Towards Coordination and Control of Multi-robot Systems

Ph.D. Thesis

Michael Melholt Quottrup

Automation & Control
Department of Electronic Systems
Aalborg University
Fredrik Bajers Vej 7C, DK-9220 Aalborg Ø, Denmark

Towards Coordination and Control of Multi-robot Systems

Ph.D. Thesis

May, 2007
ISBN 87-90664-31-0

This thesis was typeset using the $\LaTeX\,2_\varepsilon$ in `report` document class.

Figures were made in PSTricks and MATLAB®.

Simulations were performed in MATLAB®, SIMULINK®, and STATEFLOW® from The MathWorks Inc.

Model checking was performed in UPPAAL developed in collaboration between the Department of Information Technology at Uppsala University, Sweden and the Department of Computer Science at Aalborg University, Denmark.

Preface

This thesis is submitted in partial fulfillment of the requirements for the Doctor of Philosophy at Automation and Control, Department of Electronic Systems, Aalborg University, Denmark. The work has been carried out in the period from September 2002 to May 2007 under the supervision of Prof. Thomas Bak and Associate Prof. Roozbeh Izadi-Zamanabadi.

The thesis considers the coordination and control of mobile multi-robot systems.

The thesis is mainly a theoretical approach focusing on automated strategies for the coordination and control of mobile multi-robot systems using formal models and model checking techniques.

May 2007, Aalborg, Denmark
Michael Melholt Quottrup

Acknowledgement

A special thanks to Prof. T. John Koo for welcoming me to the Department of Electrical Engineering and Computer Science, Institute for Software Integrated Systems, Vanderbilt University, U.S. in the spring 2004 as a visiting researcher. Thanks for all the fruitful discussions regarding hybrid systems. The work of Prof. T. John Koo has been a great inspiration for most of the work presented in this thesis. During my stay at Vanderbilt University i have gained an insight in the various application areas of hybrid systems theory.

I am sincerely thankful for the unique opportunity to participate in the initial design phase and construction of the Vanderbilt Embedded Computing Platform for Autonomous Vehicles (VECPAV). Hopefully, the VECPAV will become the state of the art in the development and testing of coordinated strategies for automated vehicles. Also, thanks to the members at the Embedded Computing Systems Laboratory (ECSL) Abhishek Dubey, Jie Chen, Hang Su, and Xianbin Wu. Especially, thanks to Hang Su, Xianbin Wu, and Lewis F. Saettel for invaluable support in the construction of the VECPAV.

Further, thanks to Prof. Anders P. Ravn and Prof. Kim G. Larsen, at the Department of Computer Science, Aalborg University for invaluable discussions regarding timed automata and UPPAAL.

Thanks also goes to the staff at Automation & Control, Aalborg University for always making a nice social atmosphere.

The Ph.D. study is supported by the Faculty of Engineering & Science under grant no. 562/06-8-22565.

Finally, I would like to thank my girlfriend Kikki Henssel and my family for always believing in me and moral support.

Abstract

This thesis focuses on control and coordination of mobile multi-robot systems (MRS). In recent years there has been an increasing interest in systems comprised of several autonomous mobile robots. MRS can often deal with tasks that are difficult, if not impossible, to be accomplished by a single robot. In the context of MRS, one of the challenges is the need to control, coordinate and synchronize the operation of several robots to perform some specified task. This calls for new strategies and methods which allow the desired system behavior to be specified in a formal and succinct way. Temporal logics provide a formal requirement specification mechanism with the capability to define desired behaviors quantitatively due to its similarity to natural languages.

Two different frameworks for the coordination and control of MRS have been investigated.

Framework I - A network of robots is modeled as a network of multi-modal hybrid automata. A finite bisimilar quotient for each of the robots in the network is computed by forming a partition of the environment. Constructing a timed automaton, one for each robot in the network, from the finite bisimilar quotient allows coordination among the robots and timing constraints to be considered. The model checker UPPAAL is used for formal symbolic model checking against a requirement specification formulated in Computational Tree Logic (CTL) for a network of multi-modal robots. The result is a set of motion plans for the robots which satisfy the requirement specification.

Framework II - A framework for controller synthesis for a single robot, modeled as a nonlinear system with respect to requirement specification in Linear-time Temporal Logic (LTL) is presented. Dynamic feed-

back linearization is employed to obtain a linear control system for which a finite bisimilar quotient, in the form of a finite transition system can be computed. The bisimilar quotient is subsequently used for controller synthesis together with the formal specification. Since the quotient and the original linear control system are bisimilar the discrete controller synthesized for the quotient will equivalently work for the linear control system. However, a refinement of the discrete controller is necessary, resulting in a hybrid closed loop.

Resume - Danish Summary

Denne afhandling har fokus på kontrol og koordinering af mobile flerrobot systemer (MRS). I de seneste år har der været en stigende interesse for systemer, der er sammensat af adskillige autonome mobile robotter. MRS kan ofte håndtere opgaver, som kan være vanskelige, hvis ikke umulige, at løse for en enkelt robot. Indenfor MRS er en af udfordringerne behovet for at kunne kontrollere, koordinere og synkronisere operationen af adskillige robotter, for at kunne udføre en specificeret opgave. Dette kræver nye strategier og metoder, som tillader at specificere systemets ønskede opførsel på en formel og præcis måde.

To forskellige konceptuelle strukturer til koordinering og kontrol af MRS er blevet undersøgt.

Konceptuel struktur I - Et netværk af robotter er modelleret som et netværk af hybrid automater. En endelig ækvivalent kvotient er beregnet for hver enkelt robot i netværket ved at opdele robotternes omgivelser. Ved at konstruere en tidsautomat for hver enkelt robot i netværket ud fra den endelige ækvivalent kvotient, gøres det muligt at tage højde for koordinering mellem robotterne samt tidslige krav. Verifikationsværktøjet UPPAAL er brugt til formel symbolsk modelverifikation op imod en kravspecifikation formuleret i Computational Tree Logic (CTL). Resultatet er et sæt bevægelsesplaner for robotterne i netværket, som opfylder kravspecifikationen.

Konceptuel struktur II - Præsenterer en konceptuel struktur til syntese af en regulering for en enkelt robot, der er modelleret som et ulineært system ud fra en kravspecifikation i Linear-time Temporal Logic (LTL). Dynamisk tilbagekoblinglinearisering er anvendt til at opnå et lineært system, for hvilket der kan beregnes en endelig ækvivalent kvotient i form af

et endeligt transitionsystem. Den ækvivalente kvotient er efterfølgende anvendt til kontrolsyntese sammen med den formelle kravspecifikation. Da kvotienten og det oprindelige lineære system er ækvivalente, kan den diskrete kontroller, der er resultatet af syntesen, ligeledes anvendes på det lineære kontrol system. En forfining af den diskrete kontroller er nødvendig, hvilket resulterer i en hybrid lukket sløjfe.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The focus of this thesis is on the development of novel frameworks for automated deployment of mobile multi-robot systems. In particular, we want to investigate new methodologies for automated deployment of mobile multi-robot systems for a specific application domain.

1.1 Background and Motivation

In recent years there has been an increasing interest in systems comprised of several autonomous mobile robots. In the following we will refer to such system as a multi-robot system (MRS). An autonomous mobile robot is a physically independent system, equipped with sophisticated sensors and actuators, necessary and sufficient to accomplish a given task. MRS operate in a shared environment performing some specific task.

In the last decades MRS have traditionally been used in many transportation (warehouses and transshipment in harbors), industrial (assembly lines), agricultural, and military (surveillance and scouting) related tasks. What characterize these tasks are that they typically involve several subtasks that can be performed in parallel and only little amount of coordination among the robots are required.

New application areas are currently emerging such as underwater and space exploration, search and rescue, hazardous environments, and service robotics. In these challenging application areas, MRS can often deal with tasks that are difficult, if not impossible, to be accomplished by a single robot. MRS are typically well suited for various application domains which require complex tasks to be performed effectively and in a coordinated manner. Moreover, even when a single robot can achieve the given task, the possibility of deploying a MRS can improve the performance of the overall system. Also, MRS are typically employed in domains where they give rise to benefits that a single robot can not provide. Compared to a single robot there are several reasons for deploying a MRS [Arai et al., 2002].

- a task may be inherently too complex for a single robot to perform due to the fact that a single robot is spatially limited,
- a MRS can provide redundancy and contribute cooperatively to solve the assigned tasks,
- a MRS can improve the effectiveness either from the viewpoint of the performance in accomplishing certain tasks, or in the robustness and reliability of the system,
- a MRS can perform the assigned task in a more reliable, faster, or cheaper way beyond what is possible with a single robot,
- improve performance of the overall system.

Tasks, which definitely require a MRS, are tasks that typically involve spatially separate tasks and which require some sort of synchronization among the individual robots [Dudek et al., 1996]. However, there exists tasks that do not require a MRS, but can benefit significant from using a MRS. On the other hand, there may exist tasks which do not benefit from using a MRS. Tasks that are inherently not suitable for a MRS are often those which involve a single operation at a single location.

Multi-robot path planning is one of the fundamental problems of MRS and refers to finding collision free motions for each robot so that a certain task is performed. It is obvious that finding collision free motion for all robots in a MRS requires some sort of coordination. One of the most limiting characteristics of much of the existing path planning work for MRS is the computational complexity of the approaches.

From an engineering point of view MRS are inherently complex systems. The control of such complex systems poses new challenges that fall beyond the traditional approaches in control theory, e.g. stabilization and set-point regulation. One of these challenges is given by the need to control, coordinate and synchronize the operation of several robots that have to perform some specified task. This calls for new strategies and methods which allow to specify the desired system behavior in a formal and succinct way.

Temporal logic is chosen for describing the behavior of the system. Specifications containing both logical and temporal operators translate naturally to temporal logic. Temporal logic provides a formal requirement specification mechanism with the capability to define desired behaviors quantitatively and due to its similarity to natural languages it provides an intuitive and succinct way to express complex behaviors of a MRS.

1.2 Scope of Study

The scope of this study is I.) The motion coordination and planning of mobile multi-modal robots modeled as a network of hybrid automata and II.) Controller synthesis for single mobile robot systems modeled as a nonlinear system. Temporal logics such as Computational Tree Logic (CTL) and Linear-time Temporal Logic (LTL) are used as requirement specification mechanisms for expressing the desired behavior of the system. Both I.) and II.) make extensive use of bisimulations to obtain finite bisimilar quotients of the original system.

The following two frameworks comprise the study.

Framework I - Multi-robot Motion Planning It is a novel framework for the motion coordination and planning of a multi-modal robots modeled as a network of hybrid automata with respect to a requirement specification in Computational Tree Logic (CTL). CTL provides a formal requirement specification mechanism allowing to quantitatively define the desired behavior of the network of multi-modal robots. This framework presupposes an infra-structure of the multi-modal robots with feedback controllers that constraint the motion capabilities of the individual robots. This constrains the individual robot in the network to move in a planar grid where static obstacles may be present. Motion planning is performed using the model checking tool UPPAAL, given a requirement specification in CTL for the network of multi-modal robots. Path tracking the planned paths is not within the scope of this study.

Framework II - Robot Controller Synthesis A framework for controller synthesis for linear control systems with respect to a requirement specification in Linear-time Temporal Logic (LTL). Linear control systems satisfying simple controllability assumptions allow finite abstractions in the form of finite bisimilar quotients to be computed. The possibility to compute finite bisimulations of linear control systems permits a discrete controller to be synthesized. A refinement of the discrete controller results in a hybrid closed-loop combining the continuous dynamics of linear control system with the synthesized logic required to enforce the requirement specification. The framework is demonstrated for the controller synthesis for a unicycle.

1.3 Multi-Robot Systems

The study of MRS naturally extends research on single robot systems. Surveys on existing work in the field of MRS can be found in [Dudek et al., 1996, Cao et al., 1997, Parker, 2000, Arai et al., 2002, Farinelli et al., 2004]. In [Dudek et al., 1996] the authors present a taxonomy which classifies MRS based on communication and computational capabilities. Cao et al. [Cao et al., 1997] describe the theoretical issues that arise in the study of cooperation in MRS as well as identifying the primary research topics in MRS with respect to achieving the "mechanism of cooperation": Group architecture, resource conflict, origin of cooperation, learning, and geometric problems. Recently, Parker [Parker, 2000] and Arai et al. [Arai et al., 2002] have identified the primary research topics within MRS as follows.

- Biological inspirations
- Communication
- Architecture, task planning, and control
- Localization, mapping, and exploration
- Object transportation and manipulation
- Motion coordination
    - path planning
    - traffic control
    - formation generation/keeping
    - target tracking/search

    - docking behavior

- Reconfigurable robots

- Learning

As shown above motion coordination covers several sub-research topics. The work presented in this thesis is within motion coordination and path planning and control of robots.

In [Iocchi et al., 2001] and [Farinelli et al., 2004] a new taxonomy for classification of the approaches to coordination in MRS is proposed. The taxonomy is characterized by two groups of dimensions, that is the coordination and the system dimension. The term dimension refers to specific features that are grouped together in the taxonomy. The classification dimensions of the taxonomy are shown in Table 1.1.

Coordination Dimensions	**System Dimension**
Cooperation	Communication
Knowledge	Team composition
Coordination	System architecture
Organization	Team size

Table 1.1: *Classification dimensions.*

The coordination dimensions of the taxonomy are shown in Figure 1.1.

Figure 1.1 shows a hierarchical structure for the coordination dimensions of the taxonomy. The different levels of the structure are: A cooperation level, a knowledge level, a coordination level, and an organization level.

The first level of the taxonomy (cooperation level) is concerned with the ability of the system to cooperate in order to accomplish a specific task. The second level (knowledge level) is concerned with how much knowledge each robot in the system has about the presence of other robots. The third level (coordination level) is concerned with the mechanism that is used in order to achieve cooperation in the system. The fourth level (organization level) is concerned with the way the decision system is realized within the MRS.

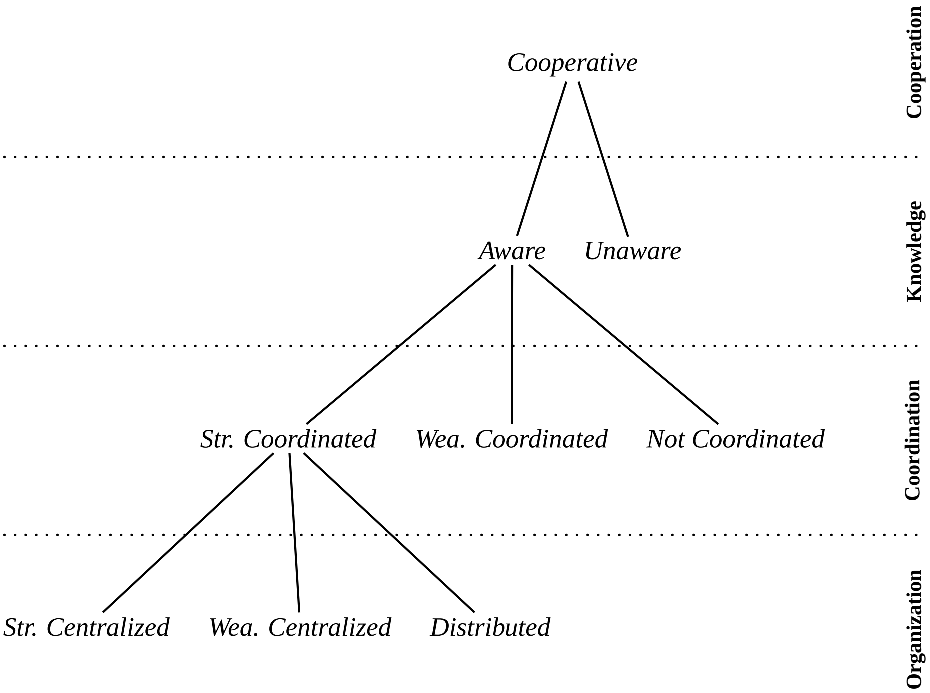


Figure 1.1: *Coordination dimensions in multi-robot systems.*

1.3.1 Coordination and Cooperation in Multi-Robot Systems

Coordination and cooperation are of fundamental importance for any MRS that is composed of several interacting mobile robots. In the following we explicitly define cooperation and coordination in a MRS. The terms cooperation (or to cooperate) and coordination (or to coordinate) are defined as follows.

> *"Coordination or to coordinate: The act of making all the actors/agents involved in a plan or activity work together in an organized way: There's absolutely no coordination between the different groups - nobody knows what anyone else is doing. To make various different things work effectively as a whole [Cambridge University Press, 2006]."*

> *"Cooperation or to cooperate: The action of cooperating, i.e. of working together towards the same end, purpose, or effect; joint operation. To work together, act in conjunction (with another actors/agents, to an end or purpose, or in a work) [Oxford University Press, 2005]."*

From the above definitions it is clear that cooperation imply that a group of actors/agents are working together to accomplish some task that is common to all actors/agents. On the other hand, coordination is related to how the action of cooperation is achieved among the actors/agents in the group. Further, coordination clearly requires that each actor/agent has some knowledge of what other actors/agents are doing. Explicit definitions of cooperation and coordination in the robotics literature are sparse. However, Iocchi et al. [Iocchi et al., 2001] defines cooperation and coordination in a MRS as follows.

DEFINITION 1.1 (COOPERATION) *Situation in which several robots operate together to perform some global task that either cannot be achieved by a single robot, or whose execution can be improved by using more than one robot, thus obtaining higher performances.*

DEFINITION 1.2 (COORDINATION) *Cooperation in which the actions performed by each robot take into account the actions executed by the other robots in such a way that the whole ends up being a coherent and high performance operation.*

Cao et al. [Cao et al., 1997] have a similar definition of what they call cooperative behavior in a MRS.

DEFINITION 1.3 (COOPERATIVE BEHAVIOR) *Given some task specified by a designer, a MRS displays cooperative behavior if, due to some underlying mechanism (i.e. the "mechanism of cooperation"), there is an increase in the total utility of the system.*

The mechanism by which coordination is achieved determines such properties as how efficient the MRS is in performing a given task and what types of coordinated tasks can be achieved.

1.4 Previous and Related Work

Several researchers have addressed the problem of coordination in a MRS. In the following we restrict previous and related work to motion coordination in MRS and to some extent to control. In particular, we focus on path planning in MRS. However, in the following we consider control of the robots as a natural part of motion coordination, and we review some of the more traditional approaches to motion coordination in MRS. Then, we review some of the approaches to motion coordination and control that employ formal methods.

In [Alami et al., 1998] multi-robot coordination is achieved by employing a plan-merging paradigm (at trajectory level) that guarantees coherent behavior of all the robots in all situations. The plan-merging paradigm enables each robot to produce a coordinated plan that is compatible with all plans executed by other robots. Ögren et al. [Ögren et al., 2002] have addressed the class of MRS for which control Lyapunov functions can be found. Their results yield an abstract and theoretically sound coordination strategy for formation control and maintenance in MRS.

1.4.1 Formal Methods in Motion Coordination

In the last decade several researchers have investigated the use of formal methods for the motion coordination and control of MRS. The methodology advocated by formal methods requires the construction of a high-level model or description of the system, typically in the form of a hybrid system. The model can then be subjected to a variety of mathematical analyses such as simulation, verification/model checking, performance evaluation, or controller synthesis. Algorithmic analysis of hybrid systems is a challenging problem since the presence of continuous variables results in an infinite state-space, and even the simplest analysis problems turn out to be undecidable. However, useful analysis can be performed for a limited class of hybrid systems such as timed automata [Alur and Dill, 1994].

Traditionally, verification tools such as NUSMV, KRONOS, UPPAAL and HYTECH, etc. have been used for checking whether a high-level model satisfies a requirement specification in some suitable temporal logic. Temporal logic is the natural framework for specifying the correctness of computer programs. Temporal logic was originally used to specify the behavior of reactive and concurrent systems [Manna and Pnueli, 1991].

An approach to formal modeling and design of communication and control strategies for a MRS was taken in [Alur et al., 1999]. The model of linear hybrid automata was used to model the high-level behavior of a robot. The verification tool HyTech was used to algorithmically analyze parameter constraints in the model.

A formalism for sequential composition of concurrent robot behaviors, based on threaded Petri nets, has been developed in [Klavins and Koditschek, 2000]. The formalism is used for the construction of automated factories such as mobile robot bucket brigades. A distributed negotiation mechanism for coordination in MRS was considered in [Gerkey and Mataric, 2002]. A hybrid control approach to action coordination and collision avoidance was taken in [Egerstedt and Hu, 2002]. In [Koo, 2001, Koo and Sastry, 2002] a hybrid automaton has been used to model the multi-modal behaviors of a robot. A computational framework for automatic generation of provably correct control laws for a fully actuated planar robot in a triangulated polygonal environment has been proposed in [Belta et al., 2005].

The use of temporal logic as a mechanism for requirement specification and controller synthesis in mobile robotic systems has been advocated as far back as [Antoniotti and Mishra, 1995]. Recently, temporal logics have been used as a specification mechanism for path planning [Fainekos et al., 2005a,b, Kloetzer and Belta, 2006a,b] and synthesis of multi-robot motion tasks [Loizou and Kyriakopoulos, 2004].

A methodology for automatically synthesizing multi-robot motion tasks based on requirement specifications in linear temporal logic (LTL) was presented by Loizou and Kyriakopoulos [Loizou and Kyriakopoulos, 2004]. The resulting closed-loop system was shown to satisfy the specifications by construction, thus ensuring correct design.

Recently, several researchers have considered the idea of developing a framework for automated deployment of single and multi-robot systems. Typically, these approaches employ temporal logic as a specification mechanism. Further, simple dynamics are assumed for the robots such that a finite abstraction (finite bisimilar quotient) can be achieved. Finally, a path or set of paths for the robot or robots are automatically generated using model checking or verification techniques.

Fainekos et al. [Fainekos et al., 2005b] have considered the problem of motion planning for a single, fully actuated robot in an polygonal environment in order to satisfy formulas expressible in LTL. First, discrete abstractions of the robots motion based on some environmental decomposition is constructed. Subsequently, discrete plans are generated that satisfy a temporal logic formula using the verification tool NUSMV. Finally, the discrete plans are translated into continuous trajectories using hybrid control.

The problem of synthesizing a controller from LTL specifications for discrete-time linear control systems with semi-linear partitions are considered by Tabuada and Pappas [Tabuada and Pappas, 2006], where it is shown that finite bisimulations exist for controllable linear control systems with properly chosen observations. The framework provided in [Tabuada and Pappas, 2006] is refined in [Fainekos et al., 2005a], where the authors study the problem of controlling a planar robot in a polygon so that its trajectory satisfies a LTL formula. It is assumed that a triangulation of the polygon is given, and vector fields are assigned in each triangle so that the produced trajectories satisfy a given LTL formula over the triangles.

A fully automated framework for control of continuous-time linear control systems from specifications given in terms of LTL formulas was provided by Kloetzer and Belta [Kloetzer and Belta, 2006a]. A single robot was used as an illustrative example. Recently, Kloetzer and Belta [Kloetzer and Belta, 2006b] have proposed a fully automated framework for motion planning of a MRS in a partitioned environment. The task requirement specification for the robots are given in terms of a LTL formula over regions of interest in the environment. In this framework, a robot is modeled as a transition system and verification methods are used to generate motion plans for the robots that satisfy the task requirement specification. Collision avoidance among the robots is achieved by allowing the robots to synchronize upon movement.

1.5 Contributions of This Work

- A novel framework for the motion planning of a network of multi-modal robots, modeled as a network of hybrid automata, with respect to formal requirement specifications in Computational Tree Logic (CTL) has been proposed. The framework presupposed an infrastructure of the multi-modal robots with feedback controllers that constraint the motion capabilities of the individual robots. This constrains the individual robot in the network to move in a planar grid where static obstacles may be present. Motion planning for the multi-modal robots in the network is feasible by abstracting each robot in the network to a timed automaton. Motion planning for the network of multi-modal robots is performed using the model checker UPPAAL.

- The motion planning part of the above framework is presented in [Quottrup et al., 2004, Andersen et al., 2004] where a network of multi-modal robots was modeled as a network of interacting timed automata. Formal composition was achieved through synchronization channels and UPPAAL was used for symbolic model checking, i.e. motion planning against a requirement specification in computational tree logic (CTL). All feasible trajectories that satisfy the specification were algorithmically analyzed. Also, the framework proposed for multi-robot motion planning developed in [Quottrup et al., 2004, Andersen et al., 2004] has been applied to a harvesting system in [Andersen and Jensen, 2004] and to multi-modal aerial robots in [Koo et al., 2006].

- The framework originally developed by Tabuada and Pappas [Tabuada and Pappas, 2006] has been extended to fit with the problem of synthesizing a controller for a mobile robot, given a requirement specification in LTL. This required some modifications which were primarily concerned with the input to the framework and with the software implementation of the linear hybrid system obtained by refinement.

1.6 Thesis Outline

The remaining chapters are outlined as follows.

Chapter 2 - Motion planning This chapter gives an introduction to various approaches for both single and multi-robot motion planning

Chapter 3 - Transition Systems and Bisimulations This chapter gives an introduction to the notions of bisimulations for two classes of transition systems with observations; transition systems and labeled transition systems.

Chapter 4 - Model Checking Networks of Timed Automata This chapter gives an introduction to the notions of model checking networks of timed automata in the model checker UPPAAL.

Chapter 5 - Multi-robot Motion Planning This chapter presents a novel framework for the motion planning of a network of multi-modal robots given a formal requirement specification in Computational Tree Logic (CTL).

Chapter 6 - Case Study I: Multi-robot Motion Planning The framework presented in chapter 5 is applied to a multi-robot system comprised of two multi-modal robots.

Chapter 7 - Robot Controller Synthesis This chapter presents a framework for synthesizing a controller for a linear control system given a formal requirement specification in Linear-time Temporal Logic (LTL).

Chapter 8 - Case Study II: Robot Controller Synthesis The framework presented in chapter 7 is applied to synthesize a controller for a mobile robot (unicycle).

Chapter 9 - Conclusions and Recommendations Presents the conclusions and recommendations for further work for the two frameworks presented in chapter 5 and 7, respectively.

Chapter 2

Motion Planning

This chapter deals with motion planning of single-robot and multi-robot mobile systems. First, a classification of motion planning algorithms is presented. Motion planning approaches for both single- and multi-robot systems are reviewed.

The research in robot motion planning can be traced back to the late 1960's [Latombe, 1991, Laumond, 1991]. In the broadest sense, motion planning refers to a robot's ability to plan its own motion. According to Hwang and Ahuja [Hwang and Ahuja, 1992] motion planning covers both path planning and trajectory planning. Path planning refers to the design of a geometric (kinematic) path of the robot only. Hence, in path planning the dynamics of a robot is not taken into account. Trajectory planning includes the design of linear and angular velocities of the robot. Further, motion planning involves such diverse aspects as finding collision-free paths among possibly moving obstacles, motion coordination of several robots, etc. Motion planning can be either static or dynamic. In static motion planning, complete information about the obstacles in the environment is known a priori. On the other hand, dynamic motion planning concerns the case where partial or no information about the obstacles are known a priori [Hwang and Ahuja, 1992].

Laumond [Laumond, 1991] classify motion planning as either holonomic or non-holonomic motion planning. Within the 1980's, motion planning was mainly considered in the context of holonomic systems. For holonomic systems all degrees of freedom (DOF) can be changed independently. In this case the existence of a collision-free path is characterized by the existence of a connected component in the configuration space. Thus, motion planning consists of building the free configuration space and searching for a path in its connected components. Latombe's book [Latombe, 1991] constitutes the reference within the domain of holonomic motion planning.

In the 1990's, researchers began to investigate motion planning in the presence of kinematic constraints, also referred to as non-holonomic motion planning. For non-holonomic systems, the degrees of freedom (DOF) are not independent. In fact, most mobile robots are subject to non-holonomic constraints. A classical example of a non-holonomic constraint is the pure-rolling without slipping constraint [Oriolo et al., 2002].

2.1 Motion Planning Algorithms

For the classification of motion planning algorithms two aspects are mainly taken into account, that is the completeness and the scope of the algorithm [Hwang and Ahuja, 1992]. The classification of motion planning algorithms is shown below.

- Completeness
  - exact
    - resolution complete
    - probabilistically complete
  - heuristic
- Scope
  - global
  - local

Exact algorithms guarantee to find a solution when one exists. However, exact algorithms are usually computational expensive. Two types of exact algorithms exist: Resolution complete and probabilistic complete. A resolution complete algorithm is guaranteed to find a collision-free path (if one exists) at a given resolution; otherwise return failure. For a probability complete algorithm the probability of finding a collision-free path (if one exists) converges to 1 as the running time goes to infinity. A long running time may be required to make the probability converge to 1. Heuristic algorithms are geared at finding a solution fast but may fail to find one for complex problems.

Further, algorithms can either have global or local scope. Algorithms with a global scope take into account all the information in the environment to plan a path from the initial to the goal configuration. Algorithms with a local scope only use information in the vicinity of the robot.

2.2 Motion Planning Problems

Any motion planning problem typically involves the following steps.

- Determine the configuration parameters[1] of the robot,
- Choose a suitable representation of the robot and possible obstacles in the environment,
- Select and apply a suitable motion planning approach to the problem at hand,
- Select and apply a search method to find a solution path to the problem.

In the basic motion planning problem [Laumond, 1991] there is one robot present in a static and known environment. The task is to compute a collision-free path that will bring the robot from its initial configuration to its desired configuration. In this context the robot is the only moving object in the environment and the dynamical properties of the robot are ignored, thus rendering the motion planning problem purely geometrical where the motions of the robot is only constrained by the static obstacles.

Various extensions of the basic motion planning problem exists and include among others the presence of dynamic obstacles in the environment, multiple-robots, unknown environment, or the presence of non-holonomic kinematic constraints, etc. Indeed, any of these extensions renders the motion planning problem more complex.

2.3 Motion Planning Approaches

In the following single-robot and multi-robot motion planning approaches will be reviewed. Typically, a motion planning approach employ some graph search

[1] For example position and orientation

method. The graph search methods applied in the context of motion planning are depth-first, breath-first, best-first, A*, D*, bidirectional search, and Dijkstra's algorithm.

2.3.1 Single-Robot

The general approaches to single-robot motion planning is outlined in the following as well as the scope that typically apply for each approach.

- Roadmap (global scope)
  - visibility graph
  - Voronoi diagram
  - freeway nets
  - silhoutte
- Cell decomposition (global scope)
  - approximate
  - exact
- Potential field (local scope)
- Sample-based (local scope)
  - rapidly-exploring random trees (RRT)
  - randomized path planner (RPP)
  - probabilistic path planner (PPP)

The roadmap and cell decomposition methods aim at capturing the global connectivity of the robot's free space into a condensed graph that is subsequently searched for a path. The roadmap method captures the connectivity of the robot's free space in a network of 1-D curves called roadmaps. Path planning is reduced to connecting the initial and goal configurations and then searching for at path. The constructed path is a concatenation of sub-paths connecting the initial configuration to the roadmap, a sub-path contained in the roadmap, and a sub-path connecting the roadmap to the goal configuration.

The cell decomposition approach typically involves three steps.

- Decomposing the robot’s free space into regions (squares, trapezoidal, triangular, polygon), called cells,

- Construct the connectivity graph that represents the adjacency relation between the cells in the free space. Nodes in the graph represent a cell and two nodes are connected by an edge if the corresponding cells are adjacent. Two cells are adjacent if they have a common edge. A graph search method (see graph search methods) is applied to find a path in the connectivity graph, connecting the initial and goal nodes. A path in the connectivity graph thus corresponds to a path of cells in the free space,

- A free path is computed by connecting the initial configuration to the goal configuration through the midpoints of the intersections of every two adjacent cells, etc. The cell decomposition can either be classified as approximate or exact. In the exact case the free space is decomposed into cells whose union is exactly the free space.

The potential field approach considers the robot as a particle moving under the action of forces generated by an artificial potential field attracting the robot towards the goal configuration and at the same time pushing it away from obstacles. Khatib pioneered the potential field. Early potential field methods had a problem with getting stuck at a local minima of the potential function other than at the goal configuration. Koditschek introduced the notion of a navigation function, a local-minimum-free potential function.

2.3.2 Multi-Robot

Several authors have tried to classify approaches to multi-robot path planning [Fujimura, 1991, Latombe, 1991, Arai and Ota, 1992, Cao et al., 1997]. Fujimura [Fujimura, 1991] classify path planning as either centralized or distributed. In centralized planning there is a global planner the makes the decisions. On the other hand, in distributed planning the individual robots plan and adjust their paths. Latombe [Latombe, 1991] classify path planning as either centralized (complete) or decoupled (not complete). The centralized approach takes into account all robots and consists of planning the co-ordinated paths of the robots as a path in their composite configuration space. In the decoupled approach the motion of one robot is planned independently of the other robots. Subsequently, in the case of resource conflicts the interactions among the path’s of the robots are taken into account. Two decoupled planning approaches are described, e.g. prioritized planning [Erdmann and Lozano-Pérez, 1986] and path coordination [O'Donnell and Lozano-Pérez, 1989]. Prioritized planning considers the path of one robot at the time according to a global priority that the robots have been assigned. The path coordination method is based on scheduling techniques for dealing with and avoiding collisions among robots.

Chapter 3

Transition Systems and Bisimulations

In this chapter we review the notions of bisimulations for two classes of transition systems: Transition systems introduced in [Tabuada and Pappas, 2006] and labeled transition systems introduced in [Pappas, 2003]. The notions of bisimulation is one of the main complexity reduction methods for the analysis and synthesis of transition systems. Bisimulations are important since they allow transition systems to be related.

3.1 Finite Quotients of Transition Systems

3.1.1 Partitions and Equivalence Relations

A partition $\mathcal{P}$ of a set A is a division of A into non-overlapping blocks or cells that cover all of A.

DEFINITION 3.1 (PARTITION) *A partition $\mathcal{P}$ of a set A is a collection of non-empty sets $\{P_i\}_{i\in I} = \mathcal{P}$, satisfying $A = \cup_{i\in I} P_i$, where $P_i \cap P_j = \emptyset$, for $i \neq j$.*

For any two elements $P_i, P_j \in A$ for $i \neq j$ the intersection $P_i \cap P_j = \emptyset$. The partition $\mathcal{P}$ is finite if I is finite and infinite otherwise. The partition $\mathcal{P}$ induces a projection map $\pi_{\mathcal{P}} : A \to \mathcal{P}$ which maps each element $a \in A$ to an unique set $\pi_{\mathcal{P}}(a) = P$.

Equivalence relations are important because they can be used to group together objects that are similar in some sense. The set A can be transformed into another set by considering each equivalence class as a single unit. Every equivalence relation can be identified with a partition and vice versa.

DEFINITION 3.2 (EQUIVALENCE RELATION) An equivalence relation $\sim$ on a set A is a binary relation on A, that is

- (Reflexivity) $a \sim a$ for all $a \in A$,
- (Symmetry) whenever $a \sim b$ then $b \sim a$,
- (Transitivity) if $a \sim b$ and $b \sim c$ then $a \sim c$.

An equivalence relation $\sim \subseteq A \times A$ on the set A induces a partition $\mathcal{P} = \{P_i\}_{i \in I}$, defined by $a, b \in P_i$ if $(a, b) \in \sim$. The elements P_i of the partition $\mathcal{P}$ are the *equivalence classes* of $\sim$. This means, that given a partition $\mathcal{P}$ of A an equivalence relation $\sim \subseteq A \times A$ having the elements of $\mathcal{P}$ can be defined. A refinement of a partition $\mathcal{P}$ is defined as follows.

DEFINITION 3.3 (REFINEMENT OF PARTITION) *Let $\mathcal{P}$ be a partition. Partition $\mathcal{P}'$ is a refinement of partition $\mathcal{P}$ when for every $P' \in \mathcal{P}'$ there exists a $P \in \mathcal{P}$ such that $\mathcal{P}' \subseteq \mathcal{P}$.*

Given a refinement $\mathcal{P}'$ of partition $\mathcal{P}$ the projection map $\pi_{\mathcal{P}'\mathcal{P}} : \mathcal{P}' \to \mathcal{P}$ is defined which maps every element $P' \in \mathcal{P}'$ to an unique element $\pi_{\mathcal{P}'\mathcal{P}}(P') = P$ such that $P' \subseteq P$.

3.1.2 Transition Systems and Bisimulations

The rationale for introducing transition systems is that linear control systems can be embedded in the class of transition systems. Transition systems will be used as an abstract model for capturing the dynamics of linear control systems. A transition system is defined as follows.

DEFINITION 3.4 (TRANSITION SYSTEM) *A transition system with observations is defined as*

$$T = \left(Q, Q^0, \longrightarrow, O, \Upsilon\right), \tag{3.1}$$

where

- *Q is a (possibly infinite) set of states,*
- *$Q^0 \subseteq Q$ is a (possibly infinite) set of initial states,*
- *$\longrightarrow \subseteq Q \times Q$ is a transition relation,*
- *O is a (possibly infinite) set of observations,*
- *$\Upsilon : Q \to O$ is an observation map assigning to each state $q \in Q$ an observation $\Upsilon(q) \in O$.*

Transition systems should be thought of as graphs with a possibly infinite number of nodes representing states and edges between nodes representing transitions. The choice of observation map Υ is natural since observations are associated with states rather than transitions. Transition system T is finite when Q and O are finite and infinite otherwise. Denote by $q \longrightarrow q'$ a pair $(q, q') \in \longrightarrow$. A state q is predecessor of a state q', and q' is a successor of q. Further, T is deadlock free if for every $q \in Q$, there exists a state $q' \in Q$ such that $q \longrightarrow q'$. Given a state $q \in Q$ it is useful to compute the set of states that can reach q in one step, that is in one transition. Denote by $\mathrm{Pre}(q)$ the set of states in Q that can reach q in one step

$$\mathrm{Pre}(q) = \left\{ q' \in Q \mid q' \longrightarrow q \right\}. \tag{3.2}$$

The Pre-*operator* is extended to sets of states $Q' \subseteq Q$ as follows

$$\mathrm{Pre}(Q') = \bigcup_{q' \in Q'} \mathrm{Pre}(q'). \tag{3.3}$$

In general, $\mathrm{Pre}^i(Q')$ for $i \geq 0$ denotes the set of states that can reach Q' in i steps. $\mathrm{Pre}^i(Q')$ is recursively defined as follows

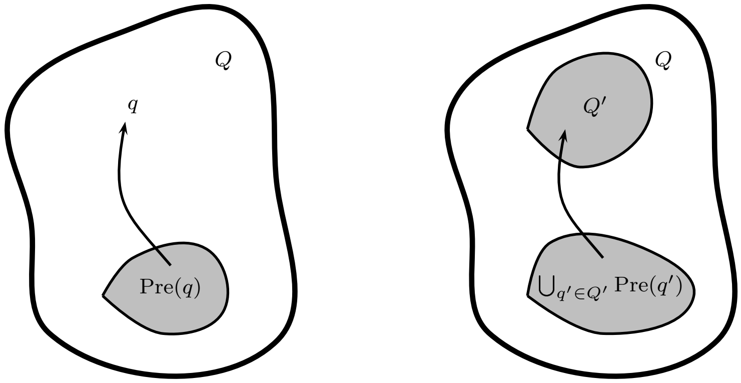
$$\mathrm{Pre}^0(Q') = Q', \tag{3.4}$$

$$\mathrm{Pre}^1(Q') = \mathrm{Pre}(Q'), \tag{3.5}$$

$$\vdots$$

$$\mathrm{Pre}^i(Q') = \mathrm{Pre}(\mathrm{Pre}^{i-1}(Q')), \quad i \geq 2. \tag{3.6}$$

The set of states in Q that can reach $q \in Q$ in one step and the set of states that can reach sets of states $Q' \subseteq Q$ in one step are graphically illustrated in Figure 3.1.



(a) *Set of states that can reach* $q \in Q$ *in one step.* (b) *Set of states that can reach* $Q' \subseteq Q$ *in one step.*

Figure 3.1: *Illustration of the* Pre*-operator.*

The notion of *bisimulation* is one of the main complexity reduction methods for the analysis and synthesis of transition systems. A bisimulation [Milner, 1989] is an equivalence relation between transition systems, associating systems which behave in the same way in the sense that one system simulates the other and vice-versa, i.e. they match each others moves while preserving the observations. Bisimulation relations can be described as symmetric simulation relations. Bisimulation relations are important since they allow transition systems to be related. Further, bisimulations preserve properties expressible in several temporal logics [Alur et al., 2000], e.g. Linear Temporal Logic (LTL) and Computational Tree Logic (CTL). Lets introduce the notion of a bisimulation relation.

DEFINITION 3.5 (BISIMULATION RELATION) *Let* $T_1 = \big(Q_1, Q_1^0, \longrightarrow_1, O, \Upsilon_1\big)$ *and* $T_2 = \big(Q_2, Q_2^0, \longrightarrow_2, O, \Upsilon_2\big)$ *be transition systems over a common set of observations* O*. Let* $\sim \subseteq Q_1 \times Q_2$ *be a relation between* Q_1 *and* Q_2*. The relation* $\sim$ *defines a bisimulation relation between* T_1 *and* T_2 *if the following holds for any pair* $(q_1, q_2) \in \sim$

- $q_1 \longrightarrow_1 q_1'$ *implies the existence of* $q_2' \in Q_2$ *satisfying* $q_2 \longrightarrow_2 q_2'$ *and* $(q_1', q_2') \in \sim$,

- $q_2 \longrightarrow_2 q_2'$ *implies the existence of* $q_1' \in Q_1$ *satisfying* $q_1 \longrightarrow_1 q_1'$ *and* $(q_1', q_2') \in\sim$,
- $q_1 \in Q_1^0$ *implies that* $q_2 \in Q_2^0$ *and* $q_2 \in Q_2^0$ *implies that* $q_1 \in Q_1^0$,
- $\Upsilon_1(q_1) = \Upsilon_2(q_2)$ *if* $q_1 \sim q_2$.

The bisimulation relation respects both observations and transitions. The existence of a bisimulation relation between T_1 and T_2 will be denoted by $T_1 \cong T_2$ and T_1 and T_2 are said to be bisimilar.

The introduction of equivalence and bisimulation relations now allow to introduce a finite bisimilar quotient that is bisimilar to transition system T.

DEFINITION 3.6 (QUOTIENT TRANSITION SYSTEM) *The quotient transition system of transition system* $T = (Q, Q^0, \longrightarrow, O, \Upsilon)$ *with respect to an equivalence relation* $\sim \subseteq Q \times Q$ *is given by*

$$T_{/\sim} = (Q_{/\sim}, Q^0_{/\sim}, \longrightarrow_{/\sim}, O, \Upsilon_{/\sim}), \tag{3.7}$$

where

- $Q_{/\sim} = \{S \subseteq Q \mid S \text{ is an equivalence class of } \sim\}$ *is a set of states,*
- $Q^0_{/\sim} = \pi_\sim(Q^0)$ *is a set of initial states,*
- $\longrightarrow_{/\sim} \subseteq Q_{/\sim} \times Q_{/\sim}$ *is a transition relation defined by* $S \longrightarrow_{/\sim} S'$ *if there exists* $q \in S$ *and* $q' \in S'$ *such that* $q \longrightarrow q'$ *in* T,
- O *is a set of observations,*
- $\Upsilon_{/\sim} : Q_{/\sim} \to O$ *is an observation map assigning to each* $S \in Q_{/\sim}$ *an observation* $\Upsilon_{/\sim}(S) = \Upsilon(q)$ *for some* $q \in S$.

The set of observations O is inherited from T. The observation map $\Upsilon_{/\sim}$ is well defined since $(q_1, q_2) \in\sim$ implies that $\Upsilon(q_1) = \Upsilon(q_2)$. If the relation $\sim$ is a bisimulation relation between T and T it follows that the graph of the projection $\pi_\sim : Q \to Q_{/\sim}$, defined by

$$\left\{(q, S) \in Q \times Q_{/\sim} \mid S = \pi_\sim(q)\right\}, \tag{3.8}$$

is a bisimulation relation between T and $T_{/\sim}$. The quotient transition system $T_{/\sim}$ is called a finite bisimilar quotient of transition system T with respect to

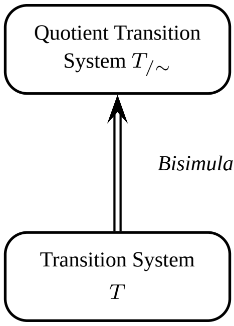


Figure 3.2: *Quotient transition system $T_{/\sim}$ of transition system T.*

equivalence relation $\sim$. The quotient transition system $T_{/\sim}$ of transition system T is graphically represented as in Figure 3.2.

Given two transition systems T_1 and T_2 the parallel composition of transition systems T_1 and T_2 with observation synchronization is natural and defined as follows.

DEFINITION 3.7 (PARALLEL COMPOSITION) *Let $T_1 = (Q_1, Q_1^0, \longrightarrow_1, O, \Upsilon_1)$ and $T_1 = (Q_2, Q_2^0, \longrightarrow_2, O, \Upsilon_2)$ be two transition systems over a common observation space O. The parallel composition of T_1 and T_2 with observation synchronization is given by*

$$T_1 \parallel_O T_2 = \left(Q_\parallel, Q_\parallel^0, \longrightarrow_\parallel, O, \Upsilon_\parallel\right), \tag{3.9}$$

where

- $Q_\parallel = \{(q_1, q_2) \in Q_1 \times Q_2 \mid \Upsilon_1(q_1) = \Upsilon_2(q_2)\}$ *is the set of states,*
- $Q_\parallel^0 = \left\{(q_1, q_2) \in Q_1^0 \times Q_2^0 \mid \Upsilon_1(q_1) = \Upsilon_2(q_2)\right\}$ *is the set of initial states,*
- $\longrightarrow_\parallel \subseteq Q_\parallel \times Q_\parallel$ *is the transition relation defined by* $(q_1, q_2) \longrightarrow_\parallel (q_1', q_2')$ *for* $(q_1, q_2), (q_1', q_2') \in Q_\parallel$ *if* $q_1 \longrightarrow_1 q_1'$ *in* T_1 *and* $q_2 \longrightarrow_1 q_2'$ *in* T_2*,*
- O *is the set of observations,*
- $\Upsilon_\parallel : Q_\parallel \to O$ *is the observation map defined by* $\Upsilon_\parallel(q_1, q_2) = \Upsilon_1(q_1) = \Upsilon_2(q_2)$.

The observation space O of $T_1 \parallel_O T_2$ is inherited from transition systems T_1 and T_2.

3.1.3 Languages of Transition Systems

Given a set S, denote by S^* the set of all finite strings obtained by concatenating elements in S. An element of S^* is given by $s_1 s_2 \ldots s_n$ with $s_i \in S$ for $i = 1, 2, \ldots, n$. The length of a string $s \in S^*$ is denoted by $|s|$. Denote by S^ω the set of all infinite strings obtained by concatenating elements in S. An element of S^ω is an infinite string $s_1 s_2 s_3 \ldots$ with $s_i \in S$ for $i \in \mathbb{N}$. Given a string s belonging to S^* or S^ω, denote by $s(i)$ the i-th element of S. A subset of S^* is called a language while a subset of S^ω is called an ω-language.

A string $s \in S^* \cup S^\omega$ is a run of T if $(s(i), s(i+1)) \in \longrightarrow$, $i = 1, 2, \ldots, |s| - 1$ for $s \in Q^\omega$ or $i \in \mathbb{N}$ for $s \in Q^*$. A run of T is initialized if $s(1) \in Q^0$. We now define the languages generated by a transition system T.

DEFINITION 3.8 (GENERATED LANGUAGE) *Let $T = \left(Q, Q^0, \longrightarrow, O, \Upsilon\right)$ be a transition system. The language generated by T is defined as*

$$L(T) = \{r \in O^* \mid r = \Upsilon(s) \text{ for some initialized run } s \text{ of } T\}.$$

The ω-language generated by T is defined as

$$L_\omega(T) = \{r \in O^\omega \mid r = \Upsilon(s) \text{ for some initialized run } s \text{ of } T\}.$$

Bisimulations preserve language equivalence. Two transition systems T_1 and T_2 are language equivalent if they generate the same language.

PROPOSITION 3.1 (LANGUAGE EQUIVALENCE) *Let T_1 and T_2 be two transition systems and $\sim$ a bisimulation relation between T_1 and T_2. The following equalities hold*

$$\begin{aligned} L(T_1) &= L(T_2), \\ L_\omega(T_1) &= L_\omega(T_2). \end{aligned}$$

Further, the languages generated by the parallel composition of T_1 and T_2, denoted by $T_1 \parallel_O T_2$ can be expressed in terms of the languages generated by T_1 and T_2

$$\begin{aligned} L\left(T_1 \parallel_O T_2\right) &= L\left(T_1\right) \cap L\left(T_2\right), \\ L_\omega\left(T_1 \parallel_O T_2\right) &= L_\omega\left(T_1\right) \cap L_\omega\left(T_2\right), \end{aligned}$$

where $L(T_1) \cap L(T_2)$ denotes language intersection.

3.1.4 Labeled Transition Systems and Bisimulations

Here, we define a labeled transition system with observations. In order to relate properties of different labeled transition systems, the definitions of simulation and bisimulation will be introduced.

DEFINITION 3.9 (TRANSITION SYSTEM (LABELED)) *A labeled transition system with observations is defined as*

$$T = (Q, \Sigma, \longrightarrow, O, \Upsilon), \tag{3.10}$$

where

- *Q is a (possibly infinite) set of states,*
- *Σ is a (possibly infinite) set of labels,*
- *$\longrightarrow \subseteq Q \times \Sigma \times Q$ is a transition relation,*
- *O is a (possibly infinite) set of observations,*
- *$\Upsilon : Q \to O$ is an observation map assigning to each $q \in Q$ an observation $\Upsilon(q) \in O$.*

The labeled transition system T is finite when Q, Σ, and O are finite and infinite otherwise. Denote by $q \xrightarrow{\sigma} p$ a triple $(q, \sigma, p) \in \longrightarrow$. A region is a subset $P \subseteq Q$ of the set of states. The σ-successor of a region P is denoted by $\mathrm{Post}_\sigma(P)$ and is defined as the set of states that can be reached from P with one σ-transition, see Figure 3.3.

The Post_σ-*operator* is defined as follows

$$\mathrm{Post}_\sigma(P) = \left\{ q \in Q \mid \exists p \in P \text{ with } p \xrightarrow{\sigma} q \right\}. \tag{3.11}$$

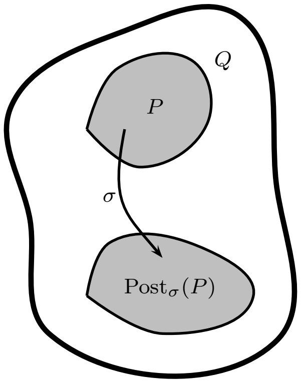


Figure 3.3: *Illustration of the* Post_σ*-operator for a region* $P \subseteq Q$.

In general, $\mathrm{Post}^i_\sigma(P)$ denotes the set of states in Q that are reachable from P using i σ-transitions. $\mathrm{Post}^i_\sigma(P)$ for $i \geq 0$ is recursively defined as

$$\begin{aligned}
\mathrm{Post}^0_\sigma(P) &= P, && (3.12)\\
\mathrm{Post}^1_\sigma(P) &= \mathrm{Post}_\sigma(P), && (3.13)\\
\mathrm{Post}^2_\sigma(P) &= \mathrm{Post}_\sigma\left(\mathrm{Post}_\sigma(P)\right), && (3.14)\\
&\vdots\\
\mathrm{Post}^i_\sigma(P) &= \mathrm{Post}_\sigma\left(\mathrm{Post}^{i-1}_\sigma(P)\right). && (3.15)
\end{aligned}$$

A relation R between Q_1 and Q_2 is a subset $R \subseteq Q_1 \times Q_2$ and we define

$$R^{-1} = \{(p,q) \in Q_2 \times Q_1 \mid (q,p) \in R\}, \tag{3.16}$$

as the inverse relation.

DEFINITION 3.10 (SIMULATION RELATION (LABELED) *Consider two labeled transition systems* $T_1 = (Q_1, \Sigma, \longrightarrow_1, O, \Upsilon_1)$ *and* $T_2 = (Q_2, \Sigma, \longrightarrow_2, O, \Upsilon_2)$ *over a common set of labels* Σ *and observations* O. *A relation* $R \subseteq Q_1 \times Q_2$ *is called a simulation relation from* T_1 *to* T_2 *if it respects both observations and transitions, that is*

- *if* $(q,p) \in R$ *then* $\Upsilon_1(q) = \Upsilon_2(p)$,

- *if* $(q,p) \in R$ *and* $q \xrightarrow{\sigma} q'$, *then* $p \xrightarrow{\sigma} p'$, *for some* $(q',p') \in R$.

DEFINITION 3.11 (BISIMULATION RELATION (LABELED)) *Let* $T_1 = (Q_1, \Sigma, \longrightarrow_1, O, \Upsilon_1)$ *and* $T_2 = (Q_2, \Sigma, \longrightarrow_2, O, \Upsilon_2)$ *be two labeled transition systems over a common set of labels* Σ *and observations* O*. A relation* $R \subseteq Q_1 \times Q_2$ *is called a bisimulation relation between* T_1 *and* T_2 *if* R *is a simulation relation from* T_1 *to* T_2 *and* R^{-1} *is a simulation relation from* T_2 *to* T_1*.*

The property of bisimulations states that equivalent states must be able perform a transition using the same label to states that are also equivalent. Given a labeled transition system $T = (Q, \Sigma, \longrightarrow, O, \Upsilon)$ and an equivalence relation $\sim \subseteq Q \times Q$ the quotient transition system is defined on the quotient space $Q_{/\sim}$. Let $\Psi : Q \to Q_{/\sim}$ be the quotient map. The definition of a quotient transition systems follows as.

DEFINITION 3.12 (QUOTIENT TRANSITION SYSTEM (LABELED)) *The quotient transition system of a labeled transition system* $T = (Q, \Sigma, \longrightarrow, O, \Upsilon)$ *with respect to an equivalence relation* $\sim \subseteq Q \times Q$ *is given by*

$$T_{/\sim} = (Q_{/\sim}, \Sigma, \longrightarrow_{/\sim}, O, \Upsilon_{/\sim}), \tag{3.17}$$

where

- $Q_{/\sim} = \{Q_{s_1} \in 2^Q \mid Q_{s_1}$ *is an equivalence class of* $\sim\}$ *is a set of states,*
- Σ *is a set of labels,*
- $\longrightarrow_{/\sim} \subseteq Q_{/\sim} \times \Sigma \times Q_{/\sim}$ *is a transition relation defined by* $Q_{s_1} \xrightarrow{\sigma}_{/\sim} Q_{s_2}$ *if there exists* $q \in Q_{s_1}$ *and* $p \in Q_{s_2}$ *such that* $q \xrightarrow{\sigma} p$ *in* T*,*
- O *is a set of observations,*
- $\Upsilon_{/\sim} : Q_{/\sim} \to O$ *is an observation map defined by* $\Upsilon_{/\sim}(\Psi(q)) = \Upsilon(q)$.

The set of labels Σ and the set of observations O of $T_{/\sim}$ are inherited from T. The transition relation $\longrightarrow_{/\sim}$ of $T_{/\sim}$ is induced from the transition relation $\longrightarrow$ of T. The observation map $\Upsilon_{/\sim}$ is well defined since the partition π induced by $\sim$ is *observation preserving* [Pappas, 2003], i.e. if $p \sim q$ then $\Upsilon(p) = \Upsilon(q)$. Thus, equivalent states have the same observation. We now define characterization [Pappas, 2003].

PROPOSITION 3.2 (CHARACTERIZATION) Consider the labeled transition system T and observation-preserving partition $\sim$ with quotient map $\Psi : Q \to Q_{/\sim}$. Then, $\sim$ is a bisimulation of T if for all states $q \in Q$ and for all labels $\sigma \in \Sigma$

$$\Psi(Post_\sigma(\Psi^{-1}(\Psi(q)))) = \Psi(Post_\sigma(q)), \tag{3.18}$$

where $\Psi^{-1}(\Psi(q))$ is the set of all states in Q that are equivalent to state q.

3.2 Summary

The notions of transitions systems and bisimulations introduced in this chapter will be used in chapters 5-8.

The class of labeled transition systems is used as an abstract model for a network of multi-modal robots where each robot is modeled as a hybrid automaton. The abstract models for the network of multi-modal robots are used for motion planning, given a requirement specification in Computational Tree Logic (see chapters 5- 6).

The class of transition systems is used as an abstract model for capturing the dynamics of a robot, modeled as a nonlinear system. The abstract model of the robot is used as a baseline for controller synthesis with respect to requirement specification in Linear-time Temporal Logic (see chapters 7-8)

Chapter 4

Model Checking Networks of Timed Automata

This chapter gives an introduction to the concept of model checking networks of timed automata using the model checker UPPAAL, *given a formal requirement specification in Computational Tree Logic (CTL).*

4.1 Networks of Timed Automata in UPPAAL

UPPAAL [Uppsala University and Aalborg University, 1995a] is an integrated tool environment for modeling, simulation and model checking real-time systems that can be modeled as a network of interacting timed automata extended with data types (bounded integers, arrays, etc.). The tool is developed in collaboration between the Department of Information Technology at Uppsala University, Sweden and the Department of Computer Science at Aalborg University, Denmark. The tool is appropriate for systems that can be modeled as a collection of non-deterministic processes with finite control structure and real-valued clocks, communicating through channels.

Given a system modeled as a network of timed automata and a requirement specification in computational Tree Logic (CTL), UPPAAL is used for model checking (verifying) the system against the specification. The result of model checking or verifying the system is a *"property satisfied/not satisfied"* and when relevant a *symbolic trace* of the system. The symbolic trace shows the trace for each automaton in the network satisfying or violating the specification. The setup is illustrated in Figure 4.1.

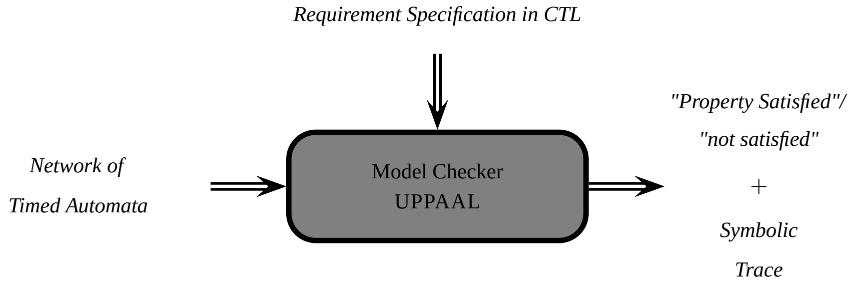


Figure 4.1: *Setup for model checking a network of timed automata using the model checker* UPPAAL*, given a formal requirement specification in Computational Tree Logic (CTL).*

Since clocks range over the non-negative reals, a timed automata can have infinitely many states and infinitely many traces. Therefore, it is not possible to visualize all these concrete traces. Instead, an infinite set of traces are visualized; a so called symbolic trace. Each symbolic state of a symbolic trace is a set of states and their delay successors described by a number of constraints on the clocks. In a given symbolic state, the active locations and the values of discrete variables are the same for all states.

4.2 Declaration of Processes

UPPAAL allows the construction of templates for the system being modeled. Subsequently, the templates are used to declare new processes.

4.3 Synchronization of Processes

In UPPAAL synchronization channels are used to synchronize processes. This is done by annotating edges in the model with synchronization labels. Synchronization labels are of the form $e!$ and $e?$, where $e!$ and $e?$ are in the sending and receiving process, respectively, and e evaluating to a channel. Two processes can synchronize on enabled edges annotated with complementary synchronization labels, i.e. two edges in different processes can synchronize if the guards of both edges are satisfied, and they have synchronization labels $e_1!$ and $e_2?$, respectively, where e_1 and e_2 evaluate to the same channel. When two processes synchronize, both edges are fired at the same time, i.e. the current location of both processes is changed. The update expression on an edge synchronizing on $e_1!$ is executed before the update expression on an edge synchronizing on $e_2?$.

Three different types of synchronization channels can be used in UPPAAL to synchronize processes.

Regular Channel Described above.

Urgent Channel Urgent channels are similar to regular channels, except that it is not possible to delay in the source state if it is possible to trigger a synchronization over an urgent channel. Clock guards are not allowed on edges synchronizing over urgent channels.

Broadcast Channel Broadcast channels allow 1-to-many synchronizations. The intuition is that an edge with synchronization label $e!$ emits a broadcast on the channel e and that any enabled edge with synchronization label $e?$ will synchronize with the emitting process. I.e. an edge with an send-synchronisation on a broadcast channel can always fire, provided that the guard is satisfied, no matter if any receiving edges are enabled. But those receiving edges, which are enabled will synchronize. Notice that clock guards are not allowed on edges receiving on a broadcast channel. The update on the receiving edges are executed left-to-right in the order the processes are given in the system definition.

4.4 Timed Automata

A timed automaton is a finite-state automaton or finite-state machine extended with a finite collection of real-valued clock variables [Alur and Dill, 1994]. All clocks are assumed to proceed at the same rate, i.e. they progress synchronously, and measure the amount of time that have elapsed since they were reset. The value of a clock may be compared with natural numbers and reset to zero. Let C be a set of real-valued variables, called clocks. Denote by $B(C)$ the set of conjunctions over simple constraints of the form $x \bowtie c$ and $x - y \bowtie c$, where $x, y \in C$, $c \in \mathbb{N}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$. Elements of $B(C)$ are called guards over C. A timed automaton is defined as [Larsen et al., 1995].

DEFINITION 4.1 (TIMED AUTOMATON) *A timed automaton is a tuple $A_T = (L, l^0, C, A, E, I)$, where*

- *L is the set of locations,*
- *$l^0 \in L$ is the initial location,*
- *C is the set of clocks,*
- *A is the set of actions, co-actions and the internal τ-action,*
- *$E \subseteq L \times A \times B(C) \times 2^C \times L$ is the set of edges between locations with an action, a guard and a set of clocks to be reset,*
- *$I : L \rightarrow B(C)$ assigns invariants to locations.*

4.4.1 Urgent and Committed Locations

When a process is in an urgent location, time is not allowed to pass, i.e. time "freezes". Semantically, urgent locations are equivalent to: Adding an extra clock x, that is reset on every incoming edge, and adding an invariant $x \leq 0$ to the location.

Like urgent locations, committed locations "freeze" time. Furthermore, if any process is in a committed location, the next transition must involve an edge from one of the committed locations. Committed locations are useful for creating atomic sequences and for encoding synchronization between more than two components. Notice that if several processes are in a committed location at the same time, then they will interleave.

4.5 Semantics of Timed Automata

A timed automaton is a finite directed graph annotated with conditions over and resets of non-negative real valued clocks $x \in C$, which satisfy the differential equation $\dot{x} = 1$. A clock valuation is a function $u : C \rightarrow \mathbb{R}_{\geq 0}$ from the set of clocks C to the non-negative reals $\mathbb{R}_{\geq 0}$. Denote by $\mathbb{R}^C$ the set of all clock valuations. A state of a timed automaton A_T is a pair (l, u), where $l \in L$ is a location of A_T and u holds the current values for the clock variables. The initial state of A_T is (l^0, u_0), where u_0 assigns zero to all clocks $x \in C$. In a timed automaton all clocks are initialized to zero by definition, that is $u_0(x) = 0$, for all clocks $x \in C$. An invariant express constraints on the clock values in order to remain in a particular state. Further, $u \in I(l)$ means that u satisfy $I(l)$. The semantics of a timed automaton is defined as follows.

DEFINITION 4.2 (SEMANTICS OF TIMED AUTOMATON) *Let $A_T = (L, l^0, C, A, E, I)$ be a timed automaton. The semantics of A_T is defined as a transition system*

$$(S, s_0, \longrightarrow), \tag{4.1}$$

where

- *$S \subseteq L \times \mathbb{R}^C$ is the set of states,*
- *$s_0 = (l^0, u_0)$ is the initial state,*
- *$\longrightarrow \subseteq S \times \{\mathbb{R}_{\geq 0} \cup A\} \times S$ is the transition relation defined by*
  - *$(l, u) \xrightarrow{d} (l, u + d)$ if $\forall d' : 0 \leq d' \leq d \implies u + d \in I(l)$,*
  - *$(l, u) \xrightarrow{a} (l', u')$ if there $\exists e = (l, a, g, r, l') \in E$ such that $u \in g$, $u' = [r \mapsto 0]u$, and $u' \in I(l)$,*

  where for $d \in \mathbb{R}_{\geq 0}$, $u + d$ maps each clock $x \in C$ to the value $u(x) + d$ and $[r \mapsto 0]u$ denotes the clock valuation which maps each clock in r to 0 and agrees with u over $C \setminus r$.

Let $A_{T_i} = (L_i, l_i^0, C, A, E_i, I_i)$, for $1 < i \leq n$ be a network of timed automata over a common set of clocks C and actions A. A location vector is given as $\bar{l} = \left[l_1, \ldots, l_n\right]^{\mathrm{T}}$. The invariant functions are composed into a common function over location vectors $I(\bar{l}) = \wedge_i I_i(l_i)$. Denote by $\bar{l}[l_i'/l_i]$ the vector where the i-th element l_i of $\bar{l}$ is replaced by l_i'. The semantics of a network of timed automata is defined as follows.

DEFINITION 4.3 (SEMANTICS OF A NETWORK OF TIMED AUTOMATA) *Let $A_{T_i} = (L_i, l_i^0, C, A, E_i, I_i)$, for $i = 1, \ldots, n$ be a network of timed automata over a common set of clocks C and actions A. The semantics is defined as a transition system*

$$(S, s_0, \longrightarrow), \tag{4.2}$$

where

- $S = (L_1 \times \ldots \times L_n) \times \mathbb{R}^C$ *is the set of states,*
- $s_0 = (\bar{l}_0, u_0)$ *is the initial state,*
- $\longrightarrow \subseteq S \times S$ *is the transition relation defined by*
  - $(\bar{l}, u) \longrightarrow (\bar{l}, u + d)$ *if* $\forall d' : 0 \leq d' \leq d \implies u + d \in I(\bar{l})$*,*
  - $(\bar{l}, u) \longrightarrow (\bar{l}[l'_i/l_i], u')$*, if there* $\exists l_i \xrightarrow{\tau g r} l'_i$ *such that* $u \in g$*,* $u' = [r \mapsto 0]u$*, and* $u' \in I(\bar{l})$*.*
  - $(\bar{l}, u) \longrightarrow (\bar{l}[l'_j/l_j, l'_i/l_i], u')$*, if there* $\exists l_i \xrightarrow{e?g_i r_i} l'_i$ *and* $l_j \xrightarrow{e!g_j r_j} l'_j$ *such that* $u \in (g_i \wedge g_j)$*,* $u' = [r_i \cup r_j \mapsto 0]u$*, and* $u' \in I(\bar{l})$*.*

4.6 Requirement Specification in Computation Tree Logic (CTL)

For timed automata, problems such as reachability and model checking requirement specifications in CTL are decidable[1] [Alur et al., 1995, 2000]. UPPAAL use a subset of Computation Tree Logic (CTL), i.e. a simplified version of CTL. Like in CTL, the query language consists of path formulas and state formulas. State formulae describe individual states, whereas path formulas quantify over paths or traces of the model. Path formulas can be classified into reachability, safety and liveness. Figures 4.2-4.4 illustrates the different path formulas supported by UPPAAL. In contrast to CTL, UPPAAL does not allow nesting of path formulas. Hence, liveness and safety properties can not be checked at the same time. A summary of the path formulas supported in UPPAAL is found in Table 4.1.

State formulae describe individual states, whereas path formulas quantify over paths or traces of the model.

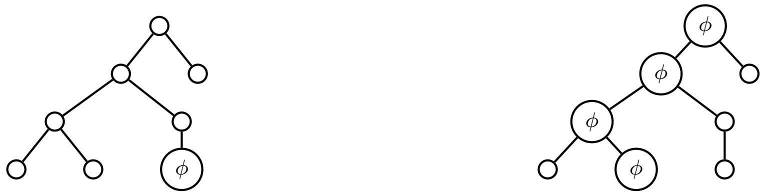
4.6.1 State and Path Formulas

A state formula is an expression that can be evaluated for a state without looking at the behavior of the model. The syntax of state formulae is a superset of that of guards, i.e., a state formula is a side-effect free expression, but in contrast to

[1] Computational feasible.

Property	Equivalent to	Description
$E\langle\rangle\phi$	N/A	*Possibly: There exists a path where state property ϕ eventually hold.*
$E[\,]\phi$	N/A	*Potentially always: There exists a path where state property ϕ always hold.*
$A\langle\rangle\phi$	$\neg E[\,]\neg\phi$	*Eventually: For all paths state property ϕ eventually hold.*
$A[\,]\phi$	$\neg E\langle\rangle\neg\phi$	*Invariantly (always): For all paths state property ϕ always hold.*
$\phi - > \varphi$	$A[\,](\phi \Rightarrow A\langle\rangle\varphi)$	*Leads to: Whenever state property ϕ holds state property φ eventually hold.*

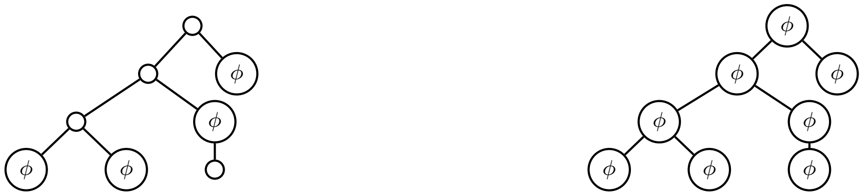
Table 4.1: *Path formulas supported in* UPPAAL.



(a) Possibly: $E\langle\rangle\phi$.

(b) Potentially always: $E[\,]\phi$.

Figure 4.2: *Path formulas using the E-operator.*



(a) Eventually: $A\langle\rangle\phi$.

(b) Invariantly: $A[\,]\phi$.

Figure 4.3: *Path formulas using the A-operator.*

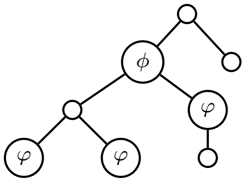


Figure 4.4: *Leads to:* $\phi -> \varphi$.

guards, the use of disjunctions is not restricted. It is also possible to test whether a particular process is in a given location using an expression on the form P.l, where P is a process and l is a location. Path formulae can be classified into reachability, safety and liveness.

4.7 Summary

The concept of model checking networks of timed automata in UPPAAL introduced in this chapter will be used in chapters 5-6. Given a network of robots modeled as a network of timed automata and a requirement specification in CTL for the network UPPAAL is used to generate a set of collision-free paths for the network of robots which satisfy the formal requirement specification.

Chapter 5

Multi-robot Motion Planning

In this chapter a framework for the motion planning of a network of multi-modal robots with respect to formal requirement specifications in Computational Tree Logic (CTL) is proposed. CTL provides a formal requirement specification mechanism allowing to quantitatively define the desired behavior of the network of multi-modal robots. The framework presented here presupposes an infrastructure of the multi-modal robots with feedback controllers that constraint the motion capabilities of the individual robots. This constrains the individual robot in the network to move in a planar grid where static obstacles may be present.

5.1 Framework

The baseline for the framework is a network of N multi-modal robots $H_1, \ldots, H_N$ and a finite partition π of the environment that conforms with the motion capabilities of the robots in the network. Each of the robots in the network is modeled as a hybrid automaton. The framework is depicted in Figure 5.1.

The intermediate steps involved in the proposed framework are briefly described below.

(1) Hybrid Automaton Model of Multi-Modal Robot A hybrid automaton H is used as a generic model for each of the multi-modal robots in the network $H_1, \ldots, H_N$.

(2) Partitioning the Environment A finite partition π of the environment is

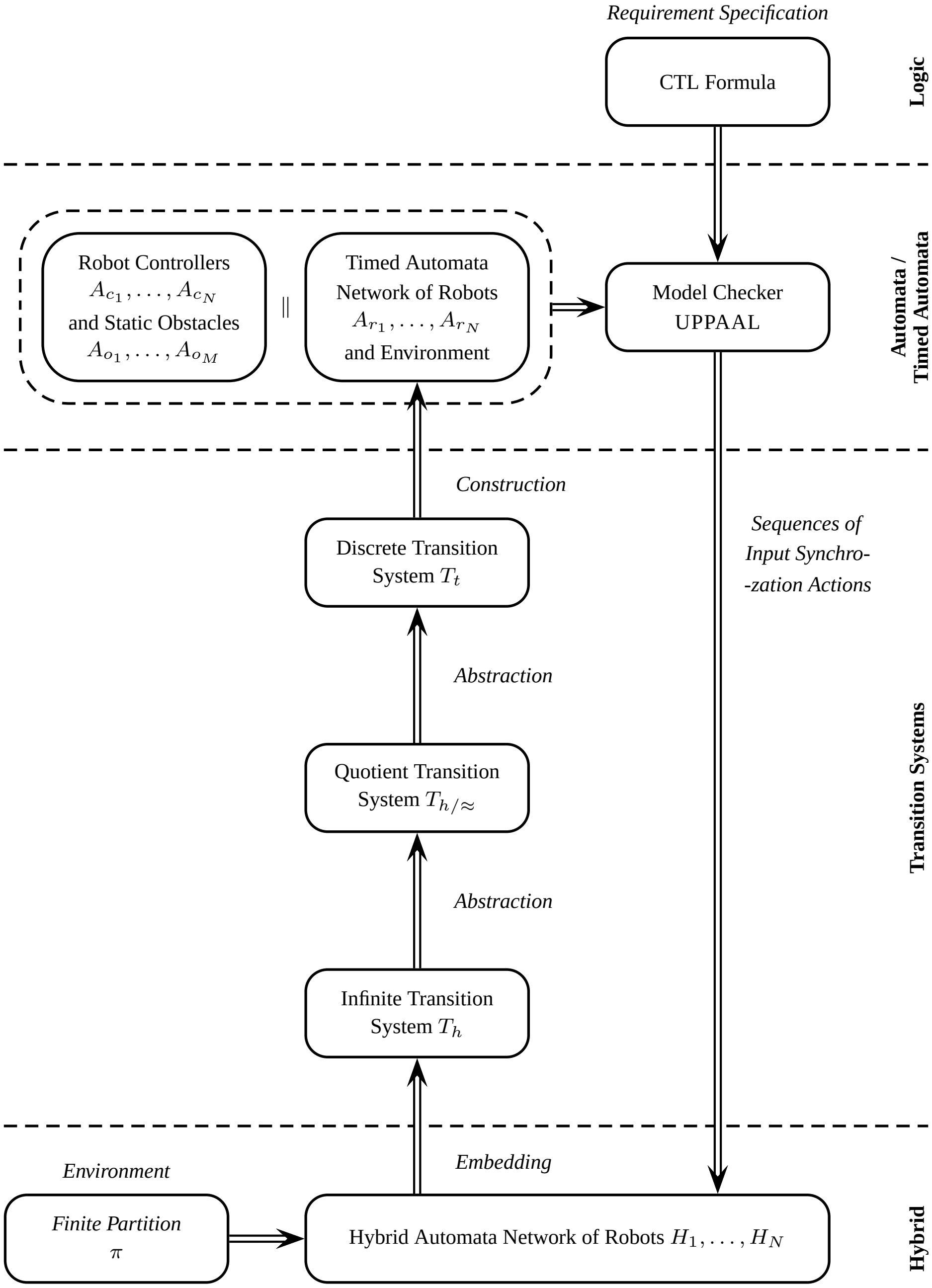

Requirement Specification
CTL Formula
Logic
Robot Controllers $A_{c_1}, \ldots, A_{c_N}$ and Static Obstacles $A_{o_1}, \ldots, A_{o_M}$
$\|$
Timed Automata Network of Robots $A_{r_1}, \ldots, A_{r_N}$ and Environment
Model Checker UPPAAL
Automata / Timed Automata
Construction
Sequences of Input Synchro- -zation Actions
Discrete Transition System T_t
Abstraction
Quotient Transition System $T_{h/\approx}$
Abstraction
Infinite Transition System T_h
Transition Systems
Embedding
Environment
Finite Partition π
Hybrid Automata Network of Robots $H_1, \ldots, H_N$
Hybrid

Figure 5.1: *Proposed framework for motion planning of a network of multi-modal robots with respect to formal requirement specification in Computational Tree Logic (CTL).*

formed. The partition conforms with the motion capabilities of the multi-modal robots in the network.

(3) Continuous Transitions Timed and time-abstract transitions are defined for describing robot movement and stopping a robot, respectively. Further, the introduction of timed and time-abstract transitions allows cyclic transitions to be defined. Cyclic transitions allow the hybrid automaton H to operate continuously by taking cyclic transitions. Cyclic transitions are essential in the process of embedding the hybrid automaton H into the class of labeled transition systems.

(4) Embedding the Hybrid Automaton The hybrid automaton H is embedded in the class of labeled transition systems. The result is a labeled transition system T_h that captures the hybrid behavior of hybrid automaton H. The labeled transition system T_h is equivalent to the hybrid automaton H with respect to desired reachability properties.

(5) Obtaining the Abstraction A finite quotient transition system $T_{h/\approx}$ is obtained from the labeled transition system T_h by considering a finite partition π of the environment. By construction, the quotient $T_{h/\approx}$ is bisimilar to transition system T_h associated with hybrid automaton H and thus the reachability properties of H are preserved in the abstraction. Further, the quotient $T_{h/\approx}$ is abstracted into a finite and discrete transition system T_t. Again, T_t is bisimilar to $T_{h/\approx}$ by construction.

(6) Constructing the Timed Automata A timed automaton A_r is constructed from transition system T_t such that the reachability properties are preserved. The construction of A_r allow timing and coordination of robots in the network to be considered. Timed automaton A_r is used as a template for instantiating each of the multi-modal robots $A_{r_1}, \ldots, A_{r_N}$ in the network. To allow the network of multi-modal robots to move concurrently a simple automaton controller template is also constructed. In this way a controller can be instantiated for each of the robots in the network. Finally, a obstacle template is constructed.

(7) Creating Process Instances Robot, controller, and obstacle process instances are instantiated from the constructed templates. Each process is instantiated with a set of specific process parameters.

(8) Requirement Specification A requirement specification is formulated in Computational Tree Logic (CTL) that express the desired behavior of the network of multi-modal robots.

(9) Motion Planning The model checker UPPAAL is used to generate a set of collision-free paths for the network of multi-modal robots in the form of sequences of synchronization inputs such that the requirement specification in CTL is satisfied. The sequences of synchronization inputs are subsequently executed by the network of multi-modal robots $H_1, \ldots, H_N$ modeled by hybrid automata.

The steps (1)-(7) are described in this chapter while steps (7)-(9) are the topic of chapter 6.

The timed automata formalism merely presents an abstraction (a high-level model) of environment, robots and associate controller, but allows composition and formal symbolic reasoning about coordinated motion planning solutions. The model checker UPPAAL is used for formal symbolic model checking against a requirement specification formulated in Computational Tree Logic (CTL) for a network of multi-modal robots. The result of the verification is a *"Property satisfied"* or *"Property not satisfied"*, meaning that the requirement specification is satisfied or not satisfied, respectively. In case the requirement specification is satisfied a symbolic trace is generated, containing the sequence of synchronization inputs required for moving the network of multi-modal robots to their final positions.

Finally, the generated sequences of input synchronization actions are executed by the network of multi-modal robots $H_1, \ldots, H_N$. Therefore, the sequences of input synchronization actions generated by the model checker are used as high-level motion plans for the network of multi-modal robots.

5.2 Modeling a Network of Multi-Modal Robots

In the following we consider a network of N robots denoted by H_i for $i = 1, \ldots, N$.

5.2.1 Assumptions

The following assumptions are made regarding the modeling of the network of multi-modal robots

- The robots move concurrently in a planar environment, i.e $X \subseteq \mathbb{R}^2$,
- Each robot is assumed to be a single point in X,
- Each of the robots is assumed to have identical motion capabilities,
- Each of the robots is assumed to move with a fixed, but possible different velocity,
- A set of static obstacles $O_1, \ldots, O_M$ may be present in the environment.

The position of robot H_i is given by

$$x_i = \begin{bmatrix} x_{i_1} & x_{i_2} \end{bmatrix}^{\mathrm{T}} \in X, \tag{5.1}$$

for $i = 1, \ldots, N$. Thus the velocity of robot H_i is given by

$$\dot{x}_i = v_i = \begin{bmatrix} v_{i_1} & v_{i_2} \end{bmatrix}^{\mathrm{T}}. \tag{5.2}$$

Thus, for a robot to move from an initial to goal position requires an appropriate sequence of inputs, where each input corresponds to a motion capability.

To simplify the notation the position of a robot in the network is denoted by x and the velocity is denoted by v.

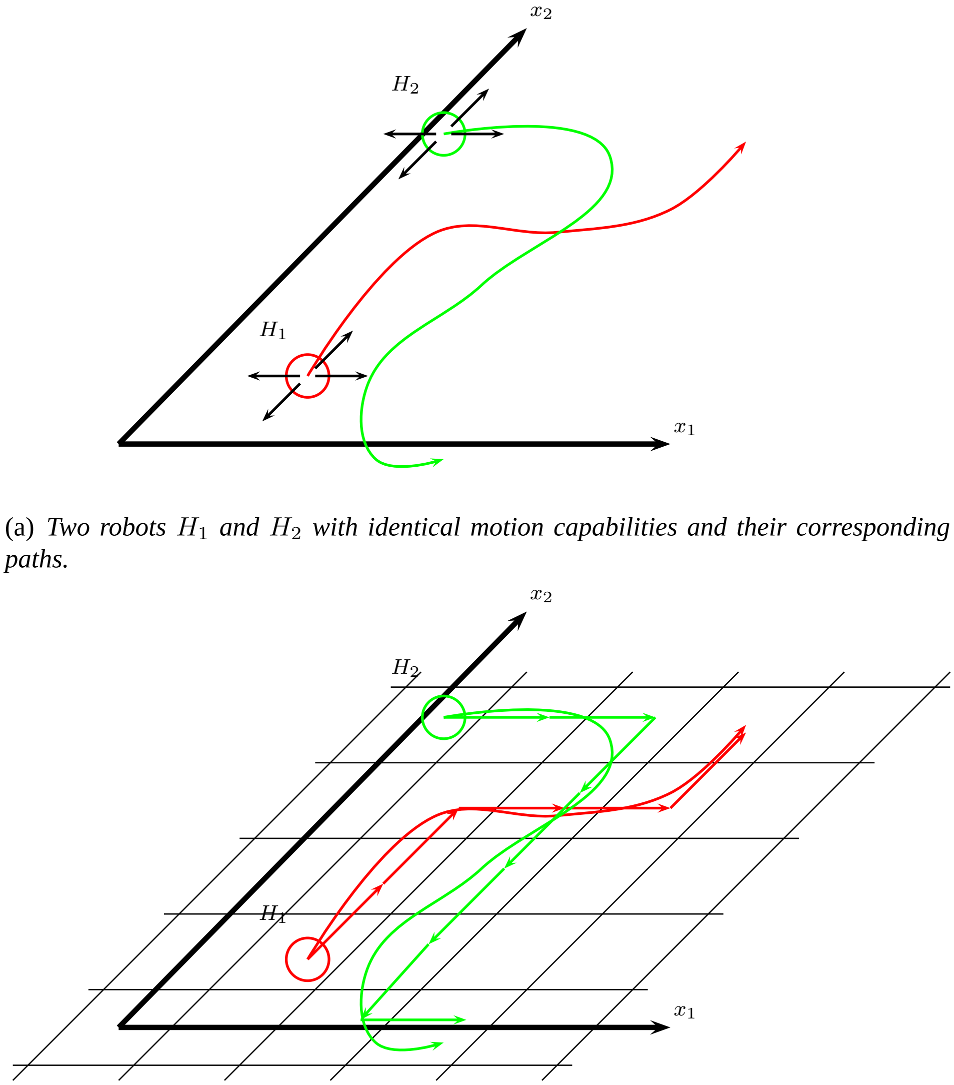
5.3 Hybrid Automaton as Generic Model

A hybrid automaton [Koo and Sastry, 2002] is used as a generic model for each of the multi-modal robots in the network $H_1, \ldots, H_N$. The hybrid automaton is defined as

$$H = (Q \times X, \Sigma, Y, Init, f, \Upsilon, I, G, R), \tag{5.3}$$

where

- $Q = \{q_1, q_2, q_3, q_4, q_5\}$ is the set of discrete states,
- $X \subseteq \mathbb{R}^2$ is the continuous state-space,



(a) *Two robots H_1 and H_2 with identical motion capabilities and their corresponding paths.*

(b) *Appropriate execution of the motion capabilities of the robots H_1 and H_2 results in paths that are similar to the original paths.*

Figure 5.2: *Network of two multi-modal robots H_1 and H_2.*

- $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$ is the set of events with input $\sigma \in \Sigma$,
- $Y \subseteq \mathbb{R}^2$ is the continuous output-space with output $y \in Y$,
- $Init$ is the initial set defined by $Init = \{q_1\} \times X$,
- f is the vector field defined by

$$\dot{x} = f(q,x) = \begin{cases} \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}} & \text{if } q = q_1, \\ \begin{bmatrix} v & 0 \end{bmatrix}^{\mathrm{T}} & \text{if } q = q_2, \\ \begin{bmatrix} -v & 0 \end{bmatrix}^{\mathrm{T}} & \text{if } q = q_3, \\ \begin{bmatrix} 0 & v \end{bmatrix}^{\mathrm{T}} & \text{if } q = q_4, \\ \begin{bmatrix} 0 & -v \end{bmatrix}^{\mathrm{T}} & \text{if } q = q_5. \end{cases}$$

  where $v = v_1 = v_2 > 0$,
- $\Upsilon : Q \times X \times \Sigma \to X$ is the observation map defined by $\Upsilon(q, x, \sigma) = y = x$ for $q \in Q$,
- I is the invariant defined by $I(q_i) = X \times \{\sigma_i\}$ for $i = 1, \ldots, 5$,
- G is the guard relation defined by

$$G(q_i, q_j) = \begin{cases} \sigma_j & \text{if } q_i = q_1 \text{ and } q_j \in Q \setminus \{q_1\}, \\ \sigma_1 & \text{if } q_i \in Q \setminus \{q_1\} \text{ and } q_j = q_1. \end{cases}$$

- R is the reset relation defined by $R(q_i, q_j, x) = \{x\}$, for $(i,j) \in \{(1,2),(2,1),(1,3),(3,1),(1,4),(4,1),(1,5),(5,1)\}$.

The hybrid automaton H is graphically represented in Figure 5.3.

The hybrid state of hybrid automata H is $(q,x) \in Q \times X$. The hybrid automaton H starts in the hybrid state $Init = \{q_1\} \times X$. Hence, the robot starts in the discrete state q_1 at an arbitrary position x. In state q_1 the vector field is $f(q_1, x) = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}$ and hence the continuous state x remains the same. In state q_1 the hybrid automaton H accepts any input from the set of events $\Sigma \setminus \{\sigma_1\}$ as defined by the guard relation G. If the input is σ_2 the guard $G(q_1, q_2)$ is enabled and the hybrid automaton H takes the transition to discrete state q_2. In q_2 only x_1 will increase since the vector field is $f(q_2, x) = \begin{bmatrix} v & 0 \end{bmatrix}^{\mathrm{T}}$. In the state $q_i \in Q \setminus \{q_1\}$ the hybrid automaton H accepts only the input σ_1 and takes the transition back to the discrete state q_1.

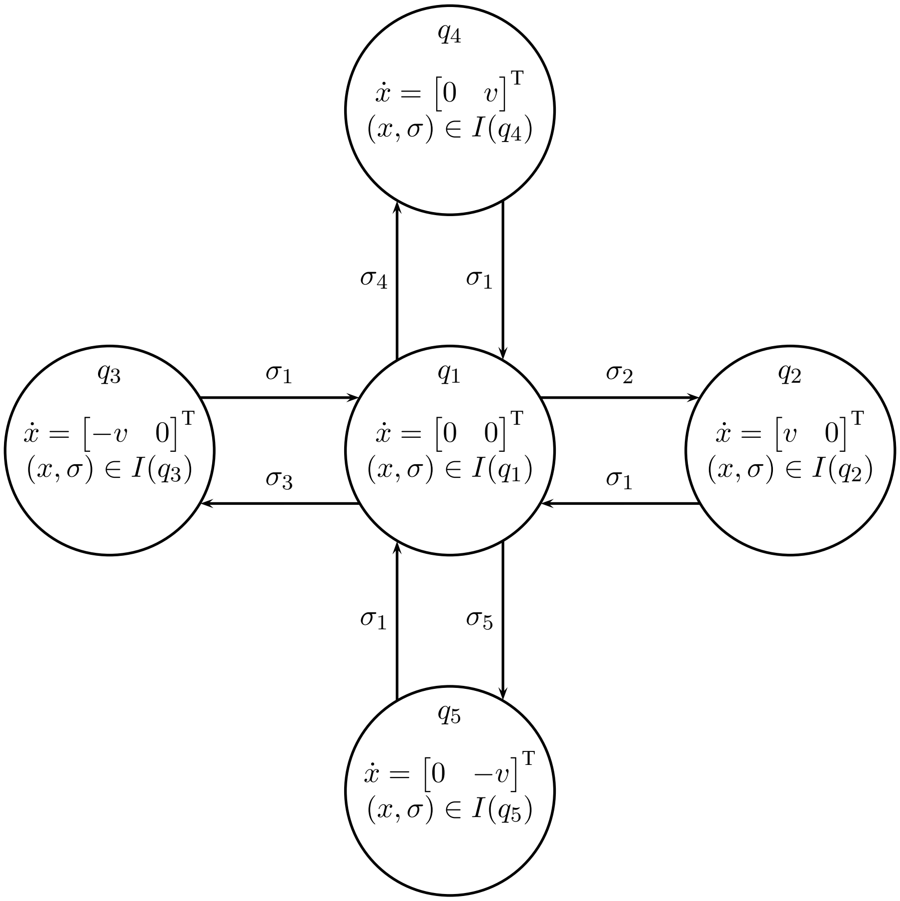

q_4
$\dot{x} = \begin{bmatrix}0 & v\end{bmatrix}^{\mathrm{T}}$
$(x,\sigma) \in I(q_4)$
σ_4
σ_1
q_3
$\dot{x} = \begin{bmatrix}-v & 0\end{bmatrix}^{\mathrm{T}}$
$(x,\sigma) \in I(q_3)$
σ_1
σ_3
q_1
$\dot{x} = \begin{bmatrix}0 & 0\end{bmatrix}^{\mathrm{T}}$
$(x,\sigma) \in I(q_1)$
σ_2
σ_1
q_2
$\dot{x} = \begin{bmatrix}v & 0\end{bmatrix}^{\mathrm{T}}$
$(x,\sigma) \in I(q_2)$
σ_1
σ_5
q_5
$\dot{x} = \begin{bmatrix}0 & -v\end{bmatrix}^{\mathrm{T}}$
$(x,\sigma) \in I(q_5)$

Figure 5.3: *A hybrid automaton H is used as a generic model for each of the multi-modal robots in the network $H_1, \ldots, H_N$.*

5.4 Partitioning the Environment

In the following we provide the partitioning of the continuous state-space X in which the robots move. Now, consider that the continuous state-space $X \subseteq \mathbb{R}^2$ is decomposed into a finite number of cells by a partition $\pi = \{X_j\}_{j=1}^{M}$, where M is the number of cells in the partition. We assume that the location of each robot or static obstacle in the continuous state space belongs to exactly one cell. The partition π of X satisfies the following two properties

$$X = \bigcup_{j \in \mathcal{I}} X_j, \quad \mathcal{I} = \{1, \ldots, M\}, \tag{5.4}$$

$$X_i \cap X_j = \emptyset, \quad \forall i \neq j, \tag{5.5}$$

where M is the number of cells in the partition. Hence, the cells of the partition π cover the continuous state space X and do not overlap.

The partition induces an equivalence relation. In this context, the induced equivalence relation $\approx$ is called *cell equivalence* and is defined over the continuous state space X. The cell equivalence relation $\approx$ is finite since it has a finite number of equivalence classes, i.e. a finite number of cells. For any two positions $x', x'' \in X$, $x' \approx x''$ (x' is equivalent to x'') if there exists $j \in \mathcal{I}$ such that $x', x'' \in X_j$.

The continuous state-space X is decomposed in such a way that it conforms with the motion capabilities of the robots. The partition π is constructed by putting a two-dimensional grid over the continuous state-space X. The boundaries of each cell $X_j \in \pi$ is parallel with exactly one possible motion direction of a robot, i.e. a motion in the x_1 or x_2-direction. The obtained partition is composed of identical cells with length $\epsilon > 0$ where each cell is defined as the Cartesian product of two half open intervals. The partition π is shown in Figure 5.4

The motion of a multi-modal robot is restricted to be from one cell in the partition to an adjacent cell. The rationale behind this restriction is twofold. First, it reduces the state-space of the system and hence it reduces the complexity of model checking the system. Second, it reduces the number of cells that needs to be occupied when the multi-modal robots are moving.

Given a cell $X_j \in \pi$ we now define

$$\mathcal{I}_j = \left\{ i \in \mathcal{I} \setminus \{j\} \mid |\partial X_j \cap \partial X_i| > 1 \right\}, \tag{5.6}$$

where ∂X_j and ∂X_i denote the boundary of cells X_j and X_i, respectively. Thus, every X_i is adjacent to X_j for $i \in \mathcal{I}_j$.

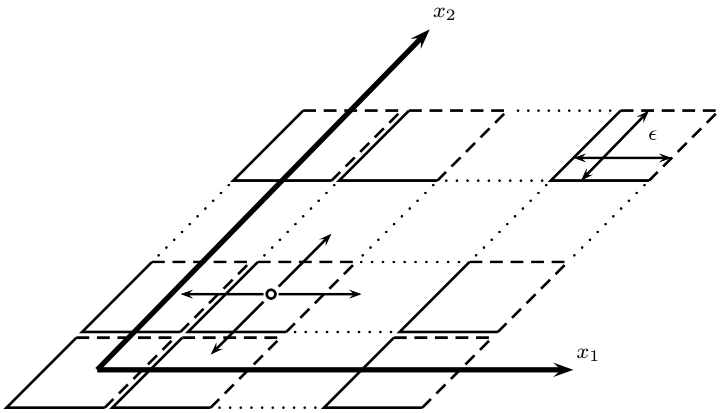


Figure 5.4: *Partition π of the continuous space X into a finite number of cells and motion capabilities of a robot.*

5.5 Cyclic Transitions

Here, we are interested in the reachability of the robots in the environment. In [Koo and Sastry, 2002] the time-abstract transitions for describing the continuous transitions of a hybrid automaton have been introduced. In this multi-robot motion planning scenario, there are multiple robots operating concurrently and there is a need to coordinate their motions in order to meet the requirement specification. However, we are only concerned about the periods of time when the robots are in motion and we are not interested in those robots that are not moving at all. This is because coordination is needed only when the robots are performing continuous transitions from one cell to another one, which takes a finite amount of time. Therefore, we now introduce two types of continuous transitions associated with hybrid automaton H, that is *timed* and *time-abstract* transitions. Timed transitions are associated with robot movement and time-abstract transitions are associated with stopping the robot. Timed and time-abstract transitions are essential in the process of embedding the hybrid automaton H into the class of labeled transition systems and subsequently for obtaining a finite quotient transition system.

Now, lets define $\phi(t, q, x_0)$ as the solution of the differential equation $\dot{x} = f(q, x)$ with $x(0) = x_0$ for $t \geq 0$.

DEFINITION 5.1 (σ_i-LABELED (TIMED) TRANSITION) Consider $x', x'' \in X$, $\rho \in \mathbb{R}_{\geq 0}$ and $\sigma_i \in \Sigma \setminus \{\sigma_1\}$, the *$\sigma_i$-labeled (timed) transition* is defined as

$$x' \xrightarrow{\sigma_i} x'' \text{ if } x'' = \phi(\rho, q_i, x').$$

This transition is defined for a fixed period of time ρ and it describes the continuous transition in discrete state $q_1 \in Q$ with input $\sigma_i \in \Sigma \setminus \{\sigma_1\}$.

DEFINITION 5.2 (σ_1-LABELED (TIME-ABSTRACT) TRANSITION) Consider $x', x'' \in X$, $\delta \in \mathbb{R}_{\geq 0}$ and $\sigma_1 \in \Sigma$, the *σ_1-labeled (time-abstract) transition* is defined as

$$x' \xrightarrow{\sigma_1} x'' \text{ if } x'' = \phi(\delta, q_1, x').$$

This transition is defined for a fixed period of time δ and it describes the continuous transition in discrete state $q_i \in Q \setminus \{q_1\}$ with input $\sigma_1 \in \Sigma$.

The introduction of timed and time-abstract transitions allows to define cyclic transitions.

DEFINITION 5.3 (σ_i-LABELED CYCLIC TRANSITION) Consider $x', x'' \in X$ and $\sigma_i \in \Sigma \setminus \{\sigma_1\}$, the *$\sigma_i$-labeled cyclic transition* is defined as

$$x' \overset{\sigma_i}{\Longrightarrow} x'' \text{ if } x' \xrightarrow{\sigma_i} x'' \xrightarrow{\sigma_1} x''.$$

Following the definition of the initial set as $Init = \{q_1\} \times X$, the cyclic transition enables a transition from q_1 to $q_i \in Q \setminus \{q_1\}$ and then back to q_1. Hence, the hybrid automaton H can operate continuously by taking the cyclic transitions. The definition of cyclic transition allows to introduce the local motion capability of a robot.

PROPOSITION 5.1 (LOCAL MOTION CAPABILITY) Consider a finite partition of the continuous state space $X \subseteq \mathbb{R}^2$ defined by $\pi = \{X_j\}_{i=1}^{M}$. Given a cell $X_j \in \pi$ and an adjacent cell $X_i \in \pi$ with $i \in \mathcal{I}_j$ there exists $\sigma_i \in \Sigma \setminus \{\sigma_1\}$ such that for all $x' \in X_j$ there exists $x'' \in X_i$ such that $x' \overset{\sigma_i}{\Longrightarrow} x''$.

Given the partition π, due to the definition of adjacent cells, there are at most four possible adjacent cells for each cell. However, for the cells at the boundary of the partition π there are at most two or three adjacent cells. For an adjacent cell, since there exists exactly one motion direction that is parallel to each boundary, one can simply pick a motion direction that will make the robot move towards the adjacent cell. Due to the simple reachability property of the multi-modal system modeled by the hybrid automata H, one can easily show that a robot could start from anywhere within the cell and could reach somewhere inside the adjacent cell in finite time. An example of a local motion capability of a robot is graphically illustrated in Figure 5.5.

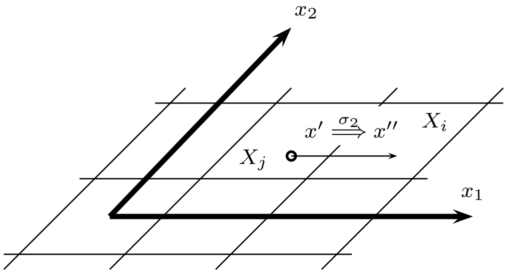


Figure 5.5: *Example of a local motion capability of a robot caused by a σ_2-labeled cyclic transition.*

Starting in cell X_j a σ_2-labeled cyclic transition will move the robot in the x_1-direction and eventually the robot will reach the adjacent cell X_i.

5.6 Embedding the Hybrid Automaton

We now introduce a labeled transition system that preserves the reachability properties of the hybrid automaton H. The hybrid automaton H is embedded in the class of labeled transition systems with observations. We shall consider a finite set of observations O associated with the finite set of cells defined by the partition $\pi = \{X_j\}_{j=1}^{M}$ of the continuous state-space X.

Define a labeled transition system associated with hybrid automaton H as

$$T_h = (Q_h, \Sigma_h, \Longrightarrow_h, O, \Upsilon_h), \tag{5.7}$$

where

- $Q_h = X$ is the set of states,
- $\Sigma_h = \Sigma \setminus \{\sigma_1\}$ is the set of labels,

- $\Longrightarrow_h \subseteq X \times \Sigma_h \times X$ is the transition relation defined by $x \overset{\sigma}{\Longrightarrow}_h x'$ if $x, x' \in X$,
- $O = \mathbb{B}^M$ is the set of observations,
- $\Upsilon_h : Q_h \to O$ is the observation map defined as

$$\Upsilon_h(x) = \begin{bmatrix} \Upsilon_{h_1}(x) & \Upsilon_{h_2}(x) & \dots & \Upsilon_{h_M}(x) \end{bmatrix}^{\mathrm{T}},$$

  where $\Upsilon_{h_j} : X \to \mathbb{B} = \{0, 1\}$, for $j = 1, \dots, M$ is defined as

$$\Upsilon_{h_j}(x) = \begin{cases} 1 & \text{if } x \in X_j, \\ 0 & \text{otherwise.} \end{cases}.$$

Transition system T_h is infinite since the set of states Q_h is defined as the continuous state space X. However, the set of observations O is finite since the partition π is finite. By constructing the observation map Υ_h according to the partition π the cell equivalence relation $\approx$ is *observation-preserving*.

5.7 Obtaining the Abstraction

The set of all equivalence classes X_j in X, given the cell equivalence relation $\approx$, is called the quotient space $X_{/\approx}$ of X by the cell equivalence relation $\approx$. The quotient space $X_{/\approx}$ is defined as

$$\begin{aligned} X_{/\approx} &= \pi \\ &= \{X_j\}_{j=1}^{M}, \end{aligned} \tag{5.8}$$

that is the set consisting of all equivalence classes X_j of cell equivalence relation $\approx$. Given the cell equivalence relation $\approx$, there is a canonical projection map[1] $\Psi_h : X \to X_{/\approx}$ defined as

$$\Psi_h(x) = X_i \text{ if } x \in X_i, \tag{5.9}$$

which sends each $x \in X$ to its equivalence class X_i. The quotient transition system obtained from the labeled transition system T_h is defined as

$$T_{h/\approx} = \left(Q_{h/\approx}, \Sigma_h, \Longrightarrow_{h/\approx}, O, \Upsilon_{h/\approx}\right), \tag{5.10}$$

where

[1] Also called quotient map. The quotient map is always surjective.

- $Q_{h/\approx} = X_{/\approx}$ is the set of states,
- $\Longrightarrow_{h/\approx} \subseteq Q_{h/\approx} \times \Sigma_h \times Q_{h/\approx}$ is the transition relation defined by $X_i \overset{\sigma}{\Longrightarrow}_{h/\approx} X_j$ in $T_{h/\approx}$ if there exists $x \in X_i$ and $x' \in X_j$ such that $x \overset{\sigma}{\Longrightarrow}_h x'$ in T_h,
- $\Upsilon_{h/\approx} : Q_{h/\approx} \to O$ is the observation map defined by $\Upsilon_{h/\approx}(\Psi_h(x)) = \Upsilon_h(x)$.

The labeled quotient transition system $T_{h/\approx}$ is finite since $Q_{h/\approx}$, Σ_h and O are finite. Note, that Σ_h and O are inherited from T_h. We can show that the cell equivalence relation $\approx$ is a bisimulation of transition system T_h associated with hybrid automaton H. Given the transition system T_h and the cell equivalence relation $\approx$, we can show that $\approx$ is a bisimulation of T_h.

THEOREM 5.1 (BISIMULATION RELATION) *Consider the transition system T_h associated with hybrid automata H and the cell equivalence relation $\approx$. Then, $\approx$ is a bisimulation of T_h.*

PROOF 1 (THEOREM 5.1) *Since $\Psi_h^{-1}(\Psi_h(x))$ is the set of all states in X that are equivalent to x, it is a cell. According to Proposition 5.1, $Post_\sigma(x) \in Post_\sigma(\Psi_h^{-1}(\Psi_h(x)))$ for all $x \in X$ and for all $\sigma \in \Sigma_h$. Furthermore, $Post_\sigma(\Psi_h^{-1}(\Psi_h(x)))$ is an adjacent cell which is also an equivalence class. Therefore, $\Psi_h(Post_\sigma(\Psi_h^{-1}(\Psi_h(x)))) = \Psi_h(Post_\sigma(x))$. Hence the result.*

THEOREM 5.2 *Consider the transition system T_h associated with hybrid automata H, the quotient transition system $T_{h/\approx}$ of T_h and the cell equivalence relation $\approx$. Then, T_h and $T_{h/\approx}$ are bisimilar.*

PROOF 2 (THEOREM 5.2) *As the states of the quotient transition system $T_{h/\approx}$ are given by the equivalence classes of cell equivalence relation $\approx$, a finite quotient transition system is obtained since the cell equivalence relation $\approx$ has a finite number of equivalence classes. Since the cell equivalence relation $\approx$ is a bisimulation of transition system T_h associated with hybrid automaton H, it can be shown that T_h is bisimilar to $T_{h/\approx}$ by the relation*

$$R_h = \{(x, X_i) \in X \times \pi \mid \Psi_h(x) = X_i\}, \tag{5.11}$$

between T_h and $T_{h/\approx}$.

Since T_h and $T_{h/\approx}$ are bisimilar, model checking properties of T_h can equivalently be performed by model checking the properties of $T_{h/\approx}$, which is discrete and finite. Therefore, the reachability problem for the multi-modal robot is decidable.

The last step in the process of obtaining a finite quotient transition system of hybrid automaton H, is to associate each cell of the partition with a discrete position. Thus, a finite transition system is introduced which has a finite set Z of discrete positions as the set of states. The midpoint of a cell in the partition coincides with a discrete position of a robot. Thus, we associate with each cell $X_i \in \pi$ a discrete position $z_i \in \mathbb{Z}^2$ of the robot. Lets define an isomorphism[2] $\Psi_t : \pi \to Z$ such that

$$\Psi_t(X_i) = z_i \text{ if } z_i \in X_i \; \forall i \in \mathcal{I}, \tag{5.12}$$

where $\mathcal{I} = \{1, \ldots, M\}$. Thus, Z is defined as

$$Z = \{z_j\}_{j=1}^{M}. \tag{5.13}$$

Now, define a labeled transition system

$$T_t = (Q_t, \Sigma_h, \Longrightarrow_t, O, \Upsilon_t), \tag{5.14}$$

where

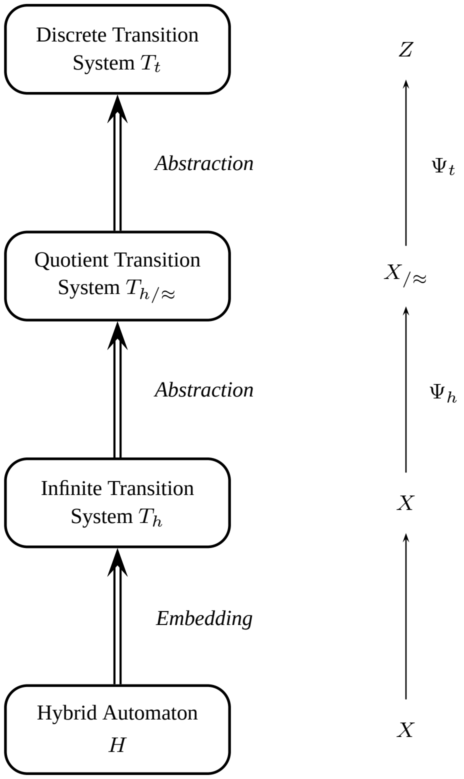
- $Q_t = Z$ is the set of discrete states,
- $\Longrightarrow_t \subseteq Q_t \times \Sigma_h \times Q_t$ is the transition relation defined by $z_i \overset{\sigma}{\Longrightarrow}_t z_j$ in T_t if there exists $\Psi_t(X_i) = z_i$ and $\Psi_t(X_j) = z_j$ such that $X_i \overset{\sigma}{\Longrightarrow}_{h/\approx} X_j$ in $T_{h/\approx}$,
- $\Upsilon_t : Q_t \to O$ is the observation map defined by $\Upsilon_t(z_i) = \Upsilon_{h/\approx}(X_i)$.

Transition system T_t is finite since Q_t, Σ_t, and O are finite. Note that Σ_h and O are inherited from $T_{h/\approx}$. Similarly, one can show that the quotient transition system $T_{h/\approx}$ is bisimilar to the finite transition system T_t by the relation

$$R_t = \left\{(X_i, z_i) \in \pi \times Q_t \mid \Psi_t(X_i) = z_i\right\}, \tag{5.15}$$

between $T_{h/\approx}$ and T_t. The relationship between the state-space of the different transition systems T_h, $T_{h/\approx}$, and T_t is shown in Figure 5.7.

[2] Informally, an isomorphism is a map that preserves sets and relations among elements, i.e a structure-preserving mapping.


Discrete Transition System T_t
Abstraction
Quotient Transition System $T_{h/\approx}$
Abstraction
Infinite Transition System T_h
Embedding
Hybrid Automaton H
Z
Ψ_t
$X_{/\approx}$
Ψ_h
X
X

(a) *Relationship between transition systems associated with linear control system* Σ.

(b) *Corresponding state-spaces.*

Figure 5.6: *Intermediate steps in the abstraction of hybrid automaton* H *to obtain a finite bisimilar quotient* T_t.

5.8 Constructing the Timed Automaton

In this section, we will show how to construct a timed automaton A_r such that the reachability properties of the finite and discrete transition system T_t are preserved. Since T_t and $T_{h/\approx}$ are bisimilar and $T_{h/\approx}$ and T_h are bisimilar the reachability properties are preserved. Therefore, any sequence of actions that can be accepted by timed automaton A_r can also be accepted by hybrid automaton H.

5.8.1 Modeling the Environment

In UPPAAL the set of discrete states Z is represented in an occupancy table. The occupancy table is represented as a two-dimensional boolean array

$$\texttt{int[0,1] Z[Z\_1][Z\_2]}, \tag{5.16}$$

where `Z_1`, `Z_2` $\in \mathbb{Z}$ define the size of the array in the x_1 and x_2-direction, respectively. Thus, elements of the array represent discrete positions, where each discrete position can be assigned the value 0 (free) or 1 (occupied). A particular element $(1,2)$ of the array `Z` is marked occupied by the assignment `Z[1][2] = 1`. By default all elements of the array `Z` are initialized to zero.

Static obstacles may be present in the environment where the robots are moveing. A static obstacle is modeled as an automaton

$$A_o = (L, l_0, E), \tag{5.17}$$

where

- $L = \{l_0, l_1\}$ is the set of locations,
- $l_0 \in L$ is the initial location,
- $E \subseteq L \times L$ is the set of edges, where an edge contains a location and a target location. The edges are defined as

$$e_{00} = (l_0, l_0),$$
$$e_{01} = (l_0, l_1).$$

Automaton A_o modeling one static obstacle is graphically shown in Figure 5.7.

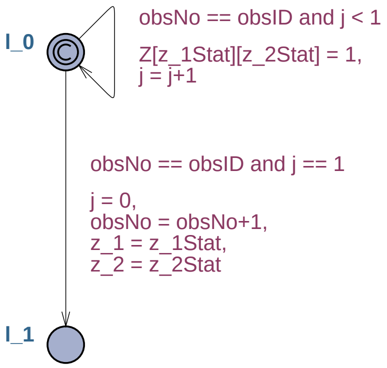


Figure 5.7: *Process template for one static obstacle.*

Automaton A_o starts in the location l_0, which is declared committed[3]. By declaring this location committed, an element in the array `Z` can be marked as occupied by an obstacle, without allowing any time delay in location l_0. When the guard `obsNo == obsId and j < 1` is enabled, the assignment `Z[z_1Stat][z_2Stat] = 1` is performed and the index variable `j` is incremented. The edge from l_0 to l_1 will then become fired since the guard `obsNo == obsID and j = 1` is satisfied, resulting in an update of index variable `j` to zero, an increment of `obsNo` and the obstacle with `obsNo == obsId` is given a static discrete position by the assignments `z_1 = z_1Stat` and `z_2 = z_2Stat`.

This automaton is used as a template for declaring static obstacles processes. Processes declared using the static obstacle template can be declared with the template parameters specified in Table 5.1.

Parameter	Type	Description
`obsID`	const int	*Unique identifier for static obstacle*
`z_1Stat`	const int	*Static position of obstacle in x_1-direction*
`z_2Stat`	const int	*Static position of obstacle in x_2-direction*

Table 5.1: *Template parameters for one static obstacle.*

[3] Committed locations are represented by Ⓒ

5.8.2 Timed Automaton Model of Robot

Recall, that the goal is to generate a set of collision-free paths for the network of multi-modal robots which satisfy a formal requirement specification in CTL and which enable the network of multi-modal robots to eventually reach their goal positions. In CTL, the *"eventually reach goal position"* property is specified as a liveness property whereas the *"collision-avoidance"* property is specified as a safety property. However, since CTL does not allow nesting of liveness and safety properties the liveness and safety properties can not be checked together. This problem is solved in two steps: I.) The collision-avoidance property is guaranteed by using a correct-by-construction principle where the collision-avoidance property is embedded in the timed automaton modeling a multi-modal robot. II.) The eventually reach goal position property is ensured for each multi-modal robot in the network by using the model checker UPPAAL.

A timed automaton template is now constructed from the finite transition system T_t. Since the multi-modal robots in the network have fixed speed $v = v_1 = v_2$ and the length of every side of the cells is $\epsilon = 1$, we can choose $\rho = \frac{1}{v}$ so that for any initial condition $x(0)$ in a cell a robot can move to an adjacent cell by taking a proper cyclic transition. The timed automaton associated with T_t is defined as

$$A_r = (L, l_0, C, A, E, I), \tag{5.18}$$

where

- $L = \{l_0, l_1, l_2, l_3, l_4, l_5\}$ is the set of locations,
- $l_0 \in L$ is the initial location,
- $C = \{\texttt{c}\}$ is the set of real-valued clock variables,
- $A = \{\texttt{sigma\_2?}, \texttt{sigma\_3?}, \texttt{sigma\_4?}, \texttt{sigma\_5?}\}$ is the set of input synchronization actions,
- $E \subseteq L \times B(C) \times A \times 2^C \times L$ is the set of edges, where an edge contains a source location, a guard to be satisfied, an synchronization action to be received, a clock variable to be reset, and a target location. The edges[4]

[4] e_{ij} denote the edge from location l_i to l_j.

are defined as

$$\begin{aligned} e_{00} &= (l_0, l_0), \\ e_{01} &= (l_0, l_1), \\ e_{i1} &= (l_i, \texttt{c == c\_m}, \texttt{c = 0}, l_1), \quad \text{for } i = 2, \ldots, 5, \\ e_{1j} &= (l_1, \texttt{sigma\_j?}, l_j), \quad \text{for } \texttt{j} = 2, \ldots, 5, \end{aligned}$$

- $I : L \to B(C)$ assigns to each location $l \in L$ an invariant $I(l)$. The invariant is defined as

$$I(l_i) \ : \ \texttt{c <= c\_m}, \quad \text{for } i = 2, \ldots, 5.$$

The timed automaton A_r modeling one multi-modal robot is graphically illustrated in Figure 5.8.

The state of the timed automaton is $(l, u) \in L \times C$, where $l \in L$ is a location and $u : C \to \mathbb{R}_{\geq 0}$ is a clock valuation function from the set of clocks $C = \{\texttt{c}\}$ to the nonnegative reals $\mathbb{R}_{\geq 0}$. Thus, u holds the current value of the clock variable c in location l.

Timed automaton A_r is equipped with a real-valued clock variable c in order to take the timing constraint related to the transition $z_i \overset{\sigma}{\Longrightarrow}_t z_j$ for some $\sigma \in \Sigma_h$ into consideration. Thus, a real-valued clock variable c is used to represent the amount of time a robot spends on moving from $z_i = \Psi_t(X_i)$ to $z_j = \Psi_t(X_j)$.

Assume that the robot can move with a fixed velocity that is given by c_m $\in \mathbb{Z}$. In the timed automaton this is modeled using an invariant on the location.

The timed automaton A_r starts in location l_0. In this location the robot is placed on its initial discrete position as specified by z_1Init and z_2Init. In location l_1 the robot can move from the initial cell to an adjacent cell in the partition given that one of the edges are fired and the associate controller sends the corresponding output synchronization action. In location l_1 the timed automaton is ready to receive an input synchronization action sigma_i! for i $= 2, \ldots, 5$ from the associated controller. If the edge e_{12} is fired and the synchronization action sigma_2! is received the timed automaton fires the edge e_{12} to location l_2. Note that the edge e_{12} is only fired if the adjacent cell is free Z[z_1+1][z_2] == 0 and within the defined partition, i.e. z_1 < Z_1. The adjacent cell is marked occupied Z[z_1][z_2]=1 when the edge e_{12} is

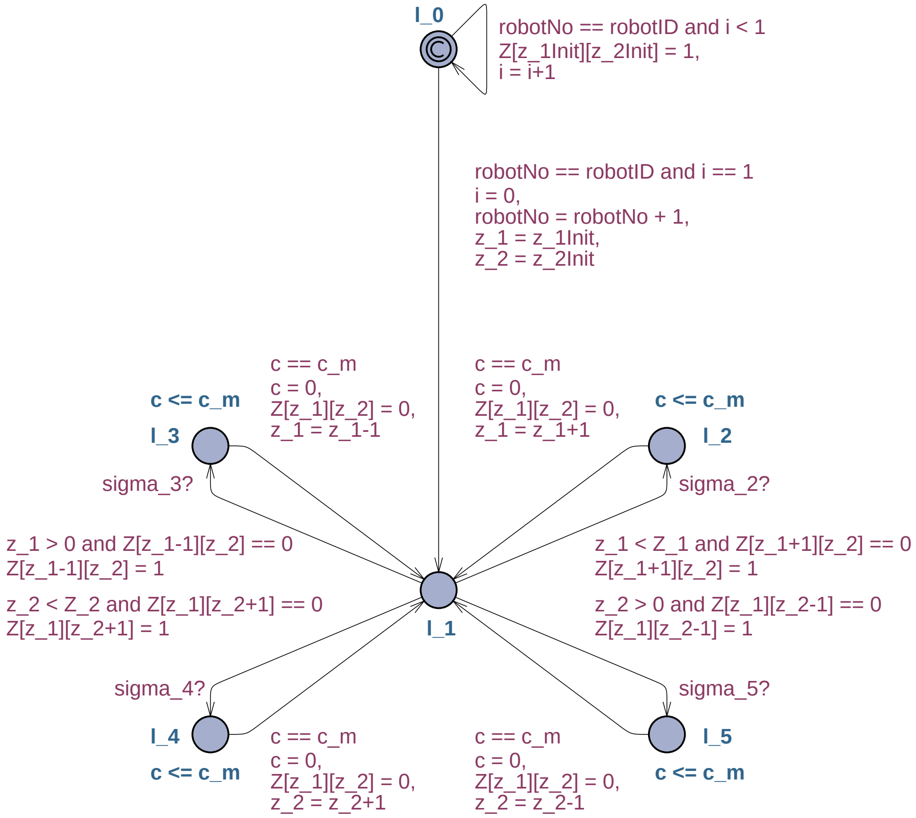


Figure 5.8: *Template for one multi-modal robot.*

fired. In location l_2 the movement towards the adjacent cell is performed for a fixed period of time, i.e. as long as the invariant `c <= c_m` is satisfied. Then, the edge e_{21} is fired when the guard `c == c_m` is enables and a transition is taken back to location l_1. When the edge e_{21} is fired the clock variable `c` is reset to zero, the previous cell is marked free `Z[Z_1][Z_2] = 0` and the discrete position of the robot is updated `z_1 = z_1 + 1`.

Processes declared using the robot template can be declared with the template parameters specified in Table 5.2.

Parameter	Type	Description
`robotID`	const int	*Unique identifier for robot*
`z_1Init`	const int	*Initial position of robot in x_1-direction*
`z_2Init`	const int	*Initial position of robot in x_2-direction*
`c_m`	const int	*Constraint on clock c*
`sigma_2`	chan	*Synchronization channel to move robot in x_1-direction*
`sigma_3`	chan	*Synchronization channel to move robot in $-x_1$-direction*
`sigma_4`	chan	*Synchronization channel to move robot in x_2-direction*
`sigma_5`	chan	*Synchronization channel to move robot in $-x_2$-direction*

Table 5.2: *Template parameters for one robot.*

5.8.3 Automaton Model of Robot Controller

A controller is associated with each timed automaton modeling a multi-modal robot. A controller for each robot is needed as the system consists of a network of concurrent robots moving in the environment. The robot controller is modeled as an automaton

$$A_c = (L, l_0, A, E), \tag{5.19}$$

where

- $L = \{l_0\}$ is the set of locations,
- $l_0 \in L$ is the initial location,
- $A = \{\texttt{sigma\_2!}, \texttt{sigma\_3!}, \texttt{sigma\_4!}, \texttt{sigma\_5!}\}$ is the set of output synchronization actions,
- $E \subseteq L \times A \times L$ is the finite set of edges, where an edge contains a source location, an output synchronization action to be send, and a target location. The edges are defined as

$$e_i = (l_0, \texttt{sigma\_i!}, l_0), \quad \text{for } \texttt{i} = 2, \ldots, 5.$$

The robot controller automaton is shown in Figure 5.9.

The automaton A_c start in the location l_0. In this location the automaton can send an output synchronization action `sigma_i!` $\in A$ can be sent to the timed

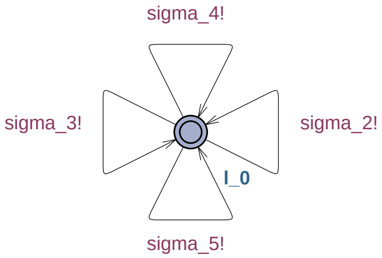


Figure 5.9: *Template for one robot controller.*

automaton A_r modeling a multi-modal robot. The set of output synchronization actions A represent all possible movements of a robot.

Complementary synchronization channels are of the form (`sigma_i!`, `sigma_i?`), where `sigma_i!` is the sender and `sigma_i?` is the receiver. Thus, a robot and its associate control can synchronize on complementary synchronization channels if their respective invariants and guards are satisfied. Thus, if the automaton A_c takes the edge

$$e_2 = (l_0, \texttt{sigma\_2!}, l_0), \tag{5.20}$$

the timed automaton A_r will take the corresponding edge

$$e_{12} = (l_1, \texttt{sigma\_2?}, l_j). \tag{5.21}$$

The automaton A_c is used as a template for declaring control process instances. Processes declared using the controller template can be declared with the template parameters specified in Table 5.3.

Parameter	**Type**	**Description**
`sigma_2`	chan	*Synchronization channel to move robot in x_1-direction*
`sigma_3`	chan	*Synchronization channel to move robot in $-x_1$-direction*
`sigma_4`	chan	*Synchronization channel to move robot in x_2-direction*
`sigma_5`	chan	*Synchronization channel to move robot in $-x_2$-direction*

Table 5.3: *Template parameters for robot controller.*

5.9 Summary

A novel framework for the motion planning of a network of multi-modal robots with respect to formal requirement specification in Compuatonal Tree Logic (CTL) was presented in this chapter. CTL enables to express the desired liveness and safety properties that the robots in the network should satisfy.

The robots were assumed to move concurrently in the environment and to have identical motion capabilities. A hybrid automaton was introduced as a generic model for each of the multi-modal robots in the network. The embedding of the hybrid automata into the class of labeled transition systems was possible by introducing cyclic transitions and by considering a finite partition of the environment that conforms with the motion capabilities of the robots.

Subsequently, the finite bisimilar quotient obtained from the hybrid automaton was used to construct a timed automaton for the robot. The quotient of the hybrid automata was obtained in such a way that the reachability properties of the hybrid automata were preserved by the timed automata. The constructed timed automaton serves as a template for instantiating new robot processes. Also, a controller and obstacle template is constructed.

The timed automata formalism merely presents an abstraction of a robot but allows composition and formal symbolic reasoning about coordinated motion planning solutions. The model checker UPPAAL is used for formal symbolic model checking against a requirement specification formulated in CTL for a network of multi-modal robots.

Chapter 6

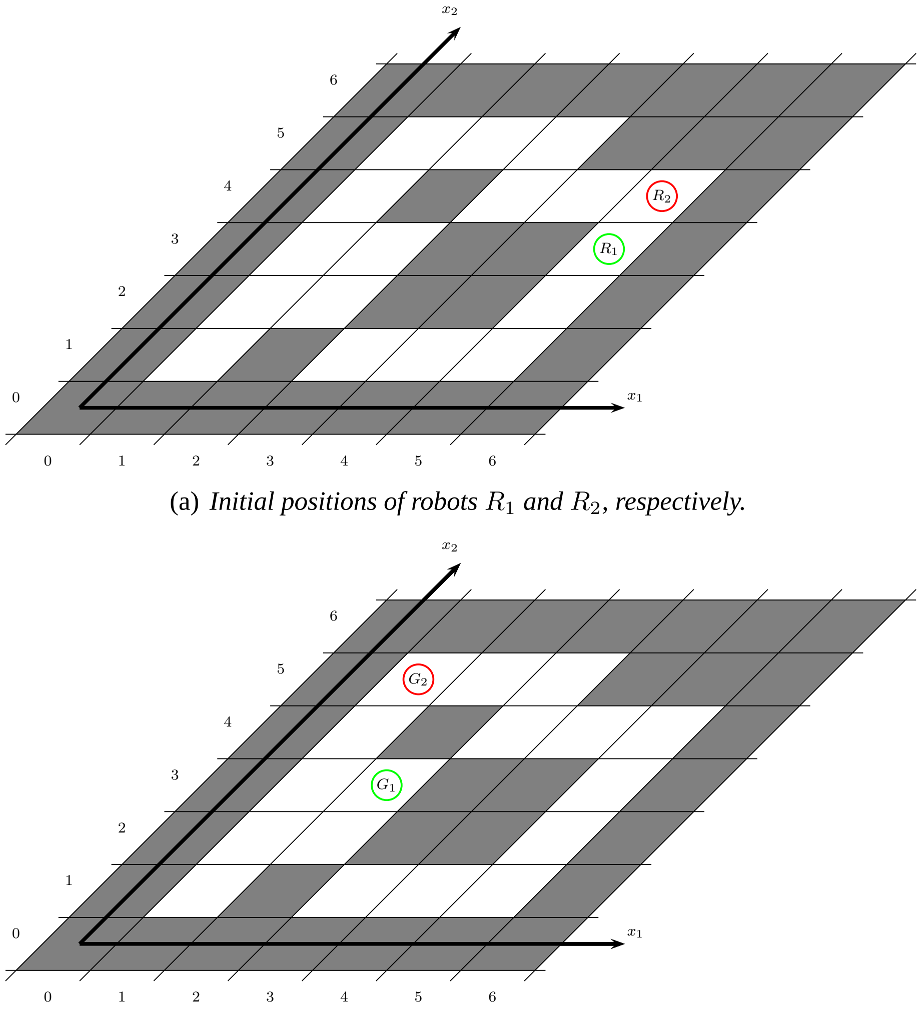
Case Study I : Multi-robot Motion Planning

In this chapter the novel framework proposed for the motion planning of a network of multi-modal robots, presented in the previous chapter will be demonstrated for a network of two multi-modal robots in a simple test scenario. The requirement specification expressing the desired liveness and safety properties for the network of multi-modal robots is formulated in Computational Tree Logic (CTL) and subsequently checked using the model checker UPPAAL. *The results of model checking the system are presented. Finally, a conclusive discussion of the novel framework proposed is given.*

6.1 Test Scenario

In the following a network of two multi-modal robots, R_1 and R_2 is considered. The network of robots is shown in Figure 6.1.

The two robots R_1 and R_2 are initially located in cell (5,3) and (5,4), respectively (See Figure 6.1.(a)). The goal positions of the two robots are marked as G_1 and G_2, respectively (See Figure 6.1.(b)). The two robots have to move from their initial to goal positions while avoiding collision with each other and the static obstacles. The system to be model checked consists of the following processes: Two robots, R_1 and R_2 two controllers, C_1, and C_2 and static obstacles $O_1, \ldots, O_{32}$, marked as grey. In UPPAAL the system is defined as `system R_1, R_2, C_1, C_2, O_1, ..., O_32;`. The global declarations for



(a) *Initial positions of robots R_1 and R_2, respectively.*

(b) *Goal positions G_1 and G_2 of robots R_1 and R_2, respectively.*

Figure 6.1: *Network with two multi-modal robots R_1 and R_2.*

the system just defined can be found in Appendix A. The network is constructed in UPPAAL, where process assignments are used to declare instances of the robot, control, and obstacle processes, respectively.

6.2 Requirement Specification in Computational Tree Logic

The requirement specification expressing the desired liveness and safety properties for the network of multi-modal robots is formulated in Computational Tree Logic (CTL) and checked using the model checker UPPAAL.

6.2.1 Liveness Properties

The liveness properties are used for the generation of a feasible motion plan, that will move the robots from their initial to goal positions, while avoiding collision among robots and obstacles.

Property 1 (Reachability) There exist a location trajectory, where the robots, R_1 and R_2 eventually reach their goal positions, G_1 and G_2?

```
E<> (R_1.z_1==2 and R_1.z_2==3 and R_2.z_1==1
and R_2.z_2==5)
```

Property 2 (Reachability with time requirement) There exist a location trajectory, where the robots, R_1 and R_2 eventually reach their goal positions, G_1 and G_2, within 15 time units?

```
E<> (R_1.z_1==2 and R_1.z_2==3 and R_2.z_1==1
and R_2.z_2==5 and time<15)
```

Property 1 express the behavior that the robots eventually will reach their goal positions, where Property 2 express the behavior that the robots eventually will reach their goal positions within 15 time units.

6.2.2 Safety Properties

The safety properties are used to check if collision avoidance is achieved among the robots when moving and static obstacles and also that the robots will move within then boundaries of the environment.

Property 3 (Collision avoidance) For all location trajectories the robots, R_1 and R_2 will never collide after they start to move, i.e. for time > 0?

```
A[] not (R_1.z_1==R_2.z_2 and R_1.z_1==R_2.z_2
and time > 0)
```

Property 4 (Bounded movement) For all location trajectories the robots, R_1 and R_2 will always move within the boundaries of the partition?

```
A[] (R_1.z_1>=0 and R_1.z_1<=Z_1 and R_1.z_2>=0
and R_1.z_2<=Z_2 and R_2.z_1>=0 and R_2.z_1<=Z_1
and R_2.z_2>=0 and R_2.z_2<=Z_2)
```

Property 3 express the requirement that collision avoidance is achieved among the robots once they start to move. Further, the requirement that the robots always move within the boundaries of the partition is expressed in Property 4.

In UPPAAL the properties Property 1-4 are verified by typing `verifyta -n0 -o0 -f trace -t2 -u model.xml query.q`, where `-n0` select automatic extrapolation, `-o0` select breadth first search order, `-f trace` write symbolic trace to file trace-n.xtr[1], `-t2` generate fastest trace, `-u` show summary[2] after verification, `model.xml` is the model to be verified against the CTL formulas as specified in the query file `query.q`. The setup is graphically illustrated in Figure 6.2.

6.3 Model Checking Results

In the following the results from the model checking of the liveness and safety properties for the system are presented.

Property 1 (Reachability) was checked and satisfied, hence there exists a location trajectory, where the robots, R_1 and R_2 eventually reach their goal positions, G_1 and G_2 (See Figure 6.6). The symbolic trace for the network of two multi-modal robots, R_1 and R_2 and their associate controllers, C_1 and C_2 is shown in Figures 6.3-6.5.

The symbolic trace in Figures 6.3-6.5 shows the fastest-time location trajectory, i.e. a trajectory with the shortest accumulated time delay. The location trajectory for the obstacles is omitted since the obstacles will not move once they are placed in the environment. The robots start in the initial location l_0. In this

[1] n denotes the symbolic trace of property n.

[2] States stored/explored.

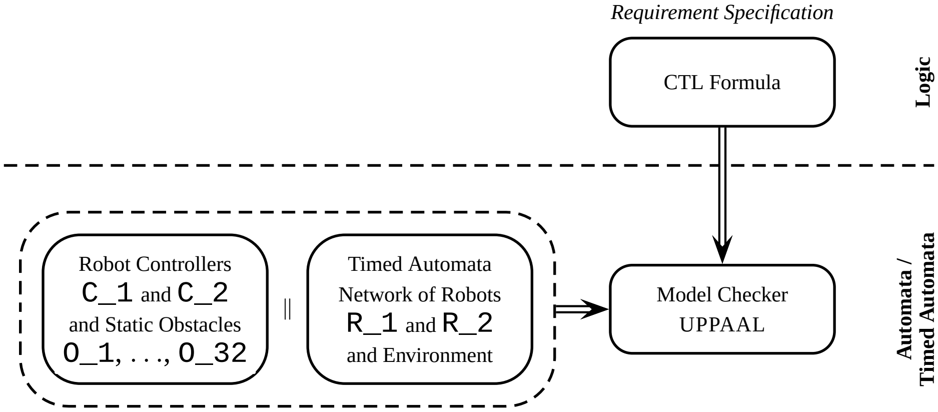


Figure 6.2: *Setup for model checking liveness and safety proterties for the network of two multi-modal robots R_1 and R_2.*

location the robots are placed in their initial position in the partition as specified in the System Declarations (See Appendix A). Once the robots are placed in the environment they enter the location l_1, where they are ready to move upon a synchronization signal from their respective controller. The dashed lines indicate a synchronization with an event between a robot and its associate controller.

Property 2 (Reachability with time requirement) was checked and satisfied, hence there exists a location trajectory, where the robots, R_1 and R_2 eventually reach their goal positions, G_1 and G_2 within 15 time units. The location trajectory is identical to the location trajectory generated for Property 1. Robot R_1 reaches its goal position after 9 time units, whereas robot R_2 reaches its goal position after 11 time units.

Property 3 (Collision avoidance) was checked and satisfied. Thus, the collision avoidance among the robots and the obstacles is guaranteed. The collision avoidance property is guaranteed by using a correct-by-construction principle by embedding the collision-avoidance property in the timed automaton template for a multi-modal robot A_r. This property is satisfied using a guard `Z[z_1+1][z_2] == 0` (is adjacent cell in x_1-direction free?) and an update `Z[z_1+1][z_2]` (occupy adjacent cell) before moving in the x_1-direction.

Property 4 (Bounded movement) was checked and satisfied, hence the robots will always move within the boundaries of the partition. This property is satisfied using the guards `z_1 < Z_1`, `z_1 > 0`, `z_2 < Z_2`, and `z_2 > 0` in the

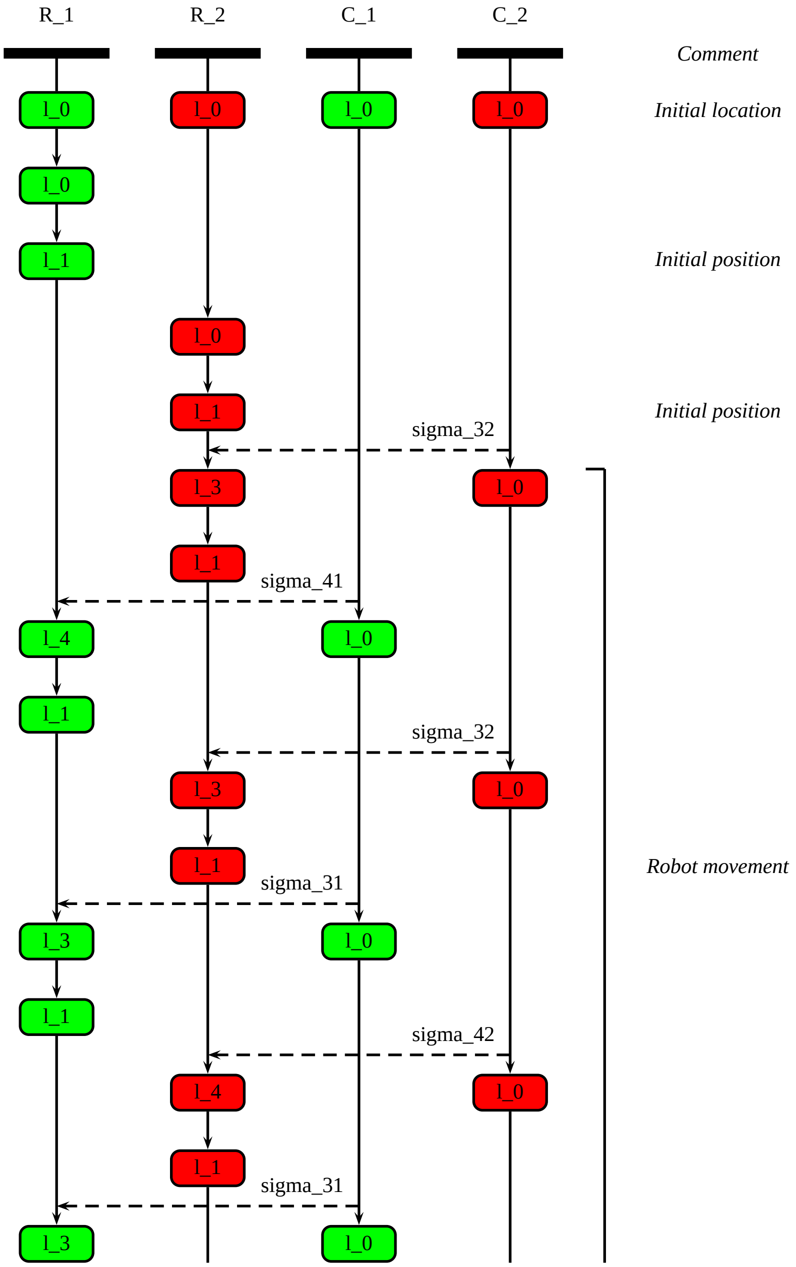


Figure 6.3: *Symbolic trace (part I) for the network of two robots, R_1 and R_2 and their associate controllers, C_1 and C_2.*

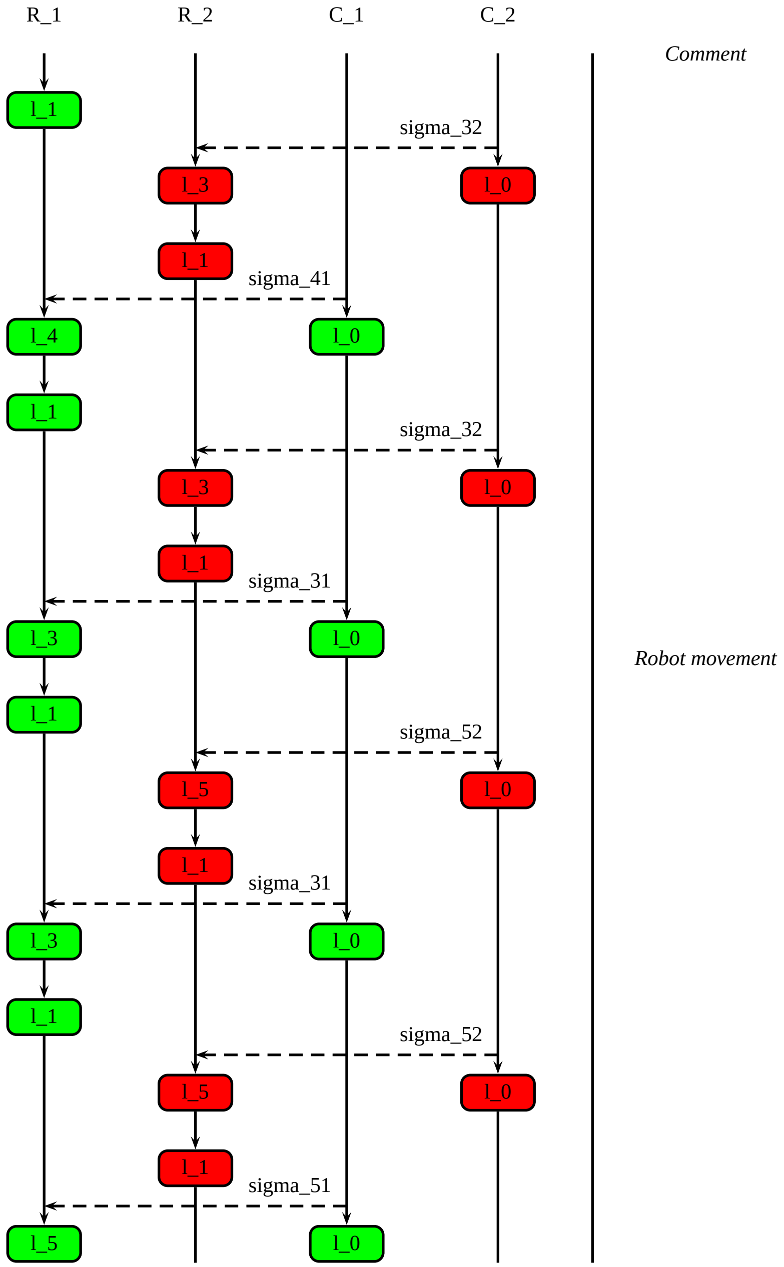


Figure 6.4: *Symbolic trace (part II) for the network of two robots, R_1 and R_2 and their associate controllers, C_1 and C_2.*

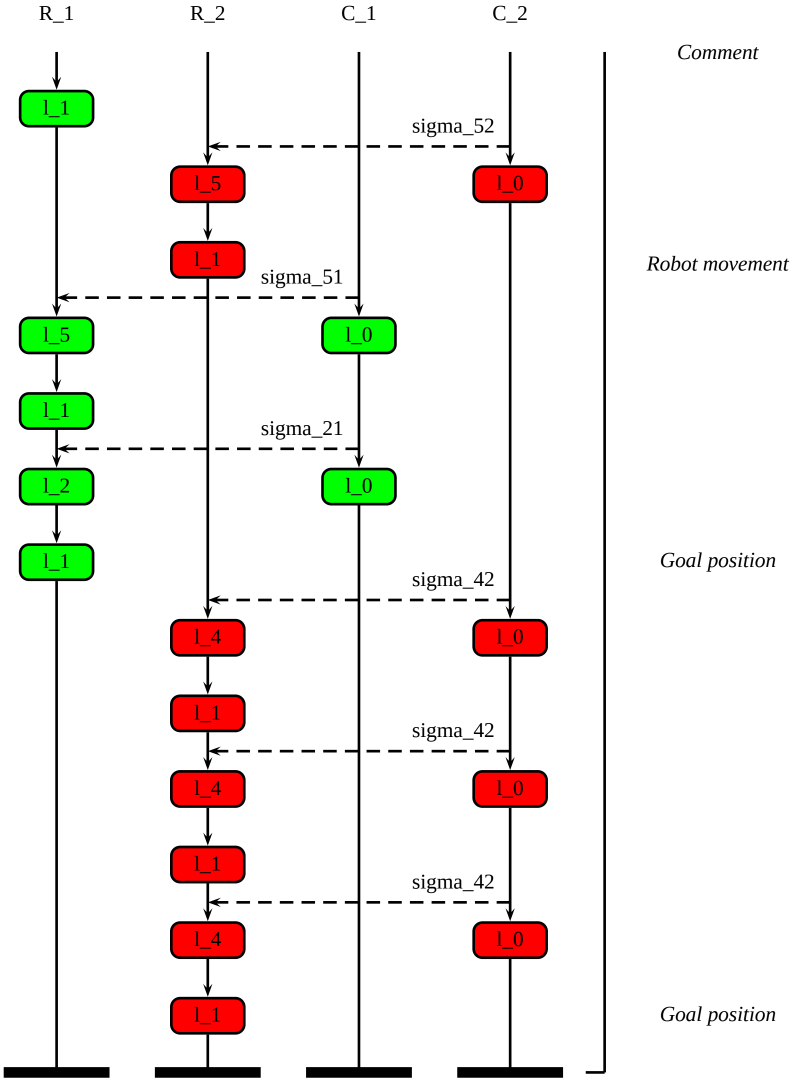
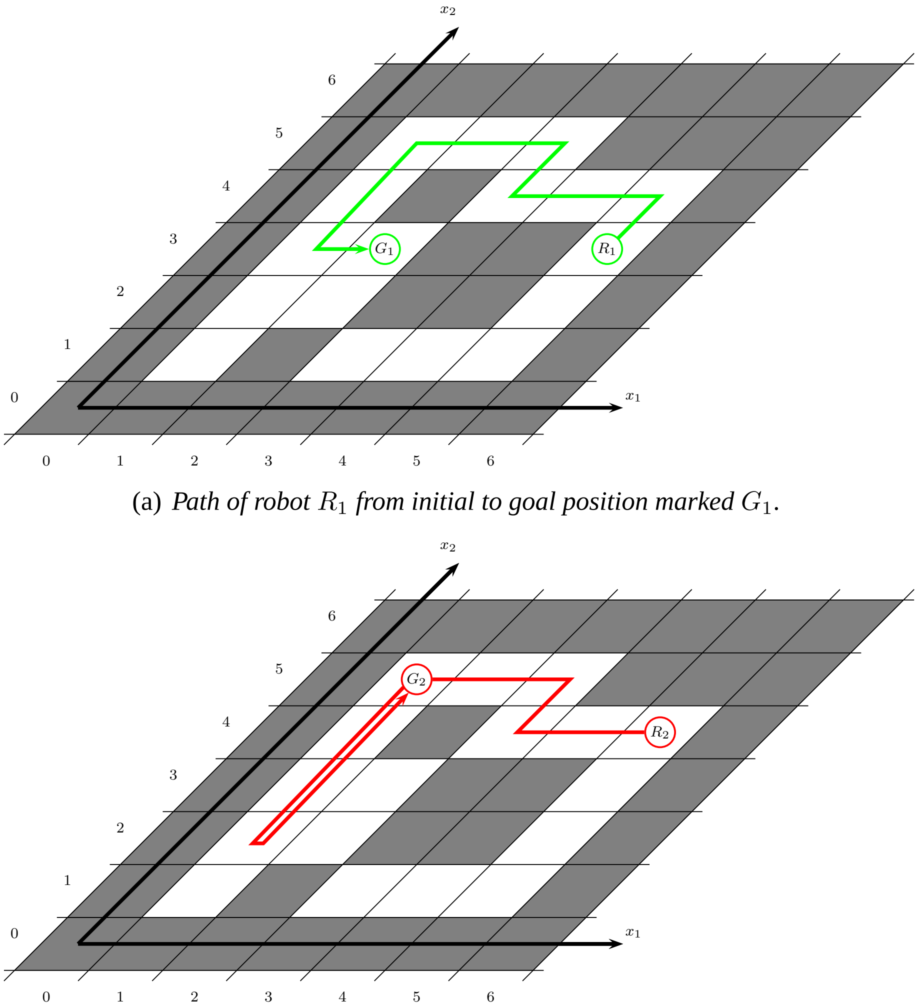


Figure 6.5: *Symbolic trace (part III) for the network of two robots, R_1 and R_2 and their associate controllers, C_1 and C_2.*



(a) *Path of robot R_1 from initial to goal position marked G_1.*

(b) *Path of robot R_2 from initial to goal position marked G_2.*

Figure 6.6: *Network with two multi-modal robots R_1 and R_2.*

timed automaton for the robot A_r (See Figure 5.8).

A summary of the results of model checking the liveness and safety properties for the system is shown in Table 6.1.

Property	**Description**	**Property satisfied**	**States stored/ explored**	**Time [s]**
1	*Reachability*		2654/2813	< 1
2	*Reachability with time requirement*		2654/2813	< 1
3	*Collision avoidance*		5059/5218	< 1
4	*Bounded movement*		5059/5218	< 1

Table 6.1: *Results from model checking liveness properties 1 and 2 and safety properties 3 and 4.*

6.4 Concluding Discussion

The novel framework presented here presupposes an infra-structure of the multi-modal robots with feedback controllers that constraint the motion capabilities of the individual robots. A natural next step would be to apply the proposed framework to networks of physical robot with more complex dynamics. However, this would require the development of a (hybrid) controller for tracking the planned paths.

Further, the novel framework presented for the motion planning of a network of multi-modal robots is also applicable to robots moving in a three-dimensional (3D) environment. This requires the hybrid automata modeling a multi-modal robot to have two additional states for moving in the x_3 and $-x_3$-direction, respectively. However, this would result in an increased computational complexity of subsequently model checking the system.

Simulations have shown that the computational complexity of model checking the system increases exponential as the number of multi-modal robots in the network and the size of the occupancy table increases. Basically, UPPAAL makes an extensive search of the state-space when model checking the system with respect to a requirement specification in CTL. In order to make the proposed framework applicable for large networks of multi-modal robots (> 3) an extensive search of the state-space should be avoided or substantially reduced.

To avoid an extensive search of state-space and thereby reducing the computational complexity of model checking the system the following approach seems feasible.

UPPAAL CORA[3] [Uppsala University and Aalborg University, 1995b] uses an extension of timed automata called Linearly Priced Timed Automata which allows to annotate the timed automaton model with the notion of cost. The idea is then be to have a low cost for moving in a direction that would bring the robot closer to its goal position, opposed to a high cost if moving in a direction that brings the robot away from the goal position. Subsequently, UPPAAL CORA could be used to find the most optimal location trajectory, i.e. a location trajectory with the lowest accumulated cost for each robot.

6.5 Summary

In this chapter the novel framework proposed for the motion planning of a network of multi-modal robots, presented in the previous chapter was demonstrated for a network of two multi-modal robots in a simple test scenario. Computational Tree Logic (CTL) was used to express the desired liveness and safety properties for the network of multi-modal robots. The model checker UPPAAL was used for checking the desired properties which all showed to be satisfied by the system. The result of model checking the system was a set of collision-free paths for the network of multi-modal robots in the form of sequences of synchronization inputs such that the requirement specification was satisfied. Subsequently, the sequences of synchronization inputs could be executed by the network of multi-modal robots $H_1, \ldots, H_N$ modeled by hybrid automata. Finally, a conclusive discussion of the novel framework proposed was given.

[3] A branch of UPPAAL for cost optimal reachability analysis.

Chapter 7

Robot Controller Synthesis

In this chapter a framework for controller synthesis for linear control systems with respect to formal requirement specification in Linear Temporal Logic (LTL) is presented. Linear control systems satisfying simple controllability assumptions allow finite abstractions in the form of finite bisimilar quotients to be computed. The possibility to compute finite bisimulations of linear control systems allows a discrete controller to be synthesized. A refinement of the discrete controller results in a hybrid closed-loop combining the continuous dynamics of linear control system with the synthesized switching logic required to implement the desired requirement specification.

The framework presented in this chapter was developed by Tabuada and Pappas [Tabuada and Pappas, 2006] and illustrated in Figure 7.1. In order to fit with the problem of synthesizing a controller for a mobile robot, given a requirement specification in LTL, it needs some modifications. These modifications are primary concerned with the input to the framework and with the implementation of the linear hybrid system obtained by refinement.

The framework in Figure 7.1 assumes a discrete-time (time-invariant) controllable linear control system Σ together with a requirement specification in the form of a LTL formula ϕ. The idea behind the framework is to synthesize a controller for the linear control system enforcing the requirement specification. This is done in a number of steps to be described later. The controller synthesis results in a closed-loop hybrid system T_H enforcing the requirement specification, i.e. $T_H \models \phi$.

To make the framework proposed by Tabuada and Pappas [Tabuada and Pap-

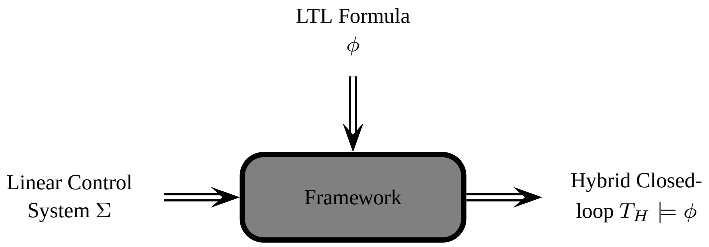

LTL Formula
ϕ
Linear Control System Σ
Framework
Hybrid Closed-loop $T_H \models \phi$

Figure 7.1: *Proposed framework for controller synthesis for linear control system Σ with respect to requirement specification in Linear-time Temporal Logic (LTL).*

pas, 2006] applicable for synthesizing a controller for a mobile robot the following modifications are necessary.

Modeling In the context of a nonholonomic mobile robot the kinematic model is appropriately represented by a nonlinear system compared to a discrete-time linear control system.

Implementation The linear hybrid system obtained from the refinement should equivalently work for the nonlinear system.

The extended framework for controller synthesis with respect to formal requirement specification is depicted in Figure 7.2.

The extended framework involves the following steps that will be described below.

(1) Modeling It is assumed that the system under consideration can be modeled as a nonlinear system Σ_*.

(2) Dynamic Feedback Linearization Using dynamic feedback linearization a dynamic compensator is obtained. This results in a continuous-time linear control system Σ_z. Subsequently, a discrete-time equivalent Σ of linear control system Σ_z is obtained as baseline for the abstraction.

(3) Requirement Specification LTL is used as a requirement specification mechanism for describing the desired behavior of transition system T_Σ associated with linear control system Σ.

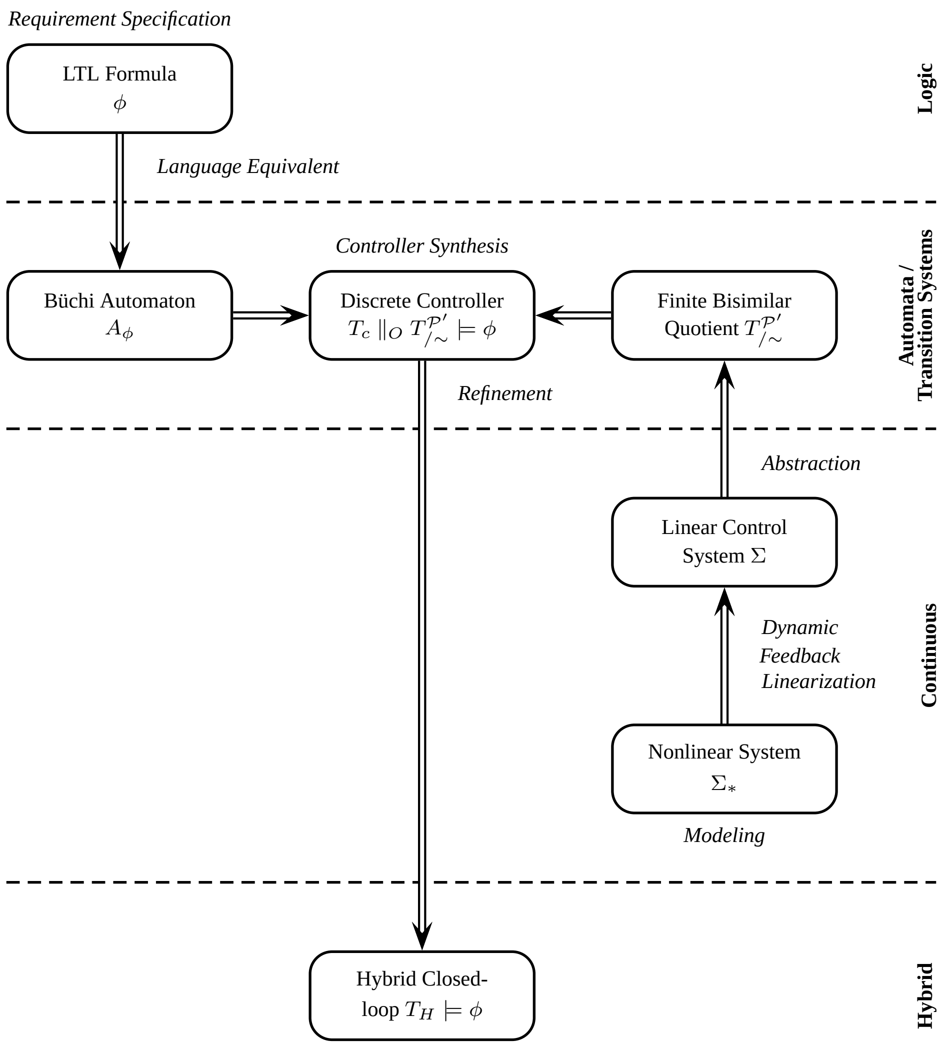


Figure 7.2: *Extended framework for controller synthesis for linear control system Σ with respect to formal requirement specification in Linear-time Temporal Logic (LTL).*

(4) Büchi Automaton The requirement specification is given in the form of a LTL formula ϕ. The LTL formula ϕ is translated into a Büchi automaton A_ϕ that is used for controller synthesis.

(5) Abstraction The linear control system Σ is embedded in the class of transition systems with observations. This results in a transition system T_Σ associated with linear control system Σ. A finite bisimilar quotient $T^{\mathcal{P}'}_{/\sim}$ of transition system T_Σ is obtained by forming a finite partition of observation-space $\mathbb{R}^m$ of the linear control system Σ.

(6) Controller Synthesis Given a finite bisimilar quotient $T^{\mathcal{P}'}_{/\sim}$ of the linear control system Σ and a requirement specification in the form of a Büchi automaton A_ϕ, a finite controller T_c for $T^{\mathcal{P}'}_{/\sim}$ is synthesized such that the parallel composition $T_c \parallel_O T^{\mathcal{P}'}_{/\sim}$ with observation synchronization enforces the requirement specification, i.e. $T_c \parallel_O T^{\mathcal{P}'}_{/\sim} \models \phi$, meaning that $T_c \parallel_O T^{\mathcal{P}'}_{/\sim}$ satisfies ϕ.

(7) Refinement The discrete model $T_c \parallel_O T^{\mathcal{P}'}_{/\sim}$ is refined with continuous inputs such that a closed-loop hybrid system T_H will satisfy the requirement specification by construction i.e. $T_H \models \phi$.

(8) Implementation Linear Hybrid System The linear hybrid system H is implemented as a simulation model for verifying the correctness of the synthesized controller (Note that this step is not shown in Figure 7.2).

7.1 Modeling

It is assumed that the system under consideration can be modeled as a generic nonlinear system without drift of the form

$$\Sigma_* \ : \quad \dot{q} = G(q)w, \tag{7.1}$$

with state $q \in \mathbb{R}^n$ and input $w \in \mathbb{R}^m$. $G(q) \in \mathbb{R}^{n\times m}$.

7.2 Dynamic Feedback Linearization

Dynamic feedback linearization of a non-linear system representing the plant is employed to obtain a linear control system. Given a nonlinear system as in

Eq. (7.1) the purpose is to find a feedback compensator of the form

$$\dot{\xi} = \alpha_1(q, \xi) + \alpha_2(q, \xi)\nu \tag{7.2}$$

$$w = \beta_1(q, \xi) + \beta_2(q, \xi)\nu, \tag{7.3}$$

with state $\xi \in \mathbb{R}^\zeta$ and input $\nu \in \mathbb{R}^m$ such that the closed-loop system in Eq. (7.1) and Eq. (7.2) is equivalent, under a state transformation $z = T(q, \xi)$ to a linear system $\Sigma_z = (A_z, B_z)$. We now review the steps involved in dynamic feedback linearization [Oriolo et al., 2002]. The first step is to choose a desired m-dimensional output $\eta = h(q)$. The output η is successively differentiated until the input $\nu \in \mathbb{R}^m$ appears in a nonsingular way. To avoid subsequent differentiation of the original inputs $w \in \mathbb{R}^m$ the concept of dynamic extension is employed, where additional integrators with state $\xi_i \in \mathbb{R}$ are added to some of the input channels of the system in Eq. (7.1). The system in Eq. (7.1) with extended state-space $\mathbb{R}^{n+\zeta}$ is then full input-state-output linearizable if the sum of the differentation orders of the output $\eta \in \mathbb{R}$ equals the dimension $n + \zeta$. The resulting closed-loop system in Eq. (7.1) and Eq. (7.2) is then equivalent to a set of m decoupled input-output chains of integrators from input $\nu_i \in \mathbb{R}$ to output $\eta_i \in \mathbb{R}$ for $i \in 1, \ldots, m$. Defining the state as $z \in \mathbb{R}^n$ the following continuous-time linear control system is obtained

$$\Sigma_z \ : \ \dot{z}(t) = A_z z(t) + B_z \nu(t), \tag{7.4}$$

with system matrices $A_z \in \mathbb{R}^{n \times n}$ and $B_z \in \mathbb{R}^{n \times m}$ and state variable $z \in \mathbb{R}^n$ and control variable $\nu \in \mathbb{R}^m$.

7.3 Requirement Specification

Temporal logic is usually used as a specification mechanism in verification of formal models. Properties about the behavior of a system over time are naturally expressible in temporal logics, such as linear-time temporal logic (LTL).

In this context, LTL is used as a specification mechanism for expressing the desired behavior of transition system $T_\Sigma^\mathcal{P}$. Each predicate $p \in \mathcal{P}$ corresponds to an element of a finite partition $\mathcal{P}$ of observation-space $\mathbb{R}^m$ of transition system T_Σ.

In this section LTL syntax and semantics are defined.

7.3.1 LTL Syntax and Semantics

The construction of LTL formulas is based on a finite set $\mathcal{P}$ of state predicates. More complex LTL formulas can then be constructed from the set $\mathcal{P}$. In this context the set $\mathcal{P}$ is identified with a finite partition of observation-space $\mathbb{R}^m$ of transition system $T_{\Sigma}^{\mathcal{P}}$. Thus, each predicate $p \in \mathcal{P}$ corresponds to an element of $\mathcal{P}$.

LTL Syntax

LTL formulas are constructed through simple formulas together with the logical connectives

- $\wedge$ (conjunction),
- $\vee$ (disjunction),
- $\neg$ (negation),
- $\Rightarrow$ (implication),

and the temporal operators

- $\circ$ (next),
- $\mathcal{U}$ (until).

LTL formulas are then recursively defined as follows.

- true, false, and p are LTL formulas for all $p \in \mathcal{P}$,
- if ϕ_1 and ϕ_2 are LTL formulas, then $\phi_1 \wedge \phi_2$ and $\neg\phi_1$ are LTL formulas,
- if ϕ_1 and ϕ_2 are LTL formulas, then $\circ\phi_1$ and $\phi_1\mathcal{U}\phi_2$ are LTL formulas.

From the until operator $\mathcal{U}$ two commonly used operators can be defined

- $\Diamond$ (eventually),
- $\Box$ (always).

Formula	**Read as**	**Description**
$\circ\phi$	*next* ϕ	*Specify that* ϕ *has to be true at the next time step.* $\cdot \longrightarrow \overset{\phi}{\cdot} \dashrightarrow \cdot \longrightarrow \cdot \dashrightarrow$
$\phi_1\mathcal{U}\phi_2$	ϕ_1 *until* ϕ_2	*Specify that* ϕ_1 *must be true until* ϕ_2 *will be true.* $\overset{\phi_1}{\cdot} \longrightarrow \overset{\phi_1}{\cdot} \dashrightarrow \overset{\phi_1}{\cdot} \longrightarrow \overset{\phi_2}{\cdot} \dashrightarrow$
$\Diamond\phi$	*eventually* ϕ	*Specify that* ϕ *has to be true at some point in time in the future.* $\cdot \longrightarrow \cdot \dashrightarrow \overset{\phi}{\cdot} \longrightarrow \cdot \dashrightarrow$
$\Box\phi$	*always* ϕ	*Specify that* ϕ *has to be true for all future time.* $\overset{\phi}{\cdot} \longrightarrow \overset{\phi}{\cdot} \dashrightarrow \overset{\phi}{\cdot} \longrightarrow \overset{\phi}{\cdot} \dashrightarrow$

Table 7.1: *Basic LTL formulas.*

These two operators are defined as

$$\Diamond\phi = \mathrm{true}\,\mathcal{U}\phi, \tag{7.5}$$

$$\Box\phi = \neg\Diamond\neg\phi. \tag{7.6}$$

The LTL formulas defined above are summarized in Table 7.1.

The until operator $\mathcal{U}$ is used to describe temporal ordering. The formula $\phi_1\mathcal{U}\phi_2$ would require ϕ_1 to be true until ϕ_2 will be true. The always operator $\Box$ defines an invariance property by requiring ϕ to hold for all $t \in \mathbb{N}$. Complex LTL formulas are constructed by nesting the temporal operators. Examples of complex LTL formulas are shown in Table 7.2.

The formula $\phi_1\mathcal{U}\Box\phi_2$ can be used to model convergence towards the operating conditions described by ϕ_2 through a particular subset of the observation-space described by ϕ_1.

LTL Semantics

An unique interpretation of LTL formulas is obtained by defining LTL semantics. In this context LTL formulas are interpreted over sequences of predicate values

Formula	Read as	Description
$\phi_1 \mathcal{U} \Box \phi_2$	*ϕ_1 until always ϕ_2*	*Specify that ϕ_2 must be true for all time, or that ϕ_1 must be true until at some later time ϕ_2 must be true for all future time.*
$\Box \Diamond \phi$	*Always eventually ϕ*	*Specify that always ϕ should eventually be satisfied.*
$\Box(\phi_1 \mathcal{U} \phi_2)$	*Always (ϕ_1 until ϕ_2)*	*Specify that $\phi_1 \mathcal{U} \phi_2$ must be true for all time, which implies that for all time either ϕ_2 must hold or ϕ_1 must hold until ϕ_2 will be true at some future time.*

Table 7.2: *Example of complex LTL formulas.*

$s \in \mathcal{P}^\omega$, where $\mathcal{P}^\omega$ denote the set of all infinite strings obtained by concatenating elements in $\mathcal{P}$. An element of $\mathcal{P}^\omega$ is an infinite string

$$s = p_1\, p_2\, p_3\, \ldots \text{ with } p_i \in \mathcal{P} \text{ for } i = \mathbb{N}. \tag{7.7}$$

Thus, for each $t \in \mathbb{N}$ only one predicate is satisfied. Let $s \in \mathcal{P}^\omega$ be a string and denote by $s(t) \models \phi$ that string s satisfies formula ϕ at time t, if formula ϕ holds at time t along trajectory s. For any $p \in \mathcal{P}$, LTL formulas ϕ_1, ϕ_2 and $t \in \mathbb{N}$ the satisfaction relation $\models$ is defined as follows.

- $s(t) \models p$ if $p = s(t)$,
- $s(t) \models \neg p$ if $p \neq s(t)$,
- $s(t) \models \phi_1 \wedge \phi_2$ if $s(t) \models \phi_1$ and $s(t) \models \phi_2$,
- $s(t) \models \circ \phi_1$ if $s(t+1) \models \phi_1$,
- $s(t) \models \phi_1 \mathcal{U} \phi_2$ if there exists $t' \geq t$ such that for all k, $t \leq k \leq t'$, $s(k) \models \phi_1$ and $s(t') \models \phi_2$,

Finally, a sequence s satisfies formula ϕ if $s(0) \models \phi$.

When a LTL formula ϕ is interpreted over an observed sequence in $L^\omega(T_\Sigma^\mathcal{P})$ and each predicate $p \in \mathcal{P}$ corresponds to a subset of observation-space $\mathbb{R}^m$, the LTL formula ϕ defines how trajectories of linear control system Σ interact with these sets. This is a convenient and formal way of expressing control requirements for discrete-time linear control systems. If every string in $L_\omega(T_\Sigma^\mathcal{P})$

satisfies formula ϕ then the transition system $T_\Sigma^{\mathcal{P}}$ satisfies ϕ, denoted by

$$T_\Sigma^{\mathcal{P}} \models \phi. \tag{7.8}$$

A similar notation is used for transition system $T_\Sigma^{\mathcal{P}'}$ even though predicates in LTL formula ϕ do not correspond to sets in the finite partition $\mathcal{P}'$ of state-space $\mathbb{R}^n$. When for every $r \in L_\omega(T_\Sigma^{\mathcal{P}'})$, where $\pi_{\mathcal{P}'\mathcal{P}}(r) \models \phi$ we say that

$$T_\Sigma^{\mathcal{P}'} \models \phi. \tag{7.9}$$

7.4 Büchi Automata

To fit with the framework proposed in [Tabuada and Pappas, 2006] the translation of LTL formula ϕ into a Büchi automaton A_ϕ is required. Given any requirement specification formulated as a LTL formula ϕ it is possible to construct a Büchi automaton A_ϕ accepting every string satisfying ϕ [Büchi, 1962]. The rationale for choosing a *language equivalence* translation is that language equivalence preserves properties expressible in LTL.

DEFINITION 7.1 (BÜCHI AUTOMATON) *A Büchi automaton is defined as*

$$A = (T_A, F) = ((Q, Q_0, \longrightarrow, O, \Upsilon), F),$$

where $T_A = (Q, Q_0, \longrightarrow, O, \Upsilon)$ is a finite transition system (See Definition 3.4) and $F \subseteq Q$ is a set of accepting states.

A Büchi automaton can be seen as a transition system extended with a mechanism for describing the behavior of strings at infinity. Thus, every Büchi automaton A necessarily carries an underlying transition system structure T_A. Thus, Büchi automata defines generated languages and ω-languages (See Definition 3.8).

Let Q^ω denote the set of all infinite strings obtained by concatenating elements in Q. A string $s \in Q^\omega$ is a run of A if $s(1) \in Q^0$, $s(i) \longrightarrow s(i+1)$ for $i \in \mathbb{N}$ and there exists infinitely many $i \in \mathbb{N}$ such that $s(i) \in F$. The language accepted by Büchi automaton A is defined as follows.

DEFINITION 7.2 (ACCEPTED LANGUAGE) *Let $A = (Q, Q_0, \longrightarrow, O, \Upsilon, F)$ be a Büchi automaton. The language accepted by A is defined as*

$$L_\omega = \{r \in Q^\omega \mid r = \Upsilon(s) \text{ for some initialized run } s \text{ of } A\}. \tag{7.10}$$

Choosing $F = Q$ we have

$$L_\omega(A) = L_\omega(T_A) = L_\omega(T) \tag{7.11}$$

Tools to automatically translate a LTL formula ϕ into a Büchi automaton exists. For further details on the translation from LTL formula ϕ into Büchi automaton the reader is referred to [Wolper, 2000].

7.5 Abstraction

In this section it is shown that finite abstractions of a discrete-time controllable linear control systems Σ can be obtained if the linear control systems satisfy certain simple controllability assumptions and by forming a finite partition of observation-space $\mathbb{R}^m$ of linear control system Σ. The finite abstraction of linear control system Σ is required for controller synthesis.

7.5.1 Linear Control Systems

Lets assume that a discrete-time (time-invariant) controllable linear control system is obtained from nonlinear system Σ_* as

$$\Sigma \ : \quad x(t+1) = Ax(t) + Bu(t), \tag{7.12}$$

where the system matrices $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are generally constant and state variable $x \in \mathbb{R}^n$ and control variable $u \in \mathbb{R}^m$ are discrete. In the following x is referred to as the state of the system and u as the input to the system. Further, $\mathbb{R}^n$ is referred to as the state-space (or set of states) of the system and $\mathbb{R}^m$ as the observation-space (or set of observations) of the system. n denotes the dimension of the system whereas m denotes the dimension of the input-space.

Brunovsky Normal Form

The Brunovsky normal form is a special form of a linear control system Σ, where the pair of matrices $(A, B) \in \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times m}$ have a special structure.

DEFINITION 7.3 (CONTROLLABILITY INDICES) *Let Σ be a linear control system as defined in Eq. (7.12). The sequence of positive integers $\kappa =$*

$(\kappa_1, \kappa_2, \ldots, \kappa_m)$, *where* $\operatorname{rank} B = m$ *are called the controllability indices of the system satisfying*

$$\kappa_1 \geq \kappa_2 \geq \ldots \geq \kappa_m,$$

such that

$$\kappa_1 + \kappa_2 + \ldots + \kappa_m = n.$$

DEFINITION 7.4 (BRUNOVSKY NORMAL FORM [SONTAG, 1998]) *Let* Σ *be a linear control system as defined in Eq. (7.12). Let* $\kappa = (\kappa_1, \kappa_2, \ldots, \kappa_m)$ *be a sequence of controllability indices. The system is in Brunovsky normal form if* A *and* B *are of the following form*

$$A = \begin{bmatrix} A_{\kappa_1} & 0 & \ldots & 0 \\ 0 & A_{\kappa_2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & A_{\kappa_m} \end{bmatrix}, \quad B = \begin{bmatrix} b_{\kappa_1} & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & b_{\kappa_2} & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & b_{\kappa_m} & 0 & \ldots & 0 \end{bmatrix},$$

where each block $A_{\kappa_i} \in \mathbb{R}^{\kappa_i \times \kappa_i}$ *and* $b_{\kappa_i} \in \mathbb{R}^{\kappa_i}$ *for* $i = 1, \ldots, m$ *are of the form*

$$A_{\kappa_i} = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix}, \quad b_{\kappa_i} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

The Brunovsky normal form Σ_κ of a controllable linear control system Σ with controllability indices $\kappa = (\kappa_1, \ldots, \kappa_m)$ is given by

$$\Sigma_\kappa \; : \; \begin{cases} x_1(t+1) & = x_2(t) \\ x_2(t+1) & = x_3(t) \\ & \vdots \\ x_{\kappa_1}(t+1) & = u_1(t) \\ x_{\kappa_1+1}(t+1) & = x_{\kappa_1+2}(t) \\ x_{\kappa_1+2}(t+1) & = x_{\kappa_1+3}(t) \\ & \vdots \\ x_{\kappa_1+\kappa_2}(t+1) & = u_2(t) \\ & \vdots \\ x_{\kappa_1+\ldots+\kappa_{m-1}+1}(t+1) & = x_{\kappa_1+\ldots+\kappa_{m-1}+2}(t) \\ x_{\kappa_1+\ldots+\kappa_{m-1}+2}(t+1) & = x_{\kappa_1+\ldots+\kappa_{m-1}+3}(t) \\ & \vdots \\ x_{\kappa_1+\ldots+\kappa_m}(t+1) & = u_m(t) \end{cases} \tag{7.13}$$

Any controllable linear control system Σ can be effectively transformed to Brunovsky normal form Σ_κ by feedback and a change of state and input coordinates as asserted in the following result.

PROPOSITION 7.1 [Brunovsky, 1970] For every controllable linear control system Σ there exists a sequence of controllability indices $\kappa = (\kappa_1, \kappa_2, \ldots, \kappa_m)$, invertible linear transformations $H \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times m}$, and linear transformation $F \in \mathbb{R}^{m\times n}$ such that the pair $(A_\kappa, B_\kappa) = \big(H(A+BF)H^{-1}, HBV\big)$ is in Brunovsky normal form Σ_κ.

The system in Brunovsky normal form Σ_κ with state $x_\kappa \in \mathbb{R}^n$ and input $u_\kappa \in \mathbb{R}^m$ is related to linear control system Σ by an invertible state/input transformation matrix (*isomorphism*) $U : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$, that is

$$\begin{bmatrix} x_\kappa \\ u_\kappa \end{bmatrix} = U \begin{bmatrix} x \\ u \end{bmatrix} = \begin{bmatrix} H & O_{n\times m} \\ F & V \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix}. \tag{7.14}$$

The computation of the state/input transformation matrix U is outlined in Algorithm 1 and described in detail in Appendix B.

Algorithm 1 State/input transformation

```
Input: Linear control system $\Sigma$ with matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$
Output: State/input transformation $U = \begin{bmatrix} H & O_{n\times m} \\ F & V \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$
Choose $\kappa = (\kappa_1, \kappa_2, \ldots, \kappa_m)$ such that $\kappa_1 \geq \kappa_2 \geq \ldots \geq \kappa_m$ and $\kappa_1 + \kappa_2 + \ldots + \kappa_m = n$, where $\operatorname{rank} B = m$
$\mathcal{C} = [b_1 \ \ldots \ b_m \ Ab_1 \ \ldots \ Ab_m \ A^2b_1 \ \ldots \ A^2b_m \ \ldots \ A^{n-1}b_1 \ \ldots A^{n-1}b_m]$
$\bar{\mathcal{C}} = [b_1 \ Ab_1 \ A^2b_1 \ \ldots \ A^{\kappa_1-1}b_1 \ b_2 \ Ab_2 \ A^2b_2 \ \ldots \ A^{\kappa_2-1}b_2 \ \ldots \ b_m \ Ab_m \ A^2b_m \ \ldots \ A^{\kappa_m-1}b_m]$
for $i = 1 : m$ do
    $\sigma_i = \Sigma_{i=1}^{m}\kappa_i$
    $d_i = \bar{\mathcal{C}}^{-1}(\sigma_i, :)$
end for
$H = \begin{bmatrix} d_1 & d_1A & \ldots & d_1A^{\kappa_1-1} & \ldots & d_m & d_mA & \ldots & d_mA^{\kappa_m-1} \end{bmatrix}^{\mathrm{T}}$
$A_c = HAH^{-1}$
$B_c = HB$
$A_m = \begin{bmatrix} A_c(\sigma_1, :) & \ldots & A_c(\sigma_m, :) \end{bmatrix}^{\mathrm{T}}$
$B_m = \begin{bmatrix} B_c(\sigma_1, :) & \ldots & B_c(\sigma_m, :) \end{bmatrix}^{\mathrm{T}}$
$F = B_m^{-1}A_mH$
$V = B_m^{-1}$
```

In order for Algorithm 1 to be defined we require that $\operatorname{rank}\mathcal{C} = n$ and $\operatorname{rank} B = m$. The matrix H is called a *similarity transformation* [Sontag, 1998].

7.5.2 Control Abstract Embedding

Discrete-time linear control systems are naturally embedded in the class of transition systems with observations. The dynamics of discrete-time linear control systems are close to transition systems due to the existence of an atomic time step. Different embeddings of both continuous and discrete-time linear systems can be found in [Pappas, 2003]. In the context of controller synthesis, a *control abstract embedding* is desirable. The transition system associated with linear control system Σ in Eq. (7.12) is defined as (See Definition 3.4)

$$T_\Sigma = (\mathbb{R}^n, \mathbb{R}^n, \longrightarrow, O, \Upsilon)\,,$$

where

- $\mathbb{R}^n$ is the set of states,

- $\mathbb{R}^n$ is the set of initial states,
- $\longrightarrow \subseteq \mathbb{R}^n \times \mathbb{R}^n$ is the transition relation defined by $x \longrightarrow x'$, if there exists an input $u \in \mathbb{R}^m$ such that $x' = Ax + Bu$,
- O is the set of observations,
- $\Upsilon : \mathbb{R}^n \to O$ is the observation map defined by $\Upsilon(x) \in O$.

Note that the set of observations O and observation map Υ are unspecified. They will be defined when a finite bisimilar quotient of transition system T_Σ is constructed. The embedding of Σ in T_Σ is *control abstract* in the sense that the input value required to perform a transition $x \longrightarrow x'$ is not explicitly captured by the transition system. However, the value of the input can be recovered from the pair (x, x') by solving $x' = Ax + Bu$ for the input $u \in \mathbb{R}^m$.

The following two assumptions must be satisfied to obtain a finite bisimilar quotient of linear control system Σ.

Assumption I (Controllability) The linear control system Σ is controllable if the columns of the controllability matrix $\mathcal{C} \in \mathbb{R}^{n \times mn}$

$$\begin{aligned}\mathcal{C} = \big[b_1 \;\; \dots \;\; b_m \;\; Ab_1 \;\; \dots \;\; Ab_m \;\; A^2b_1 \;\; \dots \;\; A^2b_m \;\; \dots \\ A^{n-1}b_1 \;\; \dots \;\; A^{n-1}b_m\big], \end{aligned} \tag{7.15}$$

are linearly independent (matrix $\mathcal{C}$ has full row rank), i.e. $\operatorname{rank} \mathcal{C} = n$. If the system is controllable then any state is reachable from any initial state, giving the system proper inputs through the variable $u(t)$.

Assumption II (Independent Inputs) A linear control system Σ has m independent inputs if the columns of matrix B are linearly independent (matrix B has full column rank), i.e. $\operatorname{rank} B = m$.

If Assumption I is satisfied it allows a decomposition of the state-space $\mathbb{R}^n$ of Σ. If the m columns of matrix B are not linearly independent ($\operatorname{rank} B = s < m$), then the same input action to the system can be accomplished with only s inputs, instead of m inputs, and hence there is a redundancy of inputs to the system. However, without loss of generality linearly dependent columns of matrix B can always be removed without destroying essential properties of Σ, in order to satisfy Assumption II.

The decomposition of state-space $\mathbb{R}^n$ results in a new controllability matrix for the system.

PROPOSITION 7.2 [Antsaklis and Michel, 1997] *Let Σ be a linear control system satisfying Assumptions I and II. Then, there exists a sequence of controllability indices $\kappa = (\kappa_1, \kappa_2, \ldots, \kappa_m)$ satisfying*

$$\begin{aligned}
&\text{span}\Big\{b_1, \ldots, b_m, Ab_1, \ldots, Ab_m, A^2 b_1, \ldots, A^2 b_m, \ldots, \\
&\qquad A^{n-1} b_1, \ldots, A^{n-1} b_m\Big\} = \qquad (7.16)\\
&\text{span}\Big\{b_1, Ab_1, A^2 b_1, \ldots, A^{\kappa_1 - 1} b_1, b_2, Ab_2, A^2 b_2, \ldots, A^{\kappa_2 - 1} b_2, \ldots, \\
&\qquad b_m, Ab_m, A^2 b_m, \ldots, A^{\kappa_m - 1} b_m\Big\}, \qquad (7.17)
\end{aligned}$$

where column $A^{\kappa_i} b_i$ is linearly dependent on the previous ones, i.e. the vectors to the left of $A^{\kappa_i} b_i$.

The decomposition of state-space $\mathbb{R}^n$ for linear control system Σ according to controllability indices $\kappa = (\kappa_1, \kappa_2, \ldots, \kappa_m)$ results in a new controllability matrix $\bar{\mathcal{C}} \in \mathbb{R}^{n \times n}$

$$\begin{aligned}
\bar{\mathcal{C}} = \Big[&b_1 \;\; Ab_1 \;\; A^2 b_1 \;\; \ldots \;\; A^{\kappa_1 - 1} b_1 \;\; b_2 \;\; Ab_2 \;\; A^2 b_2 \;\; \ldots \;\; A^{\kappa_2 - 1} b_2 \;\; \ldots \\
&b_m \;\; Ab_m \;\; A^2 b_m \;\; \ldots \;\; A^{\kappa_m - 1} b_m\Big], \qquad (7.18)
\end{aligned}$$

for the system. To obtain $\bar{\mathcal{C}}$ see Appendix B. Using the sequence of controllability indices $\kappa = (\kappa_1, \kappa_2, \ldots, \kappa_m)$ we can introduce the subspace $\mathcal{V} \cong \mathbb{R}^{n-m}$ of state-space $\mathbb{R}^n$ defined by

$$\mathcal{V} \cong \text{span}\Big\{b_1, \ldots, A^{\kappa_1 - 2} b_1, b_2, \ldots, A^{\kappa_2 - 2} b_2, \ldots, b_m, \ldots, A^{\kappa_m - 2} b_m\Big\}. \qquad (7.19)$$

Subspace $\mathcal{V}$ induces the observation map

$$\Upsilon \;:\; \mathbb{R}^n \rightarrow \mathbb{R}^n / V \cong \mathbb{R}^m, \qquad (7.20)$$

which is defined as the natural projection from state-space $\mathbb{R}^n$ to the quotient space $\mathbb{R}^n/\mathcal{V} \cong \mathbb{R}^m$. Thus, observation map Υ maps a vector $x \in \mathbb{R}^n$ into its equivalence class in $\mathbb{R}^n/\mathcal{V}$ which is identified with an observation $y \in \mathbb{R}^m$. The introduction of observation map Υ motivates the definition of transition system T_Σ with observation-space $\mathbb{R}^m$.

DEFINITION 7.5 *Let Σ be a linear control system satisfying Assumptions I and II. Transition system T_Σ associated with Σ is defined by*

$$T_\Sigma = (\mathbb{R}^n, \mathbb{R}^n, \longrightarrow, \mathbb{R}^m, \Upsilon). \tag{7.21}$$

The next step is to obtain a transition system, denoted by $T_\Sigma^{\mathcal{P}}$ which has a finite observation-space, denoted by $\mathcal{P}$. Since linear control systems are considered, a natural choice of $\mathcal{P}$ for transition system $T_\Sigma^{\mathcal{P}}$ is a finite partition of observation-space $\mathbb{R}^m$ of transition system T_Σ. The change of the observation-space $\mathbb{R}^m$ is necessarily accompanied with a change of observation map Υ. Transition system $T_\Sigma^{\mathcal{P}}$ with finite observation-space $\mathcal{P}$ is defined as

$$T_\Sigma^{\mathcal{P}} = (\mathbb{R}^n, \mathbb{R}^n, \longrightarrow, \mathcal{P}, \pi_{\mathcal{P}} \circ \Upsilon), \tag{7.22}$$

where

- $\mathcal{P} = \{P_i\}_{i\in I}$ is a finite partition of observation-space $\mathbb{R}^m$ satisfying $\cup_{i\in I} P_i = \mathbb{R}^m$ where $P_i \cap P_j = \emptyset$ for $i \neq j$,
- $\pi_{\mathcal{P}} \circ \Upsilon \;:\; \mathbb{R}^n \to \mathcal{P}$ is the observations map defined by $\pi_{\mathcal{P}}(\Upsilon(x)) = P$.

The partition of observation-space $\mathbb{R}^m$ is represented by semi-linear sets.

DEFINITION 7.6 (SEMI-LINEAR SET) *The class of semi-linear subsets of $\mathbb{R}^m$ consists of finite unions, intersections and complements of the following elementary sets*

$$\left\{x \in \mathbb{R}^m \mid f^{\mathrm{T}} x + c \sim 0\right\},$$

where $f \in \mathbb{Q}^m$, $c \in \mathbb{Q}$, and $\sim \in \{<, \leq, =, \geq, >\}$.

The next step is to obtain a transition system $T_\Sigma^{\mathcal{P}'}$ where the observation-space is identified with a finite refinement of state-space $\mathbb{R}^n$ through the finite partition $\mathcal{P}$ of observation-space $\mathbb{R}^m$. Let $\mathcal{P} = \{P_i\}_{i\in I}$ be a finite partition of observation-space $\mathbb{R}^m$ for transition system $T_\Sigma^{\mathcal{P}}$, then we have that

$$\begin{aligned} \mathcal{P}' &= \Upsilon^{-1}(\mathcal{P}) \\ &= \Upsilon^{-1}\{P_i\}_{i\in I}, \end{aligned} \tag{7.23}$$

is a finite refinement of state-space $\mathbb{R}^n$ since $\Upsilon^{-1} \;:\; \mathbb{R}^m \to \mathbb{R}^n$. Transition system $T_\Sigma^{\mathcal{P}'}$ with $\mathcal{P}' = \Upsilon^{-1}(\mathcal{P})$ being a finite refinement of state-space $\mathbb{R}^n$ is now defined as

$$T_\Sigma^{\mathcal{P}'} = \left(\mathbb{R}^n, \mathbb{R}^n, \longrightarrow, \mathcal{P}', \pi_{\mathcal{P}'}\right), \tag{7.24}$$

where

- $\mathcal{P}' = \Upsilon^{-1}(\mathcal{P})$ is a finite refinement of state-space $\mathbb{R}^n$,
- $\pi_{\mathcal{P}'} : \mathbb{R}^n \to \mathcal{P}'$ is the observation map defined by $\pi_{\mathcal{P}'}(x) = P'$.

Transition systems $T_\Sigma^{\mathcal{P}}$ and $T_\Sigma^{\mathcal{P}'}$ both admit finite abstractions in the form of finite transition systems [Tabuada and Pappas, 2006]. The finite bisimilar quotient of transition system $T_\Sigma^{\mathcal{P}'}$ with respect to equivalence relation $\sim \subseteq \mathbb{R}^n \times \mathbb{R}^n$ is defined as

$$T_{/\sim}^{\mathcal{P}'} = \left(Q_{/\sim}, Q_{/\sim}^0, \longrightarrow_{/\sim}, O, \Upsilon_{/\sim} \right), \tag{7.25}$$

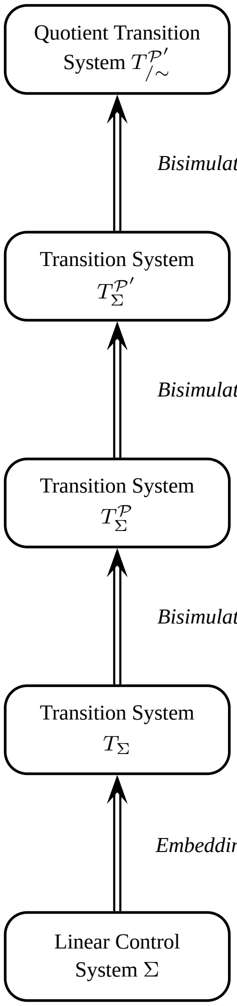
where

- $Q_{/\sim} = \{ S \subseteq \mathbb{R}^n \mid S$ is an equivalence class of $\sim \}$ is the set of states,
- $Q_{/\sim}^0 = Q_{/\sim}$ is the set of initial states,
- $\longrightarrow_{/\sim} \subseteq Q_{/\sim} \times Q_{/\sim}$ is the transition relation defined by $S \longrightarrow_{/\sim} S'$ if there exists $x \in S$ and $x' \in S'$ such that $x \longrightarrow x'$ in $T_\Sigma^{\mathcal{P}'}$,
- $O = \mathcal{P}'$ is the set of observations, where $\mathcal{P}'$ is a finite refinement of state-space partition $\Upsilon^{-1}(\mathcal{P})$,
- $\Upsilon_{/\sim} : Q_{/\sim} \to O$ is the observation map such that $\Upsilon_{/\sim}(S) = \Upsilon(x)$ for some $x \in S$.

The finite bisimilar quotient of transition system $T_\Sigma^{\mathcal{P}}$ is defined in a similar way. The intermediate steps in the abstraction of linear control system Σ to obtain a finite bisimilar quotient $T_{/\sim}^{\mathcal{P}'}$ are shown in Figure 7.5.2.

Let Σ_κ be the discrete-time linear control system in Brunovsky normal form obtained from Σ through an invertible state/input transformation U (See Eq. (7.14)). For any finite partition $\mathcal{P}$ of observation-space $\mathbb{R}^m$ of transition system T_Σ there exists a finite refinement $\mathcal{Q}$ of state-space partition $\Upsilon^{-1}(\mathcal{P})$ making the quotient $T_{/\sim}^{\mathcal{Q}}$ of T_Σ with respect to $\mathcal{Q}$ a finite bisimilar quotient if there exists a finite refinement $\mathcal{Q}_\kappa$ of state-space partition $\Upsilon_\kappa^{-1}(H(\mathcal{P}))$ making the quotient $T_{/\sim}^{\mathcal{Q}_\kappa}$ of T_{Σ_κ} with respect to $\mathcal{Q}_\kappa$ a finite bisimilar quotient. This is shown in the commutative diagram in Figure 7.5.2.

In view of Figure 7.5.2 we can assume, without loss of generality, that Σ is in Brunovsky normal form since any controllable linear system can be transformed into this form by a change of coordinates and an invertible feedback (See


Quotient Transition System $T^{\mathcal{P}'}_{/\sim}$
Bisimulation
Transition System $T^{\mathcal{P}'}_{\Sigma}$
Bisimulation
Transition System $T^{\mathcal{P}}_{\Sigma}$
Bisimulation
Transition System T_{Σ}
Embedding
Linear Control System Σ

(a) *Relationship between transition systems associated with linear control system* Σ.

$$\begin{tikzcd}
Q_{/\sim} \arrow[r, "\Upsilon_{/\sim}"] & O \\
\mathbb{R}^n \arrow[r, "\pi_{\mathcal{P}'}"] & \mathcal{P}' \arrow[ul, "\text{Refinement}"'] \\
\mathbb{R}^n \arrow[u] \arrow[r, "\pi_{\mathcal{P}} \circ \Upsilon"] & \mathcal{P} \arrow[u, "\Upsilon^{-1}"'] \\
\mathbb{R}^n \arrow[u] \arrow[r, "\Upsilon"] & \mathbb{R}^m \arrow[u, "\pi_{\mathcal{P}}"'] \\
\mathbb{R}^n \arrow[u] & \mathbb{R}^m \arrow[u]
\end{tikzcd}$$

(b) *Corresponding state and observation spaces.*

Figure 7.3: *Intermediate steps to a finite bisimilar quotient* $T^{\mathcal{P}'}_{/\sim}$.

$$\begin{tikzcd}
\mathcal{Q} \arrow[r, "H"] & \mathcal{Q}_\kappa \\
\Upsilon^{-1}(\mathcal{P}) \arrow[u, "\textit{Refinement}"'] \arrow[r, "H"] & \Upsilon_\kappa^{-1}(H(\mathcal{P})) \arrow[u, "\textit{Refinement}"'] \\
\mathcal{P} \arrow[u, "\Upsilon^{-1}"'] \arrow[r, "H"] & H(\mathcal{P}) \arrow[u, "\Upsilon_\kappa^{-1}"'] \\
\mathbb{R}^m \arrow[u, "\pi_{\mathcal{P}}"'] &
\end{tikzcd}$$

Figure 7.4: *Finite refinements $\mathcal{Q}$ and $\mathcal{Q}_\kappa$ of $\Upsilon^{-1}(\mathcal{P})$ and $\Upsilon_\kappa^{-1}(H(\mathcal{P}))$.*

Eq. (7.14)). For a linear control system in Brunovsky normal form the observation map is of the form

$$\Upsilon = \phi \circ \pi, \tag{7.26}$$

where $\pi : \mathbb{R}^n \to \mathbb{R}^m$ is a projection map such that

$$\pi(x) = \begin{bmatrix} x_1 \\ x_{\kappa_1+1} \\ \vdots \\ x_{\kappa_1+\ldots+\kappa_{m-1}+1} \end{bmatrix}, \tag{7.27}$$

and $\phi \ : \ \mathbb{R}^m \to \mathbb{R}^m$ is an arbitrary linear isomorphism.

7.5.3 Bisimulation Algorithm

In the previous section the existence of finite bisimilar quotient transitions systems for linear control systems satisfying certain assumptions was established. The following well known bisimulation algorithm can be used to compute the coarsest possible bisimulation [Bouajjani et al., 1990] provided that every set operation is effectively computable[1].

Algorithm 2 Bisimulation Algorithm

```
Input: Initial partition $\mathcal{P}' = \Upsilon^{-1}(\mathcal{P})$ of state-space $\mathbb{R}^n$ of transition system $T_\Sigma$
Output: Coarsest refinement $\mathcal{P}'$ of state-space $\mathbb{R}^n$
while $\exists P, P' \in \mathcal{P}'$ such that $\emptyset \neq P \cap \mathrm{Pre}(P') \neq P$ do
    $P_1 = P \cap \underline{\mathrm{Pre}(P')}$
    $P_2 = P \cap \overline{\mathrm{Pre}(P')}$
    $\mathcal{P}' = (\mathcal{P}' \setminus \{P\}) \cup \{P_1, P_2\}$
end while
```

Computing the coarsest bisimulation results in maximum complexity reduction. The input to the bisimulation algorithm (Algorithm 2) is transition system T_Σ associated with linear control system Σ and an initial partition $\mathcal{P}' = \Upsilon^{-1}(\mathcal{P})$ of state-space $\mathbb{R}^n$. The initial partition $\mathcal{P}'$ is based on the finite partition $\mathcal{P}$ of observation-space $\mathbb{R}^m$ of transition system T_Σ. Algorithm 2 terminates with the coarsest refinement $\mathcal{P}'$ such that $T^{\mathcal{P}'}_{/\sim}$ is a finite bisimilar quotient of transition system T_Σ. The termination of Algorithm 2 is ensured by the controllability of linear control system Σ associated with transition system T_Σ.

7.6 Controller Synthesis

The existence of finite bisimilar quotients (bisimulations) for linear control systems enables the design of controllers enforcing a given LTL specification at a discrete level. The concept of parallel composition with observation synchronization of transition systems is used to model the interconnection of the controller and the system to be controlled, i.e. the finite bisimilar quotient $T^{\mathcal{P}'}_{/\sim}$ of linear control system Σ. In the following the steps involved in controller synthesis are described.

[1] There exists an algorithm that is able to perform the set operations

To satisfy a given LTL specification ϕ the controller T_c is required to restrict the behavior of the linear control system. The notion of parallel composition with observation synchronization of transition systems is employed. Thus, the parallel composition with output synchronization of the controller T_c and the finite bisimilar quotient $T_{/\sim}^{\mathcal{P}'}$ of the linear control system Σ are required to enforce the specification.

The controller T_c is employed to restrict the behavior of transition system T_Σ such that a LTL specification ϕ is enforced. The setup in controller synthesis is shown in Figure 7.5.

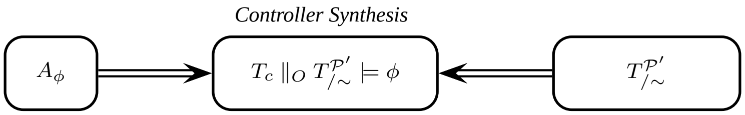

Controller Synthesis
A_ϕ
$T_c \parallel_O T_{/\sim}^{\mathcal{P}'} \models \phi$
$T_{/\sim}^{\mathcal{P}'}$

Figure 7.5: *Setup in controller synthesis.*

The baseline for controller synthesis is a Büchi automaton A_ϕ obtained from a given LTL formula ϕ and a finite bisimilar quotient $T_{/\sim}^{\mathcal{P}'}$ of linear control system Σ.

The idea is to synthesize a controller T_c that can restrict the behavior of T_Σ such that a given LTL formula ϕ is satisfied. Recall that since transition systems T_Σ and $T_{/\sim}^{\mathcal{P}'}$ are bisimilar, i.e. $T_\Sigma \cong T_{/\sim}^{\mathcal{P}'}$, then controller T_c synthesized for $T_{/\sim}^{\mathcal{P}'}$ will equivalently work for T_Σ (See Proposition 4.3 in [Tabuada and Pappas, 2006]).

Let $\mathcal{P}$ be a finite partition of observation-space $\mathbb{R}^m$ of transition system T_Σ and let $\mathcal{P}'$ be a finite refinement of state-space partition $\Upsilon^{-1}(\mathcal{P})$. Then, there exists a finite controller T_c satisfying

$$T_c \parallel_O T_\Sigma^{\mathcal{P}'} \models \phi,$$

if there exists a controller T_c satisfying

$$T_c \parallel_O T_{/\sim}^{\mathcal{P}'} \models \phi.$$

Further, $T_\Sigma^{\mathcal{P}'} \cong T_{/\sim}^{\mathcal{P}'}$ implies that $T_c \parallel_O T_\Sigma^{\mathcal{P}'} \cong T_c \parallel_O T_{/\sim}^{\mathcal{P}'}$. Thus, the parallel composition of T_c and $T_{/\sim}^{\mathcal{P}'}$ makes the controller T_c restrict the behavior of transition system $T_\Sigma^{\mathcal{P}'}$. Working with $T_\Sigma^{\mathcal{P}'}$ is preferable since the observation-space

$\mathcal{P}'$ of $T_{\Sigma}^{\mathcal{P}'}$ offers more detailed information regarding the dynamics of transition system T_{Σ} than the observation-space $\mathcal{P}$ of transition system $T_{\Sigma}^{\mathcal{P}}$. Recall that $\mathcal{P}$ is a finite partition of observation-space $\mathbb{R}^m$ whereas $\mathcal{P}' = \Upsilon^{-1}(\mathcal{P})$ is a finite refinement of state-space $\mathbb{R}^n$.

Recall that for any LTL formula ϕ it is always possible to construct a Büchi automaton A_{ϕ} recognizing every string satisfying ϕ. Given a finite bisimilar quotient $T_{/\sim}^{\mathcal{P}'}$ it is possible to construct a Büchi automaton $A_{/\sim}^{\mathcal{P}'}$ satisfying

$$L_{\omega}\left(A_{/\sim}^{\mathcal{P}'}\right) = L_{\omega}\left(T_{/\sim}^{\mathcal{P}'}\right). \tag{7.28}$$

Now, a Büchi automaton controller A_c is constructed that satisfy

$$\pi_{\mathcal{P}'\mathcal{P}}\left(\underbrace{L_{\omega}(A_c) \cap L_{\omega}\left(A_{/\sim}^{\mathcal{P}'}\right)}_{L_{\omega}\left(A_c \,\|_O\, A_{/\sim}^{\mathcal{P}'}\right)}\right) \subseteq L_{\omega}(A_{\phi}), \tag{7.29}$$

where $\pi_{\mathcal{P}'\mathcal{P}} : \mathcal{P}' \to \mathcal{P}$ is a projection map which maps every element $P' \in \mathcal{P}'$ to an unique element $\pi_{\mathcal{P}'\mathcal{P}}(P') = P$ such that $P' \subseteq P$. In the following we assume that a Büchi automaton controller A_c exists and can be modeled by a transition system T_c satisfying

$$L_{\omega}(T_c) \cap L_{\omega}\left(A_{/\sim}^{\mathcal{P}'}\right) = L_{\omega}(A_c) \cap L_{\omega}\left(A_{/\sim}^{\mathcal{P}'}\right). \tag{7.30}$$

For any Büchi automaton controller A_c enforcing a given LTL formula ϕ there exists a finite controller T_c satisfying

$$L_{\omega}\left(T_c \,\|_O\, T_{/\sim}^{\mathcal{P}'}\right) = L_{\omega}(T_c) \cap L_{\omega}\left(T_{/\sim}^{\mathcal{P}'}\right) = L_{\omega}(A_c) \cap L_{\omega}\left(A_{/\sim}^{\mathcal{P}'}\right). \tag{7.31}$$

The parallel composition of the controller T_c and the finite bisimilar quotient $T_{/\sim}^{\mathcal{P}'}$ with observation synchronization is given by (See Definition 3.7)

$$T_c \parallel_O T_{/\sim}^{\mathcal{P}'} = \left(Q_\parallel, Q_\parallel^0, \longrightarrow_\parallel, O, \Upsilon_\parallel\right), \tag{7.32}$$

where

- $Q_\parallel = \left\{(q_c, S) \in Q_c \times Q_{/\sim} \mid \Upsilon_c(q_c) = \Upsilon_{/\sim}(S)\right\}$ is a set of states,
- $Q_\parallel^0 = \left\{(q_c, S) \in Q_c^0 \times Q_{/\sim}^0 \mid \Upsilon_c(q_c) = \Upsilon_{/\sim}(S)\right\}$ is a set of initial states,
- $\longrightarrow_\parallel \subseteq Q_\parallel \times Q_\parallel$ is a transition relation defined by $(q_c, S) \longrightarrow_\parallel (q_c', S')$ for $(q_c, S), (q_c', S') \in Q_\parallel$ if $q_c \longrightarrow_c q_c'$ in T_c and $S \longrightarrow_{/\sim} S'$ in $T_{/\sim}^{\mathcal{P}'}$,
- $O \subseteq \mathcal{P}'$ is a set of observations,
- $\Upsilon_\parallel : Q_\parallel \to O$ is an observation map defined by $\Upsilon_\parallel(q_c, S) = \Upsilon_c(q_c) = \Upsilon_{/\sim}(S)$.

A state of $T_c \parallel_O T_{/\sim}^{\mathcal{P}'}$ will be denoted by $q_\parallel = (q_c, S) \in Q_c \times Q_{/\sim}$. Note that O is a subset of the finite refinement $\mathcal{P}'$ of state-space $\mathbb{R}^n$ of transition system $T_\Sigma^{\mathcal{P}'}$ since the controller T_c may restrict some transitions in the finite bisimilar quotient $T_{/\sim}^{\mathcal{P}'}$.

7.7 Refinement

In the previous section a finite controller T_c for the finite bisimilar quotient $T_{/\sim}^{\mathcal{P}'}$, enforcing a given LTL formula ϕ was synthesized. In this section the continuous inputs will be extracted from T_c that is required to enforce a LTL formula ϕ on the linear control system Σ. The explicit modeling of the control inputs available to linear control system Σ will result in a closed-loop hybrid system. The resulting closed-loop hybrid system is guaranteed to enforce the requirement specification in LTL by construction. This motivates the definition of discrete-time linear hybrid system H [Tabuada and Pappas, 2006] and the associated transition system T_H.

DEFINITION 7.7 (DISCRETE-TIME LINEAR HYBRID SYSTEM) *A discrete-time linear hybrid system is defined as*

$$H = \left(X, X^0, \{A_q, B_q\}_{q\in Q}, \delta, \mathcal{U}\right), \tag{7.33}$$

where

- *$X = \amalg_{q \in Q} \mathbb{R}^{n_q}$ is the state-space where Q is a finite set of discrete states and $n_q \in \mathbb{N}$ for each discrete state $q \in Q$,*

- *$X^0 \subseteq X$ is a set of initial states,*

- *$\{A_q, B_q\}_{q \in Q}$ is the continuous dynamics where for each discrete state $q \in Q$, the pair $(A_q, B_q) \in \mathbb{R}^{n_q \times n_q} \times \mathbb{R}^{n_q \times m_q}$ defines a discrete-time linear control system $x(t+1) = A_q x(t) + B_q u(t)$ with inputs restricted to a set $\mathcal{U}\left(q(t), x(t)\right) \subseteq \mathbb{R}^{m_q}$,*

- *$\delta : Q \times \mathbb{R}^{n_q} \to 2^Q$ is the discrete dynamics which assigns to each discrete state $q \in Q$ and continuous state $x \in \mathbb{R}^{n_q}$ the discrete successor states $\delta\left(q(t), x(t)\right) \subseteq Q$.*

State-space X is the disjoint union of the underlaying sets $\mathbb{R}^{n_q}$, denoted by the coproduct $\amalg$. Note that the definition of linear hybrid system H allows to have different continuous dynamics $x(t+1) = A_q x(t) + B_q u(t)$ for each discrete state $q \in Q$.

The discrete-time linear hybrid system H can also be embedded in the class of transition systems with observations. The embedding of linear hybrid system H in the class of transition systems with observations will allow the definition of correct implementation. Assuming the continuous dynamics of linear hybrid system H to be controllable the transition system associated with linear hybrid system H is

$$T_H = \left(X, X^0, \longrightarrow_H, O, \Upsilon\right), \qquad (7.34)$$

where

- $\longrightarrow_H \subseteq (Q \times \mathbb{R}^{n_q}) \times (Q \times \mathbb{R}^{n_q})$ is a transition relation defined by $(q, x) \longrightarrow_H (q', x')$ if $x' = A_q x + B_q u$ with $q' \in \delta(q, x)$ and $u \in \mathcal{U}(q, x)$,

- $O = \amalg_{q \in Q} O_q$ is a set of observations,

- $\Upsilon : Q \times \mathbb{R}^{n_q} \to O$ such that $\Upsilon(q, x) = \Upsilon_q(x) \in O$.

State-space X and the set of initial states X^0 are inherited from linear hybrid system H. The pair $(q, x) \in Q \times \mathbb{R}^{n_q}$ is the hybrid state of transition system T_H associated with linear hybrid system H. O_q and Υ_q are the observation-space and observation map associated with linear control system $x(t+1) =$

$A_q x(t) + B_q u(t)$ defined by the pair (A_q, B_q) for each discrete state $q \in Q$, respectively.

A linear hybrid system H is said to be a correct hybrid implementation of the closed-loop behavior $T_c \parallel_O T_\Sigma^{\mathcal{P}'}$ if $T_c \parallel_O T_\Sigma^{\mathcal{P}'}$ satisfies the transition system T_H associated with linear hybrid system H. i.e. $T_c \parallel_O T_\Sigma^{\mathcal{P}'} \models T_H$. The desired closed loop behavior $T_c \parallel_O T_\Sigma^{\mathcal{P}'}$ is the parallel composition of controller T_c and transition system $T_\Sigma^{\mathcal{P}'}$.

A correct hybrid implementation in the form of a linear hybrid system H of the desired closed loop behavior $T_c \parallel_O T_\Sigma^{\mathcal{P}'}$ (Defined in Eq. (7.32)) is obtained from

$$T_c \parallel_O T_\Sigma^{\mathcal{P}'} = \left(Q_\parallel, Q_\parallel^0, \longrightarrow_\parallel, O, \Upsilon_\parallel\right), \tag{7.35}$$

by defining the linear hybrid system as

$$H = \left(X, X^0, \{A_q, B_q\}_{q\in Q}, \delta, \mathcal{U}\right), \tag{7.36}$$

where

- $X = Q_\parallel$ is the set of states,
- $X^0 = Q_\parallel^0$ is the set of initial states,
- $A_q = A \in \mathbb{R}^{n\times n}, B_q = B \in \mathbb{R}^{n\times m}$ are the matrices of linear control system $\Sigma\ :\ x' = A_q x + B_q u$,
- $\delta\ :\ Q_\parallel \times \mathbb{R}^n \rightarrow 2^{Q_\parallel}$ is the discrete dynamics where $\delta(q_\parallel, x) = \left\{q_\parallel' \in Q_\parallel \mid q_\parallel \longrightarrow_\parallel q_\parallel'\right\}$,
- $\mathcal{U}(q_\parallel, x) = \left\{u \in \mathbb{R}^m \mid \pi_{\mathcal{P}'}(Ax + Bu) \in \Upsilon_\parallel\left(\delta\left(q_\parallel, x\right)\right)\right\}$ is the input set.

Linear hybrid system H is a control system in the sense that at every discrete state $q_\parallel \in Q_\parallel$ different future evolutions are possible under the action of different input values.

7.7.1 Determining the Input Sets

Linear hybrid system H already has all the information except the continuous inputs to be sent to linear control system Σ. Thus, all that is left to do to obtain linear hybrid system H is to compute these input sets for each discrete state $q_\| \in Q_\|$.

The semi-linear description of the sets $q_\| \in Q_\|$ induces a semi-linear description of the set of continuous inputs $\mathcal{U}(q_\|, x)$ enabling transitions in the linear control system Σ corresponding to transitions in the finite bisimilar quotient $T_{/\sim}^{\mathcal{P}'}$.

The discrete transitions in linear hybrid system H are given by $\delta : Q_\| \times \mathbb{R}^n \to 2^{Q_\|}$. Now, recall that a transition $q_\| \longrightarrow_\| q'_\|$ in $T_c \|_O T_{/\sim}^{\mathcal{P}'}$ corresponds to a pair of transitions $q_c \longrightarrow_c q'_c$ in the controller T_c and $S \longrightarrow_{/\sim} S'$ in the finite bisimilar quotient $T_{/\sim}^{\mathcal{P}'}$, respectively. Thus, from the transitions in $T_c \|_O T_{/\sim}^{\mathcal{P}'}$ we can determine which transitions in $T_{/\sim}^{\mathcal{P}'}$ are allowed by the controller T_c. Lets denote by $\pi_\| : Q_\| \to Q_{/\sim}$ the projection map recovering $S \in Q_{/\sim}$ from $q_\| = (q_c, S) \in Q_c \times Q_{/\sim}$ such that

$$\pi_\| \left(q_\|\right) = S. \tag{7.37}$$

Having determined which transitions in $T_{/\sim}^{\mathcal{P}'}$ are allowed by T_c the input set $\mathcal{U}(q_\|, x)$ can be determined for each discrete state $q_\| \in Q_\|$. From the transition $S \longrightarrow_{/\sim} S'$ and bisimilarity between bisimilar quotient transition system $T_{/\sim}^{\mathcal{P}'}$ and transition system T_Σ it follows that for any continuous state $x \in S$ the following is satisfied

$$x \longrightarrow_\Sigma x' \in S'. \tag{7.38}$$

This is equivalent to the existence of an input $u \in \mathbb{R}^m$ such that $x' = Ax + Bu$, where $x' \in S'$. From the definition of $S' \in Q_{/\sim}$ and linear control system Σ the input set $\mathcal{U}(q_\|, x)$ in discrete state $q_\| \in Q_\|$ is defined by

$$\mathcal{U}(q_\|, x) = \left\{u \in \mathbb{R}^m \mid \pi_{\mathcal{P}'}\left(Ax + Bu\right) \in \Upsilon_\| \left(\delta\left(q_\|, x\right)\right)\right\}, \tag{7.39}$$

where $\pi_{\mathcal{P}'} : \mathbb{R}^n \to \mathcal{P}'$ is an observation map such that

$$\pi_{\mathcal{P}'}(Ax + Bu) = P', \tag{7.40}$$

where $x' = Ax+Bu$. Observation P' belongs to $\Upsilon_\| \left(\delta\left(q_\|, x\right)\right)$, where $\delta : Q_\| \times \mathbb{R}^n \to 2^{Q_\|}$ defines the discrete dynamics such that

$$\delta(q_\|, x) = \left\{q'_\| \in Q_\| \mid q_\| \longrightarrow_\| q'_\|\right\}. \tag{7.41}$$

7.8 Software Implementation of Linear Hybrid System

Linear hybrid system $H = \left(Q_{\|}, Q_{\|}^0, (A, B), \delta, \mathcal{U}\right)$ defined in Eq. (7.36) can be seen as an abstract description of the embedded software required for its implementation.

The continuous elements of linear hybrid system H is a linear control system Σ_κ defined by the pair (A_κ, B_κ) which is assumed to be in Brunovsky normal form and the input sets $\mathcal{U}(q_{\|,1}, x_\kappa), \ldots, \mathcal{U}(q_{\|,i}, x_\kappa)$ for $i = 1, \ldots, |Q_{\|}|$. The discrete elements of H are the set of states $Q_{\|}$, set of initial states $Q_{\|}^0$, and $\delta : Q_{\|} \times \mathbb{R}^n \to 2^{Q_{\|}}$ defining the discrete dynamics.

The software implementation of linear hybrid system H in SIMULINK and STATEFLOW is shown in Figure 7.6.

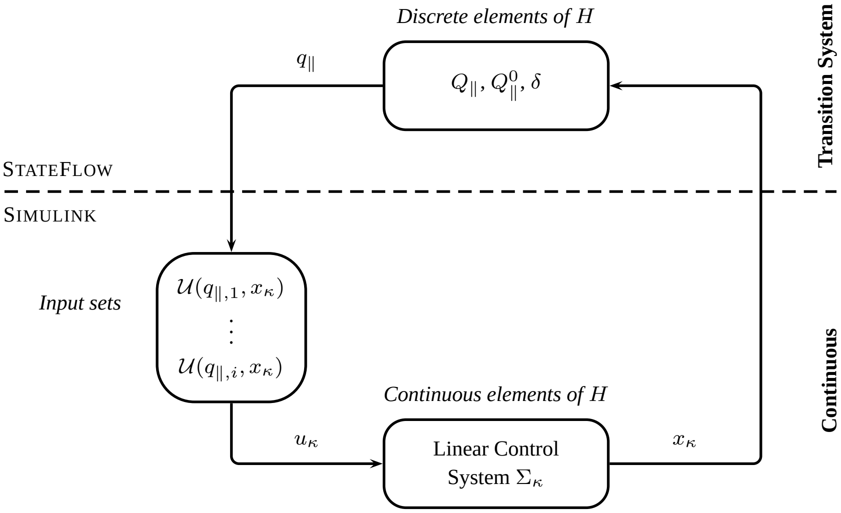

Discrete elements of H
$q_{\|}$
$Q_{\|}, Q_{\|}^0, \delta$
Transition System
STATEFLOW
SIMULINK
Input sets
$\mathcal{U}(q_{\|,1}, x_\kappa)$
$\vdots$
$\mathcal{U}(q_{\|,i}, x_\kappa)$
Continuous elements of H
Continuous
u_κ
Linear Control System Σ_κ
x_κ

Figure 7.6: *Software implementation of linear hybrid system H in* SIMULINK *and* STATEFLOW.

Transition system defined by $(Q_{\|}, Q_{\|}^0, \delta)$ starts in initial discrete state $q_0 \in Q_{\|}^0$ determined by initial continuous state $x_{\kappa,0} \in \mathbb{R}^n$. Recall, that for discrete state $q_{\|,i} \in Q_{\|}$ for $i = 1, \ldots, |Q_{\|}|$ an input set $\mathcal{U}(q_{\|,i}, x_\kappa)$ has been determined.

This means that if transition system ($Q_{\|}$, $Q^0_{\|}$, δ) is in discrete state $q_{\|,2} \in Q_{\|}$ then input set $\mathcal{U}(q_{\|,2}, x_\kappa)$ is chosen. Thus, in discrete state $q_{\|,i} \in Q_{\|}$ the input u_κ to linear control system Σ_κ has to be chosen such that $u_\kappa \in \mathcal{U}(q_{\|,i}, x_\kappa)$. If transition system ($Q_{\|}$, $Q^0_{\|}$, δ) takes a transition $q_{\|,2} \longrightarrow_{\|} q_{\|,1}$ then input set $\mathcal{U}(q_{\|,1}, x_\kappa)$ is chosen. This will generate a new input $u_\kappa \in \mathcal{U}(q_{\|,1}, x_\kappa)$ to linear control system Σ_κ and the continuous state x_κ changes according to $x'_\kappa = A_\kappa x_\kappa + B_\kappa u_\kappa$ which then becomes the new input to transition system ($Q_{\|}$, $Q^0_{\|}$, δ).

Recall that linear control system Σ_κ in Brunovsky normal form and linear control system Σ is related by the isomorphism U, defined in Eq. (7.14), that is

$$\begin{bmatrix} x \\ u \end{bmatrix} = U^{-1} \begin{bmatrix} x_\kappa \\ u_\kappa \end{bmatrix} = \begin{bmatrix} H^{-1} & 0_{n\times m} \\ -V^{-1}FH^{-1} & V^{-1} \end{bmatrix} \begin{bmatrix} x_\kappa \\ u_\kappa \end{bmatrix}. \qquad (7.42)$$

This allows linear control system in Brunovsky normal form Σ_κ from Figure 7.6 to be replaced with Σ as shown in Figure 7.7.

7.9 Summary

In this chapter a framework for controller synthesis for linear control systems with respect to formal requirement specification in Linear-time Temporal Logic (LTL) was presented. However, in order to fit with the problem of synthesizing a controller for a mobile robot, given a requirement specification in LTL, some modifications of the framework originally developed by Tabuada and Pappas [Tabuada and Pappas, 2006] were required (See steps 1,2, and 8 in the beginning of this chapter). Finally, the software implementation of linear hybrid system H in SIMULINK and STATEFLOW was presented.

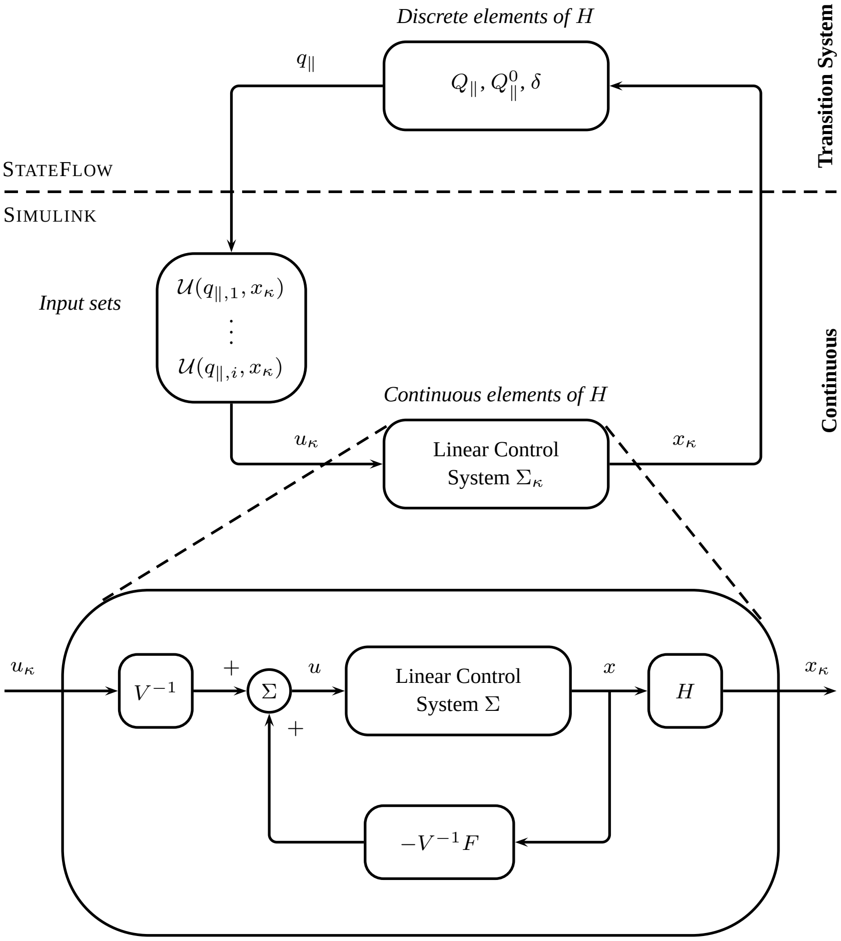

Discrete elements of H
$q_{\|}$
$Q_{\|}, Q_{\|}^{0}, \delta$
Transition System
STATEFLOW
SIMULINK
Input sets
$\mathcal{U}(q_{\|,1}, x_\kappa)$
$\vdots$
$\mathcal{U}(q_{\|,i}, x_\kappa)$
Continuous elements of H
Continuous
u_κ
Linear Control System Σ_κ
x_κ
u_κ
V^{-1}
+
Σ
+
u
Linear Control System Σ
x
H
x_κ
$-V^{-1}F$

Figure 7.7: *Software implementation of linear hybrid system H in* SIMULINK *and* STATEFLOW *where linear control system Σ_κ is replaced by Σ.*

Chapter 8

Case Study II : Robot Controller Synthesis

In this chapter the framework for controller synthesis presented in the previous chapter is applied to a simple model model of a nonholonomic wheeled mobile robot, i.e. an unicycle. The requirement specification expressing the desired behavior of the unicycle is formulated in Linear-time Temporal Logic (LTL). The results of simulating the software implementation of the linear hybrid system are presented. Finally, a conclusive discussion of the novel framework proposed is given.

8.1 Modeling the Unicycle

The simplest model of a nonholonomic wheeled mobile robot is the unicycle [Oriolo et al., 2002], i.e. a single upright wheel rolling on the plane, see Figure 8.1.

The generalized coordinates for the unicycle is defined as

$$q = \begin{bmatrix} p_1 & p_2 & \theta \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^2 \times \mathbb{S}^1, \tag{8.1}$$

where $(p_1, p_2) \in \mathbb{R}^2$ is the position (m) of the vehicle in the $\mathcal{F}^w$-frame and $\theta \in \mathbb{S}^1$ is the orientation (rad) of the vehicle. The kinematic model of the unicycle is

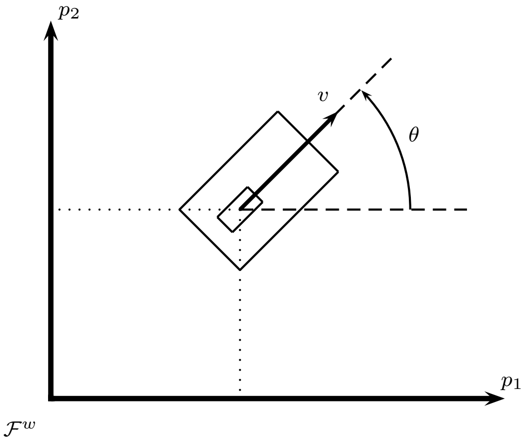


Figure 8.1: *Unicycle.*

of the form $\dot{q} = G(q)w$, that is

$$\Sigma_* \ : \ \dot{q} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega,$$

$$= \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{8.2}$$

where the inputs $v \in \mathbb{R}$ and $\omega \in \mathbb{R}$ are the linear velocity (m/s) and angular velocity (rad/s), respectively. The constraint that the wheel cannot slip in the lateral direction is expressed as

$$A(q)\dot{q} = \begin{bmatrix} \sin\theta & -\cos\theta \end{bmatrix} \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \end{bmatrix} = 0. \tag{8.3}$$

8.2 Dynamic Feedback Linearization

Define the linearizing output as

$$\eta = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}. \tag{8.4}$$

Differentiation of η with respect to time yields

$$\dot{\eta} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{8.5}$$

This shows that only the linear velocity v affect $\dot{\eta}$. Further, the angular velocity ω can not be recovered from Eq. (8.5). Therefore, we need to have the linear acceleration a as input to the system. Thus, we need to add an integrator on the linear velocity input v as shown in Figure 8.2.

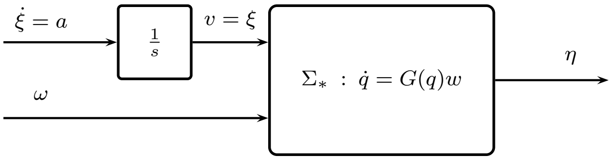


Figure 8.2: *System with modified input* $(a, \omega) \in \mathbb{R} \times \mathbb{R}$.

The state of the integrator is denoted by $\xi \in \mathbb{R}$. The modified input to the system then becomes $(a, \omega) \in \mathbb{R} \times \mathbb{R}$, where a is the linear acceleration (m/s^2) of the unicycle. The introduction of the state ξ of the integrator allows Eq. (8.5) to be rewritten as

$$\dot{\eta} = \xi \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}. \tag{8.6}$$

Differentiation of Eq. (8.6) once more then yields

$$\begin{aligned} \ddot{\eta} &= \dot{\xi} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} + \xi\dot{\theta} \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta & -\xi\sin\theta \\ \sin\theta & \xi\cos\theta \end{bmatrix} \begin{bmatrix} a \\ \omega \end{bmatrix}. \end{aligned} \tag{8.7}$$

The matrix multiplying the modified input (a, ω) is nonsingular for $\xi \neq 0$. Under the assumption that $\xi \neq 0$, we have from Eq. (8.7) that

$$\begin{aligned} \begin{bmatrix} a \\ \omega \end{bmatrix} &= \begin{bmatrix} \cos\theta & -\xi\sin\theta \\ \sin\theta & \xi\cos\theta \end{bmatrix}^{-1} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta & \sin\theta \\ -\frac{1}{\xi}\sin\theta & \frac{1}{\xi}\cos\theta \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix}, \end{aligned} \tag{8.8}$$

where $(\nu_1, \nu_2) \in \mathbb{R} \times \mathbb{R}$ is input to the dynamic compensator. Inserting Eq. (8.8) into Eq. (8.7) yields

$$\ddot{\eta} = \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix}, \tag{8.9}$$

showing that the input appears. The resulting dynamic compensator is obtained from Eq. (8.8) and has the form of Eq. (7.2), i.e.

$$\dot{\xi} = \begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix}, \tag{8.10}$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \xi \\ 0 \end{bmatrix} + \frac{1}{\xi} \begin{bmatrix} 0 & 0 \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix}. \tag{8.11}$$

The system with dynamic compensator is shown in Figure 8.3.

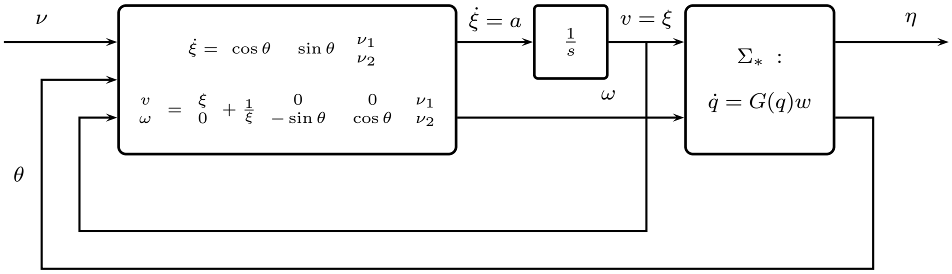

ν
$\dot{\xi} = \cos\theta \quad \sin\theta \quad \nu_1 \; \nu_2$
$v \; \omega = \xi \; 0 + \frac{1}{\xi} \quad 0 \quad 0 \quad -\sin\theta \quad \cos\theta \quad \nu_1 \; \nu_2$
$\dot{\xi} = a$
$\frac{1}{s}$
$v = \xi$
ω
$\Sigma_* :$
$\dot{q} = G(q)w$
η
θ

Figure 8.3: *Nonlinear system Σ_* with dynamic compensator.*

The dynamic compensator has a singularity at $\xi = 0$, i.e. when the unicycle is not rolling. We now apply the state transformation $z = T(q, \xi)$ to obtain

$$z_1 = p_1, \tag{8.12}$$

$$z_2 = \dot{p}_1 = \xi \cos\theta, \tag{8.13}$$

$$z_3 = p_2, \tag{8.14}$$

$$z_4 = \dot{p}_2 = \xi \sin\theta. \tag{8.15}$$

Further, we have that

$$\begin{aligned}\ddot{z}_1 &= \ddot{p}_1 = \dot{\xi}\cos\theta - \xi\dot{\theta}\sin\theta = a\cos\theta - \xi\omega\sin\theta,\\ &= \nu_1, \end{aligned} \tag{8.16}$$

$$\begin{aligned}\ddot{z}_3 &= \ddot{p}_2 = \dot{\xi}\sin\theta + \xi\dot{\theta}\cos\theta = a\sin\theta + \xi\omega\cos\theta,\\ &= \nu_2. \end{aligned} \tag{8.17}$$

Thus, the closed-loop system is equivalent to a set of decoupled input-output chains of integrators from ν_i to η_i for $i = 1, 2$ as illustrated in Figure 8.4.

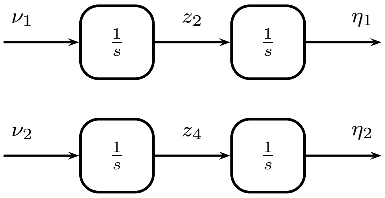


Figure 8.4: *Decoupled input-output chains of integrators.*

Defining the state as $z = \begin{bmatrix} z_1 & z_2 & z_3 & z_4 \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^4$ the following continuous-time linear control system is obtained

$$\Sigma_z \;:\; \dot{z}(t) = A_z z(t) + B_z \nu(t), \tag{8.18}$$

where matrices $A_z \in \mathbb{R}^{4\times 4}$ and $B_z \in \mathbb{R}^{4\times 2}$ are given as

$$A_z = \begin{bmatrix} 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B_z = \begin{bmatrix} 0 & 0\\ 1 & 0\\ 0 & 0\\ 0 & 1 \end{bmatrix}. \tag{8.19}$$

The system in Eq. (8.18) is controllable since the controllability matrix has full rank. The discrete equivalent[1], i.e. discrete-time linear control system is obtained as

$$\Sigma \;:\; x(t+1) = Ax(t) + Bu(t), \tag{8.20}$$

[1] Using ZOH method with a sample time of $T_s = 1s$.

where matrices $A \in \mathbb{R}^{4\times 4}$ and $B \in \mathbb{R}^{4\times 2}$ are given as

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{2} & 0 \\ 1 & 0 \\ 0 & \frac{1}{2} \\ 0 & 1 \end{bmatrix}. \tag{8.21}$$

Choosing the controllability indices[2] as $\kappa = (2, 2)$ allows to find invertible linear transformations $H \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times m}$, and linear transformation $F \in \mathbb{R}^{m\times n}$ such that the pair $(A_\kappa, B_\kappa) \in \mathbb{R}^{4\times 4} \times \mathbb{R}^{4\times 2}$ given as

$$A_\kappa = H(A - BF)H^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B_\kappa = HBV = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \tag{8.22}$$

of linear control system

$$\Sigma_\kappa \; : \; x_\kappa(t+1) = A_\kappa x_\kappa(t) + B_\kappa u_\kappa(t), \tag{8.23}$$

are in Brunovsky normal form. The system in Brunovsky normal form is related to linear control system Σ by an invertible state/input transformation matrix $U \; : \; \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$ that is computed using Algorithm 1, that is

$$\begin{bmatrix} x_\kappa \\ u_\kappa \end{bmatrix} = U \begin{bmatrix} x \\ u \end{bmatrix} = \begin{bmatrix} \underbrace{\begin{pmatrix} 1 & -\frac{1}{2} & 0 & 0 \\ 1 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & -\frac{1}{2} \\ 0 & 0 & 1 & \frac{1}{2} \end{pmatrix}}_{H} & 0_{4\times 2} \\ \underbrace{\begin{pmatrix} 1 & \frac{3}{2} & 0 & 0 \\ 0 & 0 & 1 & \frac{3}{2} \end{pmatrix}}_{F} & \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{V} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix}. \tag{8.24}$$

The controllability indices $\kappa = (2, 2)$ leads to subspace $\mathcal{V} \cong \mathbb{R}^2$ of state-space $\mathbb{R}^4$ defined by

$$\mathcal{V} \cong \operatorname{span}\{b_{\kappa,1}, b_{\kappa,2}\} = \operatorname{span}\left\{ \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\}. \tag{8.25}$$

[2] Note that controllability indices $\kappa = (3, 1)$ would equivalently work.

The observation-space is given by

$$\mathbb{R}^2 \cong \mathbb{R}^4/\mathcal{V} = \mathbb{R}^4/\operatorname{span}\left\{\begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}\right\} = \operatorname{span}\left\{\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}\right\}. \tag{8.26}$$

Subspace $\mathcal{V}$ induces the observation map $\Upsilon_{\Sigma_\kappa} : \mathbb{R}^4 \to \mathbb{R}^4/\operatorname{span}\{b_{\kappa,1}, b_{\kappa,2}\}$. Thus, the observation $y_\kappa \in \mathbb{R}^2$ is given by

$$\begin{bmatrix}y_{\kappa,1}\\y_{\kappa,2}\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_{\kappa,1}\\x_{\kappa,2}\\x_{\kappa,3}\\x_{\kappa,4}\end{bmatrix}. \tag{8.27}$$

Transition system T_{Σ_κ} associated with linear control system Σ_κ is defined by

$$T_{\Sigma_\kappa} = (\mathbb{R}^4, \mathbb{R}^4, \longrightarrow, \mathbb{R}^2, \Upsilon_{\Sigma_\kappa}), \tag{8.28}$$

where,

- $\longrightarrow \subseteq \mathbb{R}^4 \times \mathbb{R}^4$ is a transition relation defined by $x_\kappa \longrightarrow x'_\kappa$ if there exists an input $u_\kappa \in \mathbb{R}^2$ such that $x'_\kappa = A_\kappa x_\kappa + B_\kappa u_\kappa$,
- $\Upsilon_{\Sigma_\kappa} : \mathbb{R}^4 \to \mathbb{R}^2$ is an observation map defined by $\Upsilon_{\Sigma_\kappa}(x_\kappa) = y_\kappa$.

8.3 Requirement Specification

In the following a simple requirement specification is formulated in LTL. For simplicity lets assume that the unicycle starts at some initial position, given by the observation $y_\kappa \in \mathbb{R}^2$. The requirement is now to eventually bring the unicycle, i.e. the observation $y_\kappa \in \mathbb{R}^2$ to a set $[y_\kappa] \subseteq \mathbb{R}^2$ defined by

$$[y_\kappa] = \left\{ y_\kappa \in \mathbb{R}^2 \mid -\frac{1}{2} \le y_{\kappa,1} \le \frac{1}{2} \wedge -\frac{1}{2} \le y_{\kappa,2} \le \frac{1}{2} \right\}, \tag{8.29}$$

within 3 time units and stay there for all future time. The set $[y_\kappa]$ representing the desired goal position is shown in Figure 8.5. This requirement is captured in the following LTL formula

$$\phi = \Diamond_3 \Box S_1, \tag{8.30}$$

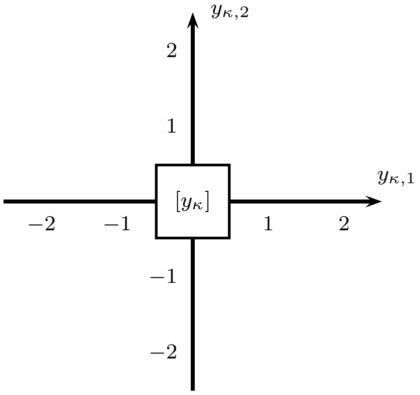


Figure 8.5: *Desired goal position represented by a set* $[y_\kappa]$.

where S_1 denotes the set defined by $p_1 \wedge p_2 \wedge p_3 \wedge p_4$ where p_i for $i = 1, \dots, 4$ are defined as

$$p_1 = y_{\kappa,1} + \frac{1}{2} > 0, \tag{8.31}$$

$$p_2 = -y_{\kappa,1} + \frac{1}{2} > 0, \tag{8.32}$$

$$p_3 = y_{\kappa,2} + \frac{1}{2} > 0, \tag{8.33}$$

$$p_4 = -y_{\kappa,2} + \frac{1}{2} > 0, \tag{8.34}$$

Thus, the initial partition of the observation-space $\mathbb{R}^2$ is

$$\mathcal{P} = \{S_1, S_2\}, \tag{8.35}$$

where S_2 denotes the compliment of S_1. The sets S_1 and S_2 cover observation-space $\mathbb{R}^2$ and are defined as

$$S_1 = \left\{ y_\kappa \in \mathbb{R}^2 \mid -\frac{1}{2} < y_{\kappa,1} < \frac{1}{2} \wedge -\frac{1}{2} < y_{\kappa,2} < \frac{1}{2} \right\}, \tag{8.36}$$

$$S_2 = \left\{ y_\kappa \in \mathbb{R}^2 \mid \left(y_{\kappa,1} \leq -\frac{1}{2} \vee y_{\kappa,1} \geq \frac{1}{2} \right) \wedge \left(y_{\kappa,2} \leq -\frac{1}{2} \vee y_{\kappa,2} \geq \frac{1}{2} \right) \right\}. \tag{8.37}$$

The initial partition of the observation-space $\mathbb{R}^2$ is shown in Figure 8.6.

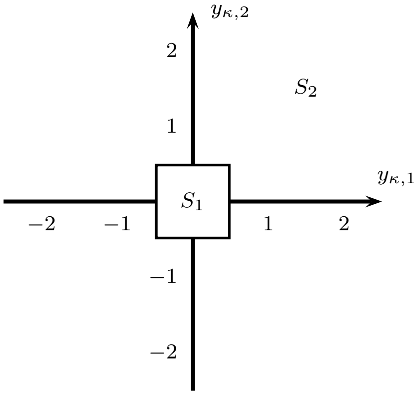


Figure 8.6: *Initial partition* $\mathcal{P} = \{S_1, S_2\}$ *of observation-space* $\mathbb{R}^2$.

8.4 Computing the Abstraction

We now compute the finite bisimilar quotient using the bisimulation algorithm (Algorithm 2). The initial partition of state-space $\mathbb{R}^4$ is

$$\begin{aligned} \mathcal{P}' &= \Upsilon^{-1}(\mathcal{P}), \\ &= \left\{\Upsilon^{-1}(S_1), \Upsilon^{-1}(S_2)\right\}, \end{aligned} \tag{8.38}$$

where

$$P_1 = \Upsilon^{-1}(S_1) = \left\{x_\kappa \in \mathbb{R}^4 \mid -\frac{1}{2} < x_{\kappa,1} < \frac{1}{2} \wedge -\frac{1}{2} < x_{\kappa,3} < \frac{1}{2}\right\}, \tag{8.39}$$

$$\begin{aligned} P_2 = \Upsilon^{-1}(S_2) = \Bigg\{x_\kappa \in \mathbb{R}^4 \mid & \left(x_{\kappa,1} \leq -\frac{1}{2} \vee x_{\kappa,1} \geq \frac{1}{2}\right) \wedge \\ & \left(x_{\kappa,3} \leq -\frac{1}{2} \vee x_{\kappa,3} \geq \frac{1}{2}\right)\Bigg\}. \end{aligned} \tag{8.40}$$

Choosing $P = P_2$ and $P' = P_1$ we compute

$$\mathrm{Pre}(P') = \left\{x_\kappa \in \mathbb{R}^4 \mid -\frac{1}{2} < x_{\kappa,2} < \frac{1}{2} \ \wedge \ -\frac{1}{2} < x_{\kappa,4} < \frac{1}{2}\right\}, \quad (8.41)$$

$$\begin{aligned} P \cap \mathrm{Pre}(P') = \Bigg\{x_\kappa \in \mathbb{R}^4 \mid \Bigg(\Bigg(x_{\kappa,1} \leq -\frac{1}{2} \vee x_{\kappa,1} \geq \frac{1}{2}\Bigg) \wedge \\ \Bigg(x_{\kappa,3} \leq -\frac{1}{2} \vee x_{\kappa,3} \geq \frac{1}{2}\Bigg)\Bigg) \wedge \\ \Bigg(-\frac{1}{2} < x_{\kappa,2} < \frac{1}{2} \ \wedge \ -\frac{1}{2} < x_{\kappa,4} < \frac{1}{2}\Bigg)\Bigg\}. \end{aligned} \quad (8.42)$$

Since $P \cap \mathrm{Pre}(P') \neq \emptyset$ and $P \cap \mathrm{Pre}(P') \neq P$, the set P_2 is split into sets P_{21} and P_{22} which are computed as follows

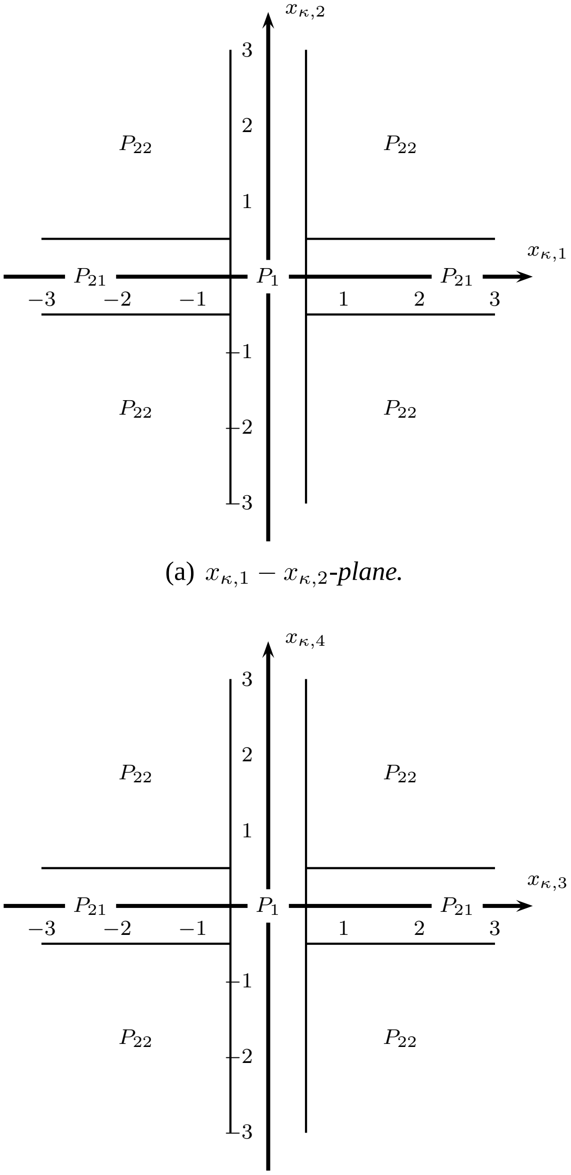
$$\begin{aligned} P_{21} &= P \cap \mathrm{Pre}(P') \\ &= \Bigg\{x_\kappa \in \mathbb{R}^4 \mid \Bigg(\Bigg(x_{\kappa,1} \leq -\frac{1}{2} \vee x_{\kappa,1} \geq \frac{1}{2}\Bigg) \wedge \\ &\Bigg(x_{\kappa,3} \leq -\frac{1}{2} \vee x_{\kappa,3} \geq \frac{1}{2}\Bigg)\Bigg) \wedge \Bigg(-\frac{1}{2} < x_{\kappa,2} < \frac{1}{2} \ \wedge \ -\frac{1}{2} < x_{\kappa,4} < \frac{1}{2}\Bigg)\Bigg\}, \end{aligned} \quad (8.43)$$

$$\begin{aligned} P_{22} &= P \cap \overline{\mathrm{Pre}(P')} \\ &= \Bigg\{x_\kappa \in \mathbb{R}^4 \mid \Bigg(\Bigg(x_{\kappa,1} \leq -\frac{1}{2} \vee x_{\kappa,1} \geq \frac{1}{2}\Bigg) \vee \Bigg(x_{\kappa,3} \leq -\frac{1}{2} \vee x_{\kappa,3} \geq \frac{1}{2}\Bigg)\Bigg) \wedge \\ &\Bigg(\Bigg(x_{\kappa,2} \leq -\frac{1}{2} \ \vee \ x_{\kappa,2} \geq \frac{1}{2}\Bigg) \wedge \Bigg(x_{\kappa,4} \leq -\frac{1}{2} \ \vee \ x_{\kappa,4} \geq \frac{1}{2}\Bigg)\Bigg)\Bigg\}. \end{aligned} \quad (8.44)$$

The refined partition is now given by

$$\mathcal{P}' = (\{P_1, P_2\} \setminus \{P_2\}) \cap \{P_{21}, P_{22}\} = \{P_1, P_{21}, P_{22}\}, \quad (8.45)$$

and shown in Figure 8.7.



(a) $x_{\kappa,1} - x_{\kappa,2}$*-plane.*

(b) $x_{\kappa,3} - x_{\kappa,4}$*-plane.*

Figure 8.7: *Refined partition* $\mathcal{P}' = \{P_1, P_{21}, P_{22}\}$ *of state-space* $\mathbb{R}^4$.

Now, choosing $P = P_1$ and $P' = P_{21}$ we compute

$$\operatorname{Pre}(P') = \left\{x_\kappa \in \mathbb{R}^4 \mid \left(x_{\kappa,2} \leq -\frac{1}{2} \vee x_{\kappa,2} \geq \frac{1}{2}\right) \wedge \left(x_{\kappa,4} \leq -\frac{1}{2} \vee x_{\kappa,4} \geq \frac{1}{2}\right)\right\}, \tag{8.46}$$

$$P \cap \operatorname{Pre}(P') = \left\{x_\kappa \in \mathbb{R}^4 \mid \left(-\frac{1}{2} < x_{\kappa,1} < \frac{1}{2} \wedge -\frac{1}{2} < x_{\kappa,3} < \frac{1}{2}\right) \wedge \left(\left(x_{\kappa,2} \leq -\frac{1}{2} \vee x_{\kappa,2} \geq \frac{1}{2}\right) \wedge \left(x_{\kappa,4} \leq -\frac{1}{2} \vee x_{\kappa,4} \geq \frac{1}{2}\right)\right)\right\}. \tag{8.47}$$

Since $P \cap \operatorname{Pre}(P') \neq \emptyset$ and $P \cap \operatorname{Pre}(P') \neq P$, the set P_1 is split into sets P_{11} and P_{12}. The sets P_{11} and P_{12} are computed as follows

$$\begin{aligned} P_{11} &= P \cap \operatorname{Pre}(P') \\ &= \left\{x_\kappa \in \mathbb{R}^4 \mid \left(-\frac{1}{2} < x_{\kappa,1} < \frac{1}{2} \wedge -\frac{1}{2} < x_{\kappa,3} < \frac{1}{2}\right) \wedge \right. \\ &\left.\left(\left(x_{\kappa,2} \leq -\frac{1}{2} \vee x_{\kappa,2} \geq \frac{1}{2}\right) \wedge \left(x_{\kappa,4} \leq -\frac{1}{2} \vee x_{\kappa,4} \geq \frac{1}{2}\right)\right)\right\}, \end{aligned} \tag{8.48}$$

$$\begin{aligned} P_{12} &= P \cap \overline{\operatorname{Pre}(P')} \\ &= \left\{x_\kappa \in \mathbb{R}^4 \mid \left(-\frac{1}{2} < x_{\kappa,1} < \frac{1}{2} \ \wedge \ -\frac{1}{2} < x_{\kappa,3} < \frac{1}{2}\right) \wedge \right. \\ &\left.\left(-\frac{1}{2} < x_{\kappa,2} < \frac{1}{2} \ \wedge \ -\frac{1}{2} < x_{\kappa,4} < \frac{1}{2}\right)\right\}. \end{aligned} \tag{8.49}$$

The refined partition is now given by

$$\mathcal{P}' = (\{P_1, P_{21}, P_{22}\} \setminus \{P_1\}) \cap \{P_{11}, P_{12}\} = \{P_{21}, P_{22}, P_{11}, P_{12}\}, \tag{8.50}$$

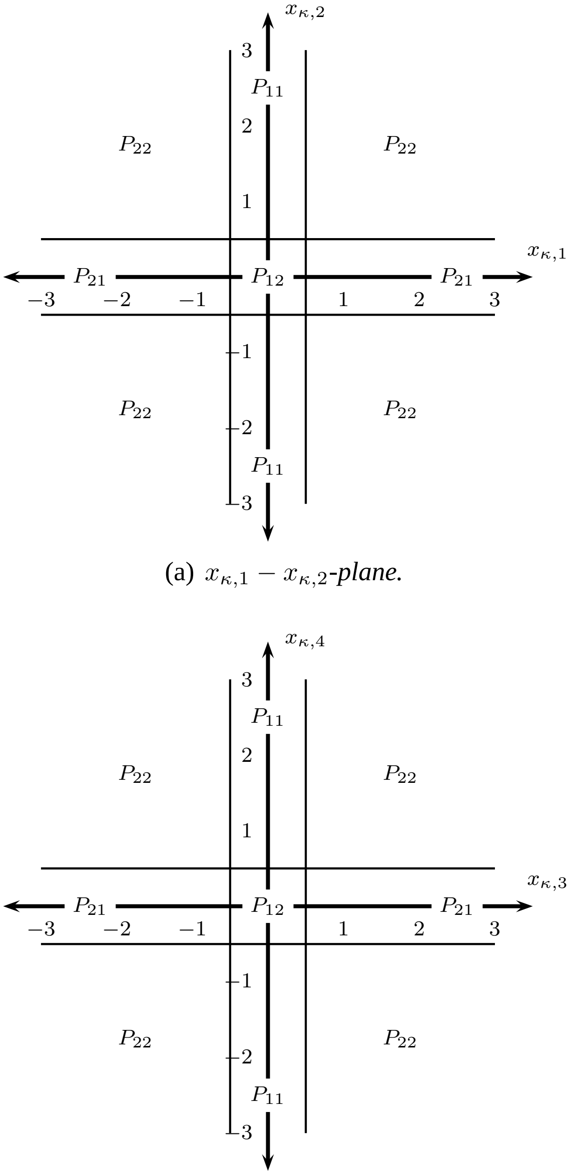
and shown in Figure 8.8.

We further compute the following to show that for any $P, P' \in \mathcal{P}'$, $P \cap \operatorname{Pre}(P') \neq \emptyset$ and $P \cap \operatorname{Pre}(P') \neq P$ are not satisfied, see Table 8.1.

The finite bisimilar quotient is now defined as

$$T_{/\sim}^{\mathcal{P}'} = \{Q_{/\sim}, Q_{/\sim}^0, \longrightarrow_{/\sim}, O, \Upsilon_{/\sim}\}, \tag{8.51}$$

where



(a) $x_{\kappa,1} - x_{\kappa,2}$*-plane.*

(b) $x_{\kappa,3} - x_{\kappa,4}$*-plane.*

Figure 8.8: *Refined partition* $\mathcal{P}' = \{P_{21}, P_{22}, P_{11}, P_{12}\}$ *of state-space* $\mathbb{R}^4$.

$\cap$	$\mathrm{Pre}(P_1)$	$\mathrm{Pre}(P_2)$	$\mathrm{Pre}(P_3)$	$\mathrm{Pre}(P_4)$
P_1	$\emptyset$	$\emptyset$	P_1	P_1
P_2	P_2	P_2	$\emptyset$	$\emptyset$
P_3	P_3	P_3	$\emptyset$	$\emptyset$
P_4	$\emptyset$	$\emptyset$	P_3	P_3

Table 8.1: *For any $P, P' \in \mathcal{P}'$, $P \cap \mathrm{Pre}(P') \neq \emptyset$ and $P \cap \mathrm{Pre}(P') \neq P$ are not satisfied.*

- $Q_{/\sim} = \{q \subseteq \mathbb{R}^4 \mid q$ is an equivalence class of $\sim \subseteq \mathbb{R}^4 \times \mathbb{R}^4\} = \{q_1, q_2, q_3, q_4\} = \{P_{21}, P_{22}, P_{11}, P_{12}\}$ defined by

$$q_1 = \Bigg\{ x_\kappa \in \mathbb{R}^4 \mid \Bigg(\Bigg(x_{\kappa,1} \leq -\frac{1}{2} \vee x_{\kappa,1} \geq \frac{1}{2} \Bigg) \wedge$$
$$\Bigg(x_{\kappa,3} \leq -\frac{1}{2} \vee x_{\kappa,3} \geq \frac{1}{2} \Bigg) \Bigg) \wedge \Bigg(-\frac{1}{2} < x_{\kappa,2} < \frac{1}{2} \ \wedge \ -\frac{1}{2} < x_{\kappa,4} < \frac{1}{2} \Bigg) \Bigg\},$$
$$q_2 = \Bigg\{ x_\kappa \in \mathbb{R}^4 \mid \Bigg(\Bigg(x_{\kappa,1} \leq -\frac{1}{2} \vee x_{\kappa,1} \geq \frac{1}{2} \Bigg) \vee$$
$$\Bigg(x_{\kappa,3} \leq -\frac{1}{2} \vee x_{\kappa,3} \geq \frac{1}{2} \Bigg) \Bigg) \ \wedge \Bigg(\Bigg(x_{\kappa,2} \leq -\frac{1}{2} \ \vee \ x_{\kappa,2} \geq \frac{1}{2} \Bigg) \wedge$$
$$\Bigg(x_{\kappa,4} \leq -\frac{1}{2} \ \vee \ x_{\kappa,4} \geq \frac{1}{2} \Bigg) \Bigg) \Bigg\},$$
$$q_3 = \Bigg\{ x_\kappa \in \mathbb{R}^4 \mid \Bigg(-\frac{1}{2} < x_{\kappa,1} < \frac{1}{2} \wedge -\frac{1}{2} < x_{\kappa,3} < \frac{1}{2} \Bigg) \ \wedge$$
$$\Bigg(\Bigg(x_{\kappa,2} \leq -\frac{1}{2} \vee x_{\kappa,2} \geq \frac{1}{2} \Bigg) \wedge \Bigg(x_{\kappa,4} \leq -\frac{1}{2} \vee x_{\kappa,4} \geq \frac{1}{2} \Bigg) \Bigg) \Bigg\},$$
$$q_4 = \Bigg\{ x_\kappa \in \mathbb{R}^4 \mid \Bigg(-\frac{1}{2} < x_{\kappa,1} < \frac{1}{2} \ \wedge \ -\frac{1}{2} < x_{\kappa,3} < \frac{1}{2} \Bigg) \wedge$$
$$\Bigg(-\frac{1}{2} < x_{\kappa,2} < \frac{1}{2} \ \wedge \ -\frac{1}{2} < x_{\kappa,4} < \frac{1}{2} \Bigg) \Bigg\},$$

- $Q^0_{/\sim} = Q_{/\sim}$,

- $\longrightarrow_{/\sim} \subseteq Q_{/\sim} \times Q_{/\sim}$ defined as $\longrightarrow_{/\sim} = \{(q_1, q_3), (q_1, q_4), (q_2, q_1), (q_2, q_2), (q_3, q_1), (q_3, q_2), (q_4, q_3), (q_4, q_4)\}$,

- $O = \{q_1, q_2, q_3, q_4\}$,

- $\Upsilon_{/\sim} : Q_{/\sim} \to O$ defined as $\Upsilon_{/\sim}(q_i) = q_i$ for $i = 1, \ldots, 4$.

Transition relation $\longrightarrow_{/\sim} \subseteq Q_{/\sim} \times Q_{/\sim}$ is determined by computing

$$\begin{aligned}
\mathrm{Pre}(q_1) &= \Bigg\{ x_\kappa \in \mathbb{R}^4 \mid \left(x_{\kappa,2} \leq -\frac{1}{2} \ \vee \ x_{\kappa,2} \geq \frac{1}{2} \right) \wedge \\
&\left(x_{\kappa,4} \leq -\frac{1}{2} \ \vee \ x_{\kappa,4} \geq \frac{1}{2} \right) \Bigg\}, \\
&= \{q_2, q_3\} && (8.52) \\
\mathrm{Pre}(q_2) &= \Bigg\{ x_\kappa \in \mathbb{R}^4 \mid \left(x_{\kappa,2} \leq -\frac{1}{2} \ \vee \ x_{\kappa,2} \geq \frac{1}{2} \right) \wedge \\
&\left(x_{\kappa,4} \leq -\frac{1}{2} \ \vee \ x_{\kappa,4} \geq \frac{1}{2} \right) \Bigg\}, \\
&= \{q_2, q_3\} && (8.53) \\
\mathrm{Pre}(q_3) &= \left\{ x_\kappa \in \mathbb{R}^4 \mid -\frac{1}{2} < x_{\kappa,2} < \frac{1}{2} \ \wedge -\frac{1}{2} < x_{\kappa,4} \geq \frac{1}{2} \right\}, \\
&= \{q_1, q_4\} && (8.54) \\
\mathrm{Pre}(q_4) &= \left\{ x_\kappa \in \mathbb{R}^4 \mid -\frac{1}{2} < x_{\kappa,2} < \frac{1}{2} \ \wedge -\frac{1}{2} < x_{\kappa,4} \geq \frac{1}{2} \right\}, \\
&= \{q_1, q_4\}. && (8.55)
\end{aligned}$$

The finite bisimilar quotient $T^{\mathcal{P}'}_{/\sim}$ is graphically represented in Figure 8.9.

8.5 Constructing the Büchi Automaton

The Büchi automaton is given as

$$A_\phi = (T_\phi, F) = ((Q, Q^0, \longrightarrow, O, \Upsilon), F), \tag{8.56}$$

where

- $Q = \{q_1, q_2, q_3, q_4\}$,
- $Q^0 = Q$,
- $\longrightarrow \subseteq Q \times Q$ defined as $\longrightarrow = \{(q_1, q_2), (q_2, q_3), (q_3, q_4), (q_4, q_1), (q_4, q_2), (q_4, q_3), (q_4, q_4)\}$,

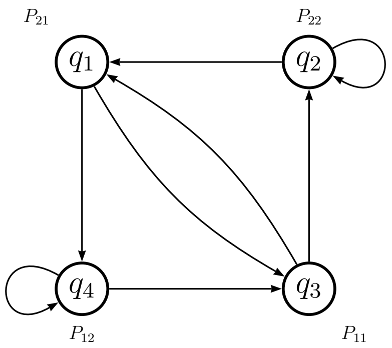


Figure 8.9: *Finite bisimilar quotient $T_{/\sim}^{\mathcal{P}'}$ of transition system T_{Σ_κ} associated with linear control system Σ_κ.*

- $O = \mathcal{P} = \{S_1, S_2\}$,
- $\Upsilon \,:\, Q \rightarrow O$ defined as

$$\Upsilon(q_i) = \begin{cases} S_1 & \text{if } i = 1, \ldots, 3 \\ S_2 & \text{if } i = 4 \end{cases}.$$

- $F = Q$.

Since $F = Q$ the Büchi automaton A_ϕ can equivalently be represented by its underlying transition system T_ϕ as shown in Figure 8.10.

8.6 Controller Synthesis

The controller is defined as

$$T_c = \left(Q_c, Q_c^0, \longrightarrow_c, O, \Upsilon_c\right), \tag{8.57}$$

where

- $Q_c = \{q_{c,1}, q_{c,2}, q_{c,3}\}$ is a set of states,
- $Q_c^0 = Q_c$ is a set of initial states,

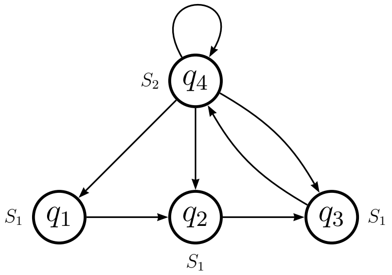


Figure 8.10: *Underlaying transition system T_ϕ corresponding to LTL formula ϕ.*

- $\longrightarrow_c \subseteq Q_c \times Q_c$ is a transition relation defined as $\longrightarrow_c = \{(q_{c,1}, q_{c,1}), (q_{c,1}, q_{c,2}), (q_{c,2}, q_{c,3}), (q_{c,3}, q_{c,1})\}$,
- $O = \{P_{11}, P_{12}, P_{21}\}$ is a set of observations,
- $\Upsilon_c : Q_c \rightarrow O$ is an observation map defined as

$$\Upsilon_c(q_{c,1}) = P_{12}, \quad \Upsilon_c(q_{c,2}) = P_{11}, \quad \Upsilon_c(q_{c,3}) = P_{21}.$$

The controller T_c is graphically illustrated in Figure 8.11.

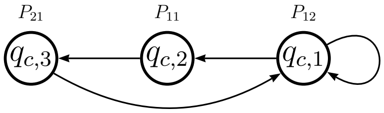


Figure 8.11: *Controller T_c.*

8.7 Refinement

We now refine that closed-loop system to obtain a linear hybrid system given by

$$H = \left(Q_{\|}, Q_{\|}^{0}, \{A_{\kappa}, B_{\kappa}\}, \delta, \mathcal{U}\right), \tag{8.58}$$

where

- $Q_{\|} = \{q_{\|,1}, q_{\|,2}, q_{\|,3}\}$ is a set of states,
- $Q_{\|}^{0} = Q_{\|}$ is a set of initial states,
- $A_{\kappa} \in \mathbb{R}^{4\times 4}, B_{\kappa} \in \mathbb{R}^{4\times 2}$ are the system matrices ghegjkhdhher in BNF of $\Sigma_{\kappa} \, : \, x_{\kappa}' = A_{\kappa}x_{\kappa} + B_{\kappa}u_{\kappa}$,
- $\delta : Q_{\|} \times \mathbb{R}^{4} \rightarrow 2^{Q_{\|}}$ is the discrete dynamics defined by

$$\begin{aligned}
\delta\left(q_{\|,1}, x_{\kappa}\right) &= \left\{q_{\|,3}\right\}, \\
\delta\left(q_{\|,2}, x_{\kappa}\right) &= \left\{q_{\|,1}\right\}, \\
\delta\left(q_{\|,3}, x_{\kappa}\right) &= \left\{q_{\|,2}, q_{\|,3}\right\}.
\end{aligned}$$

- $\mathcal{U}(q_{\|}, x_{\kappa}) = \left\{u_{\kappa} \in \mathbb{R}^{2} \mid \pi_{\mathcal{P}'}(A_{\kappa}x_{\kappa} + B_{\kappa}u_{\kappa}) \in \Upsilon_{\|}\left(\delta\left(q_{\|}, x_{\kappa}\right)\right)\right\}$ is the input set defined by

$$\begin{aligned}
\mathcal{U}\left(q_{\|,1}, x_{\kappa}\right) &= \left\{u_{\kappa} \in \mathbb{R}^{2} \mid \pi_{\mathcal{P}'}\left(A_{\kappa}x_{\kappa} + B_{\kappa}u_{\kappa}\right) \in \left\{q_{\|,3}\right\}\right\}, \\
&= \left\{u_{\kappa} \in \mathbb{R}^{2} \mid -\frac{1}{2} < u_{\kappa,1} < \frac{1}{2} \wedge -\frac{1}{2} < u_{\kappa,2} < \frac{1}{2}\right\}, \\
\mathcal{U}\left(q_{\|,2}, x_{\kappa}\right) &= \left\{u_{\kappa} \in \mathbb{R}^{2} \mid \pi_{\mathcal{P}'}\left(A_{\kappa}x_{\kappa} + B_{\kappa}u_{\kappa}\right) \in \left\{q_{\|,1}\right\}\right\}, \\
&= \left\{u_{\kappa} \in \mathbb{R}^{2} \mid -\frac{1}{2} < u_{\kappa,1} < \frac{1}{2} \wedge -\frac{1}{2} < u_{\kappa,2} < \frac{1}{2}\right\}, \\
\mathcal{U}\left(q_{\|,3}, x_{\kappa}\right) &= \left\{u_{\kappa} \in \mathbb{R}^{2} \mid \pi_{\mathcal{P}'}\left(A_{\kappa}x_{\kappa} + B_{\kappa}u_{\kappa}\right) \in \left\{q_{\|,3}, q_{\|,2}\right\}\right\}, \\
&= \Bigg\{ \left(u_{\kappa} \in \mathbb{R}^{2} \mid -\frac{1}{2} < u_{\kappa,1} < \frac{1}{2} \wedge -\frac{1}{2} < u_{\kappa,2} < \frac{1}{2}\right) \wedge \\
&\left(u_{\kappa,1} \leq -\frac{1}{2} \vee u_{\kappa,1} \geq \frac{1}{2}\right) \wedge \left(u_{\kappa,2} \leq -\frac{1}{2} \vee u_{\kappa,2} \geq \frac{1}{2}\right) \Bigg\}.
\end{aligned}$$

Each state $q_{\|} \in Q_{\|}$ is equipped with a linear control system of the form $\Sigma_{\kappa} \, : \, x_{\kappa}' = A_{\kappa}x_{\kappa} + B_{\kappa}u_{\kappa}$ with input $u_{\kappa} \in \mathcal{U}\left(q_{\|}, x_{\kappa}\right)$. Thus, in $q_{\|,3} \in Q_{\|}$ if the input is chosen such that $u_{\kappa} \in \mathcal{U}\left(q_{\|,3}, x_{\kappa}\right)$ this will result in a transition to $q_{\|}' \in \delta(q_{\|,3}, x_{\kappa})$.

8.7.1 Computing the Input Sets

In the following we show how to compute the input set $\mathcal{U}\left(q_{\|,1}, x_\kappa\right)$ for $q_{\|,1} \in Q_\|$. First, we need to determine which transitions $T^{\mathcal{P}'}_{/\sim}$ are allowed in $q_{\|,1} \in Q_\|$ by the controller T_c. Thus, we need to recover from $q_{\|,1} \in Q_\|$ the state of the finite bisimilar quotient $Q_{/\sim}$ as

$$\pi_\| \left(q_{\|,1}\right) = q_1. \tag{8.59}$$

From the transition $q_1 \longrightarrow_{/\sim} q_4$ given by

$$\pi_\| \left(q_{\|,3}\right) = q_4, \tag{8.60}$$

and bisimilarity between bisimilar quotient transition system $T^{\mathcal{P}'}_{/\sim}$ and transition system T_{Σ_κ} it follows that for any continuous state $x_\kappa \in q_1$ the following is satisfied

$$x_\kappa \longrightarrow_{\Sigma_\kappa} x'_\kappa \in q_4. \tag{8.61}$$

This is equivalent to the existence of an input $u_\kappa \in \mathbb{R}^2$ such that $x'_\kappa = A_\kappa x_\kappa + B_\kappa u_\kappa$, where $x'_\kappa \in q_4$. From the definition of $q_4 \in Q_{/\sim}$ and linear control system Σ_κ the input set $\mathcal{U}(q_{\|,1}, x_\kappa)$ in discrete state $q_{\|,1} \in Q_\|$ is defined by

$$\begin{aligned}\mathcal{U}(q_{\|,1}, x_\kappa) &= \left\{u_\kappa \in \mathbb{R}^2 \mid \pi_{\mathcal{P}'}\left(A_\kappa x_\kappa + B_\kappa u_\kappa\right) \in \left\{q_{\|,3}\right\}\right\},\\ &= \left\{u_\kappa \in \mathbb{R}^2 \mid -\frac{1}{2} < u_{\kappa,1} < \frac{1}{2} \wedge -\frac{1}{2} < u_{\kappa,2} < \frac{1}{2}\right\}.\end{aligned} \tag{8.62}$$

The other input sets are computed in a similar way.

8.8 Software Implementation of Linear Hybrid System

The software implementation of linear hybrid system H, defined in Eq. (8.58) is illustrated in Figure 8.12.

For a detailed description of the software implementation of linear hybrid system H the reader is referred to section 7.8 in the the previous chapter.

Now, since the system in Brunovsky normal form Σ_κ is related to linear control system Σ by an invertible state/input transformation matrix U, defined in Eq. (7.14), Σ_κ can be replaced with Σ as shown in Figure 8.13.

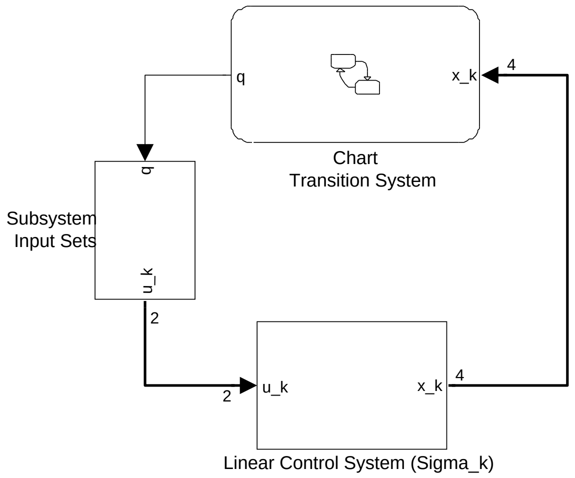


Figure 8.12: *Software implementation of linear hybrid system H in* SIMULINK *and* STATEFLOW.

Notice, that replacing Σ_κ with Σ requires the generation of a new input $u \in \mathbb{R}^2$ from the original input $u_\kappa \in \mathbb{R}^2$ and state $x_\kappa \in \mathbb{R}^4$ of Σ_κ (See Eq. (7.14)).

Further, Σ can be replaced with linear control system Σ_z which is a continuous-time linear control system. This is shown in Figure 8.14.

8.9 Simulation Results

The initial position of the unicycle is $p(0) = \begin{bmatrix}1.625 & 2.05\end{bmatrix}^{\mathrm{T}}$ (m) and the initial orientation of the unicycle is $\theta(0) = \pi$ (rad). Thus, the initial coordinates of the unicycle is $q(0) = \begin{bmatrix}1.625 & 2.05 & \pi\end{bmatrix}^{\mathrm{T}}$. The initial state of the integrator is $\xi(0) = 0.001$ (m/s).

The result of simulating the system is shown in Figure 8.15.

As shown in Figure 8.15 the vehicle start at position $eta = \begin{bmatrix}1.625 & 2.05\end{bmatrix}$ and moves towards the set representing the goal position as described in Figure 8.5.

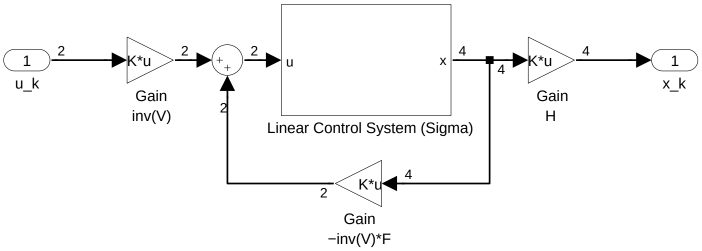


Figure 8.13: *Software implementation of hybrid linear system H where Σ_κ has been replaced with Σ.*

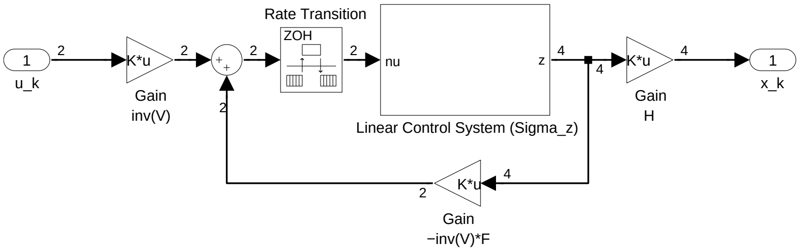


Figure 8.14: *Software implementation of hybrid linear system H where Σ has been replaced with Σ_z.*

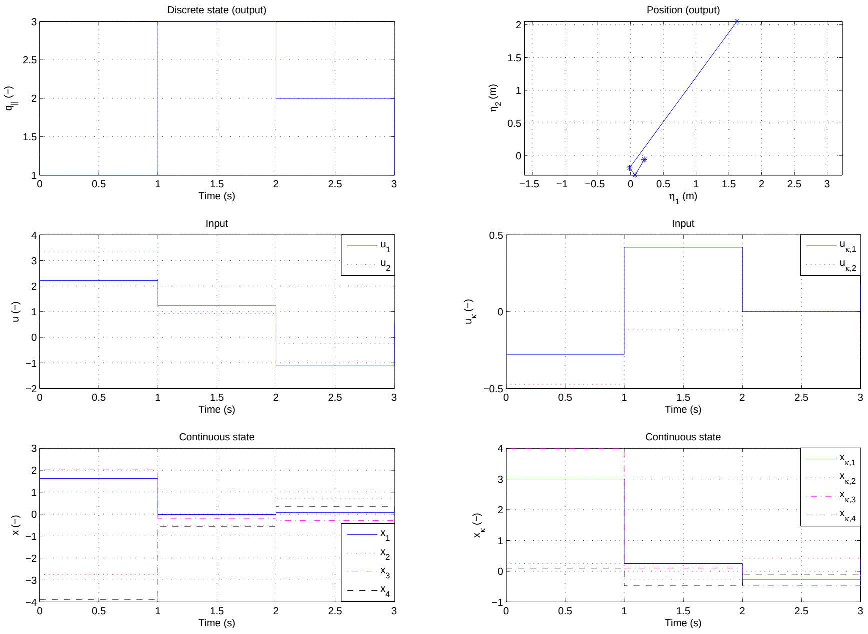


Figure 8.15: *Simulation results.*

8.10 Concluding Discussion

The novel framework presented here is applicable to a large class of nonholonomic mobile robots, including the unicycle, that can be transformed into linear systems using e.g. dynamic feedback linearization and dynamic extension.

The framework originally developed by Tabuada and Pappas [Tabuada and Pappas, 2006] was extended to fit with the problem of synthesizing a controller for a mobile robot, given a requirement specification in LTL. This required some modifications which were primarily concerned with the input to the framework and with the software implementation of the linear hybrid system obtained by refinement.

A software implementation of the hybrid linear system was not provided in [Tabuada and Pappas, 2006] for the purpose of simulating the synthesized controller and the linear control system. Therefore, to demonstrate the correctness of the synthesized controller a software implementation of the linear hybrid system was provided in SIMULINK and STATEFLOW, showing that the synthesized controller equivalently would work on the nonlinear system.

8.11 Summary

In this chapter the framework for controller synthesis presented in the previous chapter was applied to a simple model of a nonholonomic wheeled mobile robot, i.e. an unicycle. The nonlinear unicycle model was transformed into a linear control system by using dynamic feedback linearization and dynamic extension. The requirement specification expressing the desired behavior of the unicycle was formulated in Linear-time Temporal Logic (LTL). The bisimulation algorithm was used for obtaining an abstraction in the form of a finite bisimilar quotient of the linear control system. A controller was then synthesized from the Büchi automaton and the finite abstraction. Subsequently, the closed loop system of discrete nature is refined resulting in a linear hybrid system. A software implementation of linear hybrid system H in SIMULINK and STATEFLOW was provided and a simulation was performed. The results of simulating the software implementation of the linear hybrid system are presented showing that the system operate as expected. Finally, a conclusive discussion of the novel framework proposed i the previous chapter was given.

Chapter 9

Conclusions and Recommendations

The overall aim of this thesis has been to investigate automated synthesis of coordination and control of multi-robot systems. The thesis introduces two different approaches to synthesis that are potentially suitable for generation of engineering systems.

In the following we conclude on the two frameworks presented in chapter 5 and 7, respectively, that is

Framework I Motion planning of a network of multi-modal robots with respect to formal requirement specifications in Computational Tree Logic (CTL).

Framework II Controller synthesis for linear control systems with respect to formal requirement specification in Linear-time Temporal Logic (LTL).

Framework I : Multi-robot Motion Planning

- By abstracting a network of multi-modal robots, modeled as hybrid automata, to the world of timed automata symbolic reasoning about coordinated motion planning solutions was possible with respect to a formal requirement specification in Computational Tree Logic (CTL). Composition of multi-modal robots in the network modeled as timed automata was possible through the introduction of synchronization channels thus allowing a multi-modal robot and associated controller to communicate. Further, global optimal solutions was found thus minimizing the number of robot movements or accumulated time required for the network of multi-modal robots to reach their goal positions, in order to satisfy the requirement specification in CTL.

- The framework presupposes an infra-structure of the multi-modal robots with feedback controllers that constraint the motion capabilities of the individual multi-modal robots in the network. This means that the individual robots can have more complex dynamics compared to what was expressed by the hybrid automaton modeling a multi-modal robot.

- The use of Computational Tree Logic (CTL) as a requirement specification mechanism for the network of multi-modal robots allow to express requirements that cannot be specified using classical control theory. Also, CTL allows to express the desired global behavior for the network of multi-modal robots.

- The proposed framework is also applicable to multi-modal robots moving in a three-dimensional (3D) environment. To allow this, the hybrid automata, modeling a multi-modal robot, must be expanded with two additional states for moving in the x_3 and $-x_3$-direction, respectively. However, this would result in an increased computational complexity of subsequently model checking the system.

- A drawback of the proposed framework is the computational complexity of model checking the system in UPPAAL. It is a known fact that the computational complexity is exponential in the number of robots, controllers, and size of the occupancy table representing the environment.

- The framework is applicable to other application domains than robotics that is systems where planning and motion coordination is necessary, i.e. as in formation flying of satellites and aerial vehicles.

Framework II : Robot Controller Synthesis

- The framework originally developed by Tabuada and Pappas [Tabuada and Pappas, 2006] was extended to fit with the problem of synthesizing a controller for a mobile robot, given a requirement specification in LTL.

This required some modifications which were primarily concerned with the input to the framework and with the software implementation of the linear hybrid system obtained by refinement.

- A software implementation of the hybrid linear system was not provided in [Tabuada and Pappas, 2006] for the purpose of simulating the synthesized controller and the linear control system. Therefore, to demonstrate the correctness of the synthesized controller a software implementation of the linear hybrid system was provided in SIMULINK and STATEFLOW, showing that the synthesized controller equivalently would work on the nonlinear system.

- The framework is applicable to a large class of nonholonomic mobile robots, including the unicycle, that can be transformed into linear control systems using e.g. dynamic feedback linearization and dynamic extension.

9.1 Recommendations

However, two frameworks have been outlined in this thesis but the following open issues are recommendable for further investigation. The issues are prioritized as follows, with decreasing priority

Framework I : Multi-robot Motion Planning

1. The computational complexity of model checking the system in UPPAAL can be overcome by guiding the search such that an extensive search is avoided. UPPAAL CORA[1] [Uppsala University and Aalborg University, 1995b] uses an extension of timed automata called Linearly Priced Timed Automata (LPTA) which allows to annotate the timed automaton model with the notion of cost. The idea is then be to have a low cost for moving in a direction that would bring each multi-modal robot closer to its goal position, opposed to a high cost if moving in a direction that brings the robot away from the goal position. Subsequently,

[1] A branch of UPPAAL for cost optimal reachability analysis.

UPPAAL CORA could be used to find the most optimal location trajectory, i.e. a location trajectory with the lowest accumulated cost for each robot.

2. As a result of model checking the system UPPAAL generates a strategy in the form of a sequence of input synchronization actions, one for each multi-modal robot in the network, in order to satisfy a requirement specification in CTL. The sequence of input synchronization actions corresponds to a set of way-points connecting the initial and goal position. A natural next step would be the development of a controller, possibly of hybrid nature to track the set of way-points.

Framework II : Robot Controller Synthesis

1. The novel framework proposed is a step towards automated controller synthesis for mobile robots, given a nonlinear model of the system and a requirement specification in Linear-time Temporal Logic (LTL). A natural next step is the development of a tool that automatically synthesize a controller for a mobile robot, given a nonlinear model and a requirement specification in Linear-time Temporal Logic (LTL).

2. Expressing the desired behavior of the system in Linear Temporal Logic (LTL) can be complex and possibly render the framework less applicable. To make the framework more applicable to engineers unfamiliar with requirement specification in Linear-time Temporal Logic (LTL) a more "natural language" for requirement specification than LTL should be investigated. Alternatively, the possibility to translate a more "natural language" to a requirement specification in LTL should be investigated.

3. It would be desirable to expand the framework to networks of robots. If possible, this could eventually require some major modifications of the framework. The composition of robots could be performed at the level of modeling the system. Thus, a linear control system can be formed that is composed of n linear control system, i.e.

$$\Sigma = \begin{bmatrix} \Sigma_1 & & \\ & \ddots & \\ & & \Sigma_n \end{bmatrix}$$

such that system matrices are given by

$$A = \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_n \end{bmatrix}, \quad B = \begin{bmatrix} B_1 & & \\ & \ddots & \\ & & B_n \end{bmatrix}.$$

Unifying Framework I+II

The two frameworks could potentially be combined to an unifying framework for motion planning and control of networks of robots. Framework I would then be used for motion planning for the network of robots, where a feasible motion plan for the robots is generated, taking into account coordination among the robots. Subsequently, framework II would then be used for synthesizing a hybrid controller for each of the robots in the network, satisfying the motion plan generate using framework I.

Bibliography

R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Multi-robot cooperation in the MARTHA project. *IEEE Robotics & Automation Magazine*, 5: 36–47, March 1998.

R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 94:183–235, 1994.

R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, pages 3–34, 1995.

R. Alur, J. Esposito, M. Kim, V. Kumar, and I. Lee. Formal modeling and analysis of hybrid systems: A case study in multi-robot coordination. In J. Wing, J. Woodcock, and J. Davies, editors, *FM'99, LNCS 1708*, volume I, pages 212–232. Springer-Verlag, Berlin Heidelberg, 1999.

R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Procedings of the IEEE*, 88(7):971–984, July 2000.

M. S. Andersen and R. S. Jensen. Motion planning in multi-robot systems - with application to a harvesting system. *Master's Thesis. Aalborg University, Department of Control Engineering*, July 2004.

M. S. Andersen, R. S. Jensen, T. Bak, and M. M. Quottrup. Motion planning in multi-robot systems using timed automata. *Proceedings of 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles (IAV 2004), Lissabon*, July 2004.

M. Antoniotti and B. Mishra. Discrete event models + temporal logic = supervisory controller: Automatic synthesis of locomotion controllers. *IEEE Conference on Robotics & Automation*, pages 1441–1446, 1995.

P. J. Antsaklis and A. N. Michel. *Linear systems*. The McGraw-Hill Companies, Inc., New York, 2 edition, 1997.

T. Arai and J. Ota. Motion planning of multiple robots. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1761–1768, 1992.

T. Arai, E. Pagello, and L. E. Parker. Guest editorial advances in multirobot systems. *IEEE Transactions on Robotics & Automation*, 18(5):655–661, October 2002.

J. R. Büchi. On a decision method in restricted second order arithmetic. *Procedings International Congres Logic, Method and Philoophical Science, Stanford, CA*, pages 1–12, 1962.

C. Belta, V. Isler, and G. J. Pappas. Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Transactions on Robotics*, 21(5):864–874, October 2005.

A. Bouajjani, J. C. Fernandez, and N. Halbwachs. Minimal model generation. In R. C. Arkin and G. A. Bekey, editors, *CAV90 : Computer Aided Verification*, volume 531, pages 197–203. Springer Verlag, 1990.

P. Brunovsky. A classification of linear controllable systems. *Kybernetika*, 6(3): 173–188, 1970.

Cambridge University Press. Cambridge Dictionaries Online - Cambridge Advanced Learner's Dictionary. http://dictionary.cambridge.org, 2006.

Y. U. Cao, A. S. Fukunaga, and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. In R. C. Arkin and G. A. Bekey, editors, *Autonomous Robotics*, volume 4, pages 1–23. Kluwer Academic Publishers, Boston, 1997.

G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robotics*, 3:375–397, July 1996.

M. Egerstedt and X. Hu. A hybrid control approach to action coordination for mobile robots. *Automatica*, 38:125–130, January 2002.

G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas. Hybrid controllers for path planning : A temporal logic approach. *IEEE Conference on Decision & Control, Seville, Spain*, December 2005a.

G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for mobile robots. *Proceedings of the 2005 IEEE International Conference on Robotics & Automation, Barcelona, Spain*, pages 2032–2037, April 2005b.

A. Farinelli, L. Iocchi, and D. Nardi. Multi-robot systems: A classification focused on coordination. *IEEE Transactions on Systems, Man, & Cybernetics, Part B*, 34(5):2015–2028, 2004.

K. Fujimura. *Motion planning in dynamic environments*. Computer Science Workbench, Springer-Verlag, 1991.

B. P. Gerkey and M. J. Mataric. Sold: Auction methods for multirobot coordination. *IEEE Transaction on Robotics & Automation*, 18(5):758–768, October 2002.

P. Ögren, M. Egerstedt, and X. Hu. A control lyapunov function approach to multiagent coordination. *IEEE Transactions on Robotics & Automation*, 18 (5):847–851, October 2002.

Y. K. Hwang and N. Ahuja. Gross motion planning - a survey. *AMC Computing Surveys*, 24(3):221–291, September 1992.

L. Iocchi, D. Nardi, and M. Salerno. Reactivity and deliberation: A survey on multi-robot systems. In M. Hannebauer, J. Wendler, and E. Pagello, editors, *Balancing Reactivity and Social Deliberation in Multi-Agent Systems (LNAI 2103)*. Springer-Verlag, 2001.

E. Klavins and D. E. Koditschek. A formalism for the composition of concurrent robot behaviors. *Procedings of the 2000 IEEE International Conference on Robotics & Automation (ICRA'00), San Franscisco, CA*, 4:3395–3402, April 2000.

M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from LTL specifications. *9th International Workshop on Hybrid Systems: Computation & Control, Santa Barbara, CA (To Appear)*, 2006a.

M. Kloetzer and C. Belta. LTL planning for groups of robots. *IEEE International Conference on Networking, Sensing, and Control, Ft. Lauderdale, FL (To Appear)*, 2006b.

T. J. Koo. Hierarchical system architecture for multi-agent multi-modal systems. *Procedings of the 40th IEEE Conference on Decision & Control*, pages 1509–1514, December 2001.

T. J. Koo and S. Sastry. Bisimulation based hierarchical system architecture for single-agent multi-modal system. In C. J. Tomlin and M. R. Greenstreet, editors, *HSCC 2002, LNCS 2289*, pages 281–293. Springer-Verlag, Berlin Heidelberg, 2002.

T. J. Koo, M. M. Quottrup, and C. A. Clifton. Hybrid system design of multi-modal aerial robots. *Hybrid Systems: Computation & Control (HSCC), Santa Barbara, CA (Submitted)*, March 2006.

K. G. Larsen, P. Pettersson, and W. Yi. Model-checking for real-time systems. *Procedings of the 10th International Conference on Fundamentals of Computation Theory*, pages 62–88, August 1995.

J.-C. Latombe. *Robot motion planning*. Kluwer Academic Publishers, Massachusetts, 1991.

J.-P. Laumond. *Robot motion planning and control*. Springer-Verlag, London, 1991.

S. G. Loizou and K. J. Kyriakopoulos. Automatic synthesis of multi-agent motion tasks based on LTL specifications. *43rd IEEE Conference on Decision and Control*, December 2004.

Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems*. Springer-Verlag, New York, 1991.

R. Milner. *Communication and concurrency*. Prentice Hall, 1989.

G. Oriolo, A. De Luca, and M. Vendittelli. WMR control via dynamic feedback linearization: Design, implementation, and experimental validation. *IEEE Transactions on Control Systems Technology*, 10(6):835–852, November 2002.

Oxford University Press. Oxford English Dictionary - OED online. http://oed.com, 2005.

G. J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035–2047, December 2003.

L. E. Parker. Current state of the art in distributed autonomous mobile robottics. In G. Bekey L. E. Parker and J. Barhen, editors, *Distributed Autonomous Robotic System 4*, pages 3–12. Springer-Verlag, Tokyo, 2000.

M. M. Quottrup, T. Bak, and R. Izadi-Zamanabadi. Multi-robot planning: A timed automata approach. *Procedings of the 2004 IEEE on Robotics & Automation, New Orleans, LA*, pages 4417–4422, April 2004.

E. D. Sontag. *Mathematical control theory : Deterministic finite dimensional systems*. Springer, New York, 2 edition, 1998.

P. Tabuada and G. J. Pappas. Linear time logic control of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 51(12):1862–1877, December 2006.

Sweden Uppsala University and Denmark Aalborg University. UPPAAL. *http://www.uppaal.com/*, 1995a.

Sweden Uppsala University and Denmark Aalborg University. UPPAAL CORA. *http://www.cs.aau.dk/ behrmann/cora/index.html*, 1995b.

P. Wolper. Constructing automata from temporal logic formulas: A tutorial. In E. Brinksma, H. Hermanns, and J. P. Katoen, editors, *Lectures on Formal Methods and Performance Analysis, LNCS*, volume 2090. Springer-Verlag, 2000.

Appendix A

System Parameters

Global Declarations for System

```
// Declarations of global clocks, variables, constants and channels.

// global clocks
 clock time;   // measures global time for system

// global constants
 const int Z_1 = 7;   // horizontal size of Z
 const int Z_2 = 7;   // vertical size of Z
 const int N = 3;     // number of robots + 1
 const int M = 33;    // number of obstacles + 1

// global variables
 int[0,1] Z[Z_1][Z_2];   // two-dimensional integer array Z
 int[0,1] i,j;           // index variables
 int[1,N] robotNo;
 int[1,M] obsNo;

// global channels
 chan sigma_21,sigma_31,sigma_41,sigma_51;   // synchronization channels
 chan sigma_22,sigma_32,sigma_42,sigma_52;
```

Local Declarations for Robot

```
// Declarations of local clocks, variables and constants.

// local clocks
 clock c;   // measures local time for robot
// local variables
 int[0,Z_1] z_1;   // discrete horizontal position of robot in grid
 int[0,Z_2] z_2;   // discrete vertical position of robot in grid
```

Local Declaration for Obstacles

```
// Declarations of local clocks, variables and constants.

// local variables
 int[0,Z_1] z_1;   // discrete static horizontal position of obstacle in
                   grid
 int[0,Z_2] z_2;   // discrete static vertical position of obstacle in
                   grid
```

System Declarations

```
// Declarations of process assignments and system definition.

// Process assignments

// robots
 R_1 = Robot(1,5,3,1,sigma_21,sigma_31,sigma_41,
   sigma_51);
 R_2 = Robot(2,5,4,1,sigma_22,sigma_32,sigma_42,
   sigma_52);
// controls
 C_1 = Control(sigma_21,sigma_31,sigma_41,sigma_51);
 C_2 = Control(sigma_22,sigma_32,sigma_42,sigma_52);
// obstacles
```

```
 O_1 = Obstacle(1,0,0);
 O_2 = Obstacle(2,1,0);
 O_3 = Obstacle(3,2,0);
 O_4 = Obstacle(4,3,0);
 O_5 = Obstacle(5,4,0);
 O_6 = Obstacle(6,5,0);
 O_7 = Obstacle(7,6,0);
 O_8 = Obstacle(8,6,1);
 O_9 = Obstacle(9,6,2);
 O_10 = Obstacle(10,6,3);
 O_11 = Obstacle(11,6,4);
 O_12 = Obstacle(12,6,5);
 O_13 = Obstacle(13,6,6);
 O_14 = Obstacle(14,5,6);
 O_15 = Obstacle(15,4,6);
 O_16 = Obstacle(16,3,6);
 O_17 = Obstacle(17,2,6);
 O_18 = Obstacle(18,1,6);
 O_19 = Obstacle(19,0,6);
 O_20 = Obstacle(20,0,5);
 O_21 = Obstacle(21,0,4);
 O_22 = Obstacle(22,0,3);
 O_23 = Obstacle(23,0,2);
 O_24 = Obstacle(24,0,1);
 O_25 = Obstacle(25,2,1);
 O_26 = Obstacle(26,3,2);
 O_27 = Obstacle(27,4,2);
 O_28 = Obstacle(28,4,3);
 O_29 = Obstacle(29,3,3);
 O_30 = Obstacle(30,2,4);
 O_31 = Obstacle(31,4,5);
 O_32 = Obstacle(32,5,5);
// System definition
 system R_1, R_2, C_1, C_2, O_1, O_2, O_3, O_4, O_5,
 O_6, O_7, O_8, O_9, O_10, O_11, O_12, O_13, O_14,
 O_15, O_16, O_17, O_18, O_19, O_20, O_21, O_22,
 O_23, O_24, O_25, O_26, O_27, O_28, O_29, O_30,
 O_31, O_32;
```

Appendix B

Special Forms of Linear Control Systems

In this appendix we review the Controller form and Brunovsky normal form which are two special forms of both continuous-time and discrete-time linear control systems. However, the two special forms will be described for discrete-time linear control systems.

Consider a discrete-time linear control system

$$\Sigma \ : \quad x(t+1) = Ax(t) + Bu(t), \tag{B.1}$$

where matrices $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are generally constant and state variable $x \in \mathbb{R}^n$ and control variable $u \in \mathbb{R}^m$ are discrete. In the following x is referred to as the state of the system and u as the input to the system. Further, $\mathbb{R}^n$ is referred to as the state space or set of states of the system and $\mathbb{R}^m$ as the observation space or set of observations.

The computation of the Brunovsky normal form for the system (7.12) comprises two steps

- First, the linear system Σ is transformed into the controller form, denoted Σ_c with a linear change of coordinates,

- Second, the controller form Σ_c is further reduced into the Brunovsky normal form, denoted Σ' with a linear state feedback.

In the following ... The controllability matrix $\mathcal{C} \in \mathbb{R}^{n \times mn}$ for the system (B.1) is given by

$$\begin{aligned}\mathcal{C} &= \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} && \text{(B.2)}\\ &= \big[b_1 \ \dots \ b_m \ Ab_1 \ \dots \ Ab_m \ A^2b_1 \ \dots \ A^2b_m \ \dots \\ &\quad A^{n-1}b_1 \ \dots \ A^{n-1}b_m\big]. && \text{(B.3)}\end{aligned}$$

Let Σ be a linear control system as defined in Eq. B.1. The sequence of positive integers $\kappa = (\kappa_1, \kappa_2, \dots, \kappa_m)$, where $\operatorname{rank} B = m$ are called the controllability indices of the system satisfying

$$\kappa_1 \geq \kappa_2 \geq \dots \geq \kappa_m \text{ such that } \kappa_1 + \kappa_2 + \dots + \kappa_m = n. \qquad \text{(B.4)}$$

B.1 Controller Form

The decomposition of state space $\mathbb{R}^n$ for linear control system Σ according to controllability indices $\kappa = (\kappa_1, \kappa_2, \dots, \kappa_m)$ results in a new controllability matrix $\bar{\mathcal{C}} \in \mathbb{R}^{n \times n}$ given as

$$\begin{aligned}\bar{\mathcal{C}} = \Big[& b_1 \ Ab_1 \ A^2b_1 \ \dots \ A^{\kappa_1-1}b_1 \ b_2 \ Ab_2 \ A^2b_2 \ \dots \ A^{\kappa_2-1}b_2 \ \dots \\ & b_m \ Ab_m \ A^2b_m \ \dots \ A^{\kappa_m-1}b_m\Big], \qquad \text{(B.5)}\end{aligned}$$

where the vectors are arranged as follows. Select, starting from the left and moving to the right, the first n independent columns of Eq. (B.3). Recorder these columns by taking first b_1, Ab_1, A^2b_1, etc., until all columns involving b_1 have been taken; then take b_2, Ab_2, A^2b_2, etc., and lastly, take b_m, Ab_m, A^2b_m, etc., to obtain Eq. (7.17). The integer κ_i denotes the number of columns involving b_i in the set of the first n linearly independent columns found in the controllability matrix $\mathcal{C}$, when moving from left to right.

The m integers κ_i for $i = 1, \dots, m$ is called the controllability indices of the system.

Now define

$$\sigma_i = \sum_{i=1}^{m} \kappa_i, \qquad \text{(B.6)}$$

such that

$$\sigma_1 = \kappa_1, \tag{B.7}$$
$$\sigma_2 = \kappa_1 + \kappa_2, \tag{B.8}$$
$$\vdots$$
$$\sigma_m = \kappa_1 + \ldots + \kappa_m. \tag{B.9}$$

Consider $\bar{\mathcal{C}}^{-1} \in \mathbb{R}^{n \times n}$ and let d_i, where $d_i^{\mathrm{T}} \in \mathbb{R}^n$ for $i = 1, \ldots, m$ denote the σ_i-th row of $\bar{\mathcal{C}}^{-1}$. The invertible linear transformation $H \in \mathbb{R}^{n \times n}$ is a *similarity transformation* defined as

$$H = \begin{bmatrix} d_1 \\ d_1 A \\ \vdots \\ d_1 A^{\kappa_1 - 1} \\ \vdots \\ d_m \\ d_m A \\ \vdots \\ d_m A^{\kappa_m - 1} \end{bmatrix}. \tag{B.10}$$

The controller form (single input case) of a linear control system Σ is given by

$$\Sigma_c \ : \quad x(t+1) = A_c x(t) + B_c u(t), \tag{B.11}$$

where A_c and B_c are the new system matrices, defined by

$$A_c = HAH^{-1}, \tag{B.12}$$
$$B_c = HB. \tag{B.13}$$

The system matrices are of the following form

$$A_c = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ -\alpha_1 & -\alpha_2 & \cdots & -\alpha_{n-1} \end{bmatrix}, \quad B_c = \begin{bmatrix} B_i \\ B_2 \\ \vdots \\ B_m \end{bmatrix}, \tag{B.14}$$

where each block $B_i \in \mathbb{R}^{\kappa_i \times m}$ are of the form

$$B_i = \begin{bmatrix} 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & & \vdots & \vdots & & & \vdots \\ 0 & \cdots & 0 & 1 & \times & \cdots & \times \end{bmatrix}, \tag{B.15}$$

where 1 in the last row of B_i occurs at the i-th column location for $i = 1, \ldots, m$ and $\times$ denotes non-fixed entries. In (B.14) the α_i for $i = 0, \ldots, n-1$ are the coefficients of the characteristic polynomial $\alpha(s)$ of matrix A, i.e.

$$\alpha(s) = \det(sI - A) = s^n + \alpha_{n-1}s^{n-1} + \cdots + \alpha_1 s + \alpha_0. \tag{B.16}$$

B.2 Brunovsky Normal Form

At this point we will assume that the matrices A_c and B_c are in controller form. It is possible to write A_c and B_c in a systematic way

$$A_c = A' + B'A_m, \qquad B_c = B'B_m, \tag{B.17}$$

where $(A', B') \in \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times m}$ are the Brunovsky normal form of (A, B) in (B.1). The matrices $A_m \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times m}$ consists of the σ_1-th, σ_2-th, …, σ_m-th rows of A_c and B_c, respectively. The matrices A_m and B_m is that part of controllable linear control system (B.1) that can be altered by linear state feedback

$$u' = Fx + Vu. \tag{B.18}$$

Notice that

$$A_{cF} = A_c + B_cF_c, \tag{B.19}$$

is also in controller form with identical block structure as A_c for any F_c. Thus, the pair (A_{cF}, B_c) has the same controllability indices $\kappa = (\kappa_1, \ldots, \kappa_m)$ as the pair (A_c, B_c).

Selecting

$$A_d = A' + B'A_{d_m}, \tag{B.20}$$

and requiring that $A_{cF} = A_d$, implies

$$B'(A_m + B_mF_c) = B'A_{d_m}. \tag{B.21}$$

Thus, we have that

$$F_c = B_m^{-1}(A_{d_m} - A_m), \tag{B.22}$$

where B_m, A_{d_m}, and A_m are the m σ_i-th rows of B_c, A_d, and A_c, respectively and σ_i are defined in (B.6). For $A_{d_m} = 0$ which is the case for A in Brunovsky normal form, (B.25) reduces to

$$F_c = B_m^{-1} A_m. \tag{B.23}$$

The matrix F_c is related to F by

$$F = F_c H, \tag{B.24}$$

which leads to

$$F = B_m^{-1} A_m H. \tag{B.25}$$

The invertible linear transformation $V \in \mathbb{R}^{m \times m}$ is given by

$$V = B_m^{-1}. \tag{B.26}$$

The Brunovsky normal form of linear control system is given by

$$\Sigma' \ : \quad x'(t+1) = A'x'(t) + B'u'(t), \tag{B.27}$$

where A' and B' are the new system matrices and x' and u' are the new state and control variables, defined as

$$A' = H(A + BF)H^{-1}, \tag{B.28}$$
$$B' = HBV, \tag{B.29}$$
$$x' = Hx, \tag{B.30}$$
$$u' = Fx + Vu. \tag{B.31}$$

The system matrices A' and B' are of the following form

$$A' = \begin{bmatrix} A'_{\kappa_1} & 0 & \dots & 0 \\ 0 & A'_{\kappa_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A'_{\kappa_m} \end{bmatrix}, \quad B' = \begin{bmatrix} b'_{\kappa_1} & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & b'_{\kappa_2} & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & b'_{\kappa_m} \end{bmatrix}, \tag{B.32}$$

where each block $A'_{\kappa_i} \in \mathbb{R}^{\kappa_i \times \kappa_i}$ and $b'_{\kappa_i} \in \mathbb{R}^{\kappa_i}$ for $i = 1, \dots, m$ are of the form

$$A'_{\kappa_i} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \qquad B'_{\kappa_i} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{B.33}$$

The system in Brunovsky normal form Σ' with state $x' \in \mathbb{R}^n$ and input $u' \in \mathbb{R}^m$ is related to the linear control system Σ by an invertible state/input transformation matrix (*isomorphism*) $U : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$ defined as

$$\begin{bmatrix} x' \\ u' \end{bmatrix} = U \begin{bmatrix} x \\ u \end{bmatrix} = \begin{bmatrix} H & O_{n \times m} \\ F & V \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix}. \tag{B.34}$$

Index