitative abstractions of the continuously valued variables specified by polytopical regions in the value-space of the variables. Of this qualitative trajectory, in spirit of receding horizon control, the discrete command inputs calculated for time t_k are applied to the system (if subsequent numerical optimization validates $\mathbf{Q}$ to be non-spurious) and the mode sequence is utilized to simplify the subsequent numerical refinement of all other (continuous) variables.

5.2.4 Trajectory-Comparison and Spurious Behaviors

What is still left to discuss, is the influence of utilization of the principle of optimality in qualitative search on the hybrid control problem, if spurious behaviors are encountered. The problem with this is, that a qualitatively better, but spurious behavior can stop a qualitatively slightly worse, but non-spurious behavior from being investigated further. In worst case, this way all valid solutions could be missed because they were compared to qualitatively better spurious qualitative trajectories.

To illustrate this, the example search graph depicted in Figure 5.9 is utilized under the assumption that the qualitative trajectory

- $\alpha_1, \beta_1, \gamma_1$
  - is spurious
  - allows no solution of the hybrid control problem
- $\alpha_2, \beta_2, \gamma_1$
  - is a valid trajectory

  - leads to a cost function-value of 2 for the hybrid control problem

- $\alpha_1, \beta_2, \gamma_2$
  - is a valid trajectory
  - but leads to a very high cost function-value of 20 for the hybrid control problem

If standard best-first search is utilized, trajectory $\alpha_1, \beta_1, \gamma_1$ is found first, rejected and $\alpha_2, \beta_2, \gamma_1$ is found next providing a good solution to the hybrid control problem.

If, however, dynamic programming is used, of course also $\alpha_1, \beta_1, \gamma_1$ is found first, and rejected. However, $\alpha_2, \beta_2, \gamma_1$ is compared to the former trajectory after variable B and is dropped from further investigation. So $\alpha_1, \beta_2, \gamma_2$ is determined as next-best solution, providing a very bad solution to the hybrid control problem.

Preventing dynamic programming from being used would solve the problem, but significantly spoil performance of qualitative pre-selection. In fact, most of our effort on building the qualitative model would be spoiled.

So we take a closer look at the interplay between qualitative pre-selection and numerical control refinement: The idea was that trajectories were compared and eliminated from further investigation, because it was certain that they would only allow the same 'tails'. If one could now guarantee, that two partial qualitative trajectories would not only show none but the same tails in the *qualitative search graph*, but additionally could only *lead to the same numerical optimization problem*, then those could still safely be compared and only the best one needed to be investigated further.

Here, the special problem set-up for the subsequent numerical refinement-step in our hybrid control scheme comes into play. Of the qualitative trajectory $\mathbf{Q}$ only the mode-sequence and the associated transition guards on the continuous variables are needed to uniquely define the numerical optimization. (The following list is an anticipation to the following section, but it is mentioned here to provide a concise overview)

- The mode sequence specifies the time-variant continuous model
  - So the quality function is uniquely determined
- The mode-transitions specify the associated constraints on input and state.
- No other constraints but the limited range for the input-, output- and state spaces as defined for the overall hybrid model are needed.
  - So all the constraints are uniquely determined as well

It is sufficient to guarantee that some partial qualitative trajectories can only lead to the same mode-transition-sequences in order to be able to compare them and to eliminate the worse ones. This can be guaranteed, if, for variable ordering

$$V_1 \prec V_2 \prec \dots V_\alpha \prec V_{\alpha+1} \prec \dots,$$

two partial trajectories

$$\{V_1, \dots V_\alpha\}$$

share the same assignments

$$\mathbb{R}(\alpha) \subseteq \{V_1, \dots V_\alpha\},$$

and additionally specify the same mode-transition-sequence.

5.3 Numerical Control

The mode sequences specified by the solution $\mathbf{Q}$ to the qualitative control problem (5.13) allow to regard the hybrid model as time-variant continuous model and to formulate a continuous control problem (5.17). This section discusses how this formulation is obtained from the hybrid model (with specified mode sequence) and solved.

5.3.1 Formulation as Constrained Quadratic Program

To formulate the control problem, each of the model's variables and the resulting cost value c_{Rc} is expressed in terms of the inputs and the initial state. The specified mode sequence allows to regard each component of the hybrid model as

$$\begin{aligned} \mathbf{y}_k^{(i)} &= \mathbf{C}_k^{(i)}\mathbf{x}_k^{(i)} + \mathbf{D}_k^{(i)}\mathbf{u}_k^{(i)} + \mathbf{f}_k^{(i)} \\ \left(\mathbf{x}_k^{(i)}\right)' &= \mathbf{R}_k^{(i)}\,\mathbf{x}_k^{(i)} + \mathbf{r}_k^{(i)} \\ \mathbf{x}_{k+1}^{(i)} &= \mathbf{A}_{k+1}^{(i)}\left(\mathbf{x}_k^{(i)}\right)' + \mathbf{B}_{k+1}^{(i)}\mathbf{u}_k^{(i)} + \mathbf{e}_{k+1}^{(i)} \end{aligned}$$

Consecutively these equations can be used, to determine all variables $\mathbf{x}_k^{(i)}$, $\left(\mathbf{x}_k^{(i)}\right)'$, $\mathbf{y}_k^{(i)}$ in terms of the components initial states $\mathbf{x}_0^{(i)}$ and the inputs $\mathbf{u}_0^{(i)} \ldots \mathbf{u}_{N-1}^{(i)}$ only.

If the inputs $\mathbf{u}_0^{(i)} \ldots \mathbf{u}_{N-1}^{(i)}$ of the individual components are further expressed by the outputs $\mathbf{y}_k^{(j)}$ of other components and the external inputs $\mathbf{u}_k$ as given by the connection-specification $\mathcal{C}$ of the concurrent automaton, one can explicitly express each variable $\mathbf{x}_k^{(i)}$, $\mathbf{y}_k^{(i)}$, $\left(\mathbf{x}_k^{(i)}\right)'$ of all components i and for all times k in terms of the initial state

$$\mathbf{x}_0 = \begin{bmatrix} \mathbf{x}_0^{(1)} \\ \vdots \\ \mathbf{x}_0^{(L)} \end{bmatrix}$$

and the vector of inputs

$$\underline{\mathbf{u}} = \begin{bmatrix} \mathbf{u}_0 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix}.$$

This is only possible straightforwardly, if there are no algebraic loops in the model. An algebraic loop is a circular dependency among variables (see Appendix A.3). This can occur, if direct feed-through ($\mathbf{D}_k^{(i)} \neq \mathbf{0}$) through several components leads to a loop in the outline of the model structure (its causal graph)[3]. Throughout the present work, however, it is assumed that no algebraic loops are present in any combination of operational modes of the components, and so one can directly express each of the models variables v_i in terms of the initial state and input in the form

$$v_i = \mathbf{m}_{vi} \begin{bmatrix} \mathbf{x}_0 \\ \underline{\mathbf{u}} \end{bmatrix} + m_{vi}.$$

[3]Such a case requires simultaneously solving a set of equations for the variables that are involved in the loop. This can be done algorithmically in for the models here (and for more general polynomial functions) by calculation of so called Groebner Bases [15, 16].

This allows to express all range-limitations and the polytopic constraints involved in the mode-transition in terms of the initial state and input as well. More specifically, for the vector $\mathbf{v}$ of all constrained variables

$$\begin{aligned} \mathbf{H}_v\mathbf{v} &\leq \mathbf{K}_v \\ \mathbf{H}_v\left[\mathbf{M1}_v\underline{\mathbf{u}} + \mathbf{M2}_v\mathbf{x}_0 + \mathbf{m}_v\right] &\leq \mathbf{K}_v \end{aligned}$$

is expressed as

$$\mathbf{H}_v\mathbf{M1}_v\underline{\mathbf{u}} \leq \mathbf{K}_v - \mathbf{H}_v\mathbf{M2}\mathbf{x}_0 - \mathbf{H}_v\mathbf{m}_v$$

or more compactly written

$$\mathbf{H}\underline{\mathbf{u}} \leq \mathbf{K}. \tag{5.19}$$

Likewise, with notation

$$\underline{\mathbf{x}}^T = \left[\mathbf{x}_1^{(1)\,T}, \ldots, \mathbf{x}_1^{(L)\,T}, \ldots, \mathbf{x}_N^{(1)\,T}, \ldots, \mathbf{x}_N^{(L)\,T}\right]$$

and reference values

$$\underline{\mathbf{r}}_x^T = \left[\mathbf{r}_{x,1}^{(1)\,T}, \ldots, \mathbf{r}_{x,1}^{(L)\,T}, \ldots, \mathbf{r}_{x,N}^{(1)\;\;T}, \ldots, \mathbf{r}_{x,N}^{(L)\;\;T}\right]$$

$$\underline{\mathbf{r}}_u^T = \left[r_{u1,0}, \ldots, r_{uM,0}, \ldots, r_{u1,N-1}, \ldots, r_{uM,N-1}\right],$$

one can write

$$\underline{\mathbf{x}} = \mathbf{A}\mathbf{x}_0 + \mathbf{B}\underline{\mathbf{u}} + \mathbf{v}. \tag{5.20}$$

and the deviations from the reference value of x

$$\begin{aligned} \underline{\mathbf{e}} &= \underline{\mathbf{x}} - \underline{\mathbf{r}}_x \\ &= \mathbf{A}\mathbf{x}_0 + \mathbf{B}\underline{\mathbf{u}} + \mathbf{v} - \underline{\mathbf{r}}_x \end{aligned}$$

and more compactly written

$$\underline{\mathbf{e}} = \mathbf{B}\underline{\mathbf{u}} - \underline{\mathbf{r}}_1$$

This can be used to express the cost value c_{Rc} (5.1) as

$$c_{Rc} = \left[\mathbf{B}\underline{\mathbf{u}} - \underline{\mathbf{r}}_1\right]^T \mathbf{Q}_x \left[\mathbf{B}\underline{\mathbf{u}} - \underline{\mathbf{r}}_1\right] + \left[\underline{\mathbf{r}}_u - \underline{\mathbf{u}}\right]^T \mathbf{Q}_u \left[\underline{\mathbf{r}}_u - \underline{\mathbf{u}}\right]$$

what is further

$$c_{Rc} = \underline{\mathbf{u}}^T\left[\mathbf{B}\mathbf{Q}_x\mathbf{B} + \mathbf{Q}_u\right]\underline{\mathbf{u}} - 2\cdot\left[\underline{\mathbf{r}}_1^T\mathbf{Q}_x\mathbf{B} + \underline{\mathbf{r}}_u^T\mathbf{Q}_u\right]\underline{\mathbf{u}} + \underline{\mathbf{r}}_1^T\left[\mathbf{Q}_x + \mathbf{Q}_u\right]\underline{\mathbf{r}}_1$$

and again more compactly

$$c_{Rc} = \underline{\mathbf{u}}^T\mathbf{Q}\underline{\mathbf{u}} + 2\mathbf{q}^T\underline{\mathbf{u}} + \mathbf{r}_0 \tag{5.21}$$

The last term is independent of the inputs $\underline{\mathbf{u}}$. So it doesn't shift the location of the optimum and one can formulate the numerical control problem in the standard form of a quadratic program

$$\min\left(\frac{1}{2}\underline{\mathbf{u}}^T\mathbf{Q}\underline{\mathbf{u}} + \mathbf{q}^T\underline{\mathbf{u}}\right) \qquad \text{w.}r.t.\ \ \mathbf{H}\underline{\mathbf{u}} \leq \mathbf{K}. \tag{5.22}$$

5.3.2 Model Predictive Control

This quadratic program is used in Model Predictive Control (MPC) [41] to determine continuous actuation for the hybrid system. MPC is an receding horizon optimal control strategy that, at each time k, uses a model to make an N-step-ahead prediction of trajectories and optimize them with respect to given reference values by determining a sequence of inputs $\mathbf{u}(k), \mathbf{u}(k|k+1), \dots \mathbf{u}(k|k+N-1)$. MPC notation $\mathbf{u}(k|k+i)$ is used to indicate that this is the prediction of $\mathbf{u}_{k+i}$ made at time t_k.

These inputs are usually determined by formulating a linear or quadratic (5.22) optimization problem, that compares the trajectories that are predicted for the unactuated system to the reference values. The deviation of the two is then tried to be optimally counteracted by future actuation.

Of the calculated inputs $\mathbf{u}(k), \mathbf{u}(k|k+1), \dots \mathbf{u}(k|k+N-1)$, however, only the first one ($\mathbf{u}(k)$) is actually applied to the plant and the controller waits for the next sample-time t_{k+1} to again make predictions for N-step ahead trajectories and to optimize them by inputs $\mathbf{u}(k+1), \mathbf{u}(k+1|k+2), \dots \mathbf{u}(k+1|k+N)$.

To achieve offset-free tracking of constant reference values for an asymptotically stable plant (i.e. the system remains at its current mode in all components and the overall model behavior is asymptotically stable, what does not necessarily require all component models to be stable), an integrative correction of deviations for model and real plant can be introduced, by 'correcting' the given reference values. With notation

$$\mathbf{x}_k^T = \left[\mathbf{x}_k^{(1)^T}, \dots, \mathbf{x}_k^{(L)^T}\right]$$

and

$$\mathbf{r}_{x,k}^T = \left[\mathbf{r}_{x,k}^{(1)^T}, \dots, \mathbf{r}_{x,k}^{(L)^T}\right]$$

the corrected reference values are

$$\mathbf{r}_{x,\cdot,corr} = \mathbf{r}_{x,\cdot} - \left(\mathbf{x}(k) - \mathbf{x}(k-1|k)\right). \tag{5.23}$$

Stability

An important issue with respect to any controller design is stability. Concerning general hybrid systems, establishing a stability theory is yet an unresolved problem. And a through discussion about stability issues for Model Predictive Control of hybrid systems goes beyond the scope of this thesis. Only two criteria that help to achieve stability under certain constraints are presented.

The first one is very simple to explain. However, its applicability is very restricted. This stability test utilizes the quality function (5.21) itself to search for continuous actuations that lead to stable trajectories by enforcing a steady decrease in its function value. This way, the quality function serves as Lyapunov function and stability is assured. However, this stability test does not provide any measures to guarantee that at the next time-step there will exist a valid solution. So this test can mainly be used for hybrid systems, that have some kind of 'fail safe' mode that can always be reached and for which a stabilizing controller is available.

The second criterion [41] is somewhat more complicated but allows a wider applicability. To motivate this approach, one first notices that the danger of possibly encountering instability comes from the fact that MPC only performs an N-step-ahead prediction and does not make any assumptions what 'danger' may lie beyond that horizon. The idea to overcome this weakness is to reformulate the finite-horizon optimization problem (5.22) as an infinite

horizon problem. In this case, Bellman's principle of optimality applies and it is assured that the value of the optimization function does not increase with time-index k.

This may seem irrealistic at first sight, but turns out to be achievable if the qualitative controller is forced to – at the end of the receding horizon – drive the hybrid system to an operational mode (i.e. a mode for each component, of course)

- that exhibits an asymptotically stable dynamic behavior of the overall model (which not necessarily demands asymptotically stable behavior for each component)
- that is governed by a linear (not an affine) overall model
- and at which the system may remain infinitely long.

That mode is characterized by the dynamics

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k.$$

Additionally, the input remains constant at

$$\mathbf{u}(k|k+i) = 0 \quad \forall\, i \geq N$$

beyond the prediction horizon. To formulate the infinite horizon optimization problem in an extension to (5.22) the term

$$c_{\text{extra}} = \sum_{i=N+1}^{\infty} \mathbf{x}(k|k+i)^T \mathbf{Q} \mathbf{x}(k|k+i)$$

is added.

To evaluate this term, the predictions of $\mathbf{x}$ beyond the receding horizon are calculated. With $\mathbf{u}(k|k+i) = 0\ \forall\, i \geq N$ one has

$$\mathbf{x}(k|k+N+i) = \mathbf{A}^i\, \mathbf{x}(k|k+N)$$

and, hence, can write

$$c_{\text{extra}} = \mathbf{x}(k|k+N)^T \left(\sum_{i=0}^{\infty} \mathbf{A}^{T\,i}\, \mathbf{Q} \mathbf{A}^i \right) \mathbf{x}(k|k+N).$$

With notation

$$\tilde{\mathbf{Q}} = \sum_{i=0}^{\infty} \mathbf{A}^{T\,i}\, \mathbf{Q} \mathbf{A}^i,$$

one can additionally write

$$\mathbf{A}^T \tilde{\mathbf{Q}} \mathbf{A} = \sum_{i=1}^{\infty} \mathbf{A}^{T\,i}\, \mathbf{Q} \mathbf{A}^i.$$

This row converges for asymptotically stable $\mathbf{A}$ and, thus the two infinite sums can be subtracted to obtain

$$\mathbf{A}^T \tilde{\mathbf{Q}} \mathbf{A} - \tilde{\mathbf{Q}} = \mathbf{Q}, .$$

This is a Lyapunov equality and it is solved for $\tilde{\mathbf{Q}}$ to replace the definition of c_{extra}. It further has to be noticed that if $\mathbf{Q} > 0$ and $\mathbf{A}$ is stable then $\tilde{\mathbf{Q}} > 0$ as well.

This way, finally the *infinite horizon* control problem that ensures stability can be reformulated as a finite-receding horizon control problem extended by a weighting of the 'final state', i.e.

$$c_{\text{extra}} = \mathbf{x}(k|k+N)^T\, \tilde{\mathbf{Q}}\, \mathbf{x}(k|k+N).$$

5.4 Solver Interplay

This last section now discusses how a careful interplay between the two solvers – the qualitative search engine to solve the qualitative control problem and pre-select a sequence of operational mode and the numerical optimization that refines continuous actuation – ensures that a good solution to the original hybrid control problem can be found efficiently.

5.4.1 Spurious behaviors

A spurious behavior is defined to be a qualitative trajectory, that is not an abstraction of any trajectory of the hybrid system from its given initial value. To evaluate whether a pre-selected trajectory is spurious, the constrained quadratic program for the resulting time variant continuous model is specified such that each qualitative value of the pre-selected trajectory is represented as polytope-constraint on the corresponding continuously valued variable. Like shown above, utilizing the continuous model imposed by the qualitative trajectory, these polytopic constraints can be represented as polytopic constraints on the system-inputs (5.19)

$$\mathbf{H}_{\text{strict}}\underline{\mathbf{u}} \leq \mathbf{k}_{\text{strict}}.$$

If these inequalities have an empty solution, the qualitative trajectory is spurious and the numerical solver would request the next-best qualitative trajectory from the qualitative search engine. However, qualitative pre-selection merely is interested in a good solution of the original hybrid control problem. If such one could be obtained even based on a spurious behaviors, one will be satisfied with that as well.

If only the mode sequence is extracted from a qualitative trajectory to specify which continuous model is used and which polytopic guards on the continuous variables apply because of the required mode-transitions – but no additional constraints are used to force continuously valued variables to the regions specified my the respective qualitative values – a more general continuous control problem can be posed, involving polytopic constraints

$$\mathbf{H}_{\text{relaxed}}\ \underline{\mathbf{u}} \leq \mathbf{K}_{\text{relaxed}}.$$

that *only* represent constraints on continuous variables imposed by the definition of the hybrid model itself (e.g. limited actuation) and the *necessary* polytopic constraints on the required mode transitions.

With this formulation, the posed continuous control problem can be regarded to be specified for a whole set of qualitative trajectories. If this formulation does not allow variables $\underline{\mathbf{u}}$ so that all constraints are fulfilled, all qualitative trajectories that specify a common sequence of operational modes and mode transitions can together be ruled out.

So, in fact, two advantages are gained:

- The *same* optimization is performed over several qualitative trajectories without the need of any additional work
- A failed optimization does not only rule out a single qualitative trajectory, but a *whole set of qualitative trajectories* at once, what can largely help to quickly focus search onto valid solutions.

Illustration on this is provided in Figure 5.10, where control for the model

$$\begin{array}{lcll} x_{k+1} & = & x_k + u_k + 1.2 & \forall \quad k \leq 2 \\ x_{k+1} & = & x_k + u_k & \forall \quad 3 \leq k \leq 4 \end{array}$$

is calculated and actuation is limited to $-1 \leq u_k \leq 1$, the state is constrained to $-1 \leq x_k \leq 2$ and the mode-change at t_3 requires $1 \leq x_3 \leq 2$. The reference value for x is zero for all times.

Figure 5.10a shows the situation of a spurious behavior. Here, no valid solution can be found if the polytopical constraints for all qualitative values are used. Figure 5.10b shows the situation, where an alternative qualitative trajectory is provided that, however, specifies the same mode sequence. Here, a valid solution can be found, but this is not the optimal one as indicated by the dotted line. Figure 5.10c finally shows the situation, where only the necessary range- and transition constraints are used. The optimal trajectory that is possible with the specified mode sequence is found. All three optimizations use the same quality function and all constraints used in problem setup c). The former two only *add additional constraints*, what doesn't seem useful because this cannot lead better results than problem setup c).

This procedure of 'covering' all qualitative trajectories that share a common mode sequence at once in a single optimization problems has important implications on the qualitative pre-selection procedure as well, as it allows to compare partial qualitative trajectories if they lead to the same node in the overall qualitative model and specify the same mode-transition-sequence. The details about this were already pointed out in Section 5.2.4.

5.4.2 Enforcing a Certain Stability Criterion

If the first of the two stability criteria presented on page 95 is utilized – i.e. a steady decrease in the quality function of (5.21) is required – one can use solver interplay to qualitatively pre-select valid trajectories more efficiently.

In qualitative search, three different cost values are utilized

- Cost value C_{Rc}, that approximately evaluates deviations of the qualitatively abstracted continuous variables from given reference values
- Cost value C_{Rd}, that evaluates deviations from specified 'preferred operational modes'
- Cost value C_L, that gives an indication whether a qualitative trajectory is likely to be spurious.

Only the first of these values (C_{Rc}) is an abstraction of the value (c_{Rc}) of the quality function (5.21). The cost values are defined such that the one used in the qualitative model (C_{Rc}) always underestimates the one used for continuous optimization ($C_{Rc} \leq c_{Rc}$).

All three cost values, however, are used to *guide* qualitative search. So it won't always pre-select the best trajectory with respect to meeting the stability criterion.

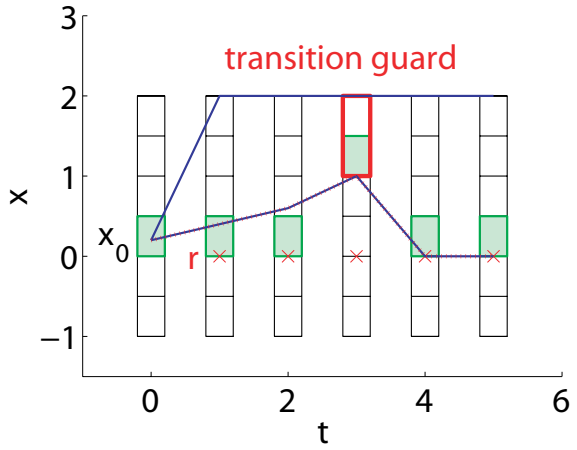
Completely dropping the other two cost values, however, is not possible with respect to C_{Rd} because this directly represents the cost value c_{Rd} used in the definition of the hybrid control problem and dropping it would change this definition. And dropping the likelihood costs can be disadvantageous because spurious behaviors would be encountered frequently.

So, still all three values are used to guide search. However, the three different values are count-up separately during search. This way, at time k, one can can stop investigating a partial qualitative trajectory further, if the corresponding cost value $C_{Rc,k}$ already exceeds last time's cost value of the numerical optimization

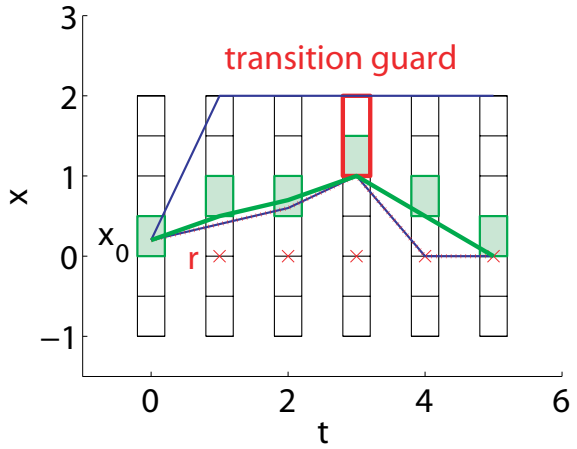
$$C_{Rc,k} \; > \; c_{Rc,k-1} \tag{5.24}$$

because then surely

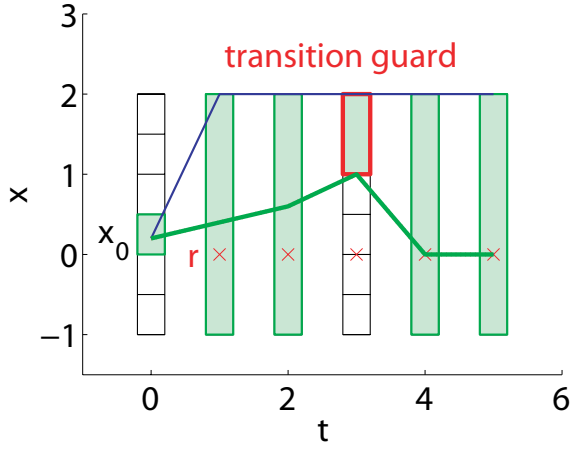
$$c_{Rc,k} \geq C_{Rc,k} > c_{Rc,k-1}.$$



(a) Qualitative pre-selection provides a spurious behavior and no solution to the numerical control problem is found



(b) Qualitative pre-selection provides another behavior that allows a – yet suboptimal – numerical solution



(c) Qualitative pre-selection provides the spurious behavior of figure a), but the numerical problem setup utilizes only the necessary constraints and so the optimal solution is found

Figure 5.10: Handling of spurious behaviors by numerical optimization

So in fact, one tries to solve the modified qualitative control problem

$$\min_{\mathbf{Q}\in\mathcal{Q},\underline{C_{Rc}<c_{Rc,k-1}}} s_L C_L + s_R \left(C_{Rc} + c_{Rd}\right) \tag{5.25}$$

5.4.3 Enforcing the Global Optimum of the Hybrid Control Task

The hybrid control scheme is designed to quickly provide suboptimal solutions to the hybrid control problem (5.6). It can, of course, be utilized to determine the optimal solution as well. The performance with respect to computation time, however, will usually degrade significantly. To determine the optimal solution, search starts with qualitative pre-selection as usual and numerical refinement of the continuous inputs is performed. This specifies the *hybrid* trajectory for $t_k \rightarrow t_{k+1}$, and the function value $c_{Rc} + c_{Rd}$ of the hybrid problem definition can be evaluated.

However, one can only be sure that this is the optimal solution with respect to a given mode-sequence, but one cannot be sure that one has pre-selected the optimal mode sequence. So further mode-sequences need to be investigated and qualitative pre-selection of alternative trajectories that lead to new mode sequences is resumed. As termination criterion for this procedure, similar to the above case of stability enforcement, a comparison of qualitative cost values $C_{Rc} + C_{Rd}$ and the results $c_{Rc} + c_{Rd}$ of already specified hybrid trajectories is used.

As cost values for the qualitative control problem are defined, such that

$$\begin{aligned} C_{Rc} &\leq c_{Rc} \\ C_{Rd} &\leq c_{Rd}, \end{aligned}$$

it is assured that the optimal solution to the hybrid control problem is found as soon as no more qualitative trajectories can be found that provide alternative mode-sequences to those already investigates and that satisfies

$$C_{Rc} + C_{Rd} \leq c^*_{Rc} + c^*_{Rd},$$

where c^*_{Rc} and c^*_{Rd} are the values related to the up-to-here best solution to the hybrid control problem.

With the intention to determine the optimal solution, it is usually a good choice to to use a weighting s_L of zero for the likelihood costs in the formulation (5.13) of the qualitative control problem. This allows to utilize the up-to-here best hybrid trajectory cost value $c_{Rc} + c_{Rd}$ as a stop-criterion to investigate partial qualitative trajectories further, as it ensures – by the utilized best-first search strategy – that the optimal solution is found as soon as a node in the search tree becomes active that has

$$C_{Rc} + C_{Rd} > c^*_{Rc} + c^*_{Rd}.$$

This search for the optimal solution of the hybrid control problem, however, usually requires computation of multiple numerical optimizations and, therefore, is computationally intensive.

5.5 Summary

Hybrid control is a difficult task for complex hybrid systems. This chapter presented a control scheme that significantly simplifies this task by breaking it up into two separate tasks, one

only operating in the discrete domain and the other purely remaining in the continuous domain.

The benefit of this procedure is, that the high complexity of the hybrid control task is first regarded on an abstracted (qualitative) level. The solution to the hybrid control task, regarding this qualitative view, is then used to specify the discrete part of the hybrid control problem and only a standard continuous optimization problem remains open for the further refinement of continuous actuation.

However, efficiency of the overall control scheme requires a very efficient qualitative reasoning among a huge (discrete) number of possible qualitative trajectories. Therefore, most emphasis is put on posing a problem formulation that is especially well suited to be solved by an efficient best-first search strategy, A^*-search. The qualitative model compiled in Chapter 4 allows very efficient operation of the A^* search strategy, because the well structured graphical representation of the qualitative model allows to build a search graph, that is not a tree-like structure where each branch has to be evaluated from root to leaf in order be compared to other branches, but partial qualitative trajectories can frequently be compared to one another and search only needs to stick to the best ones.

To allow these comparisons on behalf of the qualitative model only, it would be necessary that the we could rule out to encounter spurious behaviors. Although this *not* the case for our qualitative models it is possible to make similar statements because of the aid of subsequent numerical refinement that is utilized in the hybrid control scheme.

This numerical refinement of continuous actuation for a time-variant continuous model can be formulated as constrained quadratic program. It serves two purposes: First, it validates qualitative trajectories to be non-spurious, and second, it provides optimization for a whole set of qualitative trajectories with respect to continuous actuation at once. Especially this latter fact is the one, that allows to compare partial qualitative trajectories during qualitative pre-selection, if it can be assured that they would lead to the same MPC problem setup.

Although the two solvers (qualitative shortest path search and quadratic programming) do not need to operate simultaneously (what is the main advantage of the proposed control scheme), but operate sequentially, a careful interplay between the two is utilized to handle spurious behaviors as well as to focus search onto trajectories that meet certain stability requirements.

Even as the presented method mainly is designed to quickly provide good approximative solutions to the optimal hybrid control problem, interplay between the two solvers can be utilized to focus onto the optimal solution as well.

Chapter 6

Examples

In this chapter, all steps compilation of a qualitative model and qualitative hybrid control are summarized by an example. Then, simulation-results for compilation and qualitative hybrid control of several example systems are presented. The intention of this chapter mainly is to demonstrate applicability of the presented method to control of various different systems and to discuss difficulties and possible future improvements of the method. The MATLAB-toolbox utilized for obtaining these results only uses uncompiled code and no measures were undertaken to optimize the code with respect to computational performance. So performance of the hybrid control scheme with respect to computation time will not be discussed in much detail here.

6.1 Step-by-Step Example

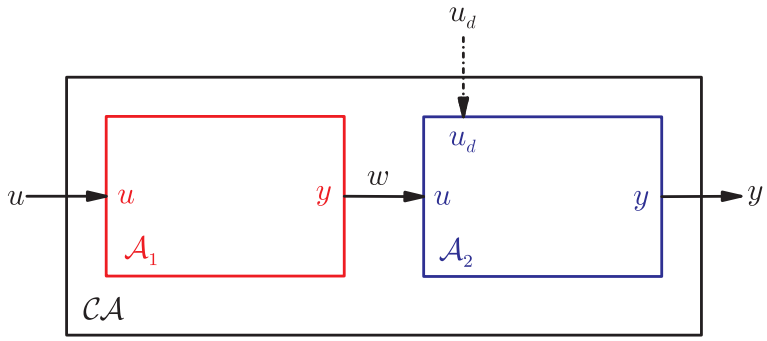
As first example to summarize qualitative modeling (utilizing the algorithms provided in the appendix) and qualitative hybrid control, the two component system depicted in Figure 6.1 is studied. Due to the simplicity of the example, many results such as the trajectory DAGs can easily be displayed.

Continuous dynamics for the components are captured by

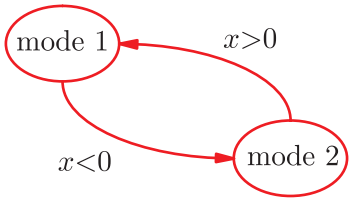
$$\begin{array}{lllrcl}
\text{component 1:} & \text{mode 1:} & x_{k+1} & = & -0.8 \cdot x_k - 0.2 \cdot u_k \\
 & & y_k & = & x_k \\
 & \text{mode 2:} & x_{k+1} & = & -0.9 \cdot x_k + 0.2 \cdot u_k \\
 & & y_k & = & x_k \\
\text{component 2:} & \text{mode 1:} & x_{k+1} & = & -0.1 \cdot x_k + u_k \\
 & & y_k & = & x_k \\
 & \text{mode 2 :} & x_{k+1} & = & -0.1 \cdot x_k - u_k \\
 & & y_k & = & x_k.
\end{array} \tag{6.1}$$

The following constraints apply:

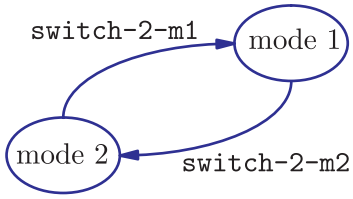
$$\begin{array}{lrcrcl}
\text{component 1:} & 0 & \leq & u & \leq & 1 \\
 & -1 & \leq & x & \leq & 1 \\
 & -1 & \leq & y & \leq & 1 \\
\text{component 2:} & -1 & \leq & u & \leq & 1 \\
 & 0 & \leq & x & \leq & 1 \\
 & 0 & \leq & y & \leq & 1.
\end{array}$$



(a) Interconnection



(b) Transition specification for component 1; Transitions do not reset the continuous state



(c) Transition specification for component 2; Transitions do not reset the continuous state

Figure 6.1: Specification of the example system

The discrete transition specifications and interconnection of the components are presented in Figure 6.1. If additionally the no-transitions are explicitly mentioned, transition guard conditions and target modes are:

component 1:	mode 1:	t_1 :	$0 \leq x \leq 1$	$\rightarrow$	m_1
		t_2 :	$-1 \leq x < 0$	$\rightarrow$	m_2
	mode 2:	t_1 :	$0 \leq x \leq 1$	$\rightarrow$	m_1
		t_2 :	$-1 \leq x < 0$	$\rightarrow$	m_2
component 2:	mode 1:	t_1 :	`switch-2-m1`	$\rightarrow$	m_1
		t_2 :	`switch-2-m2`	$\rightarrow$	m_2
	mode 2:	t_1 :	`switch-2-m1`	$\rightarrow$	m_1
		t_2 :	`switch-2-m2`	$\rightarrow$	m_2

6.1.1 Compilation

To compile the hybrid component models into a qualitative model, first qualitative variables have to be determined. Straightforwardly, we choose to abstract each continuous variable by a separate qualitative variable.

Next, the continuous domains of the variables are separated into qualitatively distinct regions. To obtain a very simple model that can easily be displayed, qualitative values for variables X_1, X_2, U, W and Y are chosen such that the just the sign of x_1, x_2, u, w and y is distinguished. That is

$$\begin{array}{llll} X_1 = \xi_1 & \longleftrightarrow & -1 \leq x^{(1)} < 0 & X_1 = \xi_2 \quad \longleftrightarrow \quad 0 \leq x^{(1)} \leq 1 \\ X_2 = \xi_1 & \longleftrightarrow & 0 \leq x^{(2)} \leq 1 & \\ U = \upsilon_1 & \longleftrightarrow & 0 \leq u \leq 1 & \\ W = \omega_1 & \longleftrightarrow & -1 \leq w < 0 & W = \omega_2 \quad \longleftrightarrow \quad 0 \leq w \leq 1 \\ Y = \mu_1 & \longleftrightarrow & 0 \leq y \leq 1 & \end{array} \tag{6.2}$$

This captures the necessary distinction of regions $x < 0$ and $x > 0$ imposed by the mode-transition specification of component 1.

Compilation of the non-deterministic Automaton

With the qualitative variables specified, the non-deterministic automaton representation of each component can be determined. The non-deterministic automaton model representing component 1 is specified by the following table. The columns of the table correspond to the qualitatively abstracted hybrid initial state, the transition guard labels (of the non-deterministic automaton) and the qualitatively abstracted hybrid target state of the transition.

$x_{1d,k}$	$X_{1,k}$	U_k	W_k	$T_{1,k}$	$x_{1d,k+1}$	$X_{1,k+1}$	Likelihood
mode 1	ξ_1	υ_1	ω_1	t_2	mode 2	ξ_2	0.19
mode 1	ξ_2	υ_1	ω_2	t_1	mode 1	ξ_1	0.20
mode 2	ξ_1	υ_1	ω_1	t_2	mode 2	ξ_2	0.19
mode 2	ξ_2	υ_1	ω_2	t_1	mode 1	ξ_1	0.20

(6.3)

The model for component 2 is

$x_{2d,k}$	$X_{2,k}$	W_k	Y_k	$u_{d,k}$	$T_{2,k}$	$x_{2d,k+1}$	$X_{2,k+1}$	Likelihood
mode 1	ξ_1	ω_1	μ_1	`switch-2-m2`	t_2	mode 2	ξ_1	0.90
mode 1	ξ_1	ω_2	μ_1	`switch-2-m1`	t_1	mode 1	ξ_1	0.90
mode 2	ξ_1	ω_1	μ_1	`switch-2-m2`	t_2	mode 2	ξ_1	0.90
mode 2	ξ_1	ω_2	μ_1	`switch-2-m1`	t_1	mode 1	ξ_1	0.90

(6.4)

Compilation of the trajectory graphs

This non-deterministic automaton model now can be represented as trajectory-DAG utilizing the algorithms in Appendix B. For this, first an ordering among the qualitative variables has to be specified. To represent the hybrid model's 'structure' in the qualitative model, this ordering among variables is derived from the hybrid model's causal graph. This can be achieved for example by utilizing the algorithm

$$\texttt{OrderedList} \leftarrow \textbf{OrderVariables}(\texttt{Graph},\texttt{Variables},\texttt{Initials},\texttt{Inputs})$$

that is given in the Appendix B. Here, `graph` is the hybrid model's causal graph as shown in Figure 6.2, but with variables replaced by their respective qualitative counterparts, and

$$\begin{aligned} \texttt{variables}: &\ [Y_k, T_{1,k}, x_{d1,k+1}, W_k, U_k, X_{1,k+1}, u_{d,k}, T_{2,k}, x_{d2,k+1}, X_{2,k+1}]\,, \\ \texttt{initials}: &\ [x_{d1,k}, X_{1,k}, x_{d2,k}, X_{2,k}]\,, \\ \texttt{inputs}: &\ [U_k, u_{d,k}] \end{aligned} \tag{6.5}$$

This way, the algorithm provides an ordering

$$Y_k \prec T_{1,k} \prec x_{d1,k+1} \prec W_k \prec U_k \prec X_{1,k+1} \prec u_{d,k} \prec T_{2,k} \prec x_{d2,k+1} \prec X_{2,k+1}. \tag{6.6}$$

that can be used consecutively for k=0,1,..., such that

$$\begin{aligned} & x_{d1,0} \prec X_{1,0} \prec x_{d2,0} \prec X_{2,0} \prec \\ & \prec Y_0 \prec T_{1,0} \prec x_{d1,1} \prec W_0 \prec U_0 \prec X_{1,1} \prec u_{d,0} \prec T_{2,0} \prec x_{d2,1} \prec X_{2,1} \prec \\ & \prec Y_1 \prec T_{1,1} \prec x_{d1,2} \prec W_1 \prec U_1 \prec X_{1,2} \prec u_{d,1} \prec T_{2,1} \prec x_{d2,2} \prec X_{2,2} \prec \dots \end{aligned}$$

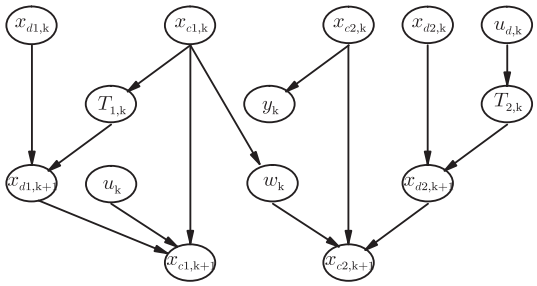


Figure 6.2: Causal graph of the example system

With the ordering of variables specified, binary decision diagram-like graphs that represent all transition specifications of each non-deterministic automaton ((6.3) and (6.4)) as paths from the root nodes of the graphs to one of their leaves can be compiled.

To generate those graphs, qualitative variables are encoded as binary-valued expressions:

$$\begin{aligned} \xi_1, \mu_1, \omega_1, \upsilon_1 &\rightarrow 0 \\ \xi_2, \omega_2, \upsilon_2 &\rightarrow 1 \\ \texttt{switch-2-m1}, \text{mode } 1 &\rightarrow 0 \\ \texttt{switch-2-m2}, \text{mode } 2 &\rightarrow 1 \end{aligned} \tag{6.7}$$

Further, the likelihood values associated to each transition of the non-deterministic automata are grouped into likelihood classes. As the likelihoods for all transitions in each component model are almost equal, only one likelihood class with value $\overline{L}$ is used. To later formulate qualitative pre-selection as shortest path search, cost values $C = -\ln(L/L_{\min})$ are used instead of the likelihoods. So $C = 0$ for all transitions.

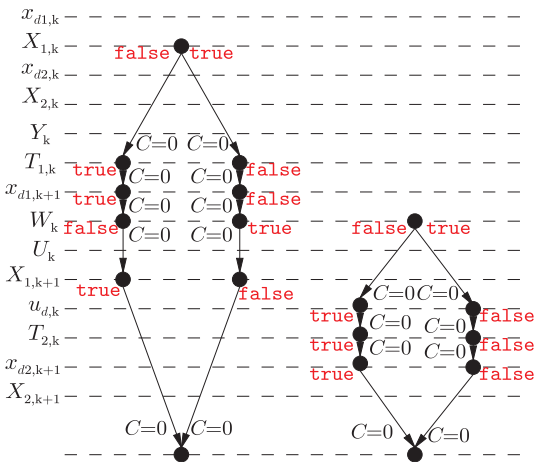


Figure 6.3: Binary graphs

The modified non-deterministic automata specifications (binary representation and likelihood classes; additionally, variables with only one qualitative value are omitted because the

qualitative model does not pose constraints on these variables) look like

$x_{1d,k}$	$X_{1,k}$	$T_{1,k}$	$x_{1d,k+1}$	W_k	$X_{1,k+1}$	C
0	0	1	1	0	1	0
0	1	0	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	1	0	0

$x_{2d,k}$	W_k	$u_{d,k}$	$T_{2,k}$	$x_{2d,k+1}$	C
0	0	1	1	1	0
0	1	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0

This table can be represented as binary trajectory graph utilizing algorithm

$$\texttt{BDAG} \leftarrow \textbf{Trans2DAG}(\texttt{BTransitions},\texttt{LikelihoodCosts},\texttt{BIndex})$$

or by first representing the table as tree and then utilizing standard BDD reduction techniques. The graphs are shown in Figure 6.3.

With binary values (6.7) again replaced by qualitative values, the final compressed trajectory-DAGs are obtained as shown in Figure 6.4.

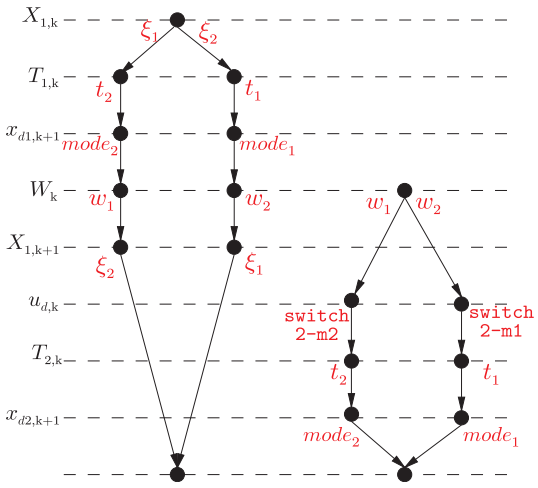


Figure 6.4: Final trajectory-DAGs (All edges have cost value 0.)

6.1.2 Hybrid Control

The control task is to let the continuous state of component 2 follow a reference trajectory. A prediction horizon of 2 steps is used and the control task is formulated as constrained quadratic optimization

$$\min \mathbf{e}^T\mathbf{e} \qquad \text{subject to} \quad \mathbf{Hu} \leq \mathbf{k}$$

where

$$\mathbf{e} = \begin{bmatrix} x_{c2,k+1} \\ x_{c2,k+2} \end{bmatrix} - \begin{bmatrix} x_{ref,k+1} \\ x_{ref,k+2} \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} u_k \\ u_{k+1} \end{bmatrix}$$

and the constraints $\mathbf{Hu} \leq \mathbf{k}$ are given by the limited range of each continuously valued variable and probably some additional constraints that need to be satisfied in order to meet the guard condition for a specific transition.

Specifically, at $k = 0$ the system is at state

$$x_{c1,0} = -0.1 \quad x_{d1,0} = \text{mode1} \qquad x_{c2,0} = 0.1 \quad x_{d2,0} = \text{mode1}$$

and trajectories shall follow the reference

$$x_{ref,i} = 0.75 \qquad i = k+1, k+2, \ldots$$

Online execution

First, the qualitative initial state has to be determined.

$$x_{c1,0} = -0.1 \rightarrow X_{1,0} = \xi_1 \qquad x_{c2,0} = 0.1 \rightarrow X_{2,0} = \xi_1$$

Next, utilizing the trajectory DAGs (Figure 6.4) consecutively for 2 time-steps, the 'best qualitative trajectory' $\mathbf{Q}$ has to be determined. This is the qualitative trajectory that leads to the smallest sum C_L of edge costs through all trajectory DAGs plus the costs C_R for deviations from the given reference values.

$$\min_{\mathbf{Q} \in \mathcal{Q}} (s_L \cdot C_L + s_R \cdot C_R)$$

For this particular example, the choice of weights s_L and s_R is irrelevant, as there is only one valid solution, anyway. This solution is shown in Figure 6.5.

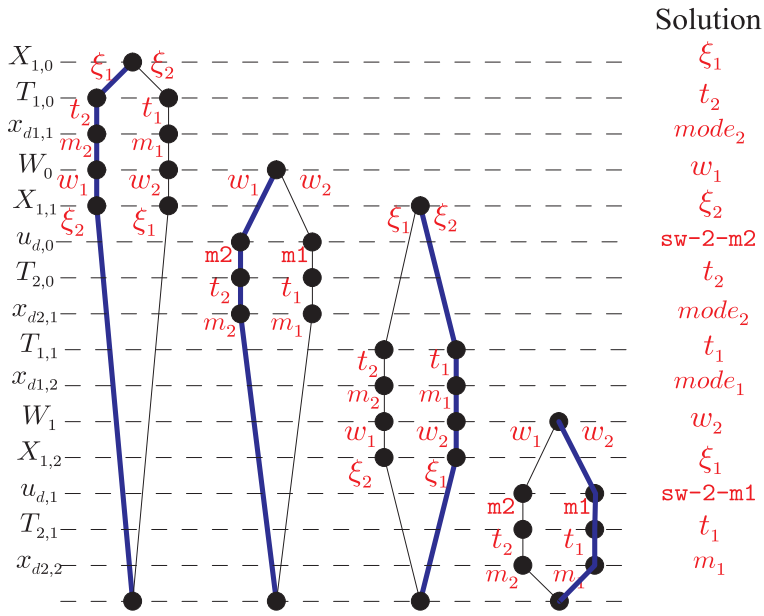


Figure 6.5: Qualitative trajectory

This so-called qualitative trajectory specifies the mode-sequence for both components, i.e.

$$\begin{aligned} x_{d1,0} &= \text{mode1}, & x_{d1,1} &= \text{mode2}, & x_{d1,2} &= \text{mode1} \\ x_{d2,0} &= \text{mode1}, & x_{d2,1} &= \text{mode2}, & x_{d2,2} &= \text{mode1} \end{aligned}$$

the necessary discrete inputs

$$u_{d,0} = \texttt{switch-2-m2} \qquad u_{d,1} = \texttt{switch-2-m1}$$

and a sequence of transitions

$$\begin{array}{ll} T_{1,0} = t_2 & T_{1,1} = t_1 \\ T_{2,0} = t_2 & T_{2,1} = t_1 \end{array}$$

From the mode sequence and the specification (6.1) of the hybrid model we have

$$\begin{array}{rcl} y_0 & = & x_{c2_0} = 0.1 \\ w_0 & = & x_{c1_0} = -0.1 \\ x_{c1_1} & = & -0.9 \cdot x_{c1_0} + 0.2 \cdot u_0 = 0.09 + 0.2 \cdot u_0 \\ x_{c2_1} & = & -0.1 \cdot x_{c2_0} - w_0 = 0.09 \\ y_1 & = & x_{c2_1} = 0.09 \\ w_1 & = & x_{c1_1} = 0.09 + 0.2 \cdot u_0 \\ x_{c1_2} & = & -0.8 \cdot x_{c1_1} - 0.2 \cdot u_0 = -0.072 - 0.16 \cdot u_0 - 0.2 \cdot u_1 \\ x_{c2_2} & = & -0.1 \cdot x_{c2_1} + w_1 = 0.081 + 0.2 \cdot u_0 \end{array}$$

and can, thus, express the deviation $\mathbf{e}$ of the state of component 2 from the reference trajectory as

$$\mathbf{e} = \begin{bmatrix} 0.09 \\ 0.081 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.2 \end{bmatrix} \cdot u_0 - \begin{bmatrix} x_{ref,k+1} \\ x_{ref,k+2} \end{bmatrix}.$$

Likewise, the constraints on the continuously valued variables – the limited range and the constraints $x_{c1,1} < 0$ and $x_{c1,2} \geq 0$ imposed by the specified transition sequence – can be expressed in terms of u_0 and u_1 as

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

So, the constrained optimization

$$\min \mathbf{e}^T\mathbf{e} \text{ subject to } \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

can be solved and one obtains

$$u_0 = 1 \qquad u_1 = ?^1.$$

Since the optimization leads to a valid solution, the qualitative trajectory is verified to be non-spurious and the determined inputs for time 0

$$u_0 = 1 \qquad u_{d,0} = \texttt{switch-2-m2}$$

[1] Any value of u_1 from 0 to 1 is valid and leads to the same value of the optimization function. The calculated value for u_1, thus, only depends on the implementation of the utilized solver.

are applied to the system.

The system then proceeds until the next sample time and hybrid control 'restarts' from the new system state.

The resulting trajectory of x_{c2} and the associated continuous input u is displayed in Figure 6.6. The discrete input is a alternating sequence of values `switch-2-m2` and `switch-2-m1`, and the modes of components 1 and 2 change at each time k.

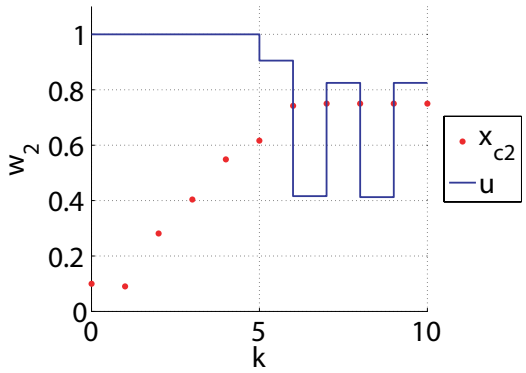


Figure 6.6: Input u and trajectory of x_{c2}

6.2 3-Tank Benchmark System

As an example to illustrate the applicability of the presented hybrid control scheme to more complex systems, the well known 'COSY' 3-Tank benchmark problem [29] is used, that has been studied for a wide range of different hybrid control strategies in literature [4, 51, 53].

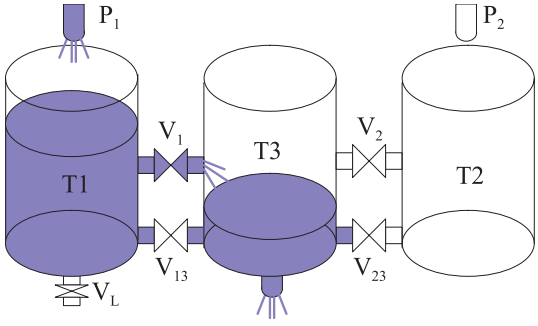


Figure 6.7: 3-tank system

This system (Figure 6.7) consists of three tanks that are connected by 4 valves that can be fully opened or fully closed by discrete control inputs. The two outer tanks can be filled by pumps that can continuously vary the flow of water q_p in the range

$$0 \leq q_p \leq 1.$$

The task is to control the water level in the middle tank. The original definition of the benchmark problem assumes that the level of this tank can only be measured qualitatively to be above or below two certain threshold values. As the presented control-scheme, however, has to assume full measurement of the systems state, this restriction has to be omitted here.

What makes control more difficult is, that valves can operate faulty, i.e. they can be stuck open or stuck closed. Additionally, there can occur a leak in tank 1.

With respect to these faults, the original benchmark problem suggests some specific test cases:

- Nominal operation: There is no fault in the system, but all valves except V_1 are kept closed. Tank two remains empty and is only used as backup for tank 1 in case of faults.
- Valve V_1 is closed and blocked
- Valve V_1 is open and blocked
- Tank 1 is leaking, i.e. valve V_L is open.

As no specific 'nominal' behavior in as distinguished with our control scheme, direct use this specification is not possible here as well. Only weights in the optimization function are used to bias the controller towards usage of tank 1.

6.2.1 Dynamical Model

The water flow through an opened valve is calculated utilizing Toricelli's law,

$$Q_{ij} = S \cdot sign(h_i - h_j) \cdot \sqrt{2g \cdot |h_i - h_j|}. \tag{6.8}$$

For all valves the flow out of tank 3 and the possible leak in tank 1, parameters are:

$$g = 9.81 \qquad S = 0.36$$

Valves V_1 and V_2 are at $h = 0.3$ above the bottom of the tank. So it has to be distinguished, whether the water levels in the neighboring tanks are above or below $h = 0.3$. If the water level in a tank is below $h = 0.3$, this tank's water height has to be set to $h = 0.3$ in Toricelli's law to calculate the flow through V_1 or V_2.

The dynamics of the tanks are modeled by

$$\frac{d}{dt}h = \frac{1}{A} \cdot \sum Q \tag{6.9}$$

with

$$A = 154.$$

6.2.2 Model Components

The 3-tank benchmark system's dynamics are different for many operational modes, i.e. the valves being open or closed, being faulty or operating correctly, tanks leaking or not, water levels above or below $h = 0.3$.

To abstract the dynamics of the overall system (3-dimensional state-space with tank water-levels being the states, 2-dimensional input space with pump flows being the inputs, many operational modes) to a single qualitative model is difficult because the effort to compile the non-deterministic automaton scales exponentially with the dimensionality of the state and input space. With the current implementation of our Qualitative Hybrid Control toolbox for Matlab, compilation of such a model was not possible because the necessary enumeration of all qualitative trajectories even exceeded the maximum matrix size allowed by Matlab.

So first an approximate model is obtained that separates the overall system into several components. Unfortunately, for this example the performance of the resulting hybrid controller significantly depends on this separation and the structure of the component models and we cannot determine in advance, how to separate the overall model in order to obtain acceptable performance. The model structure presented below has only been developed through many simulations by 'trial and error'. Moreover, the model could only be obtained utilizing some work-arounds so that, in fact, we have to confess that this example is not very well suited to be solved with the proposed qualitative modeling and hybrid control scheme. Nevertheless we include this well known example here to demonstrate our approach to hybrid control on an example that is well known in literature.

Specifically, we model the flow through each valve V_1, V_2, V_{13}, V_{23} as a separate model-component. Likewise, the three tanks T_1, T_2 and T_3 are modeled separately, as well. This is outlined in Figure 6.8.

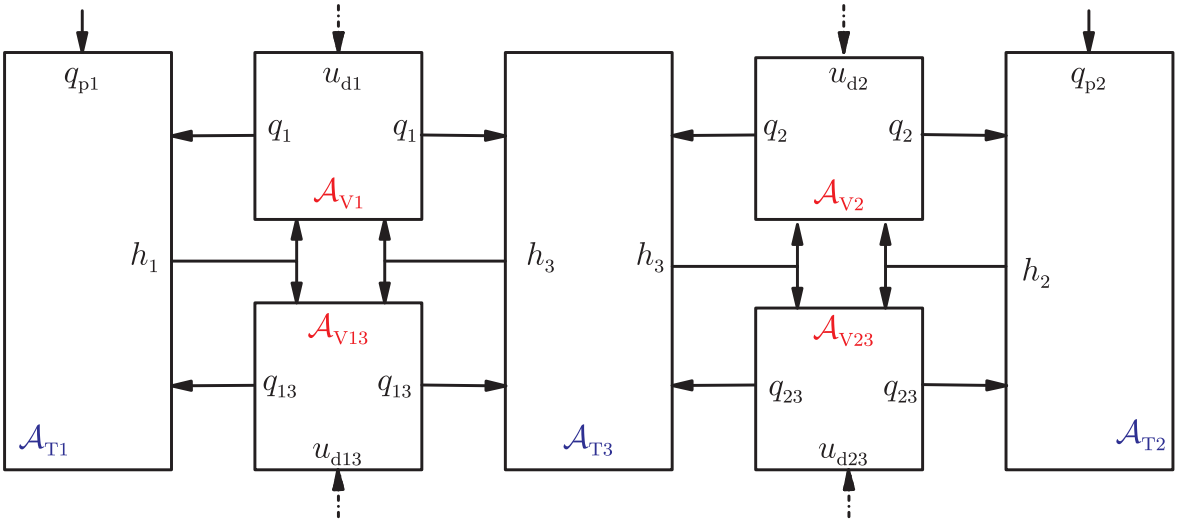


Figure 6.8: Modeling tanks and valves as separate components

Although this seems a reasonable choice, it poses some problems with respect to the hybrid control. An obvious reason for this is, that component inputs and, thus, the other component's outputs they are connected to are assumed to remain constant in between sampling times, which is not true for the real system. This assumption, however, is a reasonably close approximation of the real system behavior, if the sampling interval is small.

A probably less obvious reason is, that component outputs are updated at every sample time, but a mode change only occurs immediately *afterwards*. That means, modeling the valves just by algebraic equations that linearly approximate Toricelli's law in the `open` mode and that provide zero output in `closed` mode will delay the effect of inducing an opening or closing of a valve by one sample time, what degrades controller performance.

To overcome this, one can take some work-around, such that the components for the valves always provide an output representing the flow of water that would occur *if the valve was open*. The discrete command inputs are then utilized to change modes for the tank models, such that valve flows are considered or ignored, depending on the valves position.

This way, however, overall complexity of the model increases, because for example the second tank has to consider $4^4 = 256$ distinct modes, because each valve can either be ok and open, ok and closed, stuck open or stuck closed. Although *continuous dynamics* are the

same for many of these modes, the possible *mode transitions* are different.

This large number of necessary modes can be reduced by yet another work-around: For each valve, one more artificial component is introduced that does not model any continuous dynamics, but just decides on behalf of commands, the state of the valve and water levels, whether the tank model has to consider or ignore the water flow calculated by the (other) valve-component.

Again, here occurs the problem that this component cannot utilize an output to provide this information to the tank models, because otherwise again a delay of one sampling step would be introduced. Instead these artificial models have to be formulated such that the discrete command input is constrained to values that are valid within the current operating condition of the valve and current water levels. This will be detailed later but first the modified model structure is presented in Figure 6.9.

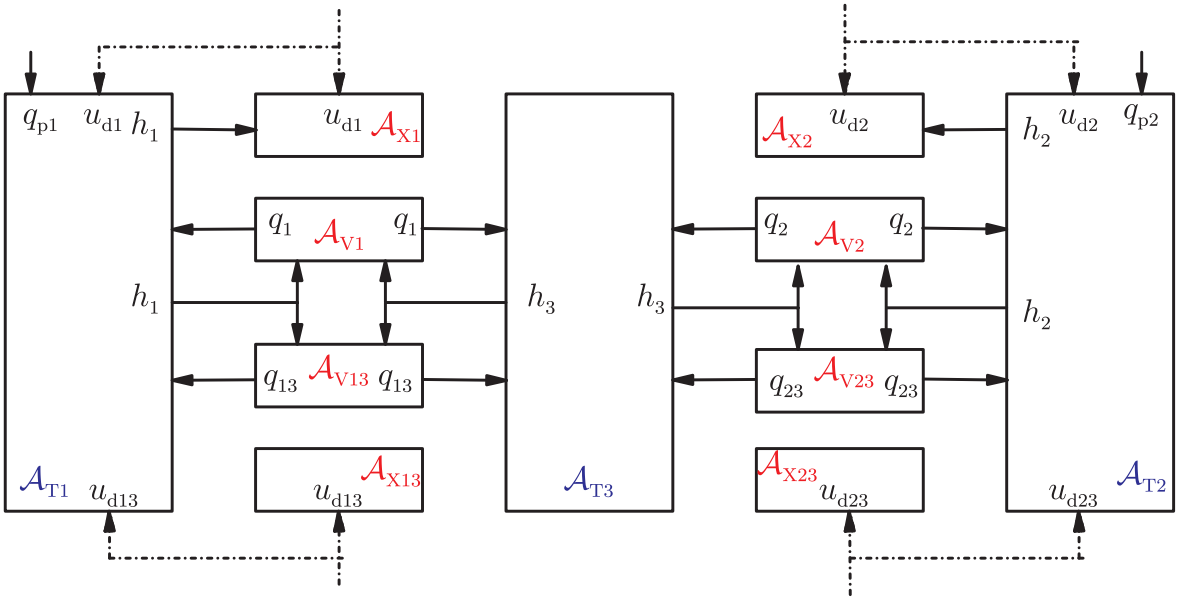


Figure 6.9: Model structure with artificial components

It can be noticed here, that the water level of tank 3 is not provided to the artificial components. To make the model simpler, it is assumed that the water level of tank 3 always is below $h_3 = 0.3$. This will let the model make significantly wrong predictions for high water levels in the middle tank. However, as the benchmark problem always specifies the target water level of tank 3 as $h_3 \leq 0.3$, this approximation should still turn out to work well.

Component models for valves

By Toricelli's Law, the flow of water through valve V_{13} is

$$Q_{13} = S \cdot sign(h_1 - h_3) \cdot \sqrt{2g \cdot |h_1 - h_3|}.$$

For our class of hybrid models, we need linear or affine approximations of this non-linear relation to model water flow through the valves. To keep the model simple, we only want to use one single linearization to approximate Toricelli's Law for the whole range of water levels $0 \leq h_1 \leq 0.6$, $0 \leq h_3 \leq 0.3$.

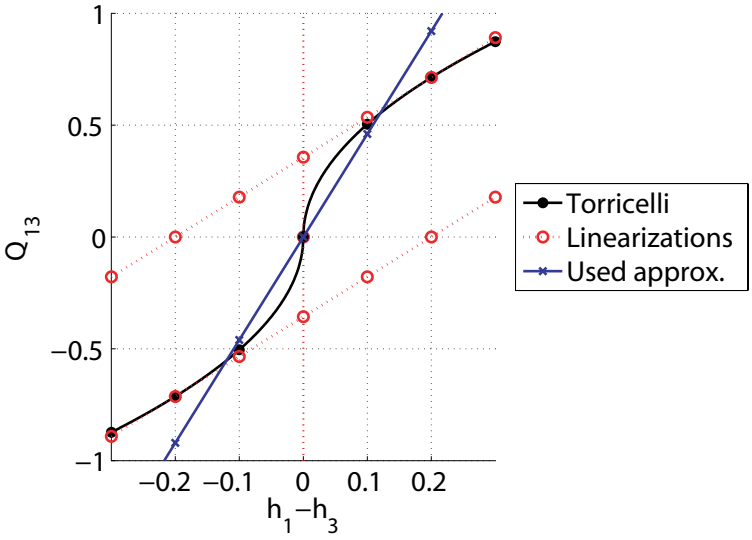


Figure 6.10: Linear model for q_{13}

Forming linearizations around an operational point does not provide sufficiently accurate approximations here, as is illustrated in Figure 6.10, so we instead model water flow through the valve linearly dependent on the difference in water height such that cases of equal water level and water levels that are different by $h_1 - h_3 = 0.12$ are captured exactly. With $g = 9.81$ and $S = 0.36$ this linear approximation is

$$Q_{13} = 4.6 \cdot (h_1 - h_3), \tag{6.10}$$

what is, again, illustrated in Figure 6.10.

This model for valve V_{13} has to be used in a sampled control system. So, with the present class of hybrid models, this would assume $Q_{13,k}$ to remain constant during sampling interval $t_k \rightarrow t_{k+1}$. If the sampling interval is sufficiently small, this is a good approximation. However, the influence of the input $q_{1,k}$ provided by pump P_1 on the water level $h_{3,k+1}$ in tank 3 is *not* captured. During the design process of this control system, degraded controller performance was observed whenever utilizing models that do not capture this influence.

So to evaluate the effect of input $q_{1,k}$ on the flow of water through valve V_{13} during the sampling interval $t_k \rightarrow t_{k+1}$, a simplified system of tanks 1 and 3 connected by valve V_{13} is utilized. By the above linearization and (6.9), this system can be modeled

$$\frac{d}{dt}\begin{bmatrix} h_1 \\ h_3 \end{bmatrix} = \frac{1}{A} \cdot \begin{bmatrix} -4.6 & 4.6 \\ 4.6 & -4.6 \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_3 \end{bmatrix} + \frac{1}{A} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} q_{p1}, \tag{6.11}$$

and the sampled model with sampling time $T_s = 10$ looks like

$$\begin{bmatrix} h_1 \\ h_3 \end{bmatrix}_{k+1} = \begin{bmatrix} 0.775 & 0.225 \\ 0.225 & 0.775 \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_3 \end{bmatrix}_k + \begin{bmatrix} 0.057 \\ 0.008 \end{bmatrix} q_{p1,k}. \tag{6.12}$$

With

$$q_{13,k} = \frac{A}{T_s}\left(h_{3,k+1} - h_{3,k}\right)$$

valve V_{13} is modeled as

$$Q_{13,k} = 3.46 \cdot h_{1,k} - 3.46 \cdot h_{3,k} + 0.12 \cdot q_{p1,k}. \tag{6.13}$$

Similarly,

$$Q_{23,k} = 3.46 \cdot h_{2,k} - 3.46 \cdot h_{3,k} + 0.12 \cdot q_{p2,k}. \tag{6.14}$$

The upper valves are modeled likewise, with the only exception that h_3 is set to $h_3 = 0.3$, so that

$$Q_{1,k} = 3.98 \cdot h_{1,k} + 0.14 \cdot q_{p1,k} - 1.19 \tag{6.15}$$

$$Q_{2,k} = 3.98 \cdot h_{2,k} + 0.14 \cdot q_{p2,k} - 1.19 \tag{6.16}$$

Artificial valve models

The above valve-models only exhibit one operational mode. So these models do not consider whether there really is water flowing through the valves or not. For reasons outlined above, this is left to other parts of the overall model, i.e. the tank models. Not to have to consider too many different modes in these tank models, the here presented artificial valve models are utilized. These models constrain the choice of command inputs such that an `open` command can only be issued if the valve is not stuck closed (and additionally for valves V_1 and V_2 if the water levels of tank 1 or, respectively, tank 2 is above 0.3). A `close` command on the other hand only is only possible if the respective valve is not stuck open. This way, after an allowed `open` command, the corresponding valve is certainly carrying flow, while after an allowed `close` command, the valve is certainly carrying no flow.

So, the commands can directly be used in the component-models representing the tanks to evaluate whether to consider or not the flow calculated by one of the component models representing the valves.

For valve V_1, allowed commands are:

failure mode	h_1	allowed commands u_{d1}
`normal operation`	≥ 0.3	`open, close`
`normal operation`	< 0.3	`close`
`stuck open`	≥ 0.3	`open`
`stuck open`	< 0.3	`close`
`stuck closed`	≥ 0.3	`close`
`stuck closed`	< 0.3	`close`

With this table it is once again pointed out that – to keep models simple – the water level h_3 of the middle tank is assumed to be $h_3 \leq 0.3$. For valve V_2, the table is the same with h_1 replaced by h_2 and u_{d1} replaced by u_{d2}.

For valves V_{13} and V_{23}, the allowed commands are

failure mode	allowed commands u_{d13}, ud_{23}
`normal operation`	`open, close`
`stuck open`	`open`
`stuck closed`	`close`

To constrain command inputs to allowed values, the three modes of operation `normal operation`, `stuck open`, `stuck closed` plus one additional mode `invalid` are modeled. Discrete dynamics are such, that the operational mode stays the same, if an allowed command is issued and that the mode changes to be `invalid` otherwise. Continuous dynamics for the invalid mode are set such that the continuous state is set to a value out of bounds. Continuous dynamics for all other modes are set such that the value is within its allowed bounds. With this, the qualitative model later on restricts a choice of command inputs to allowed values.

This is illustrated on behalf of example of valve V_{13}, where

$$x_{k+1} = 2$$

if the mode is `invalid`,

$$x_{k+1} = 0.5$$

otherwise and bounds are set to

$$0 \leq x \leq 1,$$

where this whole region is represented by qualitative value ξ_1. If, for example, mode $x_{d,k}$ is `stuck closed` and $u_{d,k}$ is `open`, the next mode $x_{d,k+1}$ would be `invalid`. However, according to the algorithms presented in Chapter 4, no transition with this combination of values would not be recorded when compiling the non-deterministic automaton, because the probability to have a continuous state $X_{k+1} = \xi_1$ here is zero. The resulting non-deterministic automaton looks like

$x_{d,k}$	$X_{d,k}$	$u_{d,k}$	$x_{d,k+1}$	X_{d_k+1}	L
`normal operation`	ξ_1	`open`	`normal operation`	ξ_1	1
`normal operation`	ξ_1	`close`	`normal operation`	ξ_1	1
`stuck open`	ξ_1	`open`	`stuck open`	ξ_1	1
`normal closed`	ξ_1	`close`	`normal closed`	ξ_1	1

and, thus, the compressed t-DAG as shown in Figure 6.11 only allows the command input to be chosen according to the allowed values. It does not constrain the choice of $u_{d,k}$ in normal operation, but forces $u_{d,k}$ to be appropriately, if the valve is stuck.

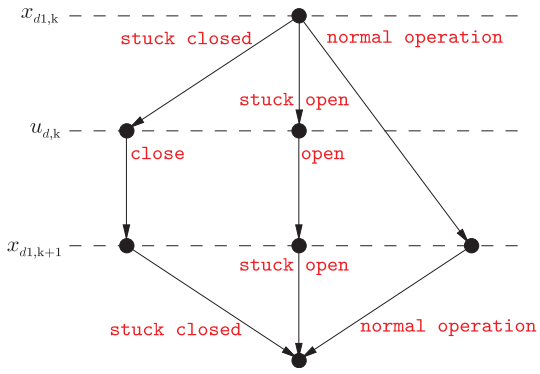


Figure 6.11: Trajectory DAG for artificial component V_{13x}

Component models for tanks

With water flows through the valves modeled as above and choices of command inputs constrained such that they indicate if a valve is truly carrying flow or not, the three tanks can be easily modeled by (6.9). With (again) sampling time $T_s = 10$ (the sampling time is required

to be the same for all components) the resulting models are:

$$h_{1,k+1} = A_{1,i} \cdot h_{1,k} + \begin{bmatrix} 0.065 & -b_{1,i} & -b_{13,i} \end{bmatrix} \begin{bmatrix} q_{p1,k} \\ Q_{1,k} \\ Q13,k \end{bmatrix} \tag{6.17}$$

$$h_{2,k+1} = 1 \cdot h_{2,k} + \begin{bmatrix} 0.065 & -b_{2,i} & -b_{23,i} \end{bmatrix} \begin{bmatrix} q_{p2,k} \\ Q_{2,k} \\ Q23,k \end{bmatrix} \tag{6.18}$$

$$h_{3,k+1} = 0.74 \cdot h_{3,k} + \begin{bmatrix} b_{1,i} & b_{2,i} & b_{13,i} & b_{23,i} \end{bmatrix} \begin{bmatrix} Q_{1,k} \\ Q_{2,k} \\ Q_{13,k} \\ Q23,k \end{bmatrix} \tag{6.19}$$

There, $A_{1,i} = 1$ if tank 1 is not leaking and $A_{1,i} = 0.74$ otherwise. Values $b_{\cdot,i} = 0.065$ if the corresponding command input is `open` and $b_{\cdot,i} = 0$ otherwise. A presentation of an enumeration of all possible combinations thereof for the three tank models is omitted.

6.2.3 Qualitative Model

The models (6.17) indicate that changes in water levels are rather small from one sample to the next. To capture this dynamic behavior, a sufficiently fine-grained separation of the value space of water levels is required. The partitioning of allowed water levels

$$0 \leq (h_1, h_2, h_3) \leq 0.6$$

is chosen to be separated into

16 equally sized intervals.

With a qualitative separation of valve flows Q_i, one mainly wants to provide an opportunity to the controller to decide which valves to open. So only a rough distinction is made here, that only distinguishes cases:

$$\begin{array}{cc} -Q_{extr} \leq q < 0 & \texttt{negative} \\ 0 \leq q < 0.4 & \texttt{small} \\ 0.4 \leq q < 0.65 & \texttt{normal} \\ 0.65 \leq q < Q_{extr} & \texttt{large}, \end{array} \tag{6.20}$$

where Q_{extr} is the calculated flow if tank 1 or 2 is full and tank 3 is empty, which is 2.77 for valves V_{13} and V_{23} and 1.58 for valves V_1 and V_2.

The input flow provided by the two pumps

$$0 \leq (q_{p1}, q_{p2}) \leq 1$$

is represented by a single qualitative value.

The resulting compressed trajectory graphs all have a size below 5000 nodes, which is a quite compact representation compared to the non-deterministic automata, the specifications of which contain up to more than 100.000 transitions. The trajectory graphs are compiled with likelihood values all combined into one single likelihood class. This significantly contributes to the compactness of the graphs. However, this way transition likelihoods cannot be used to bias qualitative search towards most likely non-spurious trajectories.

6.2.4 Control

As the hybrid control scheme does not distinguish between nominal operation and reconfiguration in case of faults, one cannot directly utilize the test cases provided with the definition of the benchmark problem.

- Nominal operation: There is no fault in the system, but all valves except V_1 are kept closed. Tank two remains empty and is only used as backup for tank 1 in case of faults.
- Valve V_1 is closed and blocked
- Valve V_1 is open and blocked
- Tank 1 is leaking, i.e. valve V_L is open.

However, the ideas behind these faults will be followed by introduction of comparable faults here. As control goal, a constant value for the water level in tank 3 is used. If this reference-value for the water level changes, this change is not known to the controller in advance.

The following test scenario is used:

- There are no faults in the system, the water levels are at $h_1 = 0.2$, $h_2 = h_3 = 0.08$. The reference-level is $h_{3,ref} = 0.13$.
- At time $t = 200$ (sample step $k = 20$), the reference level changes to $h_{3,ref} = 0.25$.
- At time $t = 400$, a fault occurs and the valve V_{13} gets stuck closed.
- At time $t = 500$, the valve becomes operational again and the reference value changes to $h_{3,ref} = 0.13$.
- At time $t = 800$, another fault occurs and valve V_{13} gets stuck open this time.
- At time $t = 1000$, the valve V_{13} becomes operational again, but tank 1 starts to leak.

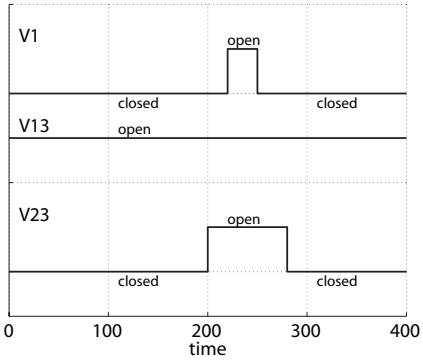
To formulate the control objective as mathematical program, we set the optimization function

$$J = 100 \cdot (h_{3,k+1} - h_{3,ref})^2 + (q_{p1,k} - 0.5)^2 + 2 \cdot q_{p2,k}^2 \tag{6.21}$$

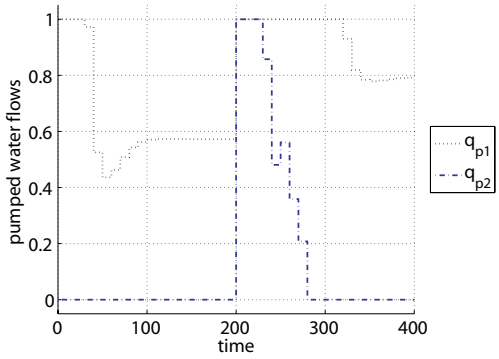
This means, a prediction horizon of just one step is used, what works well for this tank system. The different weights and reference values for the inputs (water flow through the pumps) are just included to bias the controller towards usage of tank 1, what is the nominal behavior in the original definition of the benchmark problem.

The result to simulations (the non-linear model of the 3-tank system is considered as the real system behavior, no disturbances occur) for this test scenario is displayed in Figures 6.12-6.14.

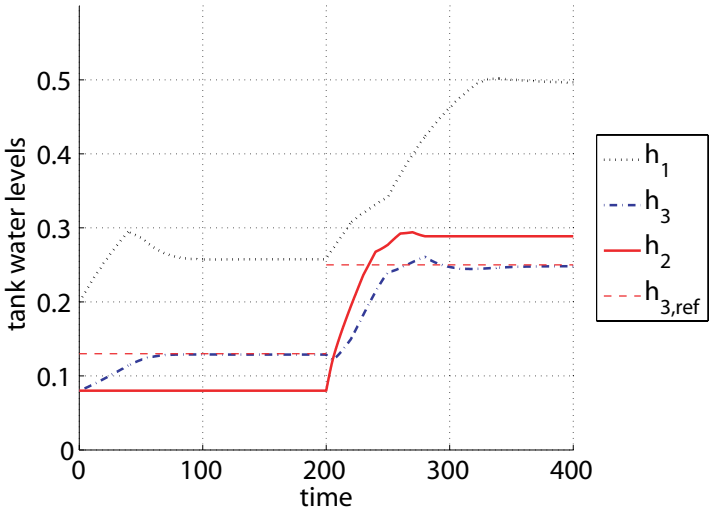
For the first few sample times, the control task is solved by qualitatively pre-selecting to keep valve V_{13} open and then numerical refinement of the continuous actuation more accurately determines the flow through pumps P_1 and P_2, such that the water level of tank 3 almost reaches its reference value. Since, in this example, not any measures are taken to obtain offset-free tracking of constant references (e.g. updating reference values by comparing predictions and measured data [41]) and since the model does not exactly capture the non-linear system behavior, the reference-value is not exactly reached. However, the deviation between linearized model and non-linear system is small in this operational point and the



(a) Valve positions



(b) Input flows



(c) mode m_2 and actuation $u = 0$

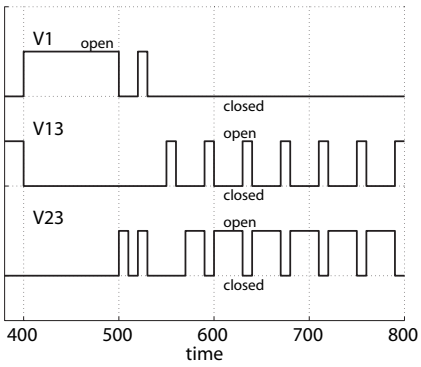
Figure 6.12: Solution to the test scenario for $0 \leq t \leq 400$

reference value $h_{3,ref} = 0.13$ is almost reached ($h_3(190) = 0.129$). When a change in water reference level to $h_{3,ref} = 0.25$ is demanded at $t = 200$, qualitative pre-selection first decides that valve V_{23} shall additionally be opened to support a more quickly rise in h_3 by filling it from tank 2 as well.

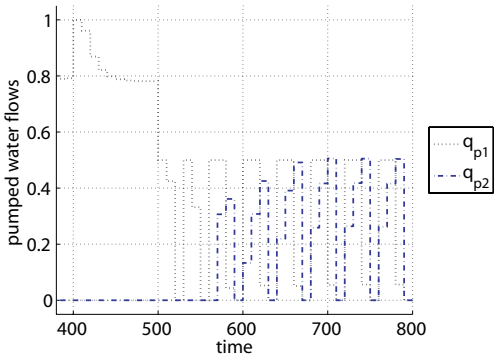
This choice is *not* optimal at $t = 200$ because the water level of tank 2 is still too low and tank 3 empties some of its content into tank 2. The rough qualitative separation of valve flows chosen when building the qualitative model, however, does not allow to correctly decide that the loss of water from tank 3 into tank 2 exceeds the additional in-flow into tank 3 due to u_{p2}. This once again illustrates, that the presented method cannot guarantee to provide optimal solutions, but nevertheless, supporting filling of tank 3 by the second pump clearly is an intuitively good solution to quickly get h_3 closely to its new reference value.

Figure 6.12 shows, how a quick rise in water level h_3 later on is further supported by additionally opening valve V_1, once this is allowed (remember the artificial valve components that constrain command inputs) when $h_1 \geq 0.3$. Finally, when h_3 is close to the reference value, the biases in weight and reference values for pumps P_1 and P_2 let qualitative pre-selection decide that level-control for tank 3 is solely performed via tank 1 again. Also at this operational point, the match between linearized model and non-linear system is good

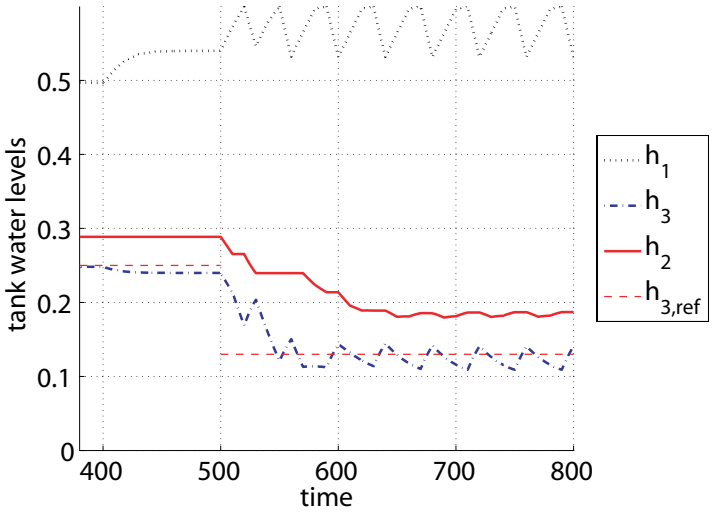
and $h_3(390) = 0.248$ which is almost $h_{3,ref} = 0.25$.



(a) Valve positions



(b) Input flows



(c) mode m_2 and actuation $u = 0$

Figure 6.13: Solution to the test scenario for $380 \leq t \leq 800$

When at $t = 400$ the valve V_{13} gets stuck closed[2], the controller has to find a reconfiguration that does not require valve V_{13} to be opened. This is not to be understood as some additional reconfiguration routine that starts acting as the fault occurs. It is just ordinary qualitative pre-selection that keeps operating as it was before. The only difference between pre-selection at time $t = 390$ (before the fault) and $t = 400$ (after the fault) is the changed initial state (V_{13} is `stuck closed` instead of `operating normally`). For the new initial state, the qualitatively best solution according to the given cost function at time $t = 400$ is to open valve V_1 and to remain controlling water level h_3 solely via tank 1. This is illustrated in Figure 6.13. Since the linearized model does not match the non-linear plant as well in this operational point, the tracking error $h_3 - h_{ref}$ increases.

A good illustration of the limitations of the presented control scheme is provided in the same Figure after $t = 500$ when the valve V_{13} becomes operational again and the reference value changes back to $h_{3,ref} = 0.13$. A solution would be to keep valve V_1 open. However, the bias weights and reference values for q_{p1} and q_{p2}, together with the rough qualitative separation of water flows through the valves, the deviations between linearized models and

[2] As full measurement of the hybrid state is assumed, the controller gets aware of that fault immediately!

non-linear plant and the short prediction horizon let the qualitative pre-selection decide to control h_3 via tank 1 at one time (thus, numerical refinement lets h_1 drop). Then, at the next sample time with a changed initial state, the qualitatively best solution corresponds to controlling water level h_3 via tank 2. Here, numerical refinement sets q_{p1} to its bias $q_{p1} = 0.5$ (because there is no influence on h_3 in this mode during the prediction horizon and h_1 increases again. The resulting next initial state, unfortunately, leads to tanks 1 and 3 being connected again,...

It is easy to find other weights and optimization functions that do not exhibit this switching performance (which certainly is not a very good performance with respect to the required level control of h_3) here. However, it is *not* easy to estimate whether these changed parameters might lead to equally bad behavior in other situations. As the control scheme does not handle different faults and control goals differently, it is not possible to tailor the controller to improved performance in certain situations without influencing its behavior in others.

On the other hand, this can be considered a benefit as well: The controller does not *need* to be tailored or be especially designed for certain situations. Although no guarantee can be given on obtaining an especially 'nice' behavior of the control system in all situations, the mixed qualitative/quantitative control scheme will at least keep the system 'near' the goal, as long as this is possible at all. This can be especially helpful, when the system under control can exhibit lots of failure modes and combinations of faults in several components, as the designer does not have to account for all these faults separately.

At time $t = 800$, the valve V_{13} gets stuck open, and – after an initial unavoidable increase in h_3 such that $h_3 > h_2$ – the controller decides qualitatively, that the control goal can be satisfied best, if additionally tank 2 is connected to tanks 1 and 3, because then the increase in h_3 can be reduced by draining off some water into tank 2. Finally, only V_{13} remains open and the reference value is reached with high accuracy.

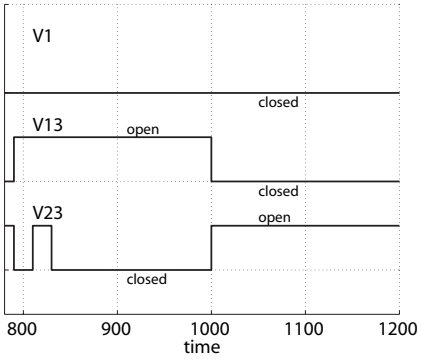
At time $t = 1000$ – the valve V_{13} is operational again – a leak in tank 1 occurs. So the tank is disconnected from tank 3 and level-control of h_3 is taken over via the back-up tank 2.

6.2.5 Result

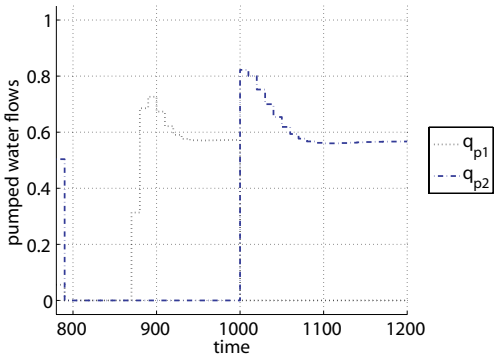
To compare the result of this test scenario to the results presented in other work [4, 51, 53] is somewhat difficult, as other work addresses the 3-tank benchmark problem in its original definition with dedicated nominal modes, fault modes and with h_3 only measurable by 2 threshold sensors.

Since in this work full state measurement is assumed, the task of detecting faults is not dealt with and also the reduced measurement-accuracy for tank 3 isn't handled. However, it has been shown that the remaining part of the benchmark reconfiguration problem can be solved with the proposed mixed qualitative/quantitative control scheme as well. The major difference to the cited work is, that the control scheme presented here does not distinguish dedicated nominal modes and fault modes, but intuitively reacts on a given initial situation, no matter whether nominal or faulty. This provides the benefit, that not all faults have to be explicitly accounted for but introduces the drawback, that controller performance cannot be improved independently for specific situations by tailoring parameters.

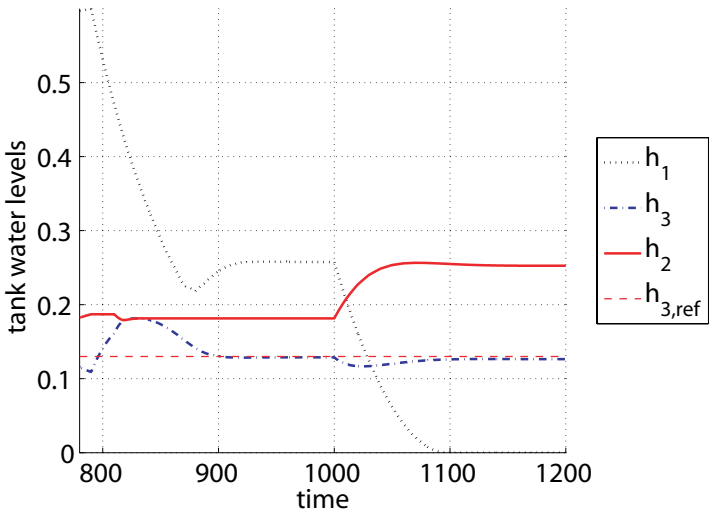
A major problem that was encountered with this example was not directly related to the control scheme itself, but originated from an earlier stage: As the control scheme is designed to operate on a specific class of hybrid systems, a given plant has to be represented within this model class. Such a representation is not unique for a given plant and it turned out that controller performance is significantly different for different hybrid model representations



(a) Valve positions



(b) Input flows



(c) mode m_2 and actuation $u = 0$

Figure 6.14: Solution to the test scenario for $780 \leq t \leq 1200$

(although all of comparable complexity).

Also, performance of the controller with respect to the computational effort necessary for qualitative pre-selection significantly depends on the adjustable parameters. As an example time $0 \leq t \leq 210$ is taken. The 'progressive best first search' algorithm as given in [31] extended by dynamic programming is utilized for qualitative pre-selection.

If the prediction horizon is set to 1, for the first 20 sample steps the algorithm for qualitative pre-selection needs a mean value of

1-step prediction horizion: 225 node expansions per sample step.

With longer prediction horizons this is

2-step prediction horizion: 392 node expansions per sample step
3-step prediction horizion: 587 node expansions per sample step.

While this is a rather moderate increase in computational effort with respect to the prediction horizon, the situation is different for the next time, immediately after the change of the reference value.

Whereas 1061 node expansions are necessary in case of a single step prediction horizon, 14505 node expansions are necessary for a prediction horizon of 2 steps. This results from the fact that a sequence of 4 spurious qualitative trajectories is predicted first. For the same reason, with a 3-step prediction horizon, no valid solution is found within reasonable time.

This degraded performance for longer prediction horizons is a property of this particular example, because for the slow-moving tank-system with reasonably short sampling time, a single step prediction horizon was chosen to obtain a very compact model. This allowed qualitative trajectory likelihoods to be combined into one single class. For longer prediction horizons, a distinction of very unlikely trajectories and a more fine-grained separation of valve-flows certainly is advised to avoid pre-selection of spurious behaviors.

The prediction horizon is not the only influence on the computational effort necessary for qualitative pre-selection. For example adding the term

$$+10 \cdot (h_{1,k+1} - 0.26)^2 + 10 \cdot (h_{2,k+1} - 0.26)^2$$

to the optimality function reduces the number of node expansions for the first 20 sample steps to 60, 110 and 142 node expansions per sample step for a prediction horizon of 1, 2 and 3 sample steps, respectively[3].

The problem with this is, that one cannot clearly estimate the influence of a change in one of the adjustable parameters on the controller performance, both in terms of the necessary computational effort and the 'quality' of the resulting hybrid trajectories.

6.3 Traction Control

An argument sometimes heard with qualitative methods in control theory is that they work well with tanks and might perform badly on other systems. As an argument against this, an example of a traction control task taken from [12] is presented. This example it is just included to demonstrate performance of the proposed control-scheme on a system that does not consist of tanks. However, the presented method is by orders of magnitude too slow to be applicable to traction control of a car in real time.

6.3.1 Hybrid Model

The article [12] demonstrates, that sufficient results for traction control can be achieved on a real car system with a rather simple hybrid model.

The basic model of the cars dynamics uses the vehicles translatorial speed v and the rotational engine speed ω as state variables. These are influenced by the inputs τ which is the torque resulting from combustion and τ_t which is the frictional torque of the tire.

The engine torque τ is the variable by which the system is actually controlled and the frictional torque τ_t of the tire is a parameter that depends on the road condition (described by the coefficient of friction μ) and the wheel slip s

$$s = \frac{w}{i} - \frac{v}{r}, \tag{6.22}$$

where i is the total drive-line gear ratio between ω and v and r is the tire radius. In [12], the non-linear dependence of τ_t on s and μ is captured as a piecewise-affine hybrid model

$$\tau_t = k_1 s + k_2 \mu + k_3$$

[3] It has to be noticed that this also changes the pre-selected mode-sequence command inputs.

where parameters k_i, i=1...3 change depending on whether

$$0.21s - 5.37\mu \leq -0.61$$

or not.

With a sampling interval of $T_s = 20ms$, gear ratio $i = 13.89$ and tire radius $r = 0.298$ the cited article provides the following piecewise affine model for the car's dynamics: If $0.21s - 5.37\mu \leq -0.61$ the model is stable:

$$\begin{bmatrix} \omega \\ v \\ \mu \end{bmatrix}_{k+1} = \begin{bmatrix} 0.983 & 0.784 & -0.354 \\ 2.31e-4 & 0.989 & 4.86e-3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega \\ v \\ \mu \end{bmatrix}_k + \begin{bmatrix} 4.83e-2 \\ 5.66e-6 \\ 0 \end{bmatrix} \tau + \begin{bmatrix} 1.09e-1 \\ -1.50e-3 \\ 0 \end{bmatrix}$$

Whereas if $0.21s - 5.37\mu > -0.61$ the model is unstable:

$$\begin{bmatrix} \omega \\ v \\ \mu \end{bmatrix}_{k+1} = \begin{bmatrix} 1.0005 & -2.18e-2 & -6.53 \\ 6.44e-6 & 1.0003 & 8.97e-2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega \\ v \\ \mu \end{bmatrix}_k + \begin{bmatrix} 4.87e-2 \\ 1.56e-7 \\ 0 \end{bmatrix} \tau + \begin{bmatrix} 0.816 \\ -1.12e-2 \\ 0 \end{bmatrix}$$

Mainly to obtain better illustrations, a new system state $[p, v, \mu]^T$ that is obtained by the invertible state transformation

$$\begin{bmatrix} p \\ v \\ \mu \end{bmatrix} = \begin{bmatrix} \frac{0.21}{i} & -\frac{0.21}{r} & -5.37 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega \\ v \\ \mu \end{bmatrix}$$

is introduced such that $p \leq -0.61$ corresponds to the stable mode and $p > -0.61$ corresponds to the unstable mode.

The car's dynamics with respect to these state variables are

$$\begin{bmatrix} p \\ v \\ \mu \end{bmatrix}_{k+1} = \begin{bmatrix} 0.974 & 1.22e-5 & -0.157 \\ 1.51e-2 & 1 & 8.62e-2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ v \\ \mu \end{bmatrix}_k + \begin{bmatrix} 7.36e-4 \\ 5.66e-6 \\ 0 \end{bmatrix} \tau + \begin{bmatrix} 2.75e-3 \\ -1.50e-3 \\ 0 \end{bmatrix}$$

for $p \leq -0.61$ and

$$\begin{bmatrix} p \\ v \\ \mu \end{bmatrix}_{k+1} = \begin{bmatrix} 1.0008 & -2.24e-5 & -0.160 \\ -4.21e-4 & 1 & 8.74e-2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ v \\ \mu \end{bmatrix}_k + \begin{bmatrix} 7.47e-4 \\ 1.56e-7 \\ 0 \end{bmatrix} \tau + \begin{bmatrix} 2.05e-2 \\ -1.12e-2 \\ 0 \end{bmatrix}$$

for $p > -0.61$.

6.3.2 Qualitative Model

According to the example definition, ω is limited to values $0 \leq \omega \leq 1000$, v is limited by $0 \leq v \leq 25$. The engine torque τ is limited by $-20 \leq \tau \leq 176$ and the maximum change $\Delta\tau$ of the engine torque from one sample time to the next is limited by $-40 \leq \Delta\tau \leq 40$. μ is assumed to be $0.2 \leq \mu \leq 0.8$ what approximately corresponds to road conditions from ice to dry concrete.

Additionally, the above model only is valid for non-negative wheel slip s

$$s = \frac{w}{i} - \frac{v}{r} \geq 0.$$

With these limitations, the newly introduced parameter p satisfies $-4.3 \leq p \leq 14.2$.

A 2-component qualitative model is compiled. Component 1 exhibits only one mode and updates the engine torque τ as

$$\tau_{k+1} = \tau_k + \Delta\tau \tag{6.23}$$

Component 2 abstracts the vehicle's dynamics. This component exhibits one out of 2 modes and a mode change occurs, if parameter p rises above or drops below -0.61.

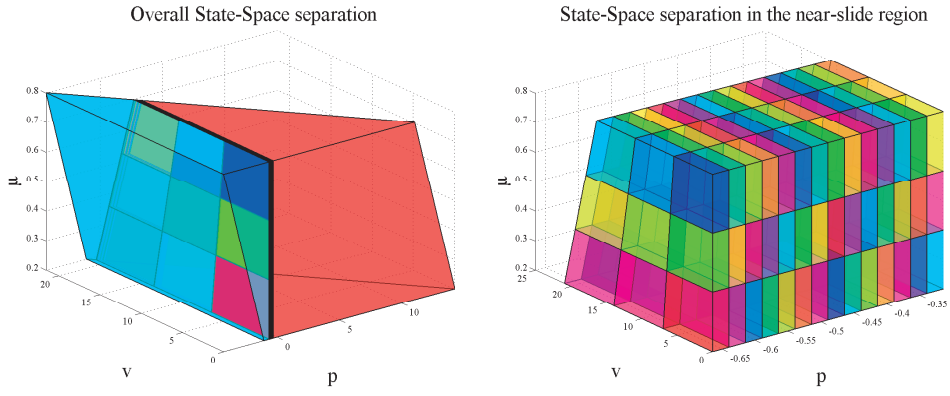
For qualitative abstraction, the value range of variables is partitioned as follows: $\Delta\tau$ is just partitioned with respect to its sign, τ is separated into qualitative regions of width 40 around the value 0. (So, in fact, there is one smaller region $140 \leq \tau \leq 176$.)

The road condition μ is just distinguished to be very slippy ($0.2 \leq \mu < 0.4$), moderately slippy ($0.4 \leq \mu < 0.6$) or good ($0.6 \leq \mu \leq 0.8$).

The only required qualitative distinction is whether parameter p is above or below the value of -0.61. To capture the dynamic behavior of the vehicle correctly, it is important to estimate whether this parameter p remains below -0.61 or can be forced to get smaller than -0.61 for certain engine torques. Due to the very small sampling interval, the vehicle dynamics only allow very small changes from one sampling to the next. Therefore, for very high or very low values of this parameter, no mode change will be predicted within a reasonably large prediction horizon. So a fine-grained qualitative separation in regions far away from $p = -0.61$ does not contribute to better qualitative predictions.

A fine grained partitioning of p is only introduced in regions where a mode change could occur within a prediction horizon of approximately 10 steps. In this region $-0.66 \leq p \leq -0.31$ very fine grained qualitative regions of width 0.025 around -0.61 are established, whereas all values $p < -0.66$ and $p > -0.31$ are combined to one respective qualitative value.

Additionally, to obtain the final qualitative separation of the vehicle-dynamics state $[p, v, \mu]^T$, each qualitatively distinct region with respect to p and μ additionally gets separated into 3 equally sized qualitative regions with respect to v. This qualitative separation is shown in Figure 6.15.



(a) Overall state space (b) Detailed display near $p = -0.61$

Figure 6.15: Separation of the state space $[p, v, \mu]^T$

For compilation of the qualitative models, transition likelihoods are separated into 2 discinct classes by utilizing Otsu's method [45].

In order to achieve good results, an extension to this qualitative model is necessary due to the particular way, deviations from given reference values are evaluated for qualitative values during pre-selection (see Section 5.1.2). I.e., for the purpose of evaluating deviations

from references, the state-space region represented by each qualitative value is represented by a bounding ball. With respect to this example this is problematic – it is anticipated here, that only parameter p shall be controlled towards a specified reference value – because most qualitative values representing the state of component 2 have wide extension in directions of μ and especially v, but only small extension in the direction of p. Therefore, the bounding ball provides a very bad approximation for the values p associated to a particular qualitative value.

This could be circumvented on the one hand by scaling p, v and μ to p', v' and μ', such that the bounding ball of the region represented by a qualitative value is mainly dependent on the region's extension in direction of p'. This, however, could introduce problems with respect to limited accuracy of numerical computations.

So another approach is followed and a third component is introduced to the model,which just copies the dynamics of variable p from component 2, i.e.

$$p_{k+1} = 0.974 \cdot p_k + \begin{bmatrix} 1.22e-5 & -0.157 & 7.36e-4 \end{bmatrix} \begin{bmatrix} v \\ \mu \\ \tau \end{bmatrix}_k + 2.75e-3$$

for $p \leq -0.61$ and

$$p_{k+1} = 1.0008 \cdot p_k + \begin{bmatrix} -2.24e-5 & -0.160 & 7.47e-4 \end{bmatrix} \begin{bmatrix} v \\ \mu \\ \tau \end{bmatrix}_k + 2.05e-2$$

for $p > -0.61$.

So, if this new artificial component 3 receives μ_k and v_k from component 2, it will make the same predictions for p as component 2, as well. However, as the state of the new component only consists of variable p, the bounding-ball is a very good approximation for evaluating deviations from a reference value.

6.3.3 Control

Traction control is investigated in two test cases

- The vehicle standing still on an icy surface with wheels rotating quickly
- The vehicle accelerating quickly on a dry surface, suddenly changing to ice

The first of these cases is also included in the article [12]. However, a comparison of results is difficult as the article deals with measured data, whereas here only simulation result without disturbances are presented. The initial condition is given by $\tau = 0$ and

$$\omega = 180 \qquad v = 0 \qquad \mu = 0.3,$$

which corresponds to the modified initial state

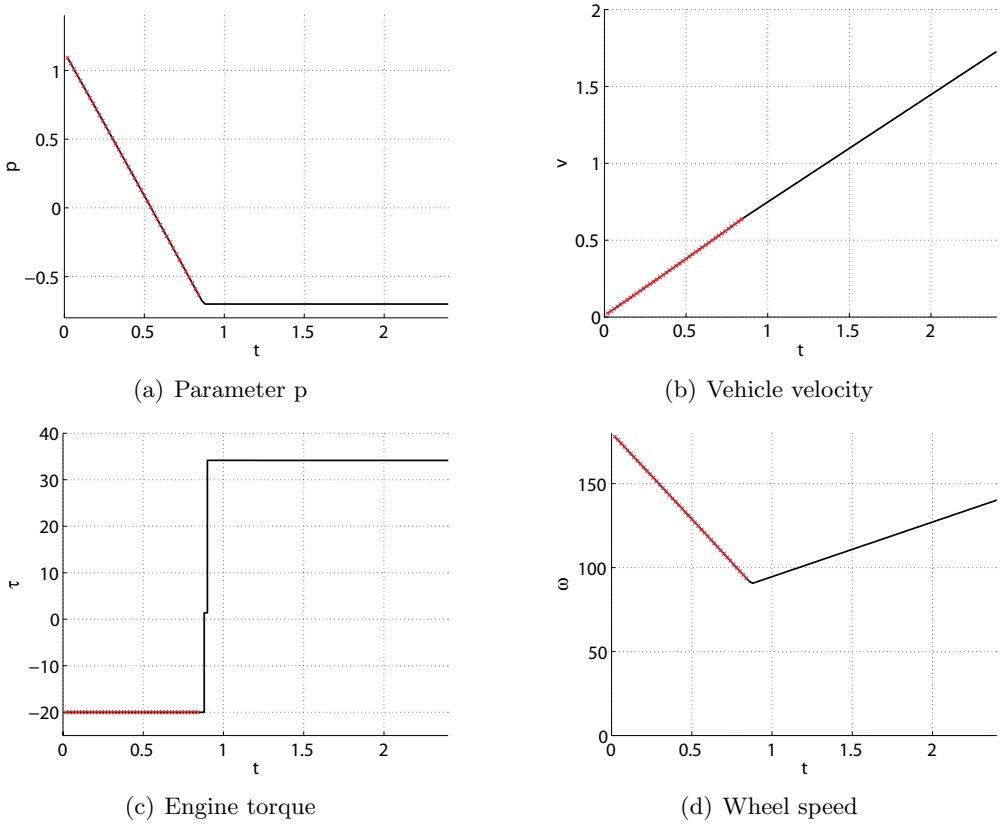
$$p = 1.13 \qquad v = 0 \qquad \mu = 0.3.$$

This indicates that components 2 and 3 are in the unstable mode.

Now, traction control is performed with a prediction horizon of length 5 and an optimality function

$$J = \sum_{i=1}^{N} (p_{k+i} + 0.7)^2$$

is used to obtain high acceleration of the car near the border of instability at p=-0.61. For qualitative pre-selection, likelihood costs are emphasized 1000 times stronger than reference costs, thus, mainly sticking to the most likely non-spurious qualitative trajectories only.



(a) Parameter p

(b) Vehicle velocity

(c) Engine torque

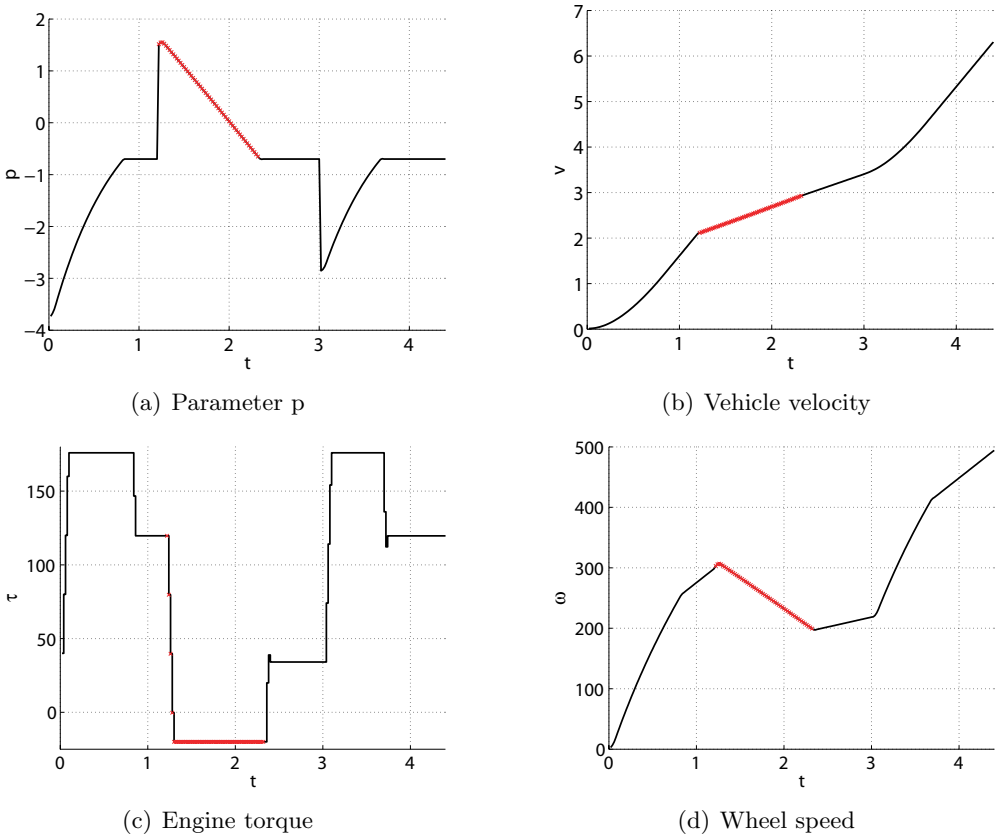
(d) Wheel speed

Figure 6.16: Solution to test scenario 1 with 5-step prediction horizon

The resulting engine torque τ that is applied to the system, and the resulting trajectories of p and v and ω are shown in Figure 6.16. Samples at times where the system is in the unstable mode are indicated by small red '×'. Results obtained for shorter or longer prediction horizons are almost equivalent. However, it once again has to be noticed that the utilized implementation of the qualitative hybrid control scheme as MATLAB toolbox is by orders of magnitude too slow to be able to calculate control for this system in real time (sample time $T_s = 0.02$).

Results obtained for the second test case are more interesting. Here, a situation is investigated where the car and the engine stand still and the road surface is dry concrete ($\mu = 0.7$). After $t = 1.2$, the road condition suddenly changes gets very slippy ($\mu = 0.2$) and changes back to $\mu = 0.7$ at $t = 3$. The control goal, again, is to keep the parameter p at value $p = -0.7$, providing high acceleration of the car while avoiding to be in the instable (slipping) mode. Adjustable parameters for this test case are the same as in for the previous case.

Two exemplary results for a prediction horizon of $N = 2$ and $N = 5$ are presented in Figures 6.17 and 6.18. Although these plots look quite similar, it can be observed different performance of the controller when parameter p gets near its reference value $p_{ref} = -0.7$ around time $t = 2.35$. With short prediction horizons, there is an overshoot in p, because if qualitative pre-selection does not predict a mode-change from instable to stable early enough,



(a) Parameter p

(b) Vehicle velocity

(c) Engine torque

(d) Wheel speed

Figure 6.17: Solution to test scenario 2 with 2-step prediction horizon

torque τ cannot be increased sufficiently early to keep p above -0.7.

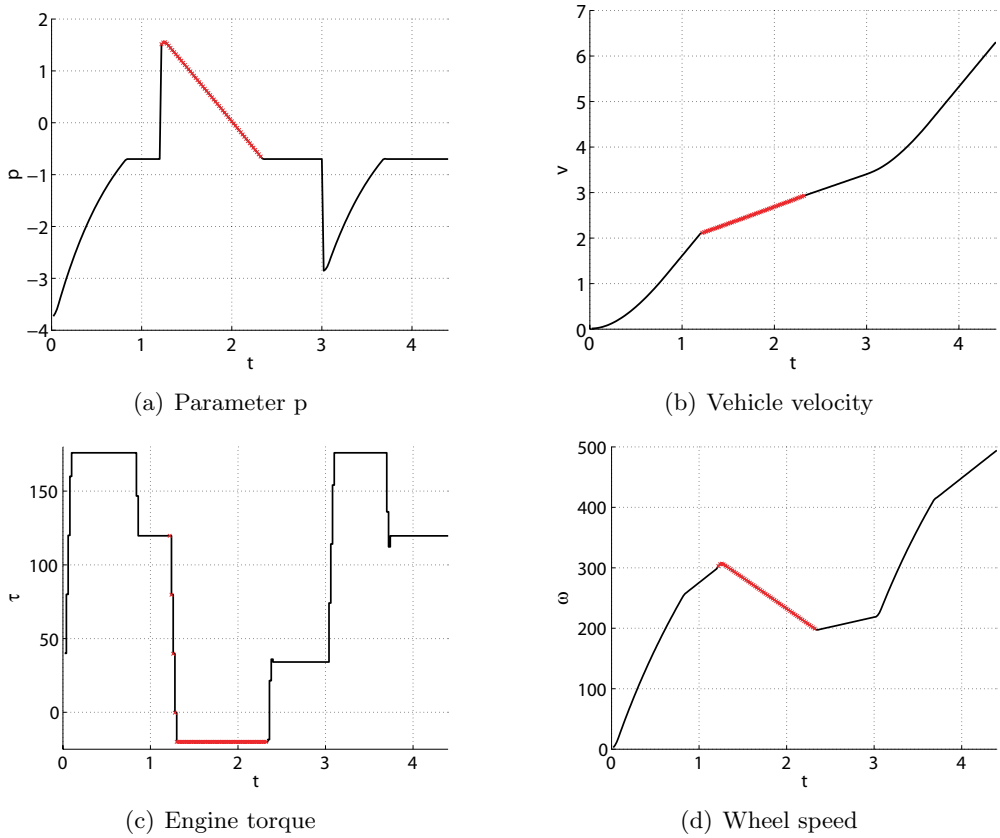
On the other hand, it is interesting to notice, that there is no further decrease of the observed overshoot in p once the prediction horizon goes beyond $N = 6$. This is, because in the qualitative model a fine-grained qualitative partitioning of values p was only provided near -0.61. If this area of fine-grained qualitative separation was increased to higher values of p, performance could be improved further by longer prediction horizons.

However, in this example much longer prediction horizons are prohibitive, anyway, because the computational effort necessary for qualitative pre-selection does not increase sub-linearly with respect to the prediction horizon as was the case for the 3-tank system. Again, the mean number of search node expansions per sampling and the same algorithm as for the 3-tank system are used to evaluate the computational effort.

Prediction horizon	Mean number-	Maximum number of node expansions
2	30	94
3	69	641
4	152	1010
5	264	2020

6.4 Summary

In this chapter first qualitative modeling and qualitative hybrid control was summarized by working out a simple example in detail. Then two more complex example systems were inves-



(a) Parameter p

(b) Vehicle velocity

(c) Engine torque

(d) Wheel speed

Figure 6.18: Solution to test scenario 2 with 5-step prediction horizon

tigated to demonstrate applicability of the hybrid control approach. As result, it is observed that qualitative hybrid control is able to handle the complexity of the COSY benchmark problem on the one hand and can handle non-tank hybrid systems on the other hand as well.

However, the examples also demonstrated that qualitative hybrid control often is no 'out-of-the-box' solution for a given example and some tailoring may be required. This tailoring can be necessary for both, building and structuring of the hybrid model itself on the one hand and tuning of adjustable parameters on the other hand. As far as this work has proceeded, no rules can be formulated yet, which could guide this tailoring.

Performance of qualitative hybrid control, both, with respect to quality of the achieved trajectories and with respect to computational efficiency, can significantly depend on a particular selection of the adjustable parameters.

Chapter 7

Conclusion

In this thesis, a hybrid control scheme was presented which allows to tackle a class of complex multi-component systems by first abstracting the complex hybrid control task to the qualitative level, where it is solved more easily. The solution to this abstracted control task then allows to reformulate the original hybrid control task as control of standard time-variant systems.

Basis for this approach is a class of especially tailored qualitative models that allow compact component-wise off-line compilation on the one hand and facilitate efficient on-line search for good solutions to the qualitative control problem on the other. This is achieved by a well-structured graphical representation of the model that is a very compact encoding of all possible qualitative trajectories each individual component may exhibit regarding a 1-step time horizon.

To keep the qualitative model reasonably small, *component-wise* abstraction and encoding of *only single time-step* qualitative trajectories were necessary. This requires, however, that qualitative on-line reasoning about possible behaviors of the hybrid model has to concurrently operate on several qualitative component models instead of operating just on a single overall model.

To make this simultaneous reasoning among the individual component models most efficient, all the model graphs are constructed to be acyclic and directed with respect to a pre-specified common ordering of variables. This allows an intuitive focussed on-line construction of those parts of the single overall model that describe behaviors of the hybrid system which promise to be valid solutions to the hybrid control task. With this graphical representation, the control task on the abstracted, qualitative, level can be formulated as shortest path search.

Furthermore, the common ordering of variables can be utilized to incorporate structural information on the hybrid model into the qualitative model, such that the resulting overall model graph is highly connected and discrete dynamic programming can be utilized to solve this shortest path search more efficiently.

In fact, qualitative pre-selection is formulated as shortest path search through a highly connected graphical representation of the overall qualitative model that is intuitively and efficiently generated on-line from the individual component models. We suggest to utilize A^*-search to perform this shortest path search. This is a best-first-search strategy and, thus, allows to investigate (and, hence, requires to generate) the overall model only at very focussed branches towards promising solutions to the control task. Further, by utilization of dynamic programming ideas it allows to exploit the structural information that was compiled into the model in a way that lead to a highly connected search graph.

The result of solving the hybrid control task on the abstracted level by qualitative pre-selection is the specification of a sequence of operational modes for each of the components, together with the associated discrete inputs that have to be applied to the system and the constraints on continuous states and inputs that have to be satisfied in order to drive the system through the pre-selected mode sequence. Continuous inputs, themselves, however are only specified on an abstracted level and require further numerical refinement.

The specified sequence of operational modes allows to regard the hybrid model as standard time-variant continuous model. Further, the constraints on the range of values for the variables in the continuous model may be expressed by linear inequalities in the hybrid systems continuous inputs. This allows to formulate the last remaining issue to solve the hybrid control task – numerical refinement of continuous inputs – as constrained quadratic program.

A well known method from control theory that handles on-line receding horizon control of constrained time-variant systems and builds upon such a mathematical program formulation is model predictive control. In the presented hybrid control scheme, model predictive control not only numerically refines continuous inputs but also ensures stability of the overall control, as it either validates a qualitatively pre selected mode sequence and determines an according stable controller or it rejects the pre-selection because of spuriosity and hands back to the qualitative pre-selection to resume path-search for the next-best qualitative trajectory.

This interplay between the two solvers, in principle, can be utilized to determine the optimal solution to the hybrid control task. However, this claim of optimality is usually dropped in trade off for a much faster operation what makes the control scheme available to more complex systems.

Applicability of the proposed method was demonstrated on examples of different complexity. The example problems were sufficiently solved by the presented method. This, however, sometimes needed non-intuitive tuning of adjustable parameters and significantly different results were obtained for different selections of these parameters.

7.1 Outlook

However, the presented qualitative modeling and hybrid control scheme is still under development and some questions still remain open and are subject to ongoing research:

- Up to now it has not yet been possible to specify rules how to choose the polytopic regions related to the qualitative values and how to cleverly choose other adjustable parameters. As examples indicate, such rules would be very advantageous, as the determined solution to a given hybrid control task can be very sensitive to these parameters.

- Full measurement of the hybrid state at each time is assumed in the present work. A combination of hybrid control and a hybrid estimation scheme developed for the same class of hybrid systems [31] will help to relax this requirement in the future.

- A main chance that is thought to further improve the presented control method is related to the search of alternative mode-sequences, if the first result of the qualitative pre-selection turns out to be spurious. In this case, qualitative search is resumed from the point where the spurious solution was found and it looks for qualitative trajectories that provide alternative mode sequences. Advanced techniques form the field of artificial intelligence could help to evaluate the reasons why the trajectory failed and could help to avoid making the same 'mistake' in another trajectory.

- It will be further investigated how the presented control scheme could be applied to an extended class of hybrid systems that does not need to specify input/output causality of each component. The proposed control scheme should be able to handle this task because qualitative pre-selection does not require any assumptions on such causality and for subsequent model predictive control causality can be determined on-line by causal analysis with the aid of the pre-selected mode sequence.
- And last but not least, some work ought to be done to extend the control framework to non-linear or asynchronously sampled hybrid systems. Non-linearity is assumed to be manageable by the qualitative modeling and pre-selection quite well. However, further investigations are needed on the interplay between qualitative pre-selection and an appropriate numerical verification and refinement of continuous actuation.

Bibliography

[1] Alur, R. and D. L. Dill: 1994, 'A Theory of Timed Automata'. *Theoretical Computer Science* **126**(2), 183–235.

[2] Alur, R. and G. J. Pappas (eds.): 2004, 'Hybrid Systems: Computation and Control, HSCC 2004', Vol. 2993 of *Lecture Notes in Computer Science*. Springer-Verlag.

[3] Andersen, H.: 1997, 'An Introduction to Binary Decision Diagrams'. Technical Report Lecture Notes for 49285 Advanced Algorithms E97, Department of Information Technology, Technikal University of Denmark, Lyngby, Denmark.

[4] Askari, J., B. Heiming, and J. Lunze: 1999, 'Controller Reconfiguration Based on a Qualitative Model: A Solution of Three-Tanks Benchmark Problem'. In: *Proceedings of the European Control Conference ECC99*. Karlsruhe, Germany. Paper ID: F1039-3.

[5] Belman, R.: 1957, *Dynamic Programming*. Princeton, NJ: Princeton University Press.

[6] Bemporad, A., F. Borrelli, and M. Morari: 2000, 'Optimal Controllers for Hybrid Systems: Stability and Piecewise Linear Explicit Form'. In: *Proceedings of the 39th IEEE Conference on Decision and Control, Sydney*, Vol. 2. pp. 1810–1815.

[7] Bemporad, A., F. Borrelli, and M. Morari: 2002a, 'Model Predictive Control Based on Linear Programming - The Explicit Solution'. *IEEE Transactions on Automatic Control* **47**(12), 1974–1985.

[8] Bemporad, A. and N. Giorgetti: 2002, 'A sat-based hybrid solver for optimal control of hybrid systems'. In: A. R. and G. Pappas (eds.): *Hybrid Systems: Computation and Control*, Vol. 2289 of *Lecture Notes in Computer Science*. pp. 126–141.

[9] Bemporad, A. and M. Morari: 1999, 'Control of systems integrating logic, dynamics and constraints'. *Automatica* **35**(3), 407–427.

[10] Bemporad, A., M. Morari, D. Vivek, and E. Pistikopoulos: 2002b, 'The explicit linear quadratic regulator for constrained systems'. *Automatica* **38**(1), 3–20.

[11] Bertsekas, D.: 1995, *Dynamic Programming and Optimal Control*, Vol. 2. Athena Scientific.

[12] Borrelli, F., A. Bemporad, M. Fodor, and D. Hrovat: 2006, 'An MPC/Hybrid System Approach to Traction Control'. *IEEE Transactions on Control Systems Technology* **14**(3), 541–552.

[13] Branicky, M.: 1995, 'Studies in hybrid systems: modeling, analysis, and control'. Ph.D. thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA.

[14] Bryant, R.: 1986, 'Graph Based Algorithms for Boolean Function Manipulation'. *IEEE Transactions on Computers* **C-35**(8), 677–691.

[15] Buchberger, B.: 1970, 'An Algorithmical Criterion for the Solvability of Algebraic Systems of Equations (German)'. *Aequationes Mathematicae* **4**(3), 347–383.

[16] Buchberger, B. and F. Winkler: 1998, 'Groebner Bases and Applications'. In: *Proceedings of the internat. conference "33 Years of Groebner Bases"*, Vol. 251 of *London Mathematical Science*.

[17] Cassandras, C., D. Pepyne, and Y. Wardi: 2001, 'Optimal control of a class of hybrid systems'. *IEEE Transactions on Automatic Control* **46**(3), 398–415.

[18] Chutinan, A. and B. Krogh: 1998, 'Computing Polyhedral Approximations to Flow Pipes for Dynamic Systems'. In: *Proceedings of the 37th IEEE Conference on Decision and Control*, Vol. 2. pp. 2089–2094.

[19] Clarke, E., O. Grumberg, and D. Peled: 1999, *Model Checking*. Cambridge, Mass.: MIT Press.

[20] De Schutter, B.: 1996, 'Max Algebraic System Theoty for Discrete Event Systems'. Ph.D. thesis, Faculty of Applied Sciences, K.U. Leuven, Leuven, Belgium.

[21] De Schutter, B. and T. van den Boom: 2001, 'Model predictive control for max-plus-linear discrete event systems'. *Automatica* **37**(7), 1049–1056.

[22] Dechter, R.: 2003, *Constraint Processing*. San Francisco, CA: Morgan Kaufmann.

[23] Even, S.: 1979, *Graph Algorithms*, Computer Software Engineering Series. Rockville, Maryland: Computer Science Press.

[24] Förstner, D., M. Jung, and J. Lunze: 2002, 'A discrete-event model of asynchronous quantised systems'. *Automatica* **8**, 1277–1286.

[25] Goldberg, A. and C. Harrelson: 2004, 'Computing the Shortest Path: A^* meets Graph Theory'. Technical Report MSR-TR-2004-24, Microsoft Research.

[26] Hamscher, W., L. Console, and J. DeKleer: 1992, *Readings in Model-Based Diagnosis*. San Mateo, Calif.: Morgan Kaufmann.

[27] Hart, P., N. Nilsson, and B. Raphael: 1968, 'A formal basis for the heuristic determination of minimum cost paths'. *IEEE Transactions on System Science and Cybernetics* **SCC-4**(2), 100–107.

[28] Heemels, W., B. De Schutter, and A. Bemporad: 2001, 'Equivalence of Hybrid Dynamical Models'. *Automatica* **37**(7), 1085–1091.

[29] Heiming, B. and J. Lunze: 1999, 'Definition of the Three-Tank Benchmark Problem for Controller Reconfiguration'. In: *Proceedings of the European Control Conference*.

[30] Hofbaur, M. and B. Williams: 2002, 'Mode Estimation of Probabilistic Hybrid Systems'. In: C. Tomlin and M. A. Greenstreet (eds.): *Hybrid Systems: Computation and Control, HSCC 2002*. pp. 253–266.

[31] Hofbaur, M. W.: 2005, *Hybrid Estimation of Complex Systems*, Vol. 319 of *Lecture Notes in Control and Information Sciences (LNCIS)*. Springer.

[32] Kerrigan, E. and D. Mayne: 2002, 'Optimal control of constrained piecewise affine systems with bounded disturbances'. In: *Proceedings of the 41st IEEE Conference on Decision and Control*, Vol. 2. Las Vegas, USA, pp. 1552–1557.

[33] Kleissl, W.: 2002, 'Structural Analysis of Hybrid Systems'. Master's thesis, Institute of Automation and Control, Graz University of Technology, Graz, Austria.

[34] Koutsoukos, X. D., P. J. Antsaklis, J. A. Stiver, and M. Lemmon: 2000, 'Supervisory Control of Hybrid Systems'. *Proceedings of the IEEE* **88**(7), 1026–1049.

[35] Kreindler, E. and P. Sarachlik: 1964, 'On the concepts of controllability and observability of linear systems'. *IEEE Transactions on Automatic Control* **9**(2), 129–136.

[36] Kuipers, B.: 1994, *Qualitative Reasoning: Modeling and Simulation with incomplete Knowledge*. Cambridge, MA: MIT Press.

[37] Kvasnica M. and Grieder, P. and Baoti, M. ć: 2004, 'Multi Parametric Toolbox (MPT)'.

[38] Lazar, M., W. Heemels, S. Weiland, and A. Bemporad: 2004, 'Stabilization Conditions for Model Predictive Control of Constrained PWA Systems'. In: *Proceedings of the 43rd IEEE Conference on Decision and Control*, Vol. 5. pp. 4595–4600.

[39] Lunze, J.: 1992, 'Qualitative modeling of continuous variable systems by means of nondeterministic automata'. *Journal of Intelligent System Engineering* **1**(1), 22–30.

[40] Lunze, J.: 1994, 'Qualitative Modeling of linear dynamical systems with quantized state measurements'. *Automatica* **30**(3), 417–431.

[41] Maciejowski, J.: 2002, *Predictive Control with Constraints*. Essex, UK: Pearson Education Ldt.

[42] Maler, O. and A. Pnueli (eds.): 2003, 'Hybrid Systems: Computation and Control, HSCC 2003', Vol. 2623 of *Lecture Notes in Computer Science*. Springer-Verlag.

[43] Mayne, D., J. Rawlings, C. Rao, and P. Sckaert: 2000, 'Constrained model predictive control: Stability and Optimality'. *Automatica* **36**(6), 789–814.

[44] Nayak, P.: 1995, *Automated Modelling of Physical Systems*. Berlin: Springer.

[45] Otsu, N.: 1979, 'A threshold selection method from gray-level histograms'. *IEEE Transactions on Systems, Man and Cybernetics* **9**(1), 62–66.

[46] Russell, S. and P. Norvig: 2003, *Artificial intelligence: a modern approach*, Prentice Hall series in artificial intelligence. NJ: Pearson Education, 2 edition.

[47] Sachenbacher, M. and P. Struss: 2003, 'Automated Qualitative Domain Abstraction'. In: G. Gottlob and T. Walsh (eds.): *Proceedings of the 18th International Conference on Artificial Intelligence IJCAI'03*. Acapulco, Mexico, pp. 382–387.

[48] Sontag, E.: 1981, 'Nonlinear regulation: The piecewise linear approach'. *IEEE Transactions on Automatic Control* **AC-26**(2), 346–358.

[49] Struss, P.: 2002, 'Automated Abstraction of Numerical Simulation Models – Theory and Practical Experience'. In: *Proceedings of the 16th international workshop on qualitative reasoning*. Spain, pp. 161–168.

[50] Travé-Massuyès, L. and R. Pons: 1997, 'Causal ordering for Multiple Mode Systems'. In: *Proceedings of the 11th Workshop on Qualitative Reasoning (QR97)*. Pavia, Italy, pp. 203–214.

[51] Tsuda, K., D. Mignone, G. Ferrari-Trecate, and M. Morari: 2001, 'Reconfiguration Strategies for Hybrid Systems'. In: *Proceedings of the 2001 American Control Conference*, Vol. 2. Arlington, Virginia, pp. 868–873.

[52] van den Boom, T. and B. de Schutter: 2001, 'Model predictive control for perturbed max-plus-linear systems: a stochastic approach'. In: *Proceedings of the 40th IEEE Conference on Decision and Control*, Vol. 5. Florida, USA. 4535–4540.

[53] Villa, J. L., M. Duque, A. Gauthier, and N. Rakoto-Ravalontsalama: 2003, 'MLD Control of Hybrid Systems: Application to the Three-TankBenchmark Problem'. *IEEE* pp. 666–671.

[54] Williams, B., M. Ingham, S. Chung, and P. Elliot: 2003, 'Model-based progrmming of intelligent embedded systems and robotic space explorers'. In: *Proceedings of the IEEE*, Vol. 91. pp. 212–237.

[55] Xu, X. and P. Antsaklis: 2001, 'An approach for solving general switched linear quadratic optimal control problems'. In: *Proceedings of the 40th IEEE Conference on Decision ad Control*, Vol. 3. pp. 2478–2483.

Appendix A

Theory

A.1 Linearization

Linearization is used to approximate a non-linear model

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{A.1}$$

for small deviations from an operational point specified by x_0 and u_0 by a linear model

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}. \tag{A.2}$$

First, the simplified case where the right-hand side in (A.1) is a scalar non-linear function in a single variable $f(x)$ is considered. If this function can be differentiated infinitely many times it can be expressed by a Taylor series expansion around a base-point $x = a$ as:

$$f(x) = \sum_{n=0}^{\infty} (x-a)^n \cdot \frac{f^{(n)}(a)}{n!} \tag{A.3}$$

where $f^{(n)}(a)$ denotes the n^{th} derivative of the function $f(x)$ evaluated at the point $x = a$, i.e.

$$f^{(n)}(a) = \left.\frac{\partial^n f(x)}{\partial x^n}\right|_{x=a} \tag{A.4}$$

If only small deviations of x from that base-point $x = a$ are considered, the function can closely be approximated by the first few terms of the sum in (A.3). Specifically, for linearization only the constant and the linear term are considered such that

$$f(x) \approx f(a) + \left.\frac{\partial f(x)}{\partial x}\right|_{x=a} \cdot (x-a). \tag{A.5}$$

If, additionally, the base point $x = a$ is a zero of the function $f(x)$, i.e.

$$f(a) = 0 \tag{A.6}$$

the function $f(x)$ around this point $x = a$ can be approximated by

$$f(x) \approx \left.\frac{\partial f(x)}{\partial x}\right|_{x=a} \cdot (x-a). \tag{A.7}$$

Extending this concept to the non-linear model (A.1), with

$$\begin{aligned}\mathbf{x} &= [x_1, \ldots, x_{N_x}]^T \\ \mathbf{u} &= [u_1, \ldots, u_{N_u}]^T\end{aligned}$$

and

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = [f_1(\mathbf{x}, \mathbf{u}), \ldots, f_{N_x}(\mathbf{x}, \mathbf{u})]^T,$$

linearized approximations of $\mathbf{f}(\mathbf{x}, \mathbf{u})$ are obtained as follows.

If the model (A.1) is linearized around a steady-state of the model indicated by

$$\mathbf{f}(\mathbf{x}_{SS}, \mathbf{u}_{SS}) = \mathbf{0} \tag{A.8}$$

(A.1) can be approximated by a linear model

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}_{SS}\mathbf{x} + \mathbf{B}_{SS}\mathbf{u} \tag{A.9}$$

where

$$\mathbf{A}_{SS} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x},\mathbf{u})}{\partial x_1}\Big|_{\mathbf{x}=\mathbf{x}_{SS},\mathbf{u}=\mathbf{u}_{SS}} & \cdots & \frac{\partial f_1(\mathbf{x},\mathbf{u})}{\partial x_{Nx}}\Big|_{\mathbf{x}=\mathbf{x}_{SS},\mathbf{u}=\mathbf{u}_{SS}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{Nx}(\mathbf{x},\mathbf{u})}{\partial x_1}\Big|_{\mathbf{x}=\mathbf{x}_{SS},\mathbf{u}=\mathbf{u}_{SS}} & \cdots & \frac{\partial f_{Nx}(\mathbf{x},\mathbf{u})}{\partial x_{Nx}}\Big|_{\mathbf{x}=\mathbf{x}_{SS},\mathbf{u}=\mathbf{u}_{SS}} \end{bmatrix} \tag{A.10}$$

$$\mathbf{B}_{SS} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x},\mathbf{u})}{\partial u_1}\Big|_{\mathbf{x}=\mathbf{x}_{SS},\mathbf{u}=\mathbf{u}_{SS}} & \cdots & \frac{\partial f_1(\mathbf{x},\mathbf{u})}{\partial u_{Nu}}\Big|_{\mathbf{x}=\mathbf{x}_{SS},\mathbf{u}=\mathbf{u}_{SS}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{Nx}(\mathbf{x},\mathbf{u})}{\partial u_1}\Big|_{\mathbf{x}=\mathbf{x}_{SS},\mathbf{u}=\mathbf{u}_{SS}} & \cdots & \frac{\partial f_{Nx}(\mathbf{x},\mathbf{u})}{\partial u_{Nu}}\Big|_{\mathbf{x}=\mathbf{x}_{SS},\mathbf{u}=\mathbf{u}_{SS}} \end{bmatrix} \tag{A.11}$$

If the model (A.1) is linearized around other values $\mathbf{x} = \mathbf{x}_0$ and $\mathbf{u} = \mathbf{u}_0$, it can be approximated by an affine model

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}_0\mathbf{x} + \mathbf{B}_0\mathbf{u} + \mathbf{e} \tag{A.12}$$

where

$$\mathbf{A} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x},\mathbf{u})}{\partial x_1}\Big|_{\mathbf{x}=\mathbf{x}_0,\mathbf{u}=\mathbf{u}_0} & \cdots & \frac{\partial f_1(\mathbf{x},\mathbf{u})}{\partial x_{Nx}}\Big|_{\mathbf{x}=\mathbf{x}_0,\mathbf{u}=\mathbf{u}_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{Nx}(\mathbf{x},\mathbf{u})}{\partial x_1}\Big|_{\mathbf{x}=\mathbf{x}_0,\mathbf{u}=\mathbf{u}_0} & \cdots & \frac{\partial f_{Nx}(\mathbf{x},\mathbf{u})}{\partial x_{Nx}}\Big|_{\mathbf{x}=\mathbf{x}_0,\mathbf{u}=\mathbf{u}_0} \end{bmatrix} \tag{A.13}$$

$$\mathbf{B} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x},\mathbf{u})}{\partial u_1}\Big|_{\mathbf{x}=\mathbf{x}_0,\mathbf{u}=\mathbf{u}_0} & \cdots & \frac{\partial f_1(\mathbf{x},\mathbf{u})}{\partial u_{Nu}}\Big|_{\mathbf{x}=\mathbf{x}_0,\mathbf{u}=\mathbf{u}_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{Nx}(\mathbf{x},\mathbf{u})}{\partial u_1}\Big|_{\mathbf{x}=\mathbf{x}_0,\mathbf{u}=\mathbf{u}_0} & \cdots & \frac{\partial f_{Nx}(\mathbf{x},\mathbf{u})}{\partial u_{Nu}}\Big|_{\mathbf{x}=\mathbf{x}_0,\mathbf{u}=\mathbf{u}_0} \end{bmatrix} \tag{A.14}$$

$$\mathbf{e} = \begin{bmatrix} f_1(\mathbf{x}_0, \mathbf{u}_0) \\ \vdots \\ f_{Nx}(\mathbf{x}_0, \mathbf{u}_0) \end{bmatrix} \tag{A.15}$$

A.2 Calculation of Transition Likelihoods

When compiling the qualitative models, a likelihood-value has to be determined for each transition of the non-deterministic automaton model. This automaton model qualitatively approximates a hybrid automaton (Section 3.3).

Such a hybrid model specifies

- Transition likelihoods to reach certain discrete states $x_{d,k+1}$ while observing the discrete outputs $y_{d,k+1}$, given discrete state $x_{d,k}$, discrete commands $u_{d,k}$, continuous state $\mathbf{x}_k$ and continuous actuation $\mathbf{u}_k$
- The continuous state $\mathbf{x}_{k+1}$ that is reached, given continuous state $\mathbf{x}'_k$, continuous input $\mathbf{u}_k$ and discrete operational mode $x_{d,k+1}$.
- The continuous output $\mathbf{y}_k$, given the continuous state $\mathbf{x}_k$, continuous input $\mathbf{u}_k$ and discrete operational mode $x_{d,k}$.

Whereas the transitions in the non-deterministic automaton model are determined in terms of qualitative values that specify

- the current discrete state $x_{d,k}$
- the current input commands $u_{d,k}$
- the assumed next discrete state $x_{d,k+1}$
- the assumed discrete outputs $y_{d,k+1}$
- a polytopic region
$$\mathbf{H}_{x1}\mathbf{x}_k \leq \mathbf{k}_{x1}$$
the current continuous state $\mathbf{x}_k$ is in
- a polytopic region
$$\mathbf{H}_u\mathbf{u}_k \leq \mathbf{k}_u$$
the current continuous inputs $\mathbf{u}_k$ are in
- a polytopic region
$$\mathbf{H}_y\mathbf{y}_k \leq \mathbf{k}_y$$
the current continuous outputs $\mathbf{y}_k$ are assumed to be in
- a polytopic region
$$\mathbf{H}_{x2}\mathbf{x}_{k+1} \leq \mathbf{k}_{x2}$$
the next continuous state $\mathbf{x}_{k+1}$ is assumed to be in.

From both these specifications, various likelihood values have to be determined:

- Likelihood value p_d specifies the mode-transition likelihood imposed by the hybrid model itself and can directly be determined from the model's transition specification T.

- Likelihood value L_{c1} compares the size of regions of states $\mathbf{x}_k$ and inputs $\mathbf{u}_k$ that actually lead to states and outputs specified by

$$\mathbf{H}_{x2}\mathbf{x}_{k+1} \leq \mathbf{K}_{x2} \quad \text{and} \quad \mathbf{H}_y\mathbf{y}_k \leq \mathbf{K}_y$$

  to the size of the regions specified by the polytopes

$$\mathbf{H}_{x1}\mathbf{x}_k \leq \mathbf{K}_{x1} \quad \text{and} \quad \mathbf{H}_u\mathbf{u}_k \leq \mathbf{K}_u$$

- Likelihood value L_{c2} compares the region of states $\mathbf{x}_{k+1}$ that actually can be reached by trajectories that satisfy the given qualitative abstraction to all those specified by

$$\mathbf{H}_{x2}\mathbf{x}_{k+1} \leq \mathbf{K}_{x2}$$

While the first of these likelihood values (c_P) can directly be looked up in the transition specification T of the hybrid model and does not require further calculations, determination of the two others requires a closer look.

Hyper-Volume of a Polytope

First of all, the hyper-volume of a polytope has to be defined. If a polytopic region $\mathbb{P}$ is defined by

$$\mathbb{P}: \quad \mathbf{H}\mathbf{x} \leq \mathbf{k} \tag{A.16}$$

a 'polytope-membership-function' $F_{\mathbb{P}}(\mathbf{x})$ is specified as

$$F_{\mathbb{P}}(\mathbf{x}) = \begin{cases} 1 & \forall\, \mathbf{x} \mid \mathbf{H}\mathbf{x} \leq \mathbf{K} \\ 0 & \text{otherwise} \end{cases} \tag{A.17}$$

With this function and $\mathbf{x} = [x_1, \ldots, x_N]^T$, the hyper-volume $V_{\mathbb{P}}$ of the polytope $\mathbb{P}$ can be calculated by

$$V_{\mathbb{P}} = \int_{x_1=-\infty}^{\infty} \ldots \int_{x_N=-\infty}^{\infty} F_{\mathbb{P}}(\mathbf{x})\ dx_N \ldots dx_1 \tag{A.18}$$

Calculation of the Input-Space Likelihood

The transition-specification of the non-deterministic automaton is only a valid abstraction for trajectories of the hybrid model that satisfy

$$\mathbf{x}_{k+1} = \mathbf{A}_{xdk+1}\mathbf{x}_k + \mathbf{B}_{xdk+1}\mathbf{u}_k + \mathbf{e}_{xdk+1} \tag{A.19a}$$
$$\mathbf{y}_k = \mathbf{C}_{xdk}\mathbf{x}_k + \mathbf{D}_{xdk}\mathbf{u}_k + \mathbf{f}_{xdk}. \tag{A.19b}$$

Additionally, the transition specification determines that

$$\mathbf{H}_{x1}\mathbf{x}_k \leq \mathbf{k}_{x1} \tag{A.20a}$$
$$\mathbf{H}_u\mathbf{u}_k \leq \mathbf{k}_u \tag{A.20b}$$
$$\mathbf{H}_y\mathbf{y}_k \leq \mathbf{k}_y \tag{A.20c}$$
$$\mathbf{H}_{x2}\mathbf{x}_{k+1} \leq \mathbf{k}_{x2} \tag{A.20d}$$

Combining (A.19) and (A.20) leads to

$$\begin{aligned}
\mathbf{H}_{x1}\mathbf{x}_k &\leq \mathbf{k}_{x1} && \text{(A.21a)}\\
\mathbf{H}_u\mathbf{u}_k &\leq \mathbf{k}_u && \text{(A.21b)}\\
\mathbf{H}_y(\mathbf{C}_{xdk}\mathbf{x}_k + \mathbf{D}_{xdk}\mathbf{u}_k + \mathbf{f}_{xdk}) &\leq \mathbf{k}_y && \text{(A.21c)}\\
\mathbf{H}_{x2}(\mathbf{A}_{xdk+1}\mathbf{x}_k + \mathbf{B}_{xdk+1}\mathbf{u}_k + \mathbf{e}_{xdk+1}) &\leq \mathbf{k}_{x2} && \text{(A.21d)}
\end{aligned}$$

what specifies a polytope $\mathbb{P}_{v1}$

$$\mathbb{P}_{v1}: \quad \mathbf{H}_{v1}\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} \leq \mathbf{k}_{v1} \tag{A.22}$$

with

$$\mathbf{H}_{v1} = \begin{bmatrix} \mathbf{H}_{x1} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_u \\ \mathbf{H}_y\mathbf{C}_{xdk} & \mathbf{H}_y\mathbf{D}_{xdk} \\ \mathbf{H}_{x2}\mathbf{A}_{xdk+1} & \mathbf{H}_{x2}\mathbf{B}_{xdk+1} \end{bmatrix} \tag{A.23a}$$

$$\mathbf{k}_{v1} = \begin{bmatrix} \mathbf{k}_{x1} \\ \mathbf{k}_u \\ \mathbf{k}_y - \mathbf{H}_y\mathbf{f}_{xdk} \\ \mathbf{k}_{x2} - \mathbf{H}_{x2}\mathbf{e}_{xdk+1} \end{bmatrix} \tag{A.23b}$$

of valid initial conditions that lead to abstracted trajectories as specified by the transition-specification of the non-deterministic automaton. This polytope is compared to the polytope $\mathbb{P}_{in}$

$$\mathbb{P}_{in}: \quad \mathbf{H}_{in}\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} \leq \mathbf{K}_{in} \tag{A.24}$$

with

$$\mathbf{H}_{in} = \begin{bmatrix} \mathbf{H}_{x1} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_u \end{bmatrix} \tag{A.25a}$$

$$\mathbf{k}_{in} = \begin{bmatrix} \mathbf{k}_{x1} \\ \mathbf{k}_u \end{bmatrix} \tag{A.25b}$$

that covers all initial conditions of the transition specification.

The likelihood value L_{c1} is then calculated as

$$L_{c1} = \frac{V_{\mathbb{P}_{v1}}}{V_{\mathbb{P}_{in}}} \tag{A.26}$$

Calculation of the Output-Space Likelihood

The polytope $\mathbb{P}_{v1}$ (A.22) specifies states $\mathbf{x}_k$ and inputs $\mathbf{u}_k$ so that the resulting trajectories have a qualitative abstraction as given by the considered transition specification of the non-deterministic automaton.

For calculating the output-space likelihood L_{c2} this polytope has to be propagated through the model

$$\mathbf{x}_{k+1} = \mathbf{A}_{xdk+1}\mathbf{x}_k + \mathbf{B}_{xdk+1}\mathbf{u}_k + \mathbf{e}_{xdk+1} \tag{A.27}$$

This results in a polytope $\mathbb{P}_{v2}$ specified by the 'polytope-membership-function'

$$F_{\mathbb{P}_{v2}}(\mathbf{x_{k+1}}) = \begin{cases} 1 & \forall \mathbf{x_{k+1}} \;\Big|\; \mathbf{x}_{k+1} = \mathbf{A}_{xdk+1}\mathbf{x}_k + \mathbf{B}_{xdk+1}\mathbf{u}_k + \mathbf{e}_{xdk+1},\;\; \mathbf{H}_{in}\begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix} < \mathbf{k}_{in} \\ 0 & \text{otherwise} \end{cases} \tag{A.28}$$

This polytope is then compared to the polytope $\mathbb{P}_{out}$

$$\mathbb{P}_{out}: \quad \mathbf{H}_{x2}\mathbf{x}_{k+1} < \mathbf{k}_{x2} \tag{A.29}$$

that covers all continuous states $\mathbf{x}_{k+1}$ of the transition specification.

The likelihood value L_{c2} is then calculated as

$$L_{c2} = \frac{V_{\mathbb{P}_{v2}}}{V_{\mathbb{P}_{out}}} \tag{A.30}$$

Example

As example, likelihood values L_{c1} and L_{c2} for the transition specification

$$\begin{aligned} \mathbf{H}_{x1} = \mathbf{H}_u = \mathbf{H}_y = \mathbf{H}_{x2} &= \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ \mathbf{K}_{x1} = \mathbf{K}_u = \mathbf{K}_y = \mathbf{K}_{x2} &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ x_{k+1} &= 1 \cdot x_k + 2 \cdot u_k \\ y_k &= 1 \cdot x_k - 1 \cdot u_k - 0.1 \end{aligned}$$

are calculated. Based on this, one straightforwardly calculates the hyper-volumes

$$\begin{aligned} V_{\mathbb{P}_{in}} &= 1 \\ V_{\mathbb{P}_{out}} &= 1 \end{aligned}$$

The valid input-space is determined by the polytope

$$\mathbb{P}_{v1}: \quad \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 2 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1.1 \\ -0.1 \\ 1 \\ 0 \end{bmatrix}$$

which after elimination of redundant inequalities is expressed as

$$\mathbb{P}_{v1}: \quad \begin{bmatrix} 0 & -1 \\ -1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \leq \begin{bmatrix} 0 \\ -0.1 \\ 1 \end{bmatrix}.$$

Its hyper-volume is

$$V_{\mathbb{P}_{v1}} = 0.135$$

leading to a likelihood value

$$L_{c1} = \frac{V_{\mathbb{P}_{v1}}}{V_{\mathbb{P}_{in}}} = 0.135$$

States x_k in polytope $\mathbb{P}_{v1}$ are propagated through the model

$$x_{k+1} = 1 \cdot x_k + 2 \cdot u_k$$

and lead to states x_{k+1} in the polytopic region

$$\mathbb{P}_{v2}: \quad \begin{bmatrix} 1 \\ -1 \end{bmatrix} x_{k+1} \leq \begin{bmatrix} 1 \\ -0.1 \end{bmatrix}$$

which contains a hyper-volume of

$$V_{\mathbb{P}_{v2}} = 0.9.$$

This gives a likelihood-value

$$L_{c1} = \frac{V_{\mathbb{P}_{v2}}}{V_{\mathbb{P}_{out}}} = 0.9$$

Illustration for these calculations is provided in Figure A.1.

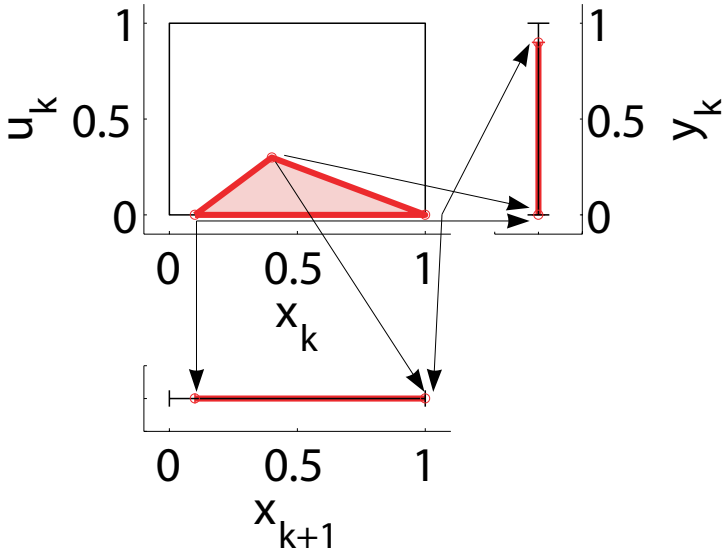


Figure A.1: Calculation of likelihood values

Approximative Computation of Hyper-Volumes

While calculation of hyper-volumes of the polytopes in the above example is accomplished fairly easy, this can be a computationally more intensive task for arbitrary polytopes of higher dimensionality. Therefore, it is sometimes advisable to perform these computations in approximative manner.

Utilization of approximations seems further justified, as not even the exactly calculated values would be used in the qualitative model as they are combined into likelihood classes anyway.

Bounding Box Approximation

The idea is to approximate arbitrary polytopes

$$\mathbb{P}: \quad \mathbf{Hx} < \mathbf{K}$$

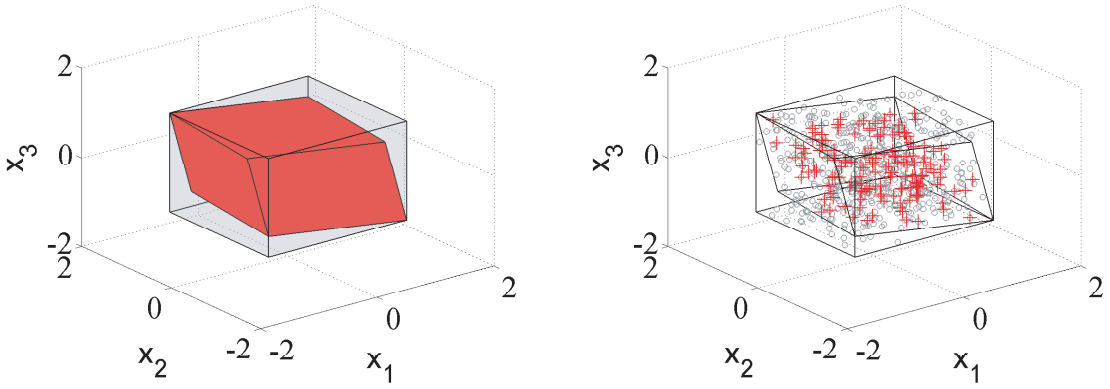
by the bounding box

$$\mathbb{B}(\mathbb{P}): \quad \begin{bmatrix} \mathbf{I} \\ -\mathbf{I} \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} \mathbf{max} \\ -\mathbf{min} \end{bmatrix} \tag{A.31}$$

where $\mathbf{I}$ represents the identity matrix and **max** is the vector of maximum Cartesian coordinates of the polytope and **min** is the vector of minimum coordinates (Figure A.2a).

The hyper-volume of the bounding box is then easily evaluated as

$$V_{\mathbb{B}(\mathbb{P})} = \prod (\mathbf{max} - \mathbf{min}) \tag{A.32}$$



(a) Bounding box approximation

(b) Estimation of the hyper-volume by sampling

Figure A.2: Approximations of a polytope's hyper-volume

Approximation by sampling

A more accurate but computationally more intensive approximation of the hyper-volume of a polytope can be achieved by sampling. The idea is to draw a number of N uniformly distributed sample-points $\mathbf{x}$ from the bounding box (with known hyper-volume $V_{B(P)}$. Next, these samples are checked against the inequalities

$$\mathbf{Hx} \leq \mathbf{K}$$

describing the polytope one is interested in and the number N_0 of sample points that satisfy the inequalities is evaluated (Figure A.2b).

The hyper-volume of the the polytope can then be approximated by

$$V_{\mathbb{P}} \approx V_{\mathbb{B}(\mathbb{P})} \frac{N_0}{N} \tag{A.33}$$

A.3 Causal Analysis

Causal analysis is an algorithmic way to determine the interdependencies of variables from a given set of 'equations' or 'relations' and to display these as directed graph.

In terms of the utilized hybrid models, these can be

- Equations with specified dependent variable, e.g.

$$x_{k+1} = a \cdot x_k + b \cdot u_k$$

- Discrete transition specifications with specified 'dependent variable'
- Algebraic constraints with unspecified 'dependent variable', e.g.

$$a + b = 0$$

- exogenous input specifications

The first type of equations is the easiest one to handle. These equations can either describe continuous dynamics of a component model

$$\mathbf{x}_{k+1} = \mathbf{A}_i \mathbf{x}_k + \mathbf{B}_i \mathbf{u}_k + \mathbf{e}_i$$

or specify component outputs

$$\mathbf{y}_k = \mathbf{C}_i \mathbf{x}_k + \mathbf{D}_i \mathbf{u}_k + \mathbf{f}_i.$$

For these equations, the dependent variables $(\mathbf{x}_{k+1}, \mathbf{y}_k)$ are directly identified and one only has to determine the independent ones. These can – with $\mathbf{x} = [x_1, \ldots, x_N]^T$ and other vectors specified accordingly – be

- the variables $x_{i,k}$, if the corresponding columns of matrix $\mathbf{A}$ (or $\mathbf{C}$, respectively) are not equal to zero in any operational mode
- the variables $u_{i,k}$, if the same holds with respect to $\mathbf{B}$ or $\mathbf{D}$, respectively
- the operational mode $x_{d,k+1}$, if any of $\mathbf{A}, \mathbf{B}, \mathbf{e}$ depend on the operational mode
- the operational mode $x_{d,k}$, if any of $\mathbf{C}, \mathbf{D}, \mathbf{f}$ depend on the operational mode

The second type can be treated similarly. The 'next' operational mode $x_{d,k+1}$ is always the dependent variable, whereas independent variables can be

- the discrete input $u_{d,k}$, if any transition guard contains a boolean condition on $u_{d,k}$.
- the continuous state $x_{i,k}$ if any transition guard contains a polytope-constraint

$$\mathbf{H}\mathbf{x}_k \leq \mathbf{K}$$

  and the corresponding column of $\mathbf{H}$ is not zero

- the continuous inputs $u_{i,k}$ if the equivalent holds for polytope-guards

$$\mathbf{H}\mathbf{u}_k \leq \mathbf{K}$$

- the current discrete state $x_{d,k}$, if the transition specifications are not the same for all 'source modes' $x_{d,k}$

The third type is somewhat more difficult to treat, as the dependent variable has to be determined by the structure imposed by other equations and exogenous inputs. These equations occur in a more general formulation of hybrid models, when inputs and outputs are not specified explicitly, but are only commonly treated as set of terminal variables

$$\mathbf{w} = [w_i, \ldots, w_N]^T$$

and input/output relations are specified as

$$\mathbf{f}_i = \mathbf{C}_i \mathbf{x}_k + \mathbf{D}_i \mathbf{w}_k.$$

With such equations, one cannot directly determine a dependent variable, but has to preliminarily indicate all variables that contribute to each equation as independent.

However, it is known from the concurrent hybrid automaton specification which variables are exogenous inputs. These are different in that they are not determined by any equation.

Once, all equations and respective variables are identified, the structure imposed by the set of equations can be evaluated. It only has to be noticed, that the same variable at different times, (i.e. $x_{1,0}, x_{1,1}, x_{1,2}, \ldots$) is treated as distinct variables in the following process.

Find Dependent Variables

If there are some equations that do not have dedicated dependent variables, these can be determined by the structure imposed by the overall set of equations through the following algorithmic procedure.

First, a so-called bipartite graph has to be built that has one side with a node for each equation and one side for with a node for each variable that occurs in any of the equations.

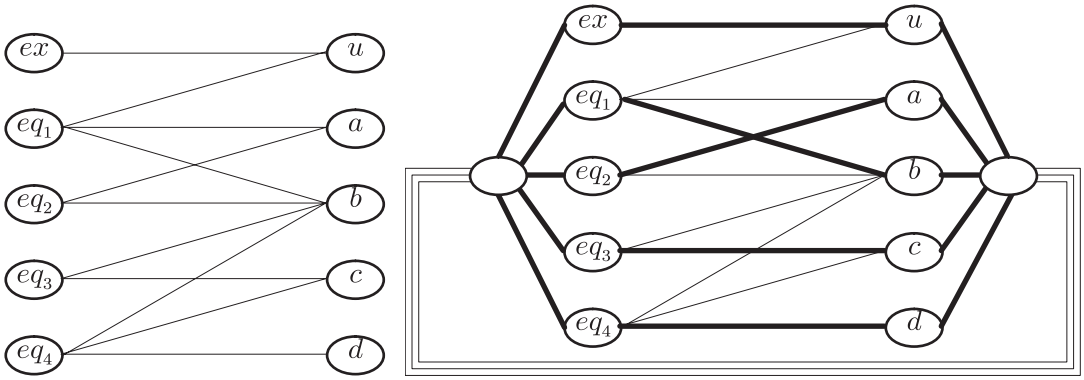
Each equation node is then connected to its dedicated dependent variable *or* to all preliminarily independent variables, if there is no dedicated dependent one. As exogenous inputs are clearly not dependent in any equation they get an extra equation node each, which they are connected to. This prevents that they will get dedicated 'dependent' in any other equation in the following process.

An example of such a graph that represents equations

$$\begin{aligned} eq_1: &\quad 0 = \quad f(a,b,u) \\ eq_2: &\quad 0 = \quad f(a,b) \\ eq_3: &\quad 0 = \quad f(b,c) \\ eq_4: &\quad 0 = \quad f(b,c,d) \\ ex: &\quad u \end{aligned}$$

is displayed in Figure A.3a. To determine a single dependent variable for each equation, a set of arcs of the graph has to be identified, such that each equation-node on the left side is connected to a single variable node on the right side and vice versa.

This task is accomplished by formulating it as maximum-flow optimization problem for a network as shown in Figure A.3b, where each arc except the big feedback-line can carry exactly one amount of flow. More detailed treatment of this causal analysis and algorithms that solve this tasks are found in [23, 44]. The flow in the network then determines the dependent variable of each equation.



(a) Bipartite Graph relating variables and equations

(b) Solution by Flow-Maximization

Figure A.3: Detection of dependent variables in a set of equations

This bipartite matching procedure additionally can detect structural deficiencies in the set of equations, e.g. conflicting situations where variables would be determined by several equations or by non at all [33]. In this work it is assumed that the hybrid automata are specified correctly and no such deficiencies exist.

For the example, a solution is displayed in Figure A.3. However, a careful look at this figure shows another possible solution that relates variable a to equation eq_1 and relates variable b to equation eq_2. Such ambiguities in the solution indicate circular interdependencies among the variables, so-called algebraic loops. The loop in this example will display even more explicitly, when the causal graph is constructed.

Causal Graph

The so-called causal graph displaying the interdependencies among the variables imposed by a set of equations can now directly be constructed by the following rules:

- create a node for the dependent variable of each equation (including the exogenous inputs)
- for each equation, draw a directed edge from all independent variables of the equation to the dependent one

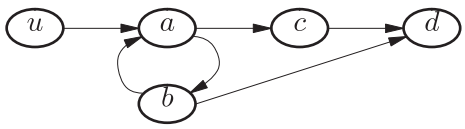


Figure A.4: Causal graph

The graph for the above example is displayed in Figure A.4. It is clear to see that there is a loop among variables a and b, which tells that a depends on b depends on a... In more complex graphs, such circular interdependencies can be detected together with the involved variables by the algorithmic detection of 'strongly connected components' [23]. Such an algebraic loop and can cause severe problems in the proposed control framework. So throughout this work systems with acyclic causal graphs have to be assumed.

A.4 Binary Decision Diagrams

Binary Decision Diagrams (BDDs) [3, 14] provide a compact encoding of boolean functions as directed acyclic graphs (DAGs).

The three main reasons – illustrated below – to select BDD-like trajectory-graphs to represent qualitative models are

- BDDs can be utilized to represent sets of particular valuations among symbolic variables as DAG.
- BDDs provide a very compact encoding of boolean functions [14].
- BDDs can be generated based on a pre-specified ordering of variables that is represented as hierarchy in the DAG

Boolean Representation of Valuations among Symbolic Variables

A boolean function is a logical expression among variables that can take values `1` (`true`) or `0` (`false`). The function itself evaluates to one of these values, depending on the valuation of the variables. These function-values depending on the valuations of the variables can be expressed in terms of a truth-table that lists an enumeration of all possible valuations of the variables and associates the corresponding function-value to each valuation.

Similarly, a set of valuations among symbolic variables can be represented as truth-table, if each symbolic value of a variable is uniquely encoded by boolean variables. All valuations among all the symbolic variables that are included in the set receive a boolean function value of `1`, all others receive `0`.

An example is used to further on support understanding of BDD generation and compactness by illustration:

set of sympolic valuations

variable:	R	S
valuations:	r_0	s_0
	r_0	s_1
	r_2	s_0
	r_3	s_0

(A.34)

representation by boolean expressions

R	B_1	B_2
r_0	`0`	`0`
r_1	`0`	`1`
r_2	`1`	`0`
r_3	`1`	`1`

S	B_3
s_0	`0`
s_1	`1`

(A.35)

associated truth table

B_1	B_2	B_3	function value
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(A.36)

First Simple BDD Construction

A BDD represents such a truth-table as DAG, where non-terminal vertices represent boolean variables and terminal nodes represent function-values.

The graph is constructed based on a pre-specified ordering of variables. The first of the ordered variables corresponds to the root node of the BDD and there is a directed edge leading from this node for each of the variable's valuations $(0, 1)$.

Each of the edges leads to one of two other nodes, each representing the variable that is second in ordering. This way, graph construction is continued until the nodes representing the last variable in ordering. The edges leaving the corresponding nodes then are connected to respective new terminal nodes that are labeled with the boolean function values. These function values are determinated from the truth table or boolean function by evaluating the valuation of boolean variables specified by the arcs along the path from the root node of the graph down to the terminal node.

This first simple construction of a BDD for the above example and a variable ordering

$$B_1 \prec B_2 \prec B_3$$

that determines variable B_1 to be ordered first is displayed in Figure A.5. The edges and terminal nodes are labeled with the corresponding boolean values. Additionally, these values are indicated by solid lines for boolean value 1 and dotted lines for boolean value 0.

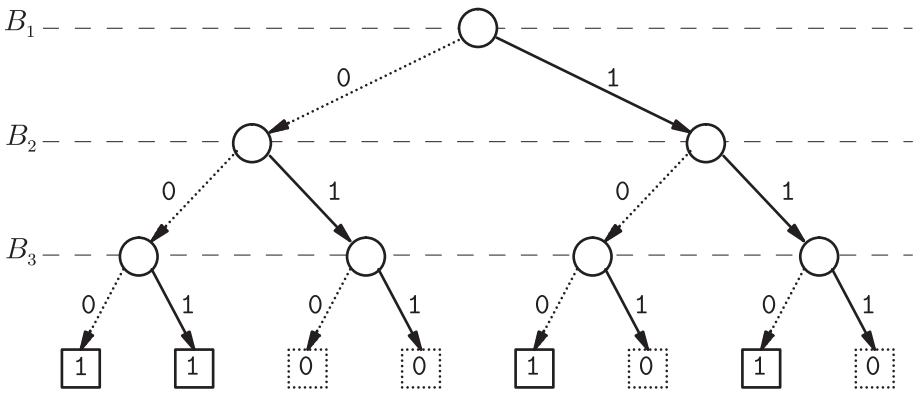


Figure A.5: Graphical representation of a boolean function

Reduction of Binary Decision Diagrams

This graphical representation of a boolean function can, now, significantly be compacted. This is achieved by three different operations.

The first step is to eliminate duplicate terminals. These are terminal nodes that represent the same truth value. To eliminate the duplicates, all edges leading to a duplicate terminal are redirected to one single node and all other duplicates of this node are removed. This operation itself does not provide much compaction, however it paves the way for the two further ones.

The result of this operation on the terminal nodes of the graph is shown for the example in Figure A.6.

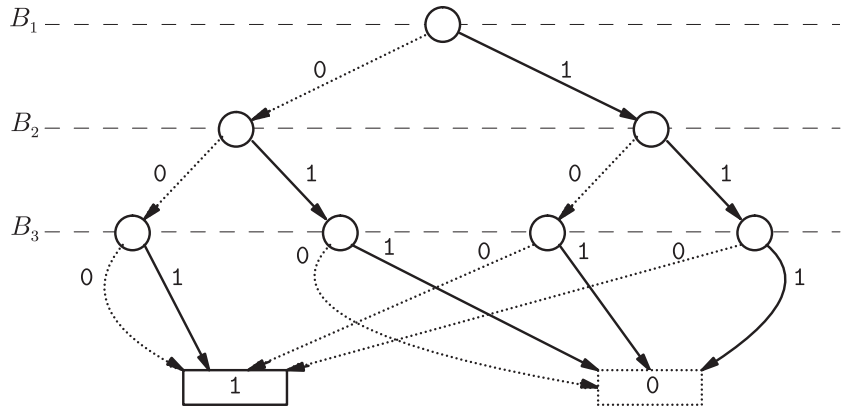


Figure A.6: Elimination of duplicate terminals

The second operation to reduce size of the BDD is the elimination of redundant tests in the graph. These are indicated by nodes that have both, their `0`-edge and their `1`-edge pointing to the same node in the graph. Such nodes indicate that the value of the variable they are representing is not relevant for particular parts of the boolean function. All edges that lead to such a node are redirected to the node's only child-node and the node and its two edges are removed from the graph.

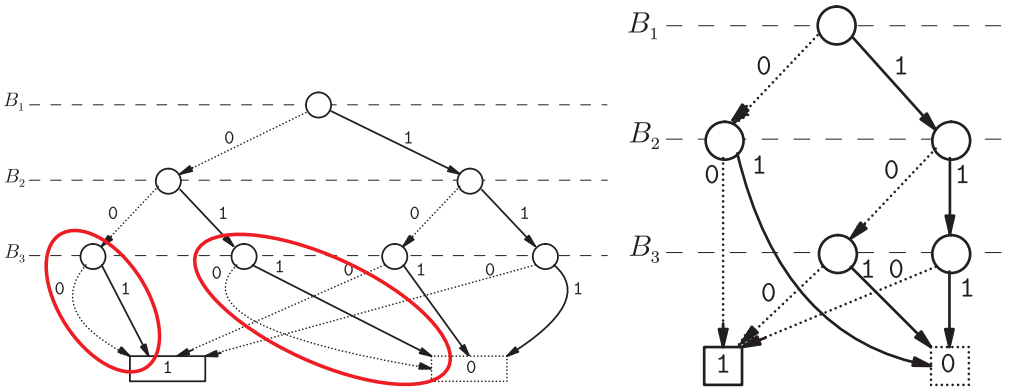
This operation is illustrated for the example in Figure A.7. Figure A.7a shows that elimination of duplicate terminals made obvious two redundant tests in the boolean function and Figure A.7b displays the BDD after the reduction.

The third operation eliminates duplicate non-terminals in the graph. These are nodes that represent the same variable and, all, have their `0`-edges leading to one common node and their `1`-edges commonly leading one other node. All edges pointing to one of these duplicate non-terminals are redirected to a single one thereof and the other duplicates, together with all their outbound edges, are removed from the graph.

The reduced example graph shows two such duplicates in Figure A.8a. The result of the reduction operation is displayed in Figure A.8b.

These operations are repeated until no further reductions are possible and the final BDD is obtained. The final reduced BDD for the example is given in Figure A.9.

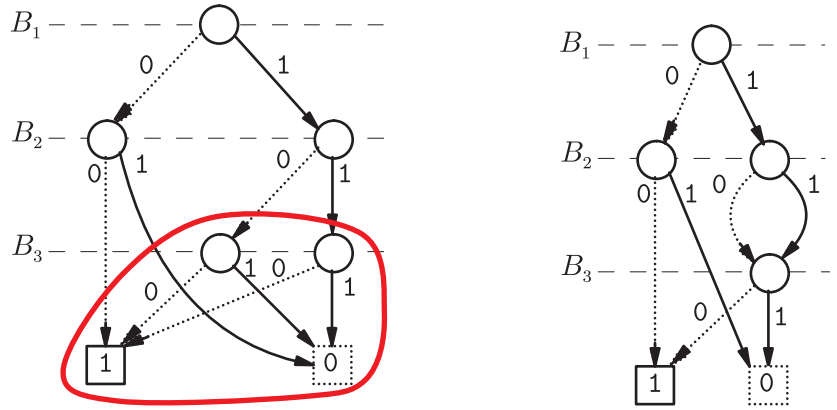
Usually, such a reduced BDD is a very compact representation of the original boolean function. However, the size of this graph largely depends on the specified ordering of variables. Computing an optimal ordering of variables for a given boolean function is a demanding task, but [19] points out that it is usually a good heuristic to order 'dependent' variables near each other.



(a) Graph indicating two redundant tests

(b) BDD after elimination of the redundant tests

Figure A.7: Elimination of redundant tests in BDDs



(a) Graph indicating two duplicate non-terminals

(b) BDD after elimination of the duplicates

Figure A.8: Elimination of duplicate non-terminals

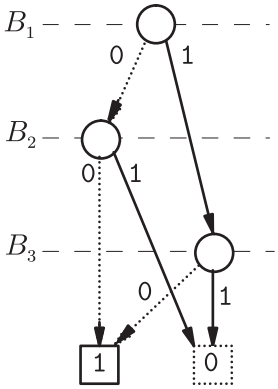


Figure A.9: Reduced Binary Decision Diagram

Appendix B

Algorithms

B.1 Ordering of Variables

The algorithms presented here are not tailored to increase their efficiency but just outlined to provide understanding of their operation. This is justified as they are only utilized for off-line compilation and not for time-critical online calculations.

It has to be noticed that in this appendix the term 'variable' will also be used for a node that represents the variable in a graph.

Algorithm: `OrderedList`← **OrderVariables**(`Graph`,`Variables`,`Initials`,`Inputs`)

Input:
Acyclic causal graph (i.e. structure describing dependencies among all variables)
Set of all variables
Set of all initial variables
Set of all external input variables

Output: Ordered list of all variables

Description: The algorithm generates an ordered list for a set of variables by utilizing a graph that represents interdependencies among the variables. It utilizes a heuristic that allows the respective next 'comparison of partial assignment' in the qualitative pre-selection procedure as soon as possible

```
OrderedList←{}
Known← Initials
Remaining← setdiff(Variables,Known)
while not empty(Remaining)
    NextVar ← GetNextVariable(Graph,Variables,Known,Inputs)
    OrderedList ← append(OrderedList,NextVar)
    Known ← append(Known,NextVar)
    Remaining ← setdiff(Remaining,NextVar)
end
```

Algorithm: `NextVar`← **GetNextVariable**(`Graph`,`Variables`,`Known`,`Inputs`)

Input:
Acyclic causal graph (i.e. structure describing dependencies among all variables)
Set of all variables
Set of all initial variables and variables already put into ordering
Set of all external input variables

Output: Next variable(s) to be added to the ordered list of all variables

Description: The algorithm utilizes the interdependencies imposed by the graph to determine the next variable in ordering. It uses the following heuristic:

Out of the set of all known variables, the graph's structure probably imposes that only a subset of these variables (plus the inputs) is needed to determine all other variables.

Further, partial assignments in qualitative search can be compared, if after specifying a new variable some other variable falls out of this set of 'needed variables'.

The strategy is now – for all these variables that are 'needed' – to evaluate the additional variables that are required to be determined until this particular variable falls out of the 'needed' set (i.e. the 'additionally needed' variables). As a criterion which of the 'needed' variables, then, is attempted to be removed from the 'needed' set, the following heuristic is used: Choose that variable for which the associated 'additionally needed' set has a minimal product of domain-sizes. The domain-size of a variable is the number of possible distinct valuations.

If the variable is chosen, one variable out of the associated needed set that is already specified by the 'known' variables (i.e. a variable that has no or only 'known' parents in the graph) is selected as next variable in the ordering.

The algorithm uses some sub-functions on the graph that are only briefly explained here:

`Outputs`←**GetSpecifiedOutputs**(`Graph`,`Known`)
Returns all children of know variables that themselves are no know variables and have no children

`Reachable`←**GetReachable**(`Graph`,`Known`,`Inputs`)
Determines all variables that are located along any path of directed edges starting at any of the known variables or inputs

`Influences`←**GetInfluences**(`Graph`,`Variables`)
Determines all variables that are located along any path of directed edges that ends at the specified variables

`Children`←**GetChildren**(`Graph`,`variable`)
Returns the children of a variable in the graph

Code:

```
if empty(GetSpecifiedOutputs(Graph,Known))
    Undetermined←setdiff(Variables,Known)
    Influences←GetInfluences(Graph,Undetermined)
    Needed←intersect(Known,Influences)
    Test←Needed
    bestValue← ∞
    while not empty(Test)
        v←extractAvariable(Test)
        C←GetChildren(Graph,v)
        Required←GetInfluences(Graph,C)
        AdditionallyNeeded←setdiff(Required,Known)
        value←product(domainSize(Additionally Needed))
        if value¡bestValue
            bestValue←value
            bestTest←v
            bestAdditional←Additionally Needed
        end
        Test←setdiff(Test,v)
    end
    NewVar←extractAvariable(bestAdditional)
else
    NewVar←GetSpecifiedOutputs(Graph,Known)
end
```

B.2 Compilation of tDAGs

Although trajectory graphs are based on Binary Decision Diagrams (BDDs) and can be constructed and manipulated very similarly, they differ in that they do not use exactly two terminal nodes representing boolean values `true` and `false`, but allow an arbitrary number of terminal nodes representing transition likelihood costs.

In principle, construction of the trajectory DAGs could be accomplished by the same basic procedure illustrated in appendix A.4. One could start with a tree-structure to represent the set of qualitative transitions – encoded as boolean truth table – and associated likelihood costs and then perform the reduction steps as presented in the appendix.

The algorithm presented here is an alternative that directly constructs tDAGs in their reduced from the specification of transition likelihoods.

Algorithm:
`tDAG` ← **Trans2TDAG**(`Transitions`,`LikelihoodCosts`,`OrderingIndex`,`Domains`)

Input:
A matrix of transition specifications, where each line represents a single transition specification and the columns represent the qualitative variables
A vector of the likelihood-costs associated to the lines of the transition-matrix
A vector of position indices associated to the columns of the matrix. This vector specifies the qualitative variables' positions in the common ordering of variables pre-specified for the overall qualitative model.
The list of possible valuations for each variable

Output: A compact, directed acyclic graph encoding the set of qualitative transition specifications. Apart from the terminal node, this graph consists of nodes that represent the qualitative variables Each of the nodes is left by a number of edges that represent particular values of the variable. Additionally, each edge is labeled with a cost value, such that the sum of edge costs along each directed path from the root node of the graph to one of its terminals matches this terminal's associated cost value.

Description: Compilation of this graph is a 3-step procedure:

- Determination of a binary representation of the transition specification
- Compilation of a Binary Decision Diagram like directed acyclic graph that represents this binary transition specification
- Compilation of a representation of this graph, that, again uses the original multi-valued qualitative variables and associated qualitative values

These steps are handled by different algorithms as follows:

`BIndex` ← **Index2BIndex**(`OrderingIndex`,`Domains`)
`BTransitions` ← **Symbol2Binary**(`Transitions`,`Domains`)
`BDAG` ← **Trans2DAG**(`BTransitions`,`LikelihoodCosts`,`BIndex`)
`tDAG` ← **BDAG2tDAG**(`BDAG`,`Domains`)

Algorithm:
`BIndex` ← **Index2BIndex**(`OrderingIndex`,`Domains`)

Input:
Vector of indices of qualitative variables corresponding to the columns of the transition matrix in the common, pre-specified ordering of variables of the overall model
Domains of the qualitative variables to decide on the necessary binary variables

Output: Vector of index numbers for the binary representations

Description: The necessary number of binary variables to represent a multi-valued symbolic variable with domain-size D is

$$\lceil \log_2(D) \rceil.$$

Accordingly, index values are adapted by inserting new values between the original index value and the next integer value

```
BIndex←{}
for i from 1 to length(OrderingIndex)
    idx←OrderingIndex(i)
    B← ⌈log2(length(Domains(i))⌉
    for plus from 0 to B-1
        BIndex←append(BIndex,idx+plus/B
    end
end
```

Algorithm:
`BTransitions` ← **Symbol2Binary**(`Transitions`,`Domains`)

Input:
Matrix of transition specifications, where each line represents a single transition specification and the columns represent the qualitative variables
Domains of the qualitative variables as a list of qualitative symbols for each qualitative variable

Output:
Matrix of transition specifications, where the columns represent the binary variables used for encoding the qualitative ones

Description: The columns of the transition matrix are adapted to the binary-variable-representation by encoding the list-index of each symbol as binary number. For this, the convention is used that the variable with the lowest index (by the definition of algorithm 'Symbol2Binary' this is corresponds to the 'leftmost' column in the binary representation of a single column of the transition matrix) is the 'most significant bit' (MSB).

```
BTransitions←{}
for col from 1 to NumberOfColumns(Transitions)
   Bcolumns←{}
   for row from 1 to NumberOfLines(Transitions)
      Symbol←Transitions(row,col)
      BinaryCode←(BinaryNumberOf(IndexInListOf(Domain(col),Symbol)
      Bcolumns←AddLineBelow(Bcolumns,BinaryCode)
   end    BTransitions←AddColumnsOnRight(BTransitions,Bcolumns)
end
```

Algorithm:
BDAG ← **Trans2DAG**(BTransitions,LikelihoodCosts,BIndex)

Input:
Binary valued matrix of transition specifications, where each line represents a single transition specification and the columns represent the binary variables used to encode the qualitative symbols
Vector of likelihood costs; each element is associated to the corresponding line of the transition matrix
Vector of (introduced) ordering-indices corresponding to the columns of the transition matrix

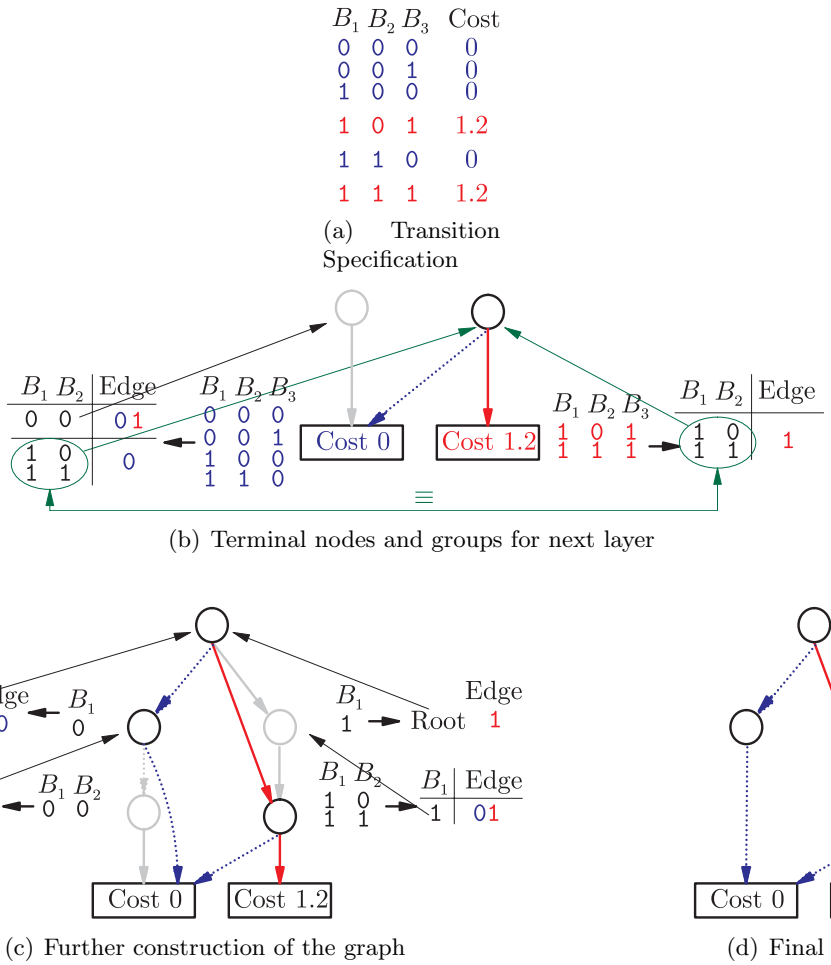
Output:
BDD-like encoding of the transition specification.

Description: This algorithm constructs a BDD-like directed acyclic graph representing the binary transition specifications. For this, the transition specification together with the likelihood values is treated similarly to a boolean truth-table.

Direct construction of the graph starts from the terminal nodes. The algorithm then determines the necessary parent nodes for these and connects them to the terminals. Then graph construction is resumed from these parent nodes to their respective parents, until the root node is reached. An illustrative example for this is provided below. (Figure B.1)

```
BDAG←CreateANodeForEachDistinctTerminalValue
for all terminal nodes
   lines←FindLinesWithTheAssociatedCostValue(LikelihoodCosts)
   AddAsExpansionInfoTo(BDAG(the terminal node),LikelihoodCosts(lines,all columns)
   AddAsIndexInfo(BDAG(the terminal node),∞)
end
LayerCounter←length(BIndex)+1 while LayerCounter¿0
LayerCounter←LayerCounter-1    for all nodes in the topmost layer
      CombinedExpansionMatrix←VerticalConcentation(ExpansionInfo(NodesInLayer))
   end    PossibleNewNodes←UnionOfLines(CombinedExpensionMatrix(all lines,all-but-
last columns))
   Group these PossibleNewNodes with respect to outbound edges an targets
      %example: a line in PossibleNewNodes is 010 and a node in the current topmost
```

```
        %   layer has an ExpansionInfo that includes the line 0100, this tells that the
        %   possible node is connected to the target node (because of the common 010
        %   by its 0-edge (because of the 0 in the last column)
    for all identified groups          AddNewNodeTo(BDAG,TheGroup)
        AddAsExpansionInfoTo(BDAG(TheGroup),PossibleNewNodes(TheGroup))
        AddAsIndexInfo(BDAG(TheGroup),BIndex(LayerCounter))
        if both edges of the group point to the same child node
            AddInfo'IsTmp'To(BDAG(TheGroup))
        end
        AddDirektedEdges(BDAG(TheGroup),Targets(TheGroup)), however
        if one of the targets is marked as 'IsTmp'
            AddDirektedEdge(BDAG(TheGroup),Child(Target(TheGroup))) instead
        end
    end
    Remove all nodes marked 'IsTmp' from the graph, except those of the topmost layer
end
```



(a) Transition Specification

(b) Terminal nodes and groups for next layer

(c) Further construction of the graph

(d) Final graph

Figure B.1: Illustration of the 'Trans2TDAG' algorithm

Algorithm:
`tDAG` ← **BDAG2tDAG**(`BDAG`,`Domains`)

Input:
Directed acyclic graph (DAG) that represents transition specifications by utilizing binary representations for the symbolic values

Output:
DAG that represents the same transition specifications but uses the original symbolic values

Description: This algorithm removes the binary representation of symbolic values that was needed for BDD generation. Additionally, it maps the cost values associated to the terminal nodes to the graphs edges, such that the sum of edge costs along each directed path from the root node of the graph to one of its terminals matches this terminal's associated cost value.

```
tDAG←BDAG(RootNode)
NodeList←BDAG(RootNode)
while not emptyNodeList
    node←NodeList(first)
    var←GetVariableOf(BDAG(node))
    symbols←Domains(var)
    for all symbols
        sym←the next symbol
        BinarySequence←Symbol2Binary(sym)
        if it is possible to follow a path according to BinarySequence from node
                % Notice: check the levels of the node along the path, as BinarySequence does
                %    not specify that exactly the sequence of edges has to be taken, but that
                %    at particular index-levels a particular edge has to be taken
            target←TargetNodeOfTheSpecifiedPath
            NodeList←union(NodeList,target)
            if not IsIn(tDAG,target)
                AddNodeTo(tDAG,target)
            end
            AddEdge(tDAG(node),tDAG(target));
            AddEdgeLabel(tDAG(node),tDAG(target),sym);
        end
    end
    NodeList←setdiff(NodeList, node);
end
Recursively assign to each node of tDAG the minimum likelihood costs of its children
for all edges in tDAG
    AssignCostValueToEdge(Cost(destination node)−Cost(source node)
        % This works because one terminal node always has value 0
end
```
