Efficient Synthesis of Safety Controllers using Symbolic Models and Lazy Algorithms

Elena Ivanova

université PARIS-SACLAY

Efficient Synthesis of Safety Controllers using Symbolic Models and Lazy Algorithms

Synthèse efficace des contrôleurs de sécurité à l'aide de modèles symboliques et d'algorithmes paresseux

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication

Spécialité de doctorat: Automatique

Graduate School : Informatique et science du numérique, Référent : Faculté des sciences d'Orsay

Thèse préparée dans la unité de recherche Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France, sous la direction de Antoine Girard, Directeur de recherche, CNRS

Thèse soutenue à Paris-Saclay, le 15 Novembre 2021, par

Elena Ivanova

Composition du jury

Cristina Maniu Professeure, CentraleSupélec	Présidente
Maria Prandini Professeure, Politecnico di Milano	Rapportrice
Luc Jaulin Professeur, ENSTA-Bretagne	Rapporteur
Anne-Kathrin Schmuck Directrice de groupe de recherche indépendant, Max Planck Institute for Software Systems	Examinatrice
Ali Zolghadri Professeur, Université de Bordeaux	Examinateur
Antoine Girard Directeur de recherche, CNRS	Directeur de thèse

THESE DE DOCTORAT

NNT : 2021UPASG088

To my grandfather

Abstract

This thesis focuses on the development of efficient abstraction-based controller synthesis approaches for cyber-physical systems (CPS). While abstraction-based methods for CPS design have been the subject of intensive research over the last decades, the scalability of these techniques remains an issue. This thesis focuses on developing lazy synthesis algorithms for safety specifications. Safety specifications consist in maintaining the trajectory of the system inside a given safe set. This specification is of the utmost importance in many engineering problems, often prioritized over other performance requirements. Lazy approaches outperform the classical synthesis algorithm [Tabuada, 2009] by avoiding computations, which are non-essential for synthesis goals. Chapter 1 motivates the thesis and discusses the state of the art. Chapter 2 structures the existing lazy synthesis approaches and emphasizes three sources of efficiency: information about a priori controllable states, priorities on inputs, and non-reachable from initial set states. Chapter 3 proposes an algorithm, which iteratively explores states on the boundary of the controllable domain while avoiding exploration of internal states, supposing that they are safely controllable a priory. A closed-loop safety controller for the original problem is then defined as follows: we use the abstract controller to push the system from a boundary state back towards the interior, while for inner states, any admissible input is valid. Chapter 4 presents an algorithm that restricts the controller synthesis computations to reachable states only while prioritizing longer-duration transitions. The original system is abstracted by a symbolic model with an adaptive grid. Moreover, a novel type of time sampling is also considered. Instead of using transitions of predetermined duration, the duration of the transitions is constrained by state intervals that must contain the reachable set. Chapter 5 is dedicated to monotone transition systems. The introduced lazy synthesis approach benefits from a monotone property of transition systems and the ordered structure of the state (input) space, and the fact that directed safety specifications are considered. The considered class of specifications is then enriched by intersections of upper and lower-closed safety requirements. Chapter 6 concludes the discussion and raises new issues for future research.

Acknowledgements

First and foremost, let me express my sincere gratitude to my Ph.D. advisor Professor Antoine Girard for his guidance and support along the way. Antoine, thank you for your trust in me and your precious advice, help, and motivation. You were always available for a discussion, and I learned so much from you. I could not have imagined having a better advisor. I mean it!

Besides, I would like to extend my special gratitude to my thesis committee members: Professor Luc Jaulin, Professor Maria Prandini, Professor Anne-Kathrin Schmuck, Professor Ali Zolghadri, and Professor Cristina Maniu. First, let me thank Professor Luc Jaulin for being part of my midterm evaluation committee and for the valuable comments he provided on this thesis. Hopefully, we will find a way to collaborate in the future! Then, I would like to thank Professor Maria Prandini for reviewing my thesis report in detail and asking deep questions during the presentation. Professor Anne-Kathrin Schmuck for the inspiration of my research on multi-scaled abstractions and a fruitful discussion after the talk. Professor Cristina Maniu for being the president of the Jury and for the support of my ideas. And Professor Ali Zolghadri for his attention to a bigger picture in addition to technical details.

I also would like to thank Professor Jana Tumova for an excellent opportunity to visit the KTH Royal Institute of Technology in August 2021. I found it amazing how the RPL group is balancing theory and applications, which gave me an understanding of what I want for my future career. Let me thank Pouria Tajvar for initiating the collaboration on safe motion planners. Christian Pek for sharing his experience of managing several projects at the same time. Also, Wei, Grace, Truls, Taras, and Alex made my visit to Stockholm unforgettable. Let me also thank Professor Dimos Dimarogonas for the possibility of giving a talk on his group's seminar.

During my stay in Paris, I have met many great people who made me feel like I belonged. Let me thank Professor Sami Tliba for giving me the opportunity to teach over the last two years. Professor Elena Panteley, Professor Antonio Loria, and Professor Luca Greco for their personal and professional support. I also want to thank Adnane Saoud for our joint work on monotone dynamical systems and my labmates: Vladimir, Kuba, Adel, Anas, Alejandro, Ralph, Vincent, Amina, and Zohra, for the inspiring discussions and the fun we have had in the last three years. I am also thankful to Felipe, Loic, Kemal, Polina, Olga, Natasha, Sergei, Oleg, Ivan, and Murad for the days and nights we spent together climbing, traveling, and discovering France. I also would like to thank my neighbors Elisa, Belatricea, Jorge, Michaelo, and Henrique for helping me to pass through 2020.

Let me also express my special thanks to my professors and friends from Lomonosov Moscow State University. First of all, I would like to thank Professor Pavel Tochilin for supervising my master thesis, believing in me, and inspiring me to get a Ph.D. degree. I am also thankful to my former classmates Nikita, Klava, Arthur, Lilia, Ruslan, and Sasha for being the best friends ever.

Last but not least, I would like to thank my family. My parents, Alexandra and Mikhail, my sister Natalia, both my aunts and my cousin Fedor. They have given me so much love and support that one lifetime would not be enough to pay it back. I also want to remember and thank my grandfather Gennadiy since he was a person I looked up to.

Aperçu de la thèse

Contrôleurs de sécurité pour systèmes cyber-physiques

La théorie du contrôle est un sous-domaine mathématique, dont l'objectif est de développer des approches efficaces pour contrôler des systemes afin qu'ils se comportent de la manière désirée. Ce manuscrit est dédié à la synthèse efficace de contrôleurs de sécurité pour les systèmes cyber-physiques (CPS) à l'aide d'approches basées sur l'abstraction.

Le CPS est un système où des éléments informatiques collaborent pour le contrôle et la commande d'entités physiques et la sécurité est souvent une propriété essentielle du CPS. Intuitivement, la sécurité exige que les «mauvaises» choses ne se produisent pas et elle est de la plus haute importance dans de nombreux problèmes d'ingénierie, souvent prioritaires par rapport à d'autres exigences de performance. Les pompes à insuline doivent protéger une personne diabétique de l'hyper ou de l'hypoglycémie [Gillis et al., 2007, Hovorka et al., 2004, Kushner et al., 2019]. Les assistants du régulateur de vitesse adaptatif doivent garder une voiture à une distance de sécurité du véhicule précédent [Alam et al., 2014, Ames et al., 2017, Darbha, 1997]. Le contrôle du climatiseur doit maintenir la température dans un bâtiment intelligent dans la plage souhaitable [Meyer et al., 2013, Thavlov and Bindner, 2015]. Dans toutes ces applications pratiques, le comportement du système contrôlé doit satisfaire aux exigences de sécurité. De plus, une spécification de sécurité apparaît souvent comme une sous-tâche d'un problème plus complexe [Chen et al., 2015], comprenant des objectifs de contrôle donnés par un automate hybride [Nilsson et al., 2016] ou une formule de logique temporelle linéaire [Tajvar et al., 2020].

Contrôle symbolique

La solution étudiée dans cette thèse pour résoudre ce problème de contrôle est basée sur des méthodes symboliques. Le principe de ces méthodes est de créer une abstraction purement discrète du système original que l'on représentera sous la forme d'un système de transitions fini et non-déterministe pour lequel un contrôleur est plus facile à synthétiser grâce aux méthodes du domaine du contrôle discret. Si une relation comportementale (simulation, bisimulation, leurs versions alternées et approchées ou une relation de raffinement de rétroac-

tion [Tabuada, 2009, Reissig et al., 2017]) entre l'abstraction et le modèle original peut être prouvée, cela signifie que tous comportements du système original peuvent être reproduits dans l'abstraction. La relation de simulation alternée implique également qu'un contrôleur discret synthétisé sur l'abstraction peut être transformé en un contrôleur du modèle original satisfaisant les mêmes spécifications. Nous parlons ainsi de contrôle hybride puisqu'un contrôleur discret est appliqué à un système continu (ou hybride). Il faut noter que ce nom ne veut pas forcément dire que cette approche ne s'utilise que pour les systèmes hybrides : elle peut être intéressante pour tous systèmes dont les dynamiques sont trop complexes pour être contrôlées avec les méthodes classiques. Le nom de méthodes symboliques s'explique par la première étape de la création de l'abstraction discrète, consistant en une partition de l'espace d'état : chaque élément de cette partition peut être vu comme un symbole représentant tous les états continus qu'il contient. Les transitions de l'abstraction symbolique sont ensuite obtenues à l'aide d'une analyse d'atteignabilité pour laquelle on prend une approximation de l'ensemble des états continus qui peuvent être atteints (avec une version échantillonnée du système de départ) à partir de ceux contenus dans un symbole.

Motivation et contributions principales

Le principal inconvénient des approches de synthèse basées sur l'abstraction est la faible évolutivité. En effet, les modèles symboliques sont couramment obtenus par discrétisation d'espaces d'état et d'entrée. Des abstractions plus précises nécessitent des paramètres d'échantillonnage plus précis et donnent lieu à des modèles symboliques avec de nombreux états et entrées. Par conséquent, la construction d'abstractions précises requises pour une synthèse réussie est souvent difficilement calculable. De plus, la complexité des algorithmes de synthèse de contrôleurs discrets dépend typiquement de la taille des modèles symboliques. Enfin, les contrôleurs de sécurité basés sur des modèles symboliques plus grands nécessitent plus de mémoire pour leur implémentation en temps réel. Un besoib de réductions de la complexité de calcul des approches de synthèse basées sur l'abstraction a motivé un nombre considérable de recherches au cours de la dernière décennie et a motivé la rédaction de cette thèse. Pour surmonter cette limitation, nous proposons d'utiliser

- des algorithmes de synthèse paresseux pour les systèmes de transitions;
- des modèles symboliques efficaces en taille.

L'idée principale des approches paresseuses pour surpasser l'algorithme de synthèse classique [Tabuada, 2009] est de restreindre les calculs à l'essentiel pour la partie synthèse [Girard et al., 2016, Hussien and Tabuada, 2018, Hsu et al., 2019, Nilson et al., 2017, Rungger and Stursberg, 2012]. Dans de telles approches, les abstractions sont calculées à la volée, parallèlement à la procédure de synthèse, et la partie inexplorée du modèle symbolique reste non calculée. Pour minimiser la taille des modèles symboliques on peut bénéficier de la stabilité incrémentale [Girard et al., 2016, Pola et al., 2010, Saoud and Girard, 2018] ou de la monotonie [Kim et al., 2017,

Sinyakov and Girard, 2020a] du système d'origine. Des abstractions efficaces ont également été proposées pour les systèmes différentiellement plats [Liu et al., 2012] et pour les systèmes présentant un structure d'interconnexion [Gruber et al., 2017]. En tant que solution pour la classe générale des systèmes, les abstractions multi-échelles sont souvent considérées [Gol et al., 2013, Girard et al., 2016, Hsu et al., 2019, Nilson et al., 2017]. Nous nous concentrerons sur eux dans le chapitre 4.

Le chapitre 2 *structure les approches de synthèse paresseuses existantes* et met l'accent sur trois sources d'efficacité: les informations sur les états contrôlables a priori [Girard et al., 2016, Koenig and Likhachev, 2005], les priorités sur les entrées [Hussien and Tabuada, 2018, Girard et al., 2016, Hsu et al., 2019, Nilson et al., 2017], et les états non accessibles depuis l'ensemble initial [Girard et al., 2016]. Le chapitre 3 présente une nouvelle technique d'échantillonnage temporel adaptatif appliquant la structure souhaitée (liée au voisin) du modèle symbolique. Un algorithme de synthèse paresseux pour les abstractions liées aux voisins est ensuite fourni, avec une extension pour les abstractions avec un échantillonnage temporel arbitraire. Dans l'esprit, l'idée est proche du résultat du théorème de Nagumo ([Blanchini, 1999]) et du principe de visée extrême ([Subbotin, 1995]). Cependant, à notre connaissance, ces idées n'ont jamais été mises en œuvre dans le cadre de la synthèse de contrôle basée sur l'abstraction. Le chapitre 4 *adapte* les idées d'exploration de modèle symbolique incrémental et de synthèse paresseuse présentées dans [Girard et al., 2016] pour les systèmes de transitions déterministes *aux abstractions non déterministes*. Un *modèle symbolique multi-échelle avec une grille adaptative est également introduit*. Le chapitre 5 présente deux classes de *systèmes de transitions monotones* et fournit des conditions suffisantes pour qu'une abstraction d'un système dynamique monotone [Angeli and Sontag, 2003] soit monotone. Ensuite, des algorithmes de synthèse paresseux pour des modèles symboliques monotones et des spécifications de sécurité dirigées (et leurs intersections) sont présentés. Le résultat est basé sur des abstractions efficaces qui ont été introduites dans [Kim et al., 2017].

Contents

1 Introduction **11**

1.1 Design of Safety Critical Cyber-Physical Systems . . . 12

1.1.1 Formal Problem Statement . . . 12

1.2 Literature Overview . . . 13

1.2.1 On Computation of Maximal Control-Invariant Sets . . . 13

1.2.2 Abstraction-Based Control Synthesis (ABCS) Approaches . . . 14

1.2.3 Over-approximations of Reachable Sets . . . 15

1.3 Illustrative Example: ABCS for Safety Specification . . . 16

1.4 Motivation and Main Contributions . . . 18

1.5 Thesis Outline . . . 19

1.5.1 Chapter 2. Safety Controller Synthesis for Finite Transition System . . . 19

1.5.2 Chapter 3. Lazy Controller Synthesis Based on Safe Set Boundary Exploration . . . 20

1.5.3 Chapter 4. Lazy Synthesis with Adaptive Symbolic Abstractions . . . 21

1.5.4 Chapter 5. Lazy Synthesis for Monotone Systems and Directed Specifications . . . 22

1.5.5 Chapter 6. Conclusion and Future Work . . . 23

1.6 Publications . . . 24

2 Safety Controller Synthesis for Finite Transition System: How to Speed-Up the Computations? **25**

2.1 Preliminaries . . . 26

2.2 Maximal Safety Controller . . . 26

2.3 Lazy Synthesis Algorithms . . . 29

2.3.1 Source of Laziness: Information about A priory Controllable States . . . 31

2.3.2 Source of Laziness: Inputs' Priorities . . . 33

2.3.3 Source of Laziness: Non Reachable from Initial Set States . . . 35

2.4 Conclusion . . . 37

3 Lazy Symbolic Controller for Continuous-Time Systems Based on Safe Set Boundary Exploration 39

3.1 Neighbor-Linked Abstractions . . . 40

3.1.1 Adaptive Time Sampling as Way to Enforce the Desired Structure . . . 40

3.1.2 Maximal Safety Controller Synthesis: Inner States as a Source of Laziness . . . 42

3.2 What if the Abstraction is not Neighbor-Linked? . . . 43

3.3 Controller Refinement for an Arbitrary Abstraction . . . 44

3.4 Numerical Illustration: Adaptive Cruise Control . . . 46

3.5 Conclusion . . . 49

4 Lazy Safety Controller Synthesis with Multiscale Adaptive-Sampling Abstractions 51

4.1 Multilayered Abstractions with Multi-scale Adaptive-Time Sampling . . . 52

4.2 Synthesis of Maximal Input-State Lazy Safety Controller . . . 55

4.2.1 From a Multilayered to an Adaptive Grid . . . 58

4.3 Controller Refinement for an Abstraction with Adaptive Grid . . . 60

4.4 Numerical Illustration: Temperature Regulation in the Building . . . 61

4.5 Conclusion . . . 63

5 Efficient Controller Synthesis for Monotone Dynamical Systems and Directed Safety Specifications 65

5.1 Monotone Dynamical Systems and Directed Safety Specifications . . . 66

5.2 Monotone Abstractions for Monotone Dynamical Systems . . . 69

5.2.1 Monotone Transition Systems . . . 69

5.2.2 Box abstraction . . . 71

5.2.3 Sparse Abstractions for (Input-)State Monotone Control Systems . . . 73

5.3 Maximal Safety Controller for Monotone Transition Systems and Directed Specifications . . . 75

5.3.1 Lower-closed and Upper-closed Sets . . . 76

5.3.2 Lazy Synthesis for State Monotone Transition Systems . . . 77

5.3.3 Lazy Synthesis for Input-State Monotone Transition Systems . . . 82

5.3.4 Controller Synthesis for Intersections of Directed Safety Specifications . . . 85

5.4 Numerical Illustration: Adaptive Cruise Control . . . 87

5.4.1 Control Objective and Numerical Results . . . 87

5.5 Conclusion . . . 91

6 Conclusion and Future Work 93

6.1 Future Research . . . 95

A Reachability Analysis for Mixed-monotone Dynamic Systems 97

List of Figures

1.1 Illustration of an abstraction-based control synthesis approach for a safety specification. 17

1.2 The left figure illustrates adaptive time-sampling techniques: while transition duration is commonly determined by a given time sampling parameter τ, we propose interrupting every transition stops just before leaving $N_A(q)$. The right figure represents the result of the synthesis procedure for an arbitrary abstraction, while the central figure shows the corresponding domain of the continuous-time controller. In the green region, we apply any admissible input, but as soon as closed-loop trajectory reaches the white area, we switch to the "abstract" controller, which steers us back to the controllable domain. 21

1.3 The left figure shows transitions with two different durations: for a control u_1 we stop before leaving a radius 1, while for a control u_2 – an interval with a radius 2. We then say that action corresponding to u_1 is less valuable than the action corresponding to u_2 since we prefer first to go further. The central figure shows a 2-layered grid: a coarser level is marked with a normal line and a finer level with a dashed line. The input up brings us from a state q' to a state q such that $q' \subset q$. The transition corresponding to input $(u, 2)$, where 2 determines the duration ends at the finest layer. The right figure then represents the same transition but with the corresponding adaptive grid. 21

1.4 The left figures show two trajectories of an input-state monotone dynamical system. The right figure provides an example of a lower-closed safety set. In all figures, a natural component-wise order on $\mathbb{R}^2$ is considered for both the input and the state spaces. 22

1.5 The left figure illustrates that if the transition system is lower input state monotone, then we for any two states $q_1 \preceq_Q q_2$ and two inputs $u_1 \preceq_U u_2$, $F(q_1, u_1) \subseteq\downarrow F(q_2, u_2)$. Right figure shows that a finite lower-closed set Q (i.e. Q s.t. $Q =\downarrow Q$) can be represented by its basis $\mathrm{Bas}(Q) = \{q_1, q_2, q_3, q_4\}$, since $Q =\downarrow \mathrm{Bas}(Q)$. 23

2.1 Illustration of Algorithm 2.1. Algorithm 2.1 iteratively removes all unsafe actions from a transition system until the system stops changing. An action is unsafe if it can not prevent the system from steering into an unsafe or blocking state. Unsafe actions are marked with a red color. Blocking states are filled with grey, the unsafe state - with black. A state q with $p(q) = 1$ is contoured with black, with $p(q) = 0$ - with red. 29

2.2 Illustration of Algorithm 2.2 execution.Blocking states are filled with grey. The unsafe state is black. A state q such that $p(q) = 1$ are contoured with black, with $p(q) = 0$ - with red. Let us remark that all transitions in the system are between neighboring states. Hence, we can restrict exploration to those states that border with blocking or unsafe states. The unexplored elements are marked with the dashed line, while the normal line denoted states already involved in computations. At the end of execution, states q_8, q_4 stay unconsidered, but we can automatically mark them as controllable, including all actions enabled at these states to the maximal safety controller. 32

2.3 Illustration of Algorithm 2.3 execution. Unsafe actions are red. Blocking states are grey. Controls u_1, u_2 have priority 2, u_3, u_4 have priority 1. If a state is contoured with black, it has priority 2, and the actions with priority 2 are enabled for this state. If with blue, it has priority 1, and the inputs with priority 1 are available. If with red, then its priority is 0. We enable lower priority transitions for a state only if it is uncontrollable with higher priority actions. Enable at the current iteration transition marked with a normal line, while action marked with a dashed line are non-available. The actions which remain untouched at step 9 are not included in the maximal input lazy safety controller. 35

2.4 Illustration Algorithm 2.4 execution. Unsafe actions are red, blocking states are grey, and the initial states are marked with a double line. The computations are restricted to states, which reachable from the initial set. States which are non-reachable from the initial set are countered with a dashed line. Hence, states q_{11}, q_{12} and q_8 are not included in the domain of the maximal state lazy safety controller. 37

3.1 Illustration of the adaptive time sampling technique (right figure) compared to a fixed time sampling approach (left figure). Black cells represent unsafe states; green and blue cells represent $F(q, u_1)$ and $F(q, u_2)$ correspondingly. The red square represents the abstract state neighborhood $N_A(q)$. While in the left figure, transition duration is determined by pre-fixed time sampling parameter τ_{fix}, in the right figure, a transition stops just before leaving $N_A(q)$ (see eq. (3.2)). 41

3.2 Illustration lazy synthesis based on boundary exploration idea. First figure illustrates the discretization of the original state space $\mathbb{R}^2$. The safe set Y is countered with blue; white cells are safe states; the unsafe state is filled with gray. Figures 2-5 illustrate the execution of Algorithm 3.2. States filled with white and green have priority 1; with grey - priority 0. White states belong to Q_{EB}. States contoured with red are blocking. The inputs set $U = \{u_1, u_2\}$. Actions corresponding to control u_1 are black, u_2 are blue. Transitions marked with red are unsafe. In opposite to normal line, the dashed line represents transitions that haven't been computed yet. Last figure illustrates a piece of trajectory of a closed-loop system (1.1),(3.5)-(3.8). White states belong to Q_B; green area corresponds to Q_I. Normal line represents mode 0, dashed line - mode 1. 45

3.3 Illustration an adaptive cruise control problem configuration with a constant time headway spacing policy. 46

3.4 Simulation results. Left figure illustrates a safety specification. Right figure represents the domain of the safety controller computed by Algorithm 3.2. 47

3.5 Simulation results. Left figures represent a disturbance realization F_1 and a safety control F_2. Right figures represent a corresponding closed-loop trajectory of the dynamic system (3.9), starting at $[20, 15, 5]$ 49

4.1 Illustration of transition relation on a 2-layered grid. Left figure illustrates (4.2) with a transition $F(q', up) = q$, and (4.3) with a transition $F(q, u)$. Unsafe states are marked with black. Right figure illustrates adaptive time-sampling techniques (see (4.4)). Transitions with two different duration are shown: for a control u_1 we stop before leaving an interval with a radius 1, while for a control u_2 – an interval with a radius 2. 55

4.2 Illustration of Algorithm 4.2 execution. Unsafe actions are red. Blocking states are grey. Controls u_1, u_2 have priority 2, u_3, u_4 have priority 1. If a state is contoured with black, it has priority 2, and the actions with priority 2 are enabled for this state. If with blue, it has priority 1, and the inputs with priority 1 are available. If with red, then its priority is 0. We enable lower priority transitions for a state only if it is uncontrollable with higher priority actions. Enable at the current iteration transition marked with a normal line, while action marked with a dashed line are non-available. Reachable from the initial set states are countered with a normal line, non-reachable with a dashed line. The actions marked with a dashed line in step 9 are not included in the maximal input lazy safety controller. . . . 57

4.3 Illustration of the difference between multilayered and adaptive grid. Left figure shows the successors $F(q, u)$ on 2-layered grid, while the right figure on the corresponding adaptive grid. 60

4.4 Illustration of the considered configuration for temperature regulation problem. 61

4.5 Simulation results. A grey area corresponds to the controllable domain: light states are controllable by the actions with priority 1; dark states - with priority 2. The white area is uncontrollable. Concerning the closed-loop trajectory, orange color corresponds to a control $\{0,0\}$, green - to a control $\{0,1\}$, violet - to a control $\{1,0\}$. 62

5.1 Illustration of Definitions 5.1.3 and 5.1.5. The left figures show two trajectories of an input-state monotone dynamical system. The right figure provides an example of a lower-closed safety set. In all figures, a natural component-wise order on $\mathbb{R}^2$ is considered for both the input and the state spaces. . 67

5.2 Illustration of Theorem 5.2.1. Given two states $q_1 \preceq_Q q_2$ and two inputs $u_1 \preceq_U u_2$, if the transition system is LISM then we have that $F(q_1,u_1) \subseteq\downarrow F(q_2,u_2)$. 70

5.3 The first two partitions satisfy Assumption 5.2.1, while the third partition does not satisfy Assumption 5.2.1. The first and third partitions satisfy Assumption 5.2.2, while the second partition does not satisfy Assumption 5.2.2. The state-space is equipped with the component-wise partial order $\preceq$ defined on $\mathbb{R}^2$. 72

5.4 Illustration of difference between box and sparse abstractions. Left: A box abstraction for a fixed disturbance interval $[\mathbf{w_1^m}, \mathbf{w_2^m}]$, $m \in \{1,\dots,M\}$. Right: box (grey) and sparse (blue) abstractions for a monotone system with a lower-closed safety specification for $M=3$. 73

5.5 Illustration of Definition 5.3.1. A lower-closed set A and its basis $\mathrm{Bas}(A) = \{a_1,a_2,a_3,a_4\}$. The state-space is equipped with the component-wise partial order $\preceq$ defined on $\mathbb{R}^2$ and we have: $A =\downarrow A =\downarrow \mathrm{Bas}(A)$. 77

5.6 Illustration of Algorithm 5.3. If $p(q)=1$, then q either blue or white. Red states are uncontrollable (i.e. $p(q)=0$). Blue states are the basis of a set Q_E. The input set $U=\{u_1,u_2\}$. 79

5.7 Illustration of lazy synthesis of the maximal safety controller for a LSM transition system with lower-closed specification. Red state are uncontrollable. White states belong to $\mathrm{Dom}(\bar{C})$. The left figure represents the $\mathrm{Dom}(\bar{C})$ and its basis $\{q_3,q_4,q_5\}$. Figures in the middle illustrate computation of the sets $Pre(\mathrm{Dom}(\bar{C},u_1,\mathrm{Dom}(\bar{C})))$ and $Pre(\mathrm{Dom}(\bar{C},u_2,\mathrm{Dom}(\bar{C})))$ by Algorithm 5.4. States filled with grey belong to Q_U. States contoured with colors belong to $Bas(\mathrm{Dom}(C)\setminus Q_U)$. States contoured with blue and yellow are controllable by inputs u_1 and u_2 correspondingly (i.e. belong to Q_C). The right figure illustrates the result of Theorem 5.3.2. States filled with yellow are controllable by u_2, with blue - by u_1, with green - by both u_1 and u_2. 81

5.8 Illustration of the reordering $U=\{u_1,\dots,u_9\}$ of the input set $U=\{1,2,3\}^2$ with respect to the component-wise partial order $\preceq$ defined on $\mathbb{R}^2$. 83

5.9 Illustration of the maximal safety controller $\bar{C}$ for the case of a total order on the input set. $U=\{u_1,u_2,u_3\}$ with $u_1 \preceq_U u_2 \preceq_U u_3$. 85

5.10 Simulation results. Maximal safety controller $\bar{C}$ for a lower-closed safety specification. 88
5.11 Simulation results. Maximal safety controller $\bar{C}$ for an intersection of lower-closed and upper-closed safety specification. 91

6.1 Left figures illustrates construction of a non-deterministic abstraction. Right figures illustrates a deterministic one. 96

List of Tables

3.1 Vehicle and safety parameters. The values are taken from [Darbha, 1997, Saoud et al., 2020]. 48

4.1 Runtime comparison when varying the number of states. 63

5.1 Runtime comparison when varying the number of states. T^s_{lm}, T_{cl}, T^s_{cl} and T^s_{3v} are the running time of Algorithm 5.5, the classical fixed point algorithm with box abstractions [Tabuada, 2009] and with sparse abstractions [Kim et al., 2017] and the lazy algorithm from [Hussien and Tabuada, 2018], respectively. 89

5.2 Memory comparison when varying the number of states. M^s_{lm}, M_{cl}, M^s_{cl} and M^s_{3v} are the required memory to implement the controller resulting from Algorithm 5.5, the classical fixed point algorithm with box abstractions [Tabuada, 2009], the classical fixed point algorithm with sparse abstractions [Kim et al., 2017] and the lazy algorithm from [Hussien and Tabuada, 2018], respectively. 89

5.3 Runtime comparison when varying the number of inputs. T^s_{lm}, T_{cl}, T^s_{cl} and T^s_{3v} are the running time of Algorithm 5.5, the classical fixed point algorithm with box abstractions [Tabuada, 2009] and with sparse abstractions [Kim et al., 2017] and the lazy algorithm from [Hussien and Tabuada, 2018], respectively. 90

5.4 Memory comparison when varying the number of states. M^s_{lm}, M_{cl}, M^s_{cl} and M^s_{3v} are the required memory to implement the controller resulting from Algorithm 5.5, the classical fixed point algorithm with box abstractions [Tabuada, 2009], the classical fixed point algorithm with sparse abstractions [Kim et al., 2017] and the lazy algorithm from [Hussien and Tabuada, 2018], respectively. 90

List of Symbols and Abbreviates

$\mathbb{N}, \mathbb{Z}, \mathbb{R}$	nonnegative integers, integers and real numbers correspondingly
$\mathbb{N}^p, \mathbb{R}^p$, $\mathbb{Z}^p$	p- dimensional vector spaces originated by $\mathbb{N}, \mathbb{Z}, \mathbb{R}$ correspondingly
$\mathcal{L}(\mathcal{T}, \mathcal{S})$	is a space of measurable function on $\mathcal{T}$ taking their values from $\mathcal{S}$
$\mathbf{s} \in \mathcal{L}(\mathcal{T}, \mathcal{S})$	is a function belonging to $\mathcal{L}(\mathcal{T}, \mathcal{S})$
$\prec_{\mathcal{S}}$	a partial order on $\mathcal{S}$
$cl(X)$	is a closure of a set X
$\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$	a control system with a dynamic given by $\dot{x} = f(x, u, w), u \in \mathcal{U}, w \in \mathcal{W}$
$x_f(t \mid x(0), \mathbf{u}, \mathbf{w})$	a trajectory of Σ_f corresponding to an initial condition $x(0)$, a control $\mathbf{u} \in \mathcal{L}(\mathcal{T}, \mathcal{U})$, and a disturbance $\mathbf{w} \in \mathcal{L}(\mathcal{T}, \mathcal{W})$
$\mathrm{Reach}(\tau \mid q, u)$	a robust to admissible disturbance reachable set, corresponding to an initial set q and a constant control function $\mathbf{u} : [0, t] \to u$ and
$\overline{\mathrm{Reach}}(\tau \mid q, u)$	an over-approximation of $\mathrm{Reach}(\tau \mid q, u)$

Transition system

$\Sigma = (Q, U, F)$	transition system
Q, U	a set of states and inputs correspondingly
F	a transition relation
$F(q, u)$	a set of successors corresponding to a state q and an input u
$Pre_F(q, u)$	set of predecessors corresponding to a state q and an input u w.r.t. F
$\mathrm{Reach}_F(A)$	set of reachable states from a set A w.r.t. F
$\mathrm{Enab}_F(q)$	enable inputs in a state q w.r.t. F
$\mathrm{Block}_F(B)$	set of blocking states in a set B w.r.t. F
$N_A(q)$	is an abstract neighborhood of a state q
Q_S, Q_{init}	is a set of safe and initial states correspondingly
C	is a controller
$\mathrm{Dom}(C)$	is a domain of a controller C

Abbreviates

TS	transition system.
ABSC	abstraction-based control synthesis.
MSC	maximal safety controller.
MILSC	maximal input-lazy safety controller.
MSLSC	maximal state-lazy safety controller.
MLSC	maximal lazy safety controller
SM	state-monotone (dynamical system)
ISM	input-state monotone (dynamical system)
LSM	lower state-monotone (transition system)
USM	upper state-monotone (transition system)
LISM	Lower input-state monotone (transition system)
UISM	Upper input-state monotone (transition system)

Chapter 1

Introduction

Cyber-physical systems (CPS) are physical and engineered systems whose operations are monitored, coordinated, controlled and integrated by a computing and communication core [Rajkumar et al., 2010]. It is clear, that they are to become ubiquitous in modern societies: autonomous vehicles, smart buildings, robots...many grand challenges await in the economically vital domains of transportation, health care, manufacturing, agriculture, energy, defense, aerospace, and buildings. The design, construction and verification of CPS pose a multitude of technical issues.


It requires a wide range of expertise from control theory to embedded software engineering to cope with the tight interactions between the computational and physical elements of a CPS. This renders the development of a CPS challenging and time-consuming. Moreover, since CPS are often safety-critical, a significant amount of effort must be invested in system integration and verification to ensure the soundness of the proposed design.

A sound design of CPS, taking into account the interactions between the computational and physical elements, requires model-based approaches. CPS models are heterogeneous: the continuous behavior is described by differential equations (in general non-linear), while the discrete behavior is formalized with finite-state automata frameworks. As pointed out in [Derler et al., 2012, Kim and Kumar, 2012], being able to deal with heterogeneity is a prerequisite to the foundation of a sound framework for CPS design. During the past decade, significant progress towards that goal has been made, notably in the area of hybrid dynamical systems. Hybrid systems are dynamical systems exhibiting both continuous and discrete behaviors. Motivated by the multiplication of “discrete” embedded computing devices interacting with the “continuous” physical world, the research on hybrid systems has

rapidly developed at the interface of computer science and control since the nineties. Each discipline has brought its models and methods, and their combination has allowed the scientific community to build the foundations of a theory of hybrid systems. The notion of hybrid automaton [Henzinger, 2000, Lygeros et al., 2003], which is one of the most commonly used mathematical models of hybrid systems, combines differential equations and finite-state automata and is a typical example of this cross-fertilization. More generally, hybrid systems research has allowed new topics at the intersection of computation and control.

Another challenging issue in developing CPS is a complex control objective that goes beyond the traditional control theory (e.g. stability, controllability, observability...). For example, one often has to address safety, reachability, fault-tolerance, or even specifications given by some logic formula or automaton describing the acceptable temporal behaviours of the system [Belta et al., 2017, Cassandras and Lafortune, 2009, Sinyakov and Girard, 2020b].

1.1 Design of Safety Critical Cyber-Physical Systems

As was mentioned above, CPS are safety-critical. Intuitively, safety requires that "bad" things do not happen and it is of the utmost importance in many engineering problems, often prioritized over other performance requirements. Insulin pumps should protect a diabetic person from hyper or hypoglycemia [Gillis et al., 2007, Hovorka et al., 2004, Kushner et al., 2019]. Adaptive cruise control assistants should keep a car at a safe distance from the previous vehicle [Alam et al., 2014, Ames et al., 2017, Darbha, 1997]. Climate control should maintain the temperature in an intelligent building into the desirable range [Meyer et al., 2013, Thavlov and Bindner, 2015]. Satellite station keeping [Weiss et al., 2018], traffic networks [Kim et al., 2017] and biochemical networks [Sontag, 2007] control, power grids design [Kader et al., 2019]... in all these practical applications the behavior of the controlled system should satisfy the safety requirements. Moreover, a safety specification often appears as a sub-task of a more complex problem [Chen et al., 2015], including control objectives given by hybrid automaton [Nilsson et al., 2016] or linear temporal logic formula [Tajvar et al., 2020]. Being a critical issue in designing CPS, safety specification is the main topic of this manuscript. However, before moving forward let us formalize the considered problem.

1.1.1 Formal Problem Statement

A control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ consists of a time domain $\mathcal{T} = [0, +\infty)$, a state space $\mathbb{R}^{n_x}$, a compact set $\mathcal{U} \subset \mathbb{R}^{n_u}$, a compact set $\mathcal{W} \subset \mathbb{R}^{n_w}$, and a non-linear function $f\colon \mathbb{R}^{n_x} \times U \times W \to \mathbb{R}^{n_x}$, such that for any control $\mathbf{u} \in \mathcal{L}(\mathcal{T}, \mathcal{U})$, any disturbance $\mathbf{w} \in \mathcal{L}(\mathcal{T}, \mathcal{W})$ and any initial condition $x(0) \in \mathbb{R}^{n_x}$ in the whole domain $\mathbb{R}^{n_x}$ there exists a unique solution $x_f(t \mid x(0), \mathbf{u}, \mathbf{w})$, $t \in \mathcal{T}$ of the following differential equation

$$\dot{x}(t) = f(x(t), u(t), w(t)) \tag{1.1}$$

in the sense of Caratheodory. The notation $\mathcal{L}(\mathcal{T}, \mathcal{S})$ is used for the space of functions $\mathbf{s}$, measurable on $\mathcal{T}$, such that $s(t) \in \mathcal{S},\ t \in \mathcal{T}$ almost everywhere. Let us also use a bold font to denote a function $\mathbf{s}\ \mathbf{in}\mathcal{L}(\mathcal{T}, \mathcal{S})$.

We then call a solution $x_f(t \mid x(0), \mathbf{u}, \mathbf{w}),\ t \in \mathcal{T}$ a trajectory of the system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ corresponding to an initial condition $x(0)$, a control function $\mathbf{u}$, and a disturbance $\mathbf{w}$. When the control inputs of system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ are generated by a state-feedback controller $u(t, x) \colon \mathcal{T} \times \mathbb{R}^{n_x} \to \mathcal{U}$, the dynamics of the closed-loop system, corresponding to a disturbance realization $\mathbf{w}$, is given by $\dot{x}(t) = f(x(t), u(t, x(t)), w(t)), t \in \mathcal{T}$, and the trajectory of closed-loop system is denoted as $x_f^u(t \mid x(0), \mathbf{w})$. The class of admissible closed-loop controls $u(t, x(t))$ must, however, be restricted so that the closed-loop system has a unique solution (in the sense of Caratheodory) in the considered domain.

Problem 1.1.1. *Given a control system $\Sigma = (\mathcal{T}, \mathbb{R}^{n_x}, U, W, f)$ find an admissible safety controller, which keeps all trajectories of the closed-loop system inside a safety set $Y \subset \mathbb{R}^{n_x}$. Here the controller is said to be admissible if it is robust against any measurable bounded disturbance $\mathbf{w} \in \mathcal{L}(\mathcal{T}, W)$ and there exists a unique solution of the closed-loop system.*

1.2 Literature Overview

1.2.1 On Computation of Maximal Control-Invariant Sets

A fundamental concept related to safety specification is (robust) control invariant sets. By definition, a subset of the state space is a control invariant set (or a viable set [Aubin and Frankowska, 1991]) if, for all initial conditions chosen among its elements, we can keep the trajectory inside the set by means of proper control action. If disturbances, in addition, are considered, then the set is known as a robust control invariant set. So, to solve a safety control problem means to find a (robust) control invariant set included in a given safe set and there is a practical interest in getting the invariant set, which is maximal by inclusion. The existence and uniqueness of such a set was, for example, studied within the viability theory [Aubin and Frankowska, 1991]. Though the problem of finding maximal control-invariant sets for arbitrary systems is intractable, there is an extensive literature on computing numerical approximations. Polytopic projection [Blanchini, 1999, Anevlavix and Tabuada, 2019]), LMI-based Lyapunov type analysis techniques [Khlebnikov et al., 2011], sum of squares programming [Papachristodoulou and Prajna, 2005, Tan and Packard, 2008], Minkowski type methods [Kolmanovsky and Gilbert, 1998] and linear programming approaches [Trodden, 2016] allow us to compute convex approximations of maximal invariant sets for linear system. For nonlinear systems the problem has been addressed with approaches based on value-function approximations [Mitchell et al., 2005, Kurzhanski and Varaiya, 2014], a control barrier functions design [Ames et al., 2017], and interval analysis techniques [Alam et al., 2014, Jaulin et al., 2001, Li and Liu, 2018a, Meyer et al., 2013]. However, the heterogeneous dynamic of CPS and demand to incorporate the safety into more difficult control specifications

have turned the attention of the considerable part of the research community to abstraction-based synthesis approaches [Belta et al., 2017, Tabuada, 2009].

1.2.2 Abstraction-Based Control Synthesis (ABCS) Approaches

In recent years, controller synthesis techniques based on abstractions have received considerable attention within the control systems community. Continuous-time abstractions serve for two reasons: simplify the dynamic of the original system [Fu et al., 2013, Dang et al., 2010] or reduce the number of dimensions [Girard and Pappas, 2007].

However, this work is dedicated to so-called symbolic model control when one creates a finite-state discrete-time abstraction on top of the original system. The latter allows to leverage computer-science control synthesis techniques and enrich the class of available specifications [Cormen et al., 2001]. We call the abstraction a symbolic model since one abstract state is a symbol representing infinitely many states of the original state space. Symbolic model control is a popular branch of the modern control theory. The dynamics of considered system varies from simple double integrators [Fainekos et al., 2009], over linear [Wongpiromsarn et al., 2009, Rungger et al., 2013] and piecewise affine [Yordanov et al., 2012] to nonlinear systems [Tabuada, 2009, Reissig et al., 2017, Girard, 2010, Liu et al., 2013, Zamani et al., 2011]. The control problems range from reach-avoid [Tabuada, 2009, Girard, 2010, Reissig et al., 2017, Zamani et al., 2011] and safety specifications [Hsu et al., 2019, Tabuada, 2009, Girard, 2010, Saoud et al., 2020], fragments of LTL [Fainekos et al., 2009, Wongpiromsarn et al., 2009, Liu et al., 2013] to full LTL [Tabuada and Pappas, 2006, Kloetzer and Belta, 2008].

Roughly speaking, control synthesis based on symbolic models is a three-step procedure. In the first step, the continuous or hybrid system (together with the specification) is lifted to an abstract domain where it is substituted by a finite transition system [Belta et al., 2017], which mimics the original dynamic. In the second step, an auxiliary problem on the abstract domain ("abstract problem") is solved using discrete methods [Cassandras and Lafortune, 2009, Cormen et al., 2001]. In the third step, the controller synthesized for the abstraction is refined to the concrete system [Belta et al., 2017, Tabuada, 2009].

Behavioural Relationships

The correctness of abstraction-based approaches is usually ensured by connecting the original system with its abstraction in terms of a system relation. The notion of simulation [Milner, 1989, Park, 1981] guarantees that an execution is possible for the abstraction if it is possible for the concrete system. The concept of bisimulation states that an execution is possible for the abstraction if and only if it is possible for the concrete system [Pappas, 2003]. Since the construction of bisimilar abstractions was possible for restricted classes of systems, the notion of simulation was relaxed by Girard and Pappas in [Girard and Pappas, 2005]. The construction of approximately bisimilar deterministic abstractions was made possible for incrementally stable systems [Angeli, 2002]. An extension called alternating

approximate bisimilarity has been proposed in [Pola et al., 2010], allowing for the construction of non-deterministic abstractions. While in the approximate alternating (bi-)simulation relations, the refined controller needs to contain the abstraction as a building block. The notion of feedback-refinement relation [Reissig et al., 2017], which in spirit are the same as over-approximation relations [Liu et al., 2013], was proposed to address this shortcoming. Finally, a first approach to construct a behavioural relationship taking into account the structural properties of dynamical systems has been recently proposed in [Kim et al., 2017], where the notion of directed alternating simulation relation was shown to be efficient in order to deal with monotone dynamical systems [Angeli and Sontag, 2003].

Toolboxes

Various tools built on abstraction-based methods are available for the purpose of control synthesis. Tools such as LTLMoP [Finucane et al., 2010], TuLip [Wongpiromsarn et al., 2011] are restricted to simple integrator dynamics and piecewise affine control systems, respectively, while Pessoa [Mazo et al., 2010], CoSyMa [Mouelhi et al., 2013], ROCS [Li and Liu, 2018b], SCOTS [Rungger and Zamani, 2016], and Co4Pro [Sinyakov and Girard, 2021] consider nonlinear systems. The differences between Pessoa, CoSyMa and ROCS, SCOTS, Co4Pro become apparent in terms of the type of symbolic model which is used to solve the synthesis problem. CoSyMa requires the original system to be incrementally stable [Angeli, 2002] and computes symbolic models that are approximately bisimilar to the original system [Girard, 2010]. Pessoa, additionally to approximately bisimilar symbolic models, supports the computation of approximately alternatingly similar symbolic models [Tabuada, 2009]. However, Pessoa can not handle nonlinear systems unless the user provides their own overapproximation function. Accepting general non-linear systems, symbolic models implemented by SCOTS are based on feedback refinement relations [Reissig et al., 2017]. The kernel of ROCS is interval branch-and-bound scheme, with limitation that customized inclusion functor has to be provided by the user. Comparing to SCOTS and ROCS the internal rechability algorithm are improved in Co4Pro to achive the better performance of abstraction-based approaches. Concerning the specifications SCOTS, Pessoa, CoSyMa, ROCS natively supports the controller design to enforce invariance and reachability. LTLMoP and TuLip support more general specifications like GR(1) [Bloem et al., 2012]. The most resent toolbox Co4Pro supports specifications given by a discrete-time hybrid automaton.

1.2.3 Over-approximations of Reachable Sets

As previously stated, the exact reachable set of linear continuous systems can only be computed in special cases [Lafferriere et al., 2001]. Thus, the reachable set has to be computed in an overapproximative way in order to verify if an unsafe set is possibly reached. Several geometric representations for linear continuous systems have been investigated: polytopes [Chutinan and Krogh, 2003], ellipsoids [Kurzhanski and Varaiya, 2014], griddy polyhedra [Asarin et al., 2000], zonotopes [Girard, 2005], or support functions [Girard and Guernic, 2008] which unify

the other mentioned representations. For linear systems with disturbances zonotopes [Girard, 2005] and support functions [Girard and Guernic, 2008] have clearly outperformed existing methods, allowing the verification of systems with more than 100 continuous state variables. The reachability analysis of non-linear or hybrid systems has been a challenging problem over the last few decades. Many existing results are based on analysis of solutions of Hamilton–Jacobi–Isaacs equation [Mitchell et al., 2005] on Taylor models [Chen et al., 2012]. However, while aiming to provide as precise approximations as possible, these approaches are not very suitable for symbolic model control where a considerable number of reachable sets should be estimated. For abstraction-based synthesis we recommend to focus on simple interval over-approximations, trading accuracy for efficiency. The literature on interval-based reachability analysis contains a wide range of methods. The main classes of approaches are summarized below. Approaches based on interval arithmetics are presented in [Jaulin et al., 2001]. In other methods, interval reachability analysis is achieved through the preservation of partial orders by the system description resulting from a monotonicity property of the dynamics [Angeli and Sontag, 2003]. This approach can be applied to both monotone systems [Moor and Raisch, 2002], as well as for mixed-monotone systems [Coogan and Arcak, 2015, Meyer et al., 2018]. Some methods focus on the propagation of the initial set over time based on an upper bound of the growth or contraction of the distance between trajectories of a continuous-time system [Maidens and Arcak, 2014, Reissig et al., 2017, Fan et al., 2016]. Methods based on differential inequalities bound the system dynamics by auxiliary differential equations whose solutions define lower and upper bounds of the reachable set for the main system [Scott and Barton, 2013, Shen and Scott, 2017]. The interested reader can find a comprehensive overview of interval-based reachability analysis techniques many of which have been implemented in [Meyer et al., 2019] in the recent book [Meyer et al., 2021].

1.3 Illustrative Example: ABCS for Safety Specification

Let us use Figure 1.1 to briefly explain how symbolic model control can be used to solve Problem 1.1.1. Given a control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ and a safe set Y we aim to compute a transition system $\Sigma = (Q, U, F)$, with finite sets of states Q and inputs U, and transition relation $F \subseteq Q \times U \times Q$ that is linked to concrete system Σ_f with feedback refinement relationship. Let us introduce a finite partitioning on the state space $\mathbb{R}^{n_x}$ and associate every element of this partitioning belonging to Y (countered with red) with a safe state $q_i \in Q, i = 1, \ldots, 24$. In the first sub-figure of Figure 1.1 the safe sets are filled with white color. The grey region is associated with an unsafe state $q_{us} \in Q$. Thus, an abstract state is considered as atomic symbol representing an infinite number of states from $\mathbb{R}^{n_x}$. We then approximate the original set of inputs $\mathcal{U}$ with a finite subset U. In our particular example $U = \{u_1, u_2\}$ and transitions corresponding to u_1 and u_2 are marked with black and blue correspondingly. Introducing a time sampling parameter τ_{fix} we then say that for all $q \in Q_S = Q \setminus \{q_{us}\}, u \in U, q' \in Q$ transition $(q, u, q') \in F$ if and only if $q' \cap \overline{\mathrm{Reach}}(\tau_{fix} \mid q, u) \neq \emptyset$, where $\overline{\mathrm{Reach}}(\tau_{fix} \mid q, u)$ is an over-approximation of the reachable set

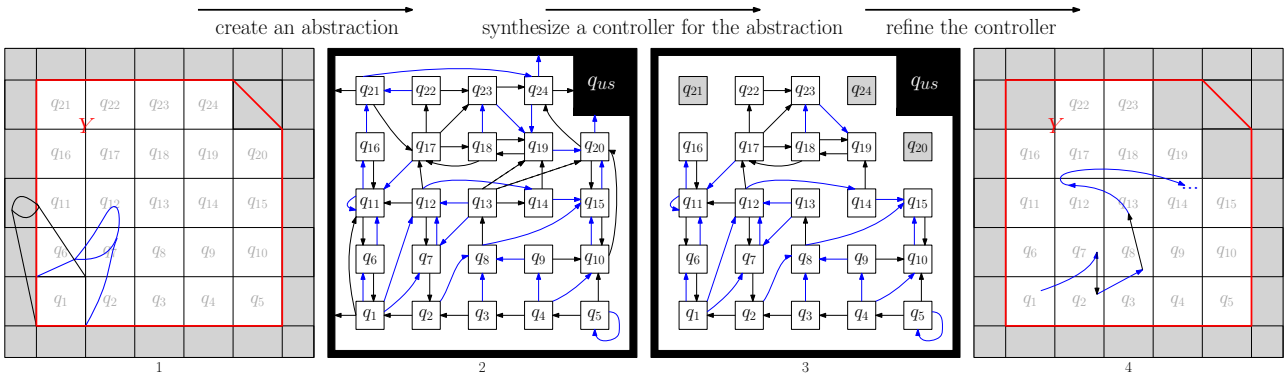


Figure 1.1: Illustration of an abstraction-based control synthesis approach for a safety specification.

$\mathrm{Reach}(\tau_{fix} \mid q, u)$, corresponding to an initial set q, a constant control function $\mathbf{u} : [0, t] \to u, u \in U$ and admissible disturbances $\mathbf{w} \in \mathcal{L}(\mathcal{T}, \mathcal{W})$, i.e.

$$\mathrm{Reach}(t \mid q, u) = \big\{x \in \mathbb{R}^{n_x} \mid \exists\, x(0) \in q \text{ and } \exists\, \mathbf{w} \in \mathcal{L}([0, t], \mathcal{W}) \text{ such that } x_f(t \mid x(0), \mathbf{u}, \mathbf{w}) = x\big\}.$$

Remark 1.3.1. *Here we use the notation* $\mathrm{Reach}(t \mid q, u)$ *instead of* $\mathrm{Reach}(t \mid q, \mathbf{u})$ *to emphasize that the reachable set corresponds to a constant control function. We will keep this logic later in the text as well.*

So, to compute transition originated in a state q_1 and corresponding to controls u_1, u_2 we have to calculate two different reachable tubes, as it is shown in the first sub-figure. We then repeat the same procedure for every $q \in Q_S, u \in U$ to compute the whole symbolic model (the second sub-figure). Let us remark that $F(q_1, u_1) = \{q_{us}, q_{11}\}$, and $F(q_1, u_2) = \{q_6, q_7, q_{12}\}$, i.e., the obtained transition system is non-deterministic. To synthesize a safety controller for the abstraction one can use the classical algorithm from [Tabuada, 2009]. According to that algorithm, all unsafe transitions are iteratively removed from the transition system until the system stops changing. The safety controller $C \colon Q \to 2^U$ is then initialised for every state q with remaining actions. A transition is unsafe if it is not safe, while safe transitions correspond to actions, preventing the system from steering into a blocking or unsafe state. For example in the second sub-figure of Figure 1.1 the transitions colored with black from states q_1, q_{21} and the transitions colored with blue from states q_{24} and q_{20} are unsafe since they steers into q_{us}. A state is blocking if there is no transitions allowing to leave this state (blocking states are grey). Hence, a safety controller C ensures that all trajectories of the controlled transition system are infinitely long and belong to the safe set $\mathrm{Dom}(C) \subseteq Q_S \subseteq Y$ (sub-figure 3 for the illustration). Here $\mathrm{Dom}(C)$ denotes the domain of the controller, i.e. $\mathrm{Dom}(C) = \{q \in Q_S \mid C(q) \neq \emptyset\}$.

We then can guarantee for the original system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ that all trajectories starting in $\mathrm{Dom}(C)$ can be maintained within a safe set Y infinitely long. Indeed, let, for example, $x(0) \in q_1$ (sub-figure 4). Since $q_1 \in \mathrm{Dom}(C)$ there is a controller $u_2 \in C(q_1)$ and if we apply input u_2 τ_{fix} seconds we end up in q_6, q_7 or in q_{12}, depending on a disturbance realization $\mathbf{w}$. Let us suppose that transition finishes at q_7. Since C is a safe controller

for the transition system states $q_7 \in \mathrm{Dom}(C)$ and there is a safe input $u_1 \in C(q_7)$ such that in τ seconds it again brings us in the $\mathrm{Dom}(C) \subseteq Y$, and so on. Hence, the real-time controller for the original system is implemented as a look-up table and, we guarantee that every τ seconds the closed-loop trajectory of Σ_f is in a safe set Y. To guarantee that it doesn't leave Y between too check-points, one either chooses τ sufficiently small, or additionally requires that $\mathrm{Reach}(t \mid q, u) \cap Y = \emptyset, t \in [0, \tau_{fix}]$ for all $q \in Q_S, u \in U$, while constructing the symbolic model.

Let us remark that if a safe controller C enables several inputs for a state, as for example in q_{12}, any of them ensure safety and we can benefit from this flexibility to achieve more difficult control objectives.

1.4 Motivation and Main Contributions

The main drawback of abstraction-based synthesis approaches is poor scalability. Indeed, symbolic models are commonly obtained through the discretization of state and input spaces. More accurate abstractions require more accurate sampling parameters and result in symbolic models with many states and inputs. Consequently, the construction of precise abstractions required for a successful synthesis is often computationally intractable. Moreover, the complexity of the discrete controller synthesis algorithms typically depends on the size of symbolic models. Finally, safety controllers based on larger symbolic models require more memory for their real-time implementation. A demand to reduce the computational complexity of abstraction-based synthesis approaches reasoned considerable amount of research over the last decade and motivated writing this thesis. There are several research directions one can explore aiming to overcome the scalability issues:

- compositional approaches as a way to overcome the curse of dimensions [Saoud, 2019];
- efficient reachability algorithms as a way to speed-up computation of symbolic models [Meyer et al., 2021].

However, let us leave these two directions out of the scope of this manuscript and focus on the following two:

- lazy synthesis algorithms for transition systems;
- size-efficient symbolic models.

The main idea of lazy approaches to outperform the classical synthesis algorithm [Tabuada, 2009] is to restrict computations to the essential for the synthesis part [Girard et al., 2016, Hsu et al., 2019, Hussien and Tabuada, 2018, Nilson et al., 2017, Rungger and Stursberg, 2012]. In such approaches, abstractions are computed on the fly, in parallel with the synthesis procedure, and the unexplored part of the symbolic model remains uncalculated. To minimize the size of symbolic models one can benefit from incremental stability [Girard et al., 2016, Pola et al., 2010, Saoud and Girard, 2018] or monotonicity [Kim et al., 2017, Sinyakov and Girard, 2020a] of the original system. Efficient abstractions were also proposed for differentially flat systems [Liu et al., 2012] and for systems exhibiting a

sparse interconnection structure [Gruber et al., 2017]. As a solution for general class of systems multi-scale abstractions are often considered [Gol et al., 2013, Girard et al., 2016, Hsu et al., 2019, Nilson et al., 2017]. We will focus on them in Chapter 4.

Chapter 2 *structures the existing lazy synthesis approaches* and emphasizes three sources of efficiency: information about a priori controllable states [Girard et al., 2016, Koenig and Likhachev, 2005], priorities on inputs [Hussien and Tabuada, 2018, Girard et al., 2016, Hsu et al., 2019, Nilson et al., 2017], and non-reachable from initial set states [Girard et al., 2016]. Chapter 3 introduces *a novel adaptive time-sampling technique* enforcing the desired (neighbor-linked) structure of the symbolic model. A lazy synthesis algorithm for neighbor-linked abstractions is then provided, with an extension for abstractions with arbitrary time-sampling. Chapter 4 *adapts* the incremental symbolic model exploration and lazy synthesis ideas presented in [Girard et al., 2016] for deterministic transition systems *to non-deterministic abstractions*. A size-efficient multi-scale *symbolic model with an adaptive grid is also introduced*. Chapter 5 introduces two classes of *monotone transition systems* and provides sufficient conditions for an abstraction of a monotone dynamic system [Angeli and Sontag, 2003] to be monotone. Then, lazy synthesis algorithms for monotone symbolic models and directed safety specifications (and their intersections) are presented.

1.5 Thesis Outline

Overall work is dedicated to efficient safety controller synthesis using the abstraction based approaches. The numerical implementations of Chapters 3 and 5 have been implemented in MATLAB, while the numerical implementations of Chapters 2 have been implemented in C++. We summarize below the results illustrated in each chapter.

1.5.1 Chapter 2. Safety Controller Synthesis for Finite Transition System

In Chapter 2, we first remind the classical procedure [Tabuada, 2009] for maximal (by inclusion) safety controller synthesis for finite transition systems (Algorithm 2.1). We then focus on lazy synthesis approaches, which outperform the classical algorithm in terms of efficiency. Structuring existing results, we emphasize three sources of laziness:

- priorities on input space [Girard et al., 2016, Hsu et al., 2019, Hussien and Tabuada, 2018];
- information about a priori controllable states [Girard et al., 2016, Koenig and Likhachev, 2005].
- non-reachable from initial set states [Girard et al., 2016, Tripakis and Altisen, 1999].

We then provide three lazy synthesis algorithms which benefit from these sources. Algorithm 2.2 avoids exploration of states and transitions which are safely controllable a priory. Algorithm 2.3 explores the lower-priority actions only

if the safety problem is unsolvable with higher-priority actions. Based on the incremental exploration of the transition system, Algorithm 2.4 restricts computations for reachable from initial set states only. Algorithm 2.2 returns the maximal safety controller but compute it lazily. The controllers returned by Algorithm 2.3 and Algorithm 2.4 do not satisfy the maximality requirement, and we call them maximal input-lazy and maximal state-lazy safety controllers correspondingly. Both these controllers are a particular case of the maximal lazy safety controller introduced in [Girard et al., 2016]. Maximal input-lazy safety controller C^* has the property that all safely controllable states are in the domain of C^*. Moreover, if the controller enables an input, it also enables all inputs that have the same priority and preserve safety. However, if several inputs can maintain safety, the controller enables only inputs with the highest priority (Definition 2.3.2). Maximal state-lazy safety controller C^* has a property that a safely controllable state is in $\mathrm{Dom}(C^*)$ if and only if it is reachable from the initial set (Definition 2.3.3). Lazy synthesis approaches explore only part of the transition system and allow us to compute the abstraction on-the-fly.

1.5.2 Chapter 3. Lazy Controller Synthesis Based on Safe Set Boundary Exploration

Chapter 3 proposes an abstraction-based approach to synthesize a safety controller solving Problem 1.1.1. The main idea consists in creating a neighbor-linked abstraction, where only transitions between neighboring states are allowed. To construct the abstraction with the desired structure, we interrupt every transition just before leaving the abstract state neighborhood (Figure 1.2 (left)). This idea differs from existing time-sampling approaches, where transitions duration is determined with a given parameter [Tabuada, 2009]. We then use Algorithm 2.2 to synthesize the maximal safety controller lazily by iteratively exploring only those states which border the uncontrollable domain (Theorem 3.1.1), since as soon a safe frontier is found, all internal states are controllable a priori with any admissible input. We then show that even when abstraction is not neighbor-linked, the controller returned by Algorithm 2.2 can be refined to solve Problem 1.1.1 by using a novel control refinement scheme that interrupts the closed-loop trajectory of the original system when it reaches a boundary state of the controllable domain (Theorem 3.3.1). Such a refining procedure ensures that the trajectory does not over-jump the safe boundary, where we have the abstract controller to push the system back towards the interior. In the internal set any admissible control is then allowed (Figure 1.2 (right)).

In spirit, the idea is close to the result of Nagumo theorem ([Blanchini, 1999]) and extreme aiming principle ([Subbotin, 1995]). In simple words, Nagumo theorem says that a convex and closed set S is a positive invariant of the system (1.1) if and only if for all $x \in \partial S$ derivative $\dot{x}$ points inside S or it is tangent to S, while in [Subbotin, 1995] the authors propose to use any control input until the trajectory reaches the boundary. However, to the best of our knowledge, these ideas has never been implemented within the abstraction-based control synthesis framework.

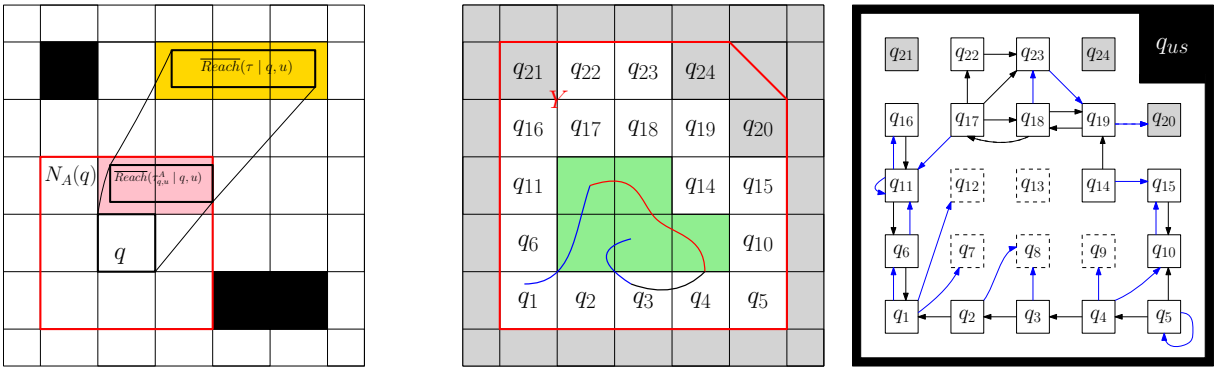


Figure 1.2: The left figure illustrates adaptive time-sampling techniques: while transition duration is commonly determined by a given time sampling parameter τ, we propose interrupting every transition stops just before leaving $N_A(q)$. The right figure represents the result of the synthesis procedure for an arbitrary abstraction, while the central figure shows the corresponding domain of the continuous-time controller. In the green region, we apply any admissible input, but as soon as closed-loop trajectory reaches the white area, we switch to the "abstract" controller, which steers us back to the controllable domain.

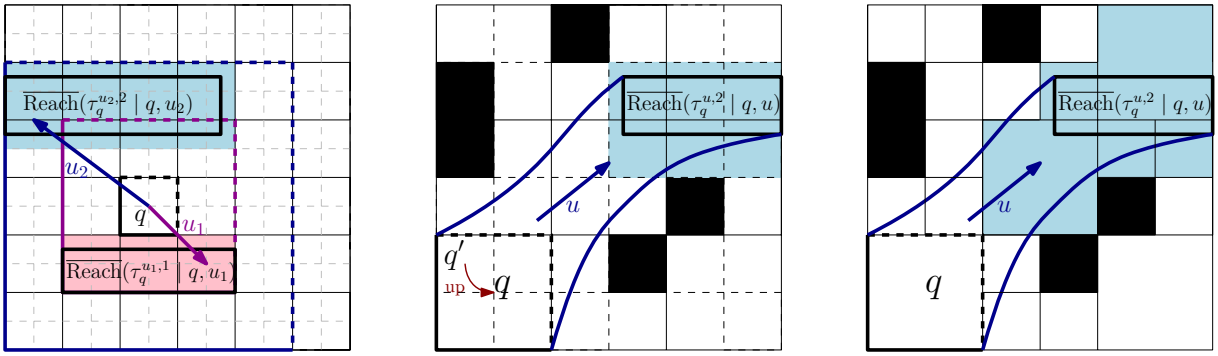


Figure 1.3: The left figure shows transitions with two different durations: for a control u_1 we stop before leaving a radius 1, while for a control u_2 – an interval with a radius 2. We then say that action corresponding to u_1 is less valuable than the action corresponding to u_2 since we prefer first to go further. The central figure shows a 2-layered grid: a coarser level is marked with a normal line and a finer level with a dashed line. The input up brings us from a state q' to a state q such that $q' \subset q$. The transition corresponding to input $(u, 2)$, where 2 determines the duration ends at the finest layer. The right figure then represents the same transition but with the corresponding adaptive grid.

1.5.3 Chapter 4. Lazy Synthesis with Adaptive Symbolic Abstractions

Chapter 4 focuses on approaches, which iteratively refine symbolic models when synthesis for coarser abstractions has not been successful [Gol et al., 2013, Girard et al., 2016, Hsu et al., 2019, Li and Liu, 2018b]. We first construct a multilayered symbolic model with multi-scale time sampling. On the state space of the concrete system, we introduce several uniform Cartesian partitions embedded one into another and combine them into a multilayered grid. We then again constrain transitions duration by intervals that must contain the reachable set. However, unlike the adaptive-time sampling in Chapter 3, neighborhoods of different sizes are used to implement the multiple time duration idea (Figure 1.3 (left)). We also introduce an input up, which allows switching from the current layer to the previous (coarser) one. We also finish all non-up transitions at the finest layer. Aiming to use a lazy synthesis

Algorithm 2.3 we define a partial order on inputs space, prioritizing transitions with a longer duration and putting the input up on top of everything (Figure 1.3 (middle)). In addition to abstraction refining, we propose to explore the symbolic model forwardly and thus restrict the controller synthesis computations to states that are reachable from the initial set only. Thus, we merge Algorithm 2.3 and Algorithm 2.4 from Chapter 2 and compute a controller inheriting properties from both maximal input-lazy and state-lazy controllers (Algorithm 4.2). Such a controller was first considered [Girard et al., 2016], but their synthesis procedure was restricted to a deterministic transition system. We overcome this issue in the present work. Let us remark that with lazy synthesis, we compute for a state q transitions with a shorter duration only if q is uncontrollable with longer duration actions. We also do not compute abstraction for states embedded in q if it is controllable. We also show in Section 4.2.1 that the multilayered grid can be replaced with a more efficient adaptive grid. (two right sub-figures of Figure 1.3 illustrate the difference between adaptive and multi-layered grid).

1.5.4 Chapter 5. Lazy Synthesis for Monotone Systems and Directed Specifications

Chapter 5 is devoted to monotone control systems and specifications given by lower-closed and upper-closed sets. I.e.,it is devoted to systems whose trajectories preserve some partial orderings on their state and input spaces [Angeli and Sontag, 2003, Kamke, 1932, Hirsch and Smith, 2004, Müller, 1927, Smith, 1995] and specifications given by sets which coincide with their lower under upper closure correspondingly, where a lower closure of a set $A \subset \mathcal{L}$ is a set $\downarrow A = \bigcup_{a \in A} \downarrow a$, where $\downarrow a = \{q \in \mathcal{L} \mid q \preceq_{\mathcal{L}} a\}$, while an upper closure is a set $\uparrow A = \bigcup_{a \in A} \uparrow a$, where $\uparrow a = \{q \in \mathcal{L} \mid a \preceq_{\mathcal{L}} q\}$. See Figure 1.4 for an illustration.

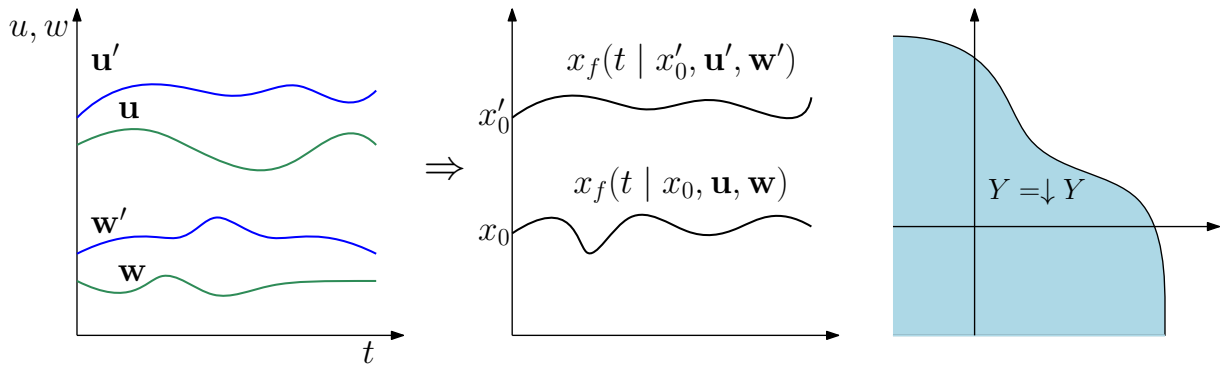


Figure 1.4: The left figures show two trajectories of an input-state monotone dynamical system. The right figure provides an example of a lower-closed safety set. In all figures, a natural component-wise order on $\mathbb{R}^2$ is considered for both the input and the state spaces.

Adapting results from [Angeli and Sontag, 2003], we provide criteria for a dynamical system to be state monotone or input-state monotone and use Example 5.1.1 to illustrate the difference between these two classes of systems. We then present sufficient conditions on state-space partitioning ensuring that the symbolic models inherit the monotonicity of the original plant (Assumptions 5.2.1, 5.2.2) and characterize state-monotone and input-state monotone

transition systems correspondingly (Theorem 5.2.1 and its corollary). We distinguish lower and upper (input-)state transition system (Definitions 5.2.1, 5.2.2), but until Section 5.3.4, we focus on lower (input-)state monotone system and lower-closed safety specifications, assuming that similar results for upper (input-)state monotone system and upper-closed safety specifications can be derived analogously. We say that a transition system $\Sigma = (Q, U, F)$ is lower state monotone if for all $q_1, q_2 \in Q$, for all inputs u enabled in q_2 if $q_1 \preceq_Q q_2$, then u is enabled in q_1 and $F(q_1, u) \subseteq\downarrow F(q_2, u)$. And it is lower input-state monotone if for all $q_1, q_2 \in Q$, for all inputs u_2 enabled in q_2 if $q_1 \preceq_Q q_2$ and $u_1 \preceq_U u_2, u_1 \in U$ then u_1 is enabled in q_1 and $F(q_1, u_1) \subseteq\downarrow F(q_2, u_2)$. See Figure 1.5 (left) for an illustration. Any

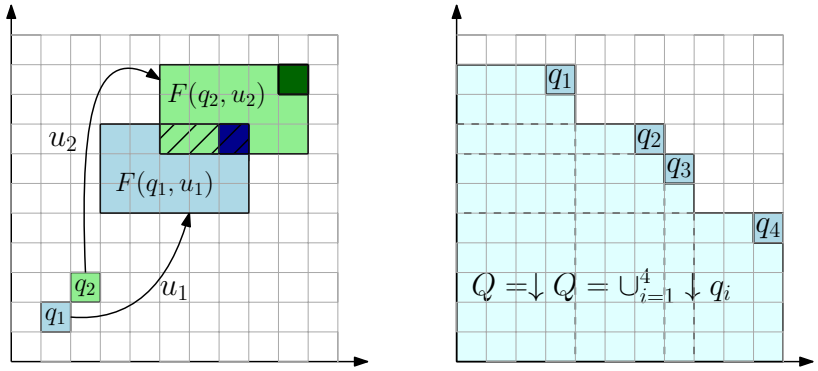


Figure 1.5: The left figure illustrates that if the transition system is lower input state monotone, then we for any two states $q_1 \preceq_Q q_2$ and two inputs $u_1 \preceq_U u_2$, $F(q_1, u_1) \subseteq\downarrow F(q_2, u_2)$. Right figure shows that a finite lower-closed set Q (i.e. Q s.t. $Q =\downarrow Q$) can be represented by its basis $\mathrm{Bas}(Q) = \{q_1, q_2, q_3, q_4\}$, since $Q =\downarrow \mathrm{Bas}(Q)$.

finite lower-closed set can be represented by its basis [Finkel and Schnoebelen, 2001] (Figure 1.5 (right)). Hence, to compute the domain of the maximal safety controller for a state-monotone transition system (Theorem 5.3.1), it is enough to iteratively explore only those states, which belong to the basis of controllable domain (Algorithm 5.3). Moreover, for lower input-state, it is sufficient to use inputs with lower priorities (Theorem 5.3.4). Let us remark that for a lower state-monotone system if a state q is controllable with an input u then all states belonging to $\downarrow q$ are controllable with an input u. Moreover if system is lower input-state monotone transition system then additionally if for a state q an input u is safe then all inputs $u' \succ u$ are safe. The latter allows us to compute the maximal safety controller for lower (input-)state monotone transition systems lazily (Theorem 5.3.2, Algorithm 5.4 and Algorithm 5.5). We then enrich the class of considered specification by intersections of lower-closed and upper-closed sets in Section 5.3.4. Let us remark that instead of using classical box abstraction we use efficient sparse abstractions proposed in [Kim et al., 2017], where the sparse transition relation $F_S(q, u) = \max(F(q, u)), q \in Q, u \in U$. In Figure 1.5 (left) sparse successors are marked with dark blue and green for transitions $F(q_1, u_1)$ and $F(q_2, u_2)$.

1.5.5 Chapter 6. Conclusion and Future Work

In the last chapter, we summarize the results of the thesis and propose several directions for future research.

1.6 Publications

The work presented in this thesis led to several papers either published or accepted.

Journal paper:

- Ivanova E., Saoud A. and Girard A. **Lazy Controller Synthesis for Monotone Transition Systems and Directed Safety Specifications**. Automatica, 2021.

International conference:

- Ivanova E. and Girard A. **Lazy Symbolic Controller for Continuous-Time Systems Based on Safe Set Boundary Exploration**. IFAC Conference on Analysis and Design of Hybrid Systems, Brussel, Belgium, July 2021.
- Ivanova E. and Girard A. **Lazy Safety Controller Synthesis with Multiscale Adaptive-Sampling Abstractions of Nonlinear Systems**. IFAC World Congress, Berlin, Germany, July 2020.
- Saoud A., Ivanova E. and Girard A. **Efficient Synthesis for Monotone Transition Systems and Directed Safety Specifications**. IEEE Conference on Decision and Control, Nice, France, December 2019.

Chapter 2

Safety Controller Synthesis for Finite Transition System: How to Speed-Up the Computations?

As mentioned in the introduction, the successful synthesis requires quite precise abstractions with many states and inputs to be stored and a considerable number of transitions to be computed. Moreover, the complexity of discrete methods used for abstract controller synthesis grows with the size of the symbolic model making synthesis computationally intractable. A demand to overcome a scalability issue of symbolic model approaches reasoned a wave of research in the last decade and motivated writing this thesis. Various methods to reduce the computational effort were proposed. In [Kim et al., 2015, Meyer et al., 2017, Swikir and Zamani, 2019] the authors use compositional approaches as a common way to tackle the curse of dimensions. However, this research branch is out of the scope of this manuscript, and the interested reader is referred to [Saoud, 2019] and reference therein. Other works benefit from incremental stability [Girard et al., 2016, Pola et al., 2010, Saoud and Girard, 2018] or monotonicity [Kim et al., 2017, Sinyakov and Girard, 2020a] to minimize the size of symbolic models. Efficient abstractions were also proposed for differentially flat systems [Liu et al., 2012] and for systems exhibiting a sparse interconnection structure [Gruber et al., 2017]. As a solution for general class of systems multi-scale abstractions were introduced [Gol et al., 2013, Girard et al., 2016, Hsu et al., 2019, Li and Liu, 2018b, Nilson et al., 2017]. We will return in more detail to all these ideas in the following chapters.

This Chapter is devoted to lazy synthesis approaches for a safety specification. In such approaches, abstractions are computed on-the-fly, restricting computations only to the essential for the synthesis part. Section 2.1 introduces the notion of a finite transition system. Section 2.2 formulates a safety control problem for a transition system and provides a classical algorithm synthesizing maximal safety controller. Section 2.3 starts with a literature review

of lazy approaches. Then Subsection 2.3.1 introduces the lazy synthesis algorithm accelerating maximal safety controller computations if the information of a priory controllable states is available. The lazy algorithm in Subsection 2.3.2 benefits from a partial order introduced on the input set. Subsection 2.3.3 proposes to restrict the computations for reachable from initial states only while using incremental search to explore the transition system. All algorithms in Section 2.3 are applicable for non-deterministic transition systems. Section 2.4 summarises the ideas of this Chapter.

2.1 Preliminaries

Let us start with the main definitions.

Definition 2.1.1. *A transition system is a tuple* $\Sigma = (Q, U, F)$*, consisting of a set of states* Q*, a set of inputs* U*, and a transition relation* $F \subseteq Q \times U \times Q$*. If sets* Q, U *are finite* $\Sigma = (Q, U, F)$ *is said to be a finite transition system.*

In this thesis, we only work with finite transition systems, and from now, we will shortly call them transition systems for simplicity. For every transition $(q, u, q') \in F$ the state q is named *u-predecessor* of q' and similarly the state q' is named *u-successor* of q. For the set of all u-predecessors of the state q the notation $F^{-1}(q, u)$ is used, while the set of all u-successors of a state q is denoted by $F(q, u)$. If there is $q \in Q,\ u \in U$ such that $|\, F(q, u)\,| > 1$, then the transition system is called *non-deterministic*, otherwise it is *deterministic*. Since $F(q, u)$ may be empty let us introduce a set $\mathrm{Enab}_F(q) = \{u \in U \mid F(q, u) \neq \emptyset\}$ of all enabled inputs at a state $q \in Q$. If $\mathrm{Enab}_F(q) = \emptyset$, then q is said to be *blocking*, otherwise it is *non-blocking*. We also use notation $\mathrm{Block}_F(Q')$ to describe the set of all blocking states in a set $Q' \subseteq Q$. If $\mathrm{Block}_F(Q) = \emptyset$, then the transition system is called *non-blocking*.

Definition 2.1.2. *A* trajectory *of a transition system* $\Sigma = (Q, U, F)$ *is a finite or infinite sequence of transitions* $q_0 \xrightarrow{u_0} q_1 \xrightarrow{u_1} q_2 \xrightarrow{u_2} q_3 \xrightarrow{u_3} \ldots$*, s.t.* $q^i \in Q, u^i \in U$ *and* $q^{i+1} \in F(q^i, u^i)$ *for all* $i \geq 0$.

Definition 2.1.3. *A* controller *for a transition system* $\Sigma = (Q, U, F)$ *is a map* $C \colon Q \to 2^U$*, such that* $C(q) \subseteq \mathrm{Enab}_F(q)$ *for every* $q \in Q$*. Let us use notation* $\mathrm{Dom}(C) = \{q \in Q \mid C(q) \neq \emptyset\}$ *for a domain of controller* C*. If* $\mathrm{Dom}(C) = \emptyset$ *the controller is called trivial, otherwise non-trivial.*

2.2 Maximal Safety Controller

Problem 2.2.1 (Safety Specification)**.** *For a given transition system* $\Sigma = (Q, U, F)$ *and a safe set* $Q_S \subseteq Q$*, synthesize a non-trivial controller* C *such that* $\mathrm{Dom}(C) \subseteq Q_S$ *and for any trajectory* $q_0 \xrightarrow{u_0} q_1 \xrightarrow{u_1} q_2 \xrightarrow{u_2} q_3 \xrightarrow{u_3} \ldots$*, such that* $q_0 \in \mathrm{Dom}(C)$ *and* $u_i \in C(q_i),\ i \geq 0$ *the following is satisfied:* $q_i \in \mathrm{Dom}(C)$ *for all* $i \geq 0$*.*

A desired controller C is then called a safety controller for transition system $\Sigma = (Q, U, F)$ and a safe set $Q_S \subseteq Q$.

Definition 2.2.1. *A safety controller for a transition system $\Sigma = (Q, U, F)$ and a safe set $Q_S \subseteq Q$ is a controller C such that the following two properties hold*

1. $\mathrm{Dom}(C) \subseteq Q_S$;
2. *for all $q \in \mathrm{Dom}(C)$ for all $u \in C(q)$ the following inclusion is satisfied $F(q, u) \subseteq \mathrm{Dom}(C)$.*

Definition 2.2.2. *For a given transition system $\Sigma = (Q, U, F)$ and a controller C, the controlled transition system $\Sigma/C = (\mathrm{Dom}(C), U, F_C)$ is a transition system with the reduced transition relation $F_C \subseteq F$, such that $(q, u, q') \subseteq F_C$ if and only if $u \in C(q)$.*

Hence, if C is a non-trivial safety controller, then the controlled transition system $\Sigma/C = (\mathrm{Dom}(C), U, F_C)$ is non-blocking and all trajectories of this system belong to the safe set Q_S.

Lemma 2.2.1. *For a given transition system $\Sigma = (Q, U, F)$, a safe set $Q_S \subseteq Q$ there exists the unique maximal safety controller $\bar{C}$ such that for any safety controller C the following hold*

1. $\mathrm{Dom}(C) \subseteq \mathrm{Dom}(\bar{C})$;
2. *for all $q \in \mathrm{Dom}(C)$ the inclusion $C(q) \subseteq \bar{C}(q)$ is satisfied.*

Obviously, to solve Problem 2.2.1 means to synthesize a non-trivial safety controller and the maximal safety controller $\bar{C}$ is the most complete solution one can get. That justifies the following notion of controllability.

Definition 2.2.3. *Let Q_S be a safe set. A state $q \in Q$ of transition system $\Sigma = (Q, U, F)$ is* safety controllable *if and only if $q \in \mathrm{Dom}(\bar{C})$. The set of safety controllable states is denoted $\mathrm{Cont}(\Sigma, Q_S)$.*

The algorithm, proposed in [Tabuada, 2009], is the most common way to compute the maximal safety controller. According to this algorithm, all unsafe transitions are iteratively removed from the transition system until the system stops changing. A transition is safe if it corresponds to an action, preventing the system from steering into a blocking or unsafe state. Otherwise, it is unsafe. The maximal safety controller is initialized to permit all transitions of the resulting system. This simple idea admits various implementations, so let me briefly comment on Algorithm 2.1 (on page 31) and prove that it indeed returns the maximal safety controller. Let a function $p\colon Q \to \{0, 1\}$ indicates whether a state $q \in Q$ is controllable ($p(q) = 1$) or not ($p(q) = 0$), and the reduced transition relation F_p is defined as follows: $(q, u, q') \in F_p$ if and only if $p(q) \neq 0$ and for all $q'' \in F(q, u)$ the equality $p(q'') \neq 0$ is satisfied. First, all safe states in Q_S are marked as controllable, while all states beyond the safe set as uncontrollable (lines 2-5). Then we iteratively explore all potentially controllable states, marking as uncontrollable those which are blocking (lines 6-11). Updating the function p we automatically remove from the transition relation F_p all unsafe transitions. When procedure stops, the reduced transition relation is used to initialize the controller C (lines 12-15).

Remark 2.2.1. *In the definition of the the reduced transition relation F_p we write $p(q'') \neq 0$ instead of $p(\text{"}q\text{"}) = 1$ to ease comparison of the result of this section with the results of the following sections.*

Lemma 2.2.2. *Let Algorithm 2.1 executes for transition systems $\Sigma_1 = (Q, U, F^1)$, $\Sigma_2 = (Q, U, F^2)$ with safety specifications Q_S^1, Q_S^2 correspondingly, and let C^1, C^2 be returned controllers. Let $Q_S^1 \subseteq Q_S^2$ and $\mathrm{Enab}_{F^1}(q) \subseteq \mathrm{Enab}_{F^2}(q)$ for all $q \in Q$. If for all $q \in Q,\ u \in \mathrm{Enab}_{F^1}(q)$ the inclusion holds $F^2(q,u) \subseteq F^1(q,u)$ then $C^1(q) \subseteq C^2(q)$ for all $q \in Q$.*

Proof. The fact that $C^1(q) \subseteq C^2(q)$ for all $q \in Q$ means that $\mathrm{Enab}_{F^1_{p_1}}(q) \subseteq \mathrm{Enab}_{F^2_{p_2}}(q)$ for all $q \in Q$ when the algorithm reach the line 12. Under assumptions of the lemma about transition relations F^1 and F^2 the latter is equivalent to $p_1(q) \leq p_2(q)$ (see the definitions of the reduced transition relations $F^1_{p_1}$ and $F^2_{p_2}$). Let us now prove that inequality holds. Since $Q_S^1 \subseteq Q_S^2$ at line 5 the inequality $p_{1,0}(q) \leq p_{2,0}(q)$ obviously holds for all $q \in Q$ and this give me induction base. Let $p_{1,i}(q) \leq p_{2,i}(q)$ at the beginning of i iteration of the loop 6-11. Then $Q^1_{E,i} \subseteq Q^2_{E,i}$ and since the transition relation $F^1_{p_{1,i}}$ is included in $F^2_{p_{2,i}}$ the set B_i^2 is included in B_i^1 from where $p_{1,i+1}(q) \leq p_{2,i+1}(q)$. Let us also remark that if at some iteration $\tilde{i}$ the set $B_{\tilde{i}}^2$ is already empty, while $B_{\tilde{i}}^1$ is not then the inequality $p_{1,i}(q) \leq p_{2,\tilde{i}}(q),\ q \in Q$ is also obviously holds for all $i \geq \tilde{i}$ until B_i becomes empty as well. □

Theorem 2.2.1. *Let C computed by Algorithm 2.1. Then, C is the maximal safety controller.*

Proof. Let us first remark that since the set Q is finite and with every iteration of the loop, all blocking states are marked as uncontrollable and then never explore again, Algorithm 2.1 always ends its execution. Moreover since $F_p(q,u) \subseteq F(q,u)$ for all $q \in Q,\ u \in U$ the map C is a controller for the transition system $\Sigma = (Q, U, F)$. Let us prove that it is the maximal safety one.

Safety. Let us first show that a state $q \in Q$ belongs to $\mathrm{Dom}(C)$ if and only if $p(q) \neq 0$. If $q \in \mathrm{Dom}(C)$ then $\mathrm{Enab}_{F_p}(q) \neq 0$, consequently, from the definition of F_p it follows that $p(q) \neq 0$. Let for a state $q \in Q$, $p(q)$ is equal to 0, when the execution reaches the line 12. It means that the state q was in Q_E at the last iteration of the loop 6-11 and since the exit condition from the loop is $B = \emptyset$ the state $q \notin \mathrm{Block}_{F_p}(Q_E)$, i.e. there exists $u \in \mathrm{Enab}_{F_p}$, consequently, $q \in \mathrm{Dom}(C)$. Hence, from the definition of F_p it follows that for all $q \in \mathrm{Dom}(C)$ the inclusion $F_p(q,u) \subseteq \mathrm{Dom}(C)$ is satisfied for all $u \in \mathrm{Enab}_{F_p}$, i.e. for all $q \in \mathrm{Dom}(C)$ and for all $u \in C(q)$ the inclusion $F(q,u) \subseteq \mathrm{Dom}(C)$. To prove that $\mathrm{Dom}(C) \subset Q_S$ let me suppose the opposite: there exists $q \in \mathrm{Dom}(C) \cap (Q \setminus Q_S)$. Obviously, if some state q was marked as uncontrollable (i.e. $p(q) = 0$) it stays uncontrollable until the end of Algorithm 2.1 execution. Hence, if $q \in Q \setminus Q_S$ then $p(q) = 0$, but this contradicts to $q \in \mathrm{Dom}(C)$.

Maximality. Let $\bar{C}$ is the maximal safety controller for a transition system $\Sigma = (Q, U, F)$ and a safe set Q_S. Since C is a safety controller then for all $q \in Q$ the following holds $C(q) \subseteq \bar{C}(q)$. To prove the opposite let me run Algorithm 2.1 for transition system $\Sigma = (Q, U, F_{\bar{C}})$ and a safe set $\mathrm{Dom}(\bar{C})$. In this case, the returned controller $\tilde{C}$ obviously coincide with the maximal safety controller $\bar{C}$. Since $\mathrm{Dom}(\bar{C}) \subseteq Q_S$ and for all $q \in Q$ the set $\mathrm{Enab}_{F_{\bar{C}}}(q) = \mathrm{Enab}_F(q)$, moreover for all $u \in \mathrm{Enab}_F(q)$ sets of successors $F_{\bar{C}}(q,u), F(q,u)$ are equal, from Lemma 2.2.2 it follows that $\bar{C}(q) \subseteq C(q)$. This ends the proof. □

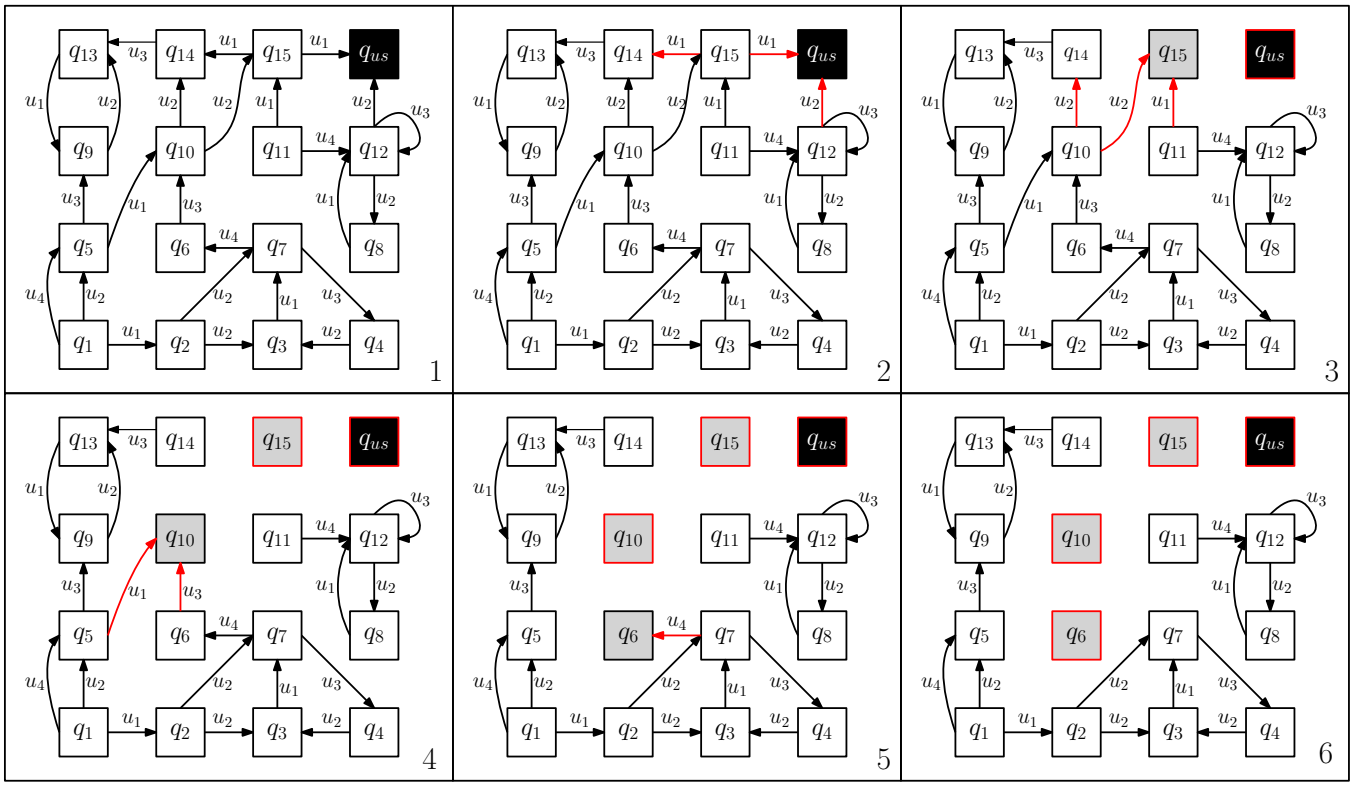


Figure 2.1: Illustration of Algorithm 2.1. Algorithm 2.1 iteratively removes all unsafe actions from a transition system until the system stops changing. An action is unsafe if it can not prevent the system from steering into an unsafe or blocking state. Unsafe actions are marked with a red color. Blocking states are filled with grey, the unsafe state - with black. A state q with $p(q) = 1$ is contoured with black, with $p(q) = 0$ - with red.

Figure 2.1 illustrates the execution of Algorithm 2.1.Starting at step 1 with a simple transition system, we first indicate which state is unsafe. Then at step 3, the unsafe set is initialized with priority 0. Consequently, all actions steering to q_{us} are removed from the system. At steps 4-6, we continue iteratively delete inputs, which can not prevent safety, until at step 6, we get a non-blocking transition system, all trajectories of which belong to $Q_S = \{q_1, \dots, q_{15}\}$. Though Algorithm 2.1 is sound and complete, it explores all states and all transitions making computations too labor-intensive for many real-world problems, where a considerable number of states and inputs should be explored. In the next section, we introduce more efficient safety synthesis algorithms and discuss their correctness and completeness with respect to the classical synthesis approach.

2.3 Lazy Synthesis Algorithms

The obvious way to reduce the computational burden is to avoid computations, which are non-essential for synthesis goals. This is a purpose for lazy synthesis algorithms [Girard et al., 2016, Nilson et al., 2017, Hsu et al., 2019, Hussien and Tabuada, 2018, Rungger and Stursberg, 2012, Kader et al., 2019]. Unlike the classical algorithm, in lazy approaches, the system is abstracted parallel with the controller synthesis. Hence, one can avoid calculating

the part of the abstraction, which is not involved in the synthesis procedure. The worst-case complexity of lazy approaches usually coincides with the classical synthesis's complexity or can be even higher, but lazy algorithms are more efficient in practice. Lazy synthesis usually requires additional assumptions about a given transition system, and the following three different sources of laziness are mainly considered in the literature:

- **priorities on input space**. In [Hussien and Tabuada, 2018], once a safe input is found for a state, the other controls are not explored for this state. The authors start with an empty transition relation and iteratively add new transitions essential for the synthesis purpose. At every iteration, the precomputed part of the symbolic model is explored. Then, one randomly chooses N states from uncontrollable states, where N is a parameter given by the user. For each chosen state, one randomly picks an unexplored input and adds the corresponding tradition to the abstraction. The algorithm returns only the domain of the maximal safety controller and not the whole maximal safety controller. So, in some sense, the authors introduce a random order on the set of inputs and try to solve the problem using as higher priorities actions as possible. In [Girard et al., 2016, Nilson et al., 2017, Hsu et al., 2019, Kader et al., 2019] the authors use multi-scale abstractions and prioritize actions with longer duration. The idea to give different inputs different priorities is common while dealing with more general specifications as well. For example, many path-finding algorithms explore the most promising actions [Hart et al., 1968, Rungger and Stursberg, 2012] first. In cases when specification can not be satisfied, the least-violating [Girard and Eqtami, 2021, Tůmová et al., 2012] actions are often favored.
- **non-reachable from initial set states**. In [Girard et al., 2016, Tripakis and Altisen, 1999] it is also proposed to restrict computations only for states, which are reachable from the initial set. To do so, the authors implement the incremental transition system exploration, which in turn enables the usage of their approach while dealing with more general control problems, such as, for instance, a reach-avoid specification [Hsu et al., 2019, Reissig et al., 2017].
- **information about a priory controllable states**. If, in addition, the incrementally explored transition system is deterministic, one can mark a state as controllable as soon as a safe loop-path has been found for this state [Girard et al., 2016]. These a priory controlled states need not be investigated in future iterations of the synthesis algorithm. Skipping exploration of a part of the transition system is also popular when slightly changing specifications are considered [Koenig and Likhachev, 2005].

We provide three lazy synthesis algorithms Algorithm 2.2, 2.3 and 2.4 which benefit from these sources. Algorithm 2.2 avoid exploration of states and transitions which are safely controllable a priory. Algorithm 2.3 explores the lower-priority actions only if the safety problem is unsolvable with higher-priority actions. Based on the incremental exploration of the transition system, Algorithm 2.4 restricts computations for reachable from initial set states only. Algorithm 2.2 returns the maximal safety controller, but compute it lazily.The controllers returned by Algorithm 2.3 and 2.4 do not satisfy the maximality requirement, and we call them lazy controllers. Still, lazy controllers are retaining other important properties serving as a compromise between maximality and efficiency.

Algorithm 2.1: ClassicalSynthesisMSC(Σ, Q_S)

Input: $\Sigma = (Q, U, F)$ and a safe set Q_S
Output: MS controller C

begin
  for $q \in Q_S$ **do**
    $p(q) := 1$;
  for $q \in Q \setminus Q_S$ **do**
    $p(q) := 0$;
  repeat
    $Q_E := \{q \in Q \mid p(q) \neq 0\}$;
    $B := \mathrm{Block}_{F_p}(Q_E)$;
    for $q \in B$ **do**
      $p(q) := p(q) - 1$;
  until $B = \emptyset$;
  for $q \in Q_E$ **do**
    $C(q) := \mathrm{Enab}_{F_p}(q)$;
  for $q \in Q \setminus Q_E$ **do**
    $C(q) := \emptyset$;
  return C;

Algorithm 2.2: LazySynthesisMSC(Σ, Q_S, I_{EB})

Input: $\Sigma = (Q, U, F)$, a safe set Q_S, an indicator I_{EB}
Output: MS controller C

begin
  for $q \in Q_S$ **do**
    $p(q) := 1$;
  for $q \in Q \setminus Q_S$ **do**
    $p(q) := 0$;
  repeat
    $Q_{EB} := \{q \in Q \mid p(q) \neq 0 \text{ and } I_{EB}(q) = 1\}$;
    $B := \mathrm{Block}_{F_p}(Q_{EB})$;
    for $q \in B$ **do**
      $p(q) := p(q) - 1$;
  until $B = \emptyset$;
  for $q \in \{q \in Q \mid p(q) \neq 0\}$ **do**
    $C(q) := \mathrm{Enab}_{F_p}(q)$;
  for $q \in Q \setminus \{q \in Q \mid p(q) \neq 0\}$ **do**
    $C(q) := \emptyset$;
  return C;

Algorithm 2.3: MILSC($\Sigma, Q_S, \preceq_U$)

Input: $\Sigma = (Q, U, F)$ and a safe set Q_S
Output: MILS controller C.

begin
  for $q \in Q_S$ **do**
    $p(q) := N$;
  for $q \in Q \setminus Q_S$ **do**
    $p(q) := 0$;
  repeat
    $Q_E := \{q \in Q \mid p(q) \neq 0\}$;
    $B := \mathrm{Block}_{F_{p,\preceq_U}}(Q_E)$;
    for $q \in B$ **do**
      $p(q) := p(q) - 1$;
  until $B = \emptyset$;
  for $q \in Q_E$ **do**
    $C(q) := \mathrm{Enab}_{F_{p,\preceq_U}}(q)$;
  for $q \in Q \setminus Q_E$ **do**
    $C(q) := \emptyset$;
  return C;

Algorithm 2.4: MSLSC(Σ, Q_S, Q_{init})

Input: $\Sigma = (Q, U, F)$, a safe set Q_S, an initial set Q_{init}
Output: MSLS controller C.

begin
  for $q \in Q_S$ **do**
    $p(q) := 1$;
  for $q \in Q \setminus Q_S$ **do**
    $p(q) := 0$;
  repeat
    $Q_E^R := \mathrm{Reach}_{F_p}(\{q \in Q_{init} \mid p(q) \neq 0\})$;
    $B := \mathrm{Block}_{F_p}(Q_E^R)$;
    for $q \in B$ **do**
      $p(q) := p(q) - 1$;
  until $B = \emptyset$;
  for $q \in Q_E^R$ **do**
    $C(q) := \mathrm{Enab}_{F_p}(q)$;
  for $q \in Q \setminus Q_E^R$ **do**
    $C(q) := \emptyset$;
  return C;

The detailed description of Algorithm 2.2, 2.3, and 2.4 is provided in Sections 2.3.1, 2.3.2 and 2.3.3 correspondingly. We also illustrate the proposed approaches with Figures 2.2, 2.3 and 2.4 correspondingly.

2.3.1 Source of Laziness: Information about A priory Controllable States

This section introduces a lazy Algorithm 2.2, which benefits from information about a priory controllable states.

Definition 2.3.1. *Consider $\Sigma = (Q, U, F)$ and a set $Q_E \subseteq Q$. We say that a set $Q_{EB} \subseteq Q_E$ is an essential basis of set Q_E if for all $q \in Q_E \setminus Q_{EB}$ the set $\mathrm{Enab}_F(q) \neq \emptyset$ and for all $u \in \mathrm{Enab}_F(q)$ the following holds $F(q, u) \subseteq Q_E$.*

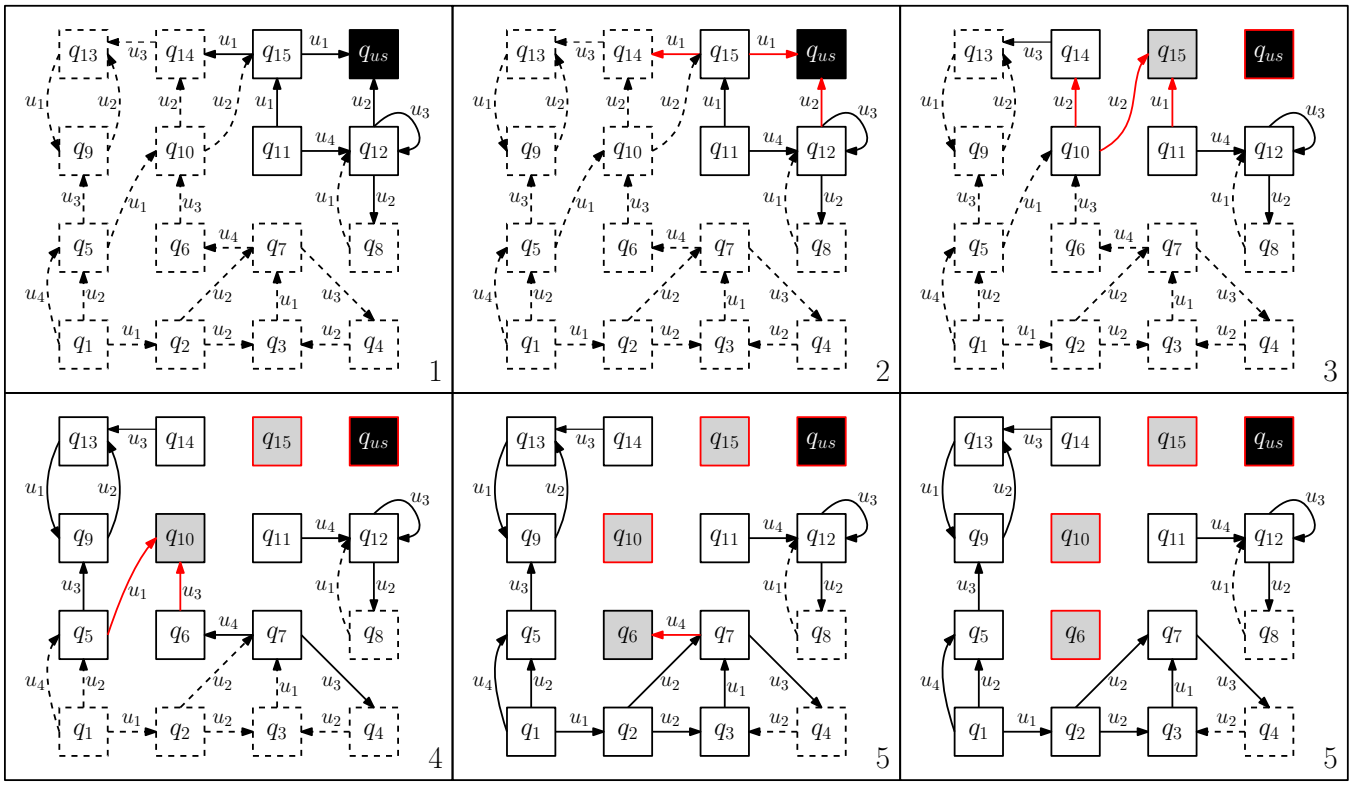


Figure 2.2: Illustration of Algorithm 2.2 execution.Blocking states are filled with grey. The unsafe state is black. A state q such that $p(q) = 1$ are contoured with black, with $p(q) = 0$ - with red. Let us remark that all transitions in the system are between neighboring states. Hence, we can restrict exploration to those states that border with blocking or unsafe states. The unexplored elements are marked with the dashed line, while the normal line denoted states already involved in computations. At the end of execution, states q_8, q_4 stay unconsidered, but we can automatically mark them as controllable, including all actions enabled at these states to the maximal safety controller.

Hence, if at some iteration of Algorithm 2.1 a set Q_{EB} is an essential basis of a set Q_E, then all states $q \in Q \setminus Q_E$ are controllable a priory at this iteration and the following result is true.

Theorem 2.3.1. *Let a set $Q_E = \{q \in Q \mid p(q) \neq 0\}$ and let at every iteration of the loop 6-11 of Algorithm 2.2 the set $Q_{EB} = \{q \in Q_E \mid I_{EB}(q) = 1\}$ is essential basis of Q_E then the controller C computed by Algorithm 2.2 is the maximal safety controller.*

Proof. Indeed, from Definition 2.3.1 and definition of the reduced transition relation F_p it follows that for all $q \in Q_E$ such that $I_{EB}(q) = 0$ the set $\mathrm{Enab}_{F_p}(q) \neq 0$, consequently $q \notin B$ at every iteration of the loop 6-11. □

Hence, in the general case, Algorithm 2.2 does not return a safety controller. However, if the information function $I_{EB} : Q \to \{0, 1\}$ distinguishes a priory controllable on the current iteration states from those which should be explored, then Algorithm 2.2 synthesize the maximal safety controller. The idea is simple, but the main intrigue is in determining the information function I_{EB}, which is different from an indicator of a set $Q_E = \{q \in Q \mid p(q) \neq 0\}$. For instance, in our illustrative example (see Figure 2.2), all states are linked only with their neighbors. Hence, iteratively explore only those states that surround blocking or unsafe states is enough. The latter speed up computation

in comparison with the classical approach. Moreover, states q_4, q_8 were never involved in computations, and the corresponding part of the abstraction may remain uncalculated. Of course, in a general system, transitions between non-neighboring states are possible. Still, in Chapter 3, we show how one can ensure that the abstraction (constructed for the original dynamic system) has the desirable structure by using adaptive time sampling techniques. Another way to get necessary information about a priory controllable states is to benefit from the monotonicity of the systems as we do in Chapter 5.

2.3.2 Source of Laziness: Inputs' Priorities

The ideas of this subsection and the following one were first introduced in [Girard et al., 2016]. However, they were restricted to deterministic transition systems. The results provided in this manuscript are applicable for non-deterministic systems as well.

Assumption 2.3.1. *Let the set of inputs U be split into N non-intersecting subsets $U = U_1 \cup U_2 \ldots \cup U_N$ and for all $u' \in U_i,\ u'' \in U_j,\ i < j,\ i, j \in \{1, \ldots, N\}$ it holds that $u' \prec u''$, while inputs u', u'' belonging to the same subset are considered as equivalent $u' \simeq u''$.*

I.e., it is supposed that a set of inputs U is equipped with a partial order. Then if for some state several inputs preserve safety, it is reasonable to keep only those that have the highest priority. Here we say that an input $u \in U$ has higher priority than $u' \in U$ if and only if $u \succ u'$.

Definition 2.3.2. *For a given transition system $\Sigma = (Q, U, F)$, a safety specification $Q_S \subseteq Q$ a maximal inputs-lazy safety (MILS) controller C^* is a safety controller satisfying the following properties*

1. $\mathrm{Dom}(C^*) = \mathrm{Cont}(\Sigma, Q_S)$;
2. *for all states $q \in \mathrm{Dom}(C^*)$:*
   (a) *if $u \in C^*(q)$ then for all $u' \in \mathrm{Enab}_F(q)$ such that $u' \simeq u$, it holds that $u' \in C^*(q)$ if and only if $F(q, u') \subseteq \mathrm{Cont}(\Sigma, Q_S)$;*
   (b) *if $u \in C^*(q)$, then for all $u' \in \mathrm{Enab}_F(q)$ with $u \prec u'$, it holds that $F(q, u') \cap \mathrm{Cont}(\Sigma, Q_S) \neq F(q, u')$.*

A MILS controller is a particular case of the maximal lazy safety (MLS) controller introduced in [Girard et al., 2016]. It corresponds to a situation when an initial set coincides with a safe set. The MLS controller exists for any transition system, and it is unique [Girard et al., 2016]. Consequently, a MILS controller exists and is unique, as well. The term maximal refers to the fact that all safety controllable states are in $\mathrm{Dom}(C^*)$, and if the controller enables an input, it also enables all inputs which have the same priority and preserve safety. The term inputs-lazy refers to the fact that while several inputs can maintain safety, the controller enables only inputs with the highest priority. Hence, C^* represents a trade-off between maximal permissiveness and efficiency. Intuitively, the MILS controller C^* can be

obtained from the maximal safety controller $\bar{C}$ by keeping, for every state, only those enabled controls which have a higher priority. Of course, it is not the best way to find C^* since it needs first to compute $\bar{C}$.

The main idea of a more efficient algorithm for computing the MILS controller is to explore inputs with a lower priority only if we failed to find a safe input with a higher priority. For its implementation, we introduce for every state $q \in Q$ a notion of a state priority $p(q)$ and define for a given $p\colon Q \to \{0, \ldots, N\}$ a reduced transition relation $F_{p,\preceq_U}$ such that $(q, u, q') \in F_{p,\preceq_U}$ if and only if $p(q) \in \{1, \ldots, N\}$, $u \in U_{p(q)}$, $(q, u, q') \in F$ and for all $q'' \in F(q, u)$ the equality $p(q'') \neq 0$ is satisfied. Intuitively, this means that for states with priority from 1 to N, only transitions with the same priority inputs are considered, while states with priority 0 are blocking. Starting with the highest priority for states in a safe set Q_S and with the lowest one for the others (lines 2-5), we iteratively decreasing by one point the priority of all blocking but controllable states (line 10) in the main block of Algorithm 2.3 (lines 6-11) until the reduced transition relation $F_{p,\preceq_U}$ stops changing. Then $F_{p,\preceq_U}$ is used to initialize the controller C. Let us remark that if all inputs have the same priority, Algorithm 2.3 coincides with Algorithm 2.1.

Theorem 2.3.2. *Let C^* computed by Algorithm 2.3. Then, C^* is the MILS controller.*

Proof. The proof of the fact that $\mathrm{Dom}(C^*) = \{q \in Q \mid p(q) \neq 0\}$ also as the fact that C^* is a safety controller repeats the proof of analogous result from the Theorem 2.2.1. Since C^* is a safety controller $\mathrm{Dom}(C^*) \subseteq \mathrm{Cont}(\Sigma, Q_S)$. To prove that $\mathrm{Cont}(\Sigma, Q_S) \subseteq \mathrm{Dom}(C^*)$ let me run the Algorithm 2.3 for the transition system $\Sigma = (Q, U, F)$ and a safe set $\mathrm{Cont}(\Sigma, Q_S)$ and let the controller C_1^* is the result of the execution. Since the result analogous to Lemma 2.2.2 can be proven for Algorithm 2.3 as well the following inclusion holds $C_1^*(q) \subseteq C^*(q)$ for all $q \in Q$. Consequently $\mathrm{Dom}(C_1^*) \subseteq \mathrm{Dom}(C^*)$ and since $\mathrm{Dom}(C_1^*) = \mathrm{Cont}(\Sigma, Q_S)$ and we get what we want.

Let us now prove that the second property of Definition 2.3.2 is satisfied. For any $q \in \mathrm{Dom}(C)$ a control $u \in C(q)$ if and only if $u \in \mathrm{Enab}_{F_{p,\preceq_U}}(q)$. Then for any $u' \in \mathrm{Enab}_F(q) \cap U_{p(q)}$ we have: if $u' \in \mathrm{Enab}_{F_{p,\preceq_U}}(q)$ then $F(q, u') = F_{p,\preceq_U}(q, u') \subseteq \mathrm{Dom}(C) \subseteq \mathrm{Cont}(\Sigma, Q_S)$, because C is a safety controller; if $F(q, u') \subseteq \mathrm{Cont}(\Sigma, Q_S)$ then there is no $q' \in F(q, u')$ such that $p(q') = 0$ and $u' \in \mathrm{Enab}_{F_{p,\preceq_U}}(q)$. So, we have (2a). The (2b) follows from the fact that we decrease the priority p of a state q, if either $\mathrm{Enab}_F(q) \cap U_{p(q)} = \emptyset$, or for all $u \in \mathrm{Enab}_F(q) \cap U_{p(q)}$ there exists $q' \in F(q, u)$ such that $p(q') = 0$. □

Figure 2.3 illustrates the execution of Algorithm 2.3 for a simple transition system. In opposite to previous examples, now actions have different priorities, and we enable for a state actions u_3, u_4, only if it is uncontrollable with actions u_1, u_2. At the beginning (step 3), all states except q_{us} get priority 2. Then, if a state q is blocking concerning its current priority at some step, we decrease $p(q)$ by one point. The latter either enable for this state actions with a lower priority (see, for example, a state q_{14} at step 6), or delete all unsafe inputs steering to state q (if q get a 0 as, for example, q_{15} at step 7).

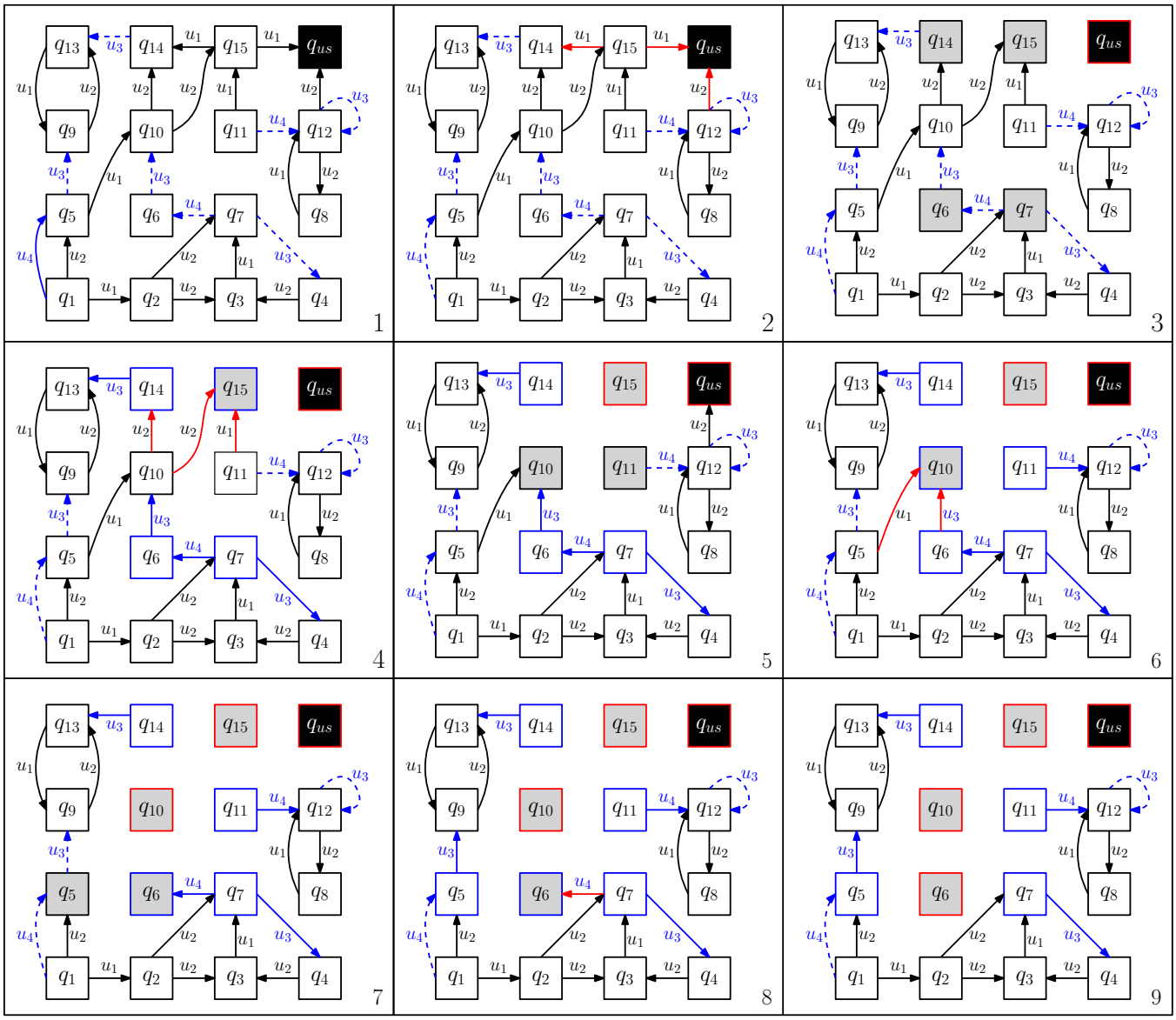


Figure 2.3: Illustration of Algorithm 2.3 execution. Unsafe actions are red. Blocking states are grey. Controls u_1, u_2 have priority 2, u_3, u_4 have priority 1. If a state is contoured with black, it has priority 2, and the actions with priority 2 are enabled for this state. If with blue, it has priority 1, and the inputs with priority 1 are available. If with red, then its priority is 0. We enable lower priority transitions for a state only if it is uncontrollable with higher priority actions. Enable at the current iteration transition marked with a normal line, while action marked with a dashed line are non-available. The actions which remain untouched at step 9 are not included in the maximal input lazy safety controller.

2.3.3 Source of Laziness: Non Reachable from Initial Set States

Let us suppose that the initial set $Q_{init} \subseteq Q$ is fixed. In this case, we have no interest in providing a controller for non-reachable from Q_{init} states.

A state $q' \in Q$ is *reachable* from the state q, if $q' = q$ or there exists a trajectory connecting them. The set of all reachable states from the state q is denoted by $\mathrm{Reach}_F(q)$. This definition could be naturally extended for a subset

Q' of the set of states Q: $\mathrm{Reach}_F(Q') = \cup_{q \in Q'} \mathrm{Reach}_F(q)$.

Definition 2.3.3. *For a given transition system $\Sigma = (Q, U, F)$, a safety specification $Q_S \subseteq Q$, and a fixed initial set $Q_{init} \subseteq Q$ a maximal states-lazy safety (MSLS) controller C^* is a safety controller satisfying the following properties*

1. $Q_{init} \cap \mathrm{Cont}(\Sigma, Q_S) \subseteq \mathrm{Dom}(C^*)$;
2. *for all states $q \in \mathrm{Dom}(C^*)$ the input $u \in C^*(q)$ if and and only if $F(q, u) \subseteq \mathrm{Cont}(\Sigma, Q_S)$.*
3. $\mathrm{Dom}(C^*) \subseteq \mathrm{Reach}_{F_{C^*}}(Q_{init} \cap \mathrm{Dom}(C^*))$;

An MSLS controller is a particular case of the maximal lazy safety controller introduced in [Girard et al., 2016]. It corresponds to a situation when all inputs have the same priorities. Since the MLS controller exists for any transition system and it is unique [Girard et al., 2016] an MSLS controller exists and is unique, as well. From Definition 2.3.2 it follows that all controllable safety states in Q_S, which are reachable from Q_{init} are in $\mathrm{Dom}(C^*)$ and this explain the term maximal in the name of C^*. At the same time, the controller C^* is called state-lazy since non-reachable from the initial set states are not in the domain. The MSLS controller C^* can be obtained from the maximal safety controller $\bar{C}$ by removing the states that are not reachable from initial states, but this is obviously not the most efficient way to find C^*. The main Algorithm 2.4 from Algorithm 2.1 by line 7, thereby restricting exploration to states, which are reachable from the initial set. Let us remark that if Q_{init} coincides with Q_S, then Algorithm 2.4 returns the maximal safety controller.

Lemma 2.3.1. *Let $Q_{zero} = \{q \in Q \mid p(q) = 0\}$. While running the Algorithm 2.4, the intersection $Q_{zero} \cap \mathrm{Cont}(\Sigma, Q_S)$ is always empty.*

Proof. Let us use induction to prove this fact. Before the loop 8-12, the set Q_{zero} consists only of unsafe (see line 6) and, as a consequence, uncontrollable states. In the loop, Q_{zero} updates only when some state $q \in B \subseteq Q$ gets at line 11 a prioryty zero. Supposing that $Q^i_{zero} \cap \mathrm{Cont}(\Sigma, Q_S) = \emptyset$ let us show that $Q^{i+1}_{zero} \cap \mathrm{Cont}(\Sigma, Q_S) = \emptyset$, where $Q^{i+1}_{zero} = Q^i_{zero} \cup \{q\}$. So, let us show, that $q \notin \mathrm{Cont}(\Sigma, Q_S)$. Since $p(q)$ got value 0 either $\mathrm{Enab}_F(q) = \emptyset$, or for all $u \in \mathrm{Enab}_F(q)$ there exists $q' \in F(q, u) \setminus \{q\}$ such that $p(q') = 0$, i.e $q' \in Q^i_{zero}$ and $q' \notin \mathrm{Cont}(\Sigma, Q_S)$. Hence, $\bar{C}(q)$ is empty and q is uncontrollable. □

Theorem 2.3.3. *Let C computed by Algorithm 2.4. Then, C is the MSLS controller.*

Proof. We start with safety. Since $\mathrm{Dom}(C)$ coincide with the set Q^R_E and all unsafe states are non-reachable we have $\mathrm{Dom}(C) \subseteq Q_S$. Moreover, from the definition of Reach operator we have that for all $q \in \mathrm{Dom}(C)$ and for all $u \in U$ it holds $F_p(q, u) \subseteq \mathrm{Dom}(C)$. Combining with the fact that for all $q \in \mathrm{Dom}(C)$ and for all $u \in \mathrm{Enab}_{F_p}(q)$ the set $F_p(q, u)$ coincides with $F(q, u)$, the second requirement from Definition 2.2.1 is satisfied.

Now, let us show that all properties of MSLS controller are satisfied (see Definition 2.3.3). Let $q \in \mathrm{Cont}(\Sigma, Q_S) \cap Q_{init}$. It is equivalent to $q \in Q_{init}$ and $q \in \mathrm{Cont}(\Sigma, Q_S)$. From where, using the Lemma 2.3.1, we get that $q \in Q_{init}$

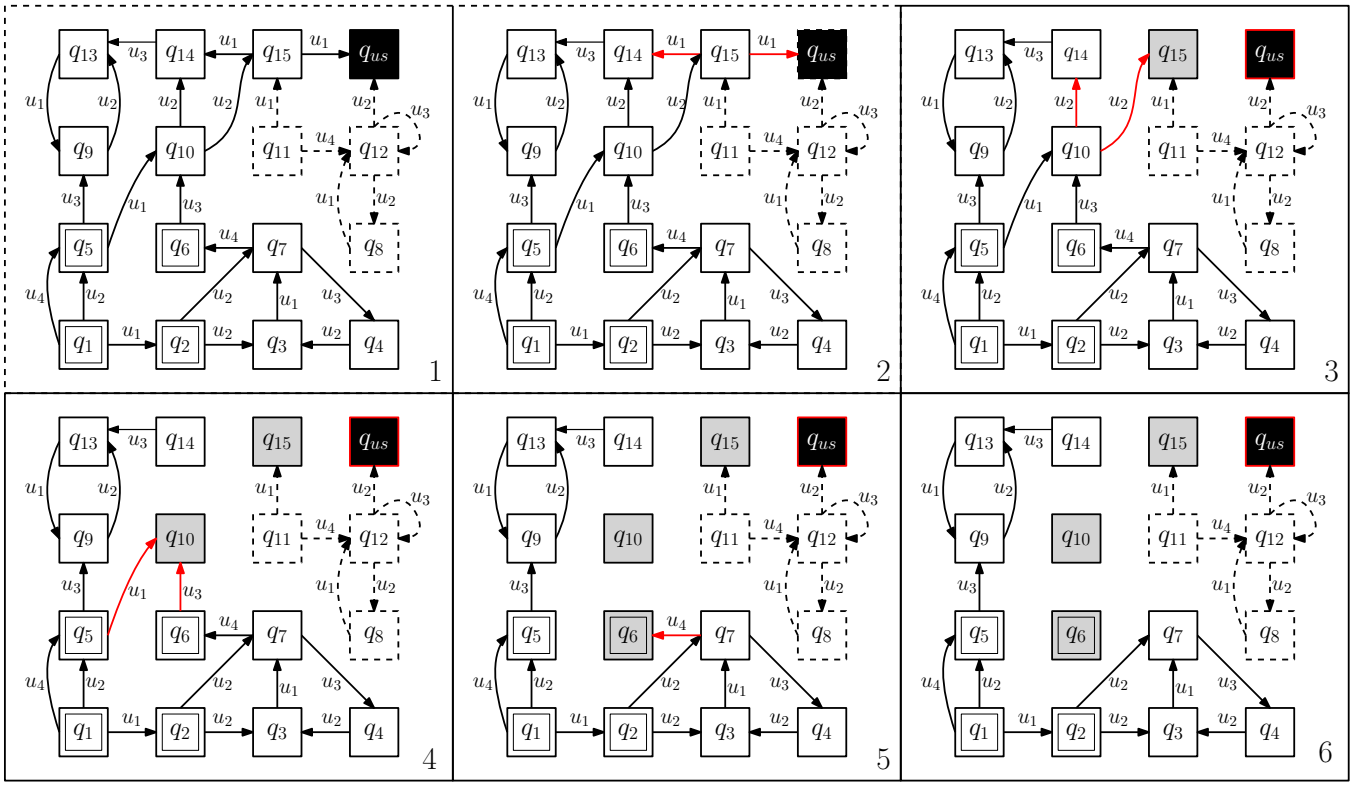


Figure 2.4: Illustration Algorithm 2.4 execution. Unsafe actions are red, blocking states are grey, and the initial states are marked with a double line. The computations are restricted to states, which reachable from the initial set. States which are non-reachable from the initial set are countered with a dashed line. Hence, states q_{11}, q_{12} and q_8 are not included in the domain of the maximal state lazy safety controller.

and $p(q) \neq 0$. Consequently, $q \in Q_E^R = \mathrm{Dom}(C)$ (see line 7,13), and the first property is satisfied. If a state $q \in \mathrm{Dom}(C^*)$ then the set $F(q,u) \subseteq \mathrm{Cont}(\Sigma, Q_S)$ for all $u \in C^*(q)$ since C^* is a safety controller. If for an input $u' \in U$ the set $F(q,u') \subseteq \mathrm{Cont}(\Sigma, Q_S)$ then there is no $q' \in F(q,u')$ such that $p(q') = 0$ and $u' \in \mathrm{Enab}_{F_p}(q)$ and we have the second property. The third property follows straightforwardly from line 7. □

Figure 2.4 provides an illustration of Algorithm 2.4. Assuming the set of initial states is known we restrict all computations to states, which are reachable from $Q_{init} = \{q_1, q_2, q_5, q_6\}$.

2.4 Conclusion

This Chapter structured the existing lazy safety approaches and provided three lazy synthesis algorithms applicable for a general class of non-deterministic transition systems. Algorithm 2.2 avoid exploration of states and transitions which are safely controllable a priory. Algorithm 2.3 explores the lower-priority actions only if the safety problem is unsolvable with higher-priority actions. Based on the incremental exploration of the transition system, Algorithm 2.4 restricts computations for reachable from initial set states only. Let us remark that Algorithms 2.2, 2.3 and 2.4 can be

easily combined in one in order to get as maximal gain in efficiency as possible. Later in the manuscript, we provide examples of applying these algorithms in abstraction-based control synthesis approaches. In Chapter 3, using adaptive time-sampling techniques, we construct abstractions where transitions only between neighboring states are allowed and then use the Algorithm 2.2 for efficient synthesis of the maximal safety controller. In Chapter 4, we combine Algorithms 2.3 and 2.4 to synthesise a maximal input-state lazy safety controller for a multi-scale symbolic model. In Chapter 5, we adopt ideas of Algorithms 2.2 and 2.3 to synthesize the maximal input-lazy safety controller for monotone transition systems. We also prove that for this particular class of systems, the maximal safety controller can be reconstructed from the maximal input-lazy safety controller without any computation.

Chapter 3

Lazy Symbolic Controller for Continuous-Time Systems Based on Safe Set Boundary Exploration

In this Chapter, we propose an abstraction-based approach to synthesize a safety controller solving Problem 1.1.1. The main idea consists in creating an abstraction with a particular geometrical structure where only transitions between neighboring states are allowed. Let us call such transition systems neighbor-linked. Though neighbor-linked abstractions often appear in path-finding problems [Hart et al., 1968], in the general case, symbolic models constructed for an arbitrary dynamic system do not have the desired property. Indeed, in the existing methods, [Mazo et al., 2010, Mouelhi et al., 2013, Wongpiromsarn et al., 2011], a time is usually sampled with a given frequency, and it is the local speed of the controlled dynamic system, which determines how far away from the initial symbolic state a transition ends. However, if all trajectories of the original plant are continuous, we can force the desired structure by interrupting transitions just before leaving the abstract state neighborhood. Related to event-triggered control approaches [Liu et al., 2014, Peng and Li, 2018] such adaptive time sampling technique is a contribution of this thesis.

As it was already briefly discussed in Chapter 2, we then can benefit from the neighbor-linked structure of the proposed abstraction and synthesize the maximal safety controller lazily by iteratively exploring only those states which border with unsafe or blocking states. The lazy synthesis is more efficient than the classical approach; moreover, it does not calculate the unexplored part of the abstraction. Unfortunately, using adaptive time sampling, we lose the flexibility to discretize time and state-space independently. But what if we can restrict exploration to boundary states even when arbitrary time sampling is used to get a symbolic model? Let us run Algorithm 3.2 for an arbitrary transition system. Of course, in this case, there is no guarantee that the synthesized controller is safe, but

it still can be refined to solve Problem 1.1.1. Indeed, when a closed-loop trajectory of the original system reaches a boundary state of the controllable domain, we have the abstract controller to push the system back towards the interior where any input, which is permissible for the original system, can be applied. In spirit, this idea is close to the Nagumo theorem result ([Blanchini, 1999]) and extreme aiming principle ([Subbotin, 1995]), but to the best of our knowledge, it has never been implemented within the abstraction-based control synthesis framework.

This Chapter is organized as follows. In Section 3.1 we introduce a novel adaptive time-sampling technique and use it to construct neighbor-linked symbolic models for the original system. The particular structure of the abstractions allows synthesizing the maximal safety controller lazily, benefiting from the information about a priori controllable states. We then extend the proposed approach towards arbitrary transition systems from Section 3.2 by introducing a novel control refinement scheme in Section 3.3. In Section 3.4 we consider an illustrative example (adaptive cruise control problem) to show the benefits of the approach.

3.1 Neighbor-Linked Abstractions

In Problem 1.1.1, we aim to synthesize a controller for a continuous-time control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ that maintains all closed-loop trajectories within a safety set $Y \subset \mathbb{R}^{n_x}$. Let us also remind that the dynamic of $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ is given by eq.(1.1): $\dot{x}(t) = f(x(t), u(t), w(t))$, $t \in \mathcal{T}, u(t) \in \mathcal{U} \subset \mathbb{R}^{n_u}, w(t) \in \mathcal{W} \subset \mathbb{R}^{n_w}$, where $\mathcal{U}, \mathcal{W}$ are compact sets. Let us suppose for this Chapter that set Y is a compact as well. To solve the Problem 1.1.1 we use abstraction-based synthesis techniques, and this section is devoted neighbor-linked abstractions for continuous dynamical systems. To enforce the abstraction's desired structure, we first introduce a novel time sampling technique allowing interrupt transitions before leaving an abstract state neighborhood. Then we define an information function that distinguishes boundary states of a set from the inner states and proves that at every iteration of lazy synthesis Algorithm 3.2 it is enough to explore only those states which border with the uncontrollable domain.

3.1.1 Adaptive Time Sampling as Way to Enforce the Desired Structure

Let us first abstract the original system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ with a transition system $\Sigma = (Q, U, F)$, which have a particular geometric structure: only transitions between neighboring states are allowed.

We first introduce a finite partitioning $Q_S = \{q_1, \ldots, q_n\}$ on an internal approximation S of the safety set Y such that $S = \cup_{i=1}^{n} q_i$, $q_i \cap q_j = \emptyset$ for all $i \neq j$ and define a set of abstract states Q, as follows $Q = Q_S \cup \{q_{us}\}$. Here $q_{us} = \mathbb{R}^{n_x} \setminus S$ is an unsafe state. We also formally define a neighborhood of a state $q \in Q$, as follows

$$N_A(q) = \{q' \in Q \setminus \{q\} \mid \mathrm{cl}(q) \cap q' \neq \emptyset \text{ or } q \cap \mathrm{cl}(q') \neq \emptyset\},$$

where $\mathrm{cl}(q)$ denotes a closure of a set $q \subset \mathbb{R}^{n_x}$. Thus, depending on the context, a symbol q represents an abstract

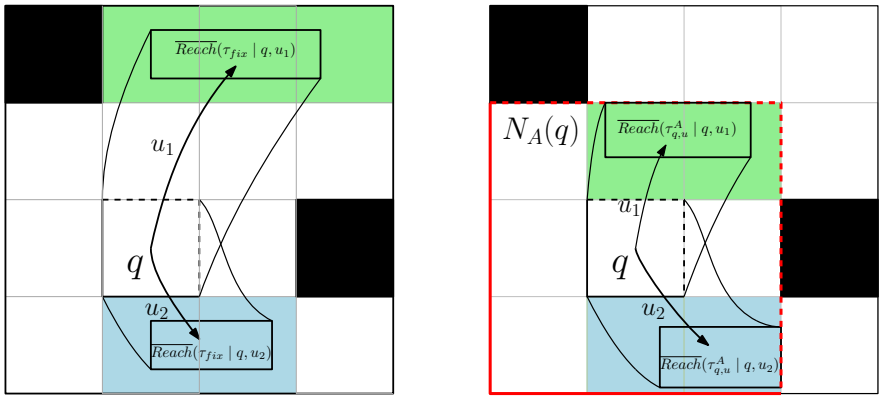


Figure 3.1: Illustration of the adaptive time sampling technique (right figure) compared to a fixed time sampling approach (left figure). Black cells represent unsafe states; green and blue cells represent $F(q, u_1)$ and $F(q, u_2)$ correspondingly. The red square represents the abstract state neighborhood $N_A(q)$. While in the left figure, transition duration is determined by pre-fixed time sampling parameter τ_{fix}, in the right figure, a transition stops just before leaving $N_A(q)$ (see eq. (3.2)).

state or a subset of the space $\mathbb{R}^{n_x}$, corresponding to this abstract state. Then for every $q \in Q_S$ we chose a finite number of admissible inputs $U_S(q) = \{u_1, \dots, u_m\} \subseteq \mathcal{U}$ and use reachability analysis techniques to compute corresponding transitions. Let $\mathrm{Reach}(t \mid q, u_S)$ be a reachable set

$$\mathrm{Reach}(t \mid q, u_S) = \big\{x \in \mathbb{R}^{n_x} \mid \exists\, x(0) \in q \text{ and } \exists\, \mathbf{w} \in \mathcal{L}^\infty([0,t], \mathcal{W}) \text{ such that } x_f(t \mid x(0), \mathbf{u}, \mathbf{w}) = x\big\}.$$

corresponding to an initial set q, a constant control function $\mathbf{u} : [0,t] \to u_S, u_S \in U_S(q)$ and admissible disturbances $\mathbf{w} \in \mathcal{L}(\mathcal{T}, \mathcal{W})$. We, then define a set $U = \cup_{q \in Q_S} U_S(q)$ and say that for every $q \in Q, u \in U$ transition $(q, u, q') \in F$ if and only if

$$q \in Q_S, u \in U_S(q), q' \in Q,\ q' \cap \mathrm{Reach}(\tau^A_{q,u} \mid q, u) \neq \emptyset, \tag{3.1}$$

and for all $t \in [0, \tau^A_{q,u}]$ the condition of a collision avoidance $\mathrm{Reach}(t \mid q, u) \cap (\mathbb{R}^{n_x} \setminus Y) = \emptyset$ is satisfied. Here, we use the adaptive time-sampling and a transition duration defined as $\tau^A_{q,u} = \min(\tau, \tau_{q,u} - \varepsilon)$, where τ is a given parameter which determines the maximal evolution time, while $\tau_{q,u}$ is a moment of time

$$\tau_{q,u} = \inf_{t \in [0, +\infty)} \Big\{ \mathrm{Reach}(t \mid q, u) \not\subseteq N_A(q) \Big\} \tag{3.2}$$

when the reachable set leaves the neighborhood of $N_A(q)$, we chose $\varepsilon < \tau_{q,u}$ arbitrary small to stop evolution just before leaving. In contrast, τ should be big enough since it only manages situations when a solution stucks within the box. Figure 3.1 illustrates the difference between fixed time sampling and adaptive time sampling techniques. Both subfigures show two transitions originating at a state q and corresponding to two different controls u_1 and u_2. However, transitions in the left figure have a pre-fixed duration τ_{fix}, while in the right, transitions interrupt just before leaving the abstract state neighborhood.

Algorithm 3.1: ClassicalSynthesisMSC(Σ, Q_S)

Input: $\Sigma = (Q, U, F)$ and a safe set Q_S
Output: Maximal Safety Controller C

1 **begin**
2 **for** $q \in Q_S$ **do**
3 $p(q) := 1$;
4 **for** $q \in Q \setminus Q_S$ **do**
5 $p(q) := 0$;
6 **repeat**
7 $Q_E := \{q \in Q \mid p(q) \neq 0\}$;
8 $B := \mathrm{Block}_{F_p}(Q_E)$;
9 **for** $q \in B$ **do**
10 $p(q) := p(q) - 1$;
11 **until** $B = \emptyset$;
12 **for** $q \in Q_E$ **do**
13 $C(q) := \mathrm{Enab}_{F_p}(q)$;
14 **for** $q \in Q \setminus Q_E$ **do**
15 $C(q) := \emptyset$;
16 **return** C;

Algorithm 3.2: LazySynthesisMSC(Σ, Q_S, I_{EB})

Input: $\Sigma = (Q, U, F)$, a safe set Q_S, an indicator I_{EB}
Output: Maximal Safety Controller C

1 **begin**
2 **for** $q \in Q_S$ **do**
3 $p(q) := 1$;
4 **for** $q \in Q \setminus Q_S$ **do**
5 $p(q) := 0$;
6 **repeat**
7 $Q_{EB} := \{q \in Q \mid p(q) \neq 0 \text{ and } I_{EB}(q) = 1\}$;
8 $B := \mathrm{Block}_{F_p}(Q_{EB})$;
9 **for** $q \in B$ **do**
10 $p(q) := p(q) - 1$;
11 **until** $B = \emptyset$;
12 **for** $q \in \{q \in Q \mid p(q) \neq 0\}$ **do**
13 $C(q) := \mathrm{Enab}_{F_p}(q)$;
14 **for** $q \in Q \setminus \{q \in Q \mid p(q) \neq 0\}$ **do**
15 $C(q) := \emptyset$;
16 **return** C;

When the dynamic of the original system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ is too complex to find exact attendance sets its over-approximations $\overline{\mathrm{Reach}}(\tau^A_{q,u} \mid q, u)$ are commonly used instead. Obviously, to get a whole symbolic model, one should estimate quite a lot of reachable sets and we recommend to use simple interval over-approximations [Meyer et al., 2021, Maidens and Arcak, 2014, Zamani et al., 2011, Reissig et al., 2017] trading accuracy for efficiency. Moreover, interval estimations match well with grid-aligned partitions, which are often used in symbolic model control [Mazo et al., 2010, Mouelhi et al., 2013, Wongpiromsarn et al., 2011]. A transition is safe if it corresponds to an action preventing the system from steering into a blocking or unsafe state. Otherwise, it is unsafe.

3.1.2 Maximal Safety Controller Synthesis: Inner States as a Source of Laziness

In this section, we aim to synthesize the maximal safety controller for the constructed transition system. I.e., we are looking for the maximal by inclusion controller C, ensuring that all closed-loop trajectories of $\Sigma = (Q, U, F)$ starting in $\mathrm{Dom}(C)$ are infinite and belong to the safe set Q_S.

To compute the desired controller one can iteratively remove all unsafe transitions from $\Sigma = (Q, U, F)$ until the abstraction stops changing [Tabuada, 2009]. Let a function $p\colon Q \to \{0, 1\}$ indicates whether a state $q \in Q$ is controllable ($p(q) = 1$) or not ($p(q) = 0$), and the reduced transition relation F_p is defined as follows: $(q, u, q') \in F_p$ if and only if $p(q) \neq 0$ and for all $q'' \in F(q, u)$ the equality $p(q'') \neq 0$ is satisfied. Then, Algorithm 3.1 returns the maximal safety controller. This statement was proved in Chapter 2, where the classical synthesis Algorithm 3.1 was referred to as Algorithm 2.1. However, since the abstraction $\Sigma = (Q, U, F)$ is neighbor-linked we can synthesize the controller more efficiently. Let the information function I_{EB} be defined by (3.3). Then, in opposite to Algorithm 3.1, Algorithm 3.2 explores only boundary states of a set $Q_E = \{q \in Q \mid p(q) \neq 0\}$ at every iteration of the loop 6-11.

Tacking into account that Algorithm 3.2 coincides with Algorithm 2.2 from Chapter 2 and it is repeated here to improve the readability of the text, let us now prove the correctness of the proposed lazy synthesis approach.

Theorem 3.1.1. *For a given transition system $\Sigma = (Q, U, F)$ and a safe set $Q_S = Q \setminus \{q_{us}\}$ let at every iteration of the loop 6-11 of Algorithm 3.2 for all $q \in Q_E = \{q \in Q \mid p(q) \neq 0\}$ the information function is defined as follows*

$$I_{EB}(q) = \begin{cases} 0 & \text{if } N_A(q) \subseteq Q_E \\ 1 & \text{if } N_A(q) \not\subseteq Q_E \end{cases} \tag{3.3}$$

Then the controller C returned by Algorithm 3.2 is the maximal safety controller for transition system $\Sigma = (Q, U, F)$ and a safe set $Q_S = Q \setminus \{q_{us}\}$.

Proof. Indeed, $F(q, U) \subseteq N_A(q)$ for any $q \in Q_E$. Consequently, at every iteration of the loop 6-11 of the Algorithm 3.2, the set $Q_{EB} = \{q \in Q_E \mid I_{EB}(q) = 1\}$ is an essential basis of $Q_E = \{q \in Q \mid p(q) \neq 0\}$ (see Definition 2.3.1) and by Theorem 2.3.1, C is the maximal safety controller. □

3.2 What if the Abstraction is not Neighbor-Linked?

In the previous section, we construct a neighbor-linked symbolic model for the original system. That allows us to use efficient Algorithm 3.2 instead of the classical synthesis approach Algorithm 3.1 for the maximal safety controller computation. However, with the transition relation defined in Section 3.1.1, we lost flexibility in scaling time sampling parameters independently from space sampling parameters. In this section, we consider a more general case. Let us define a transition relation $F^* \subseteq Q \times U \times Q$ as follows for every $q \in Q, u \in U$ a transition $(q, u, q') \in F^*$ if and only

$$q \in Q_S,\ u \in U_S(q),\ q' \in Q,\ q' \cap \overline{\mathrm{Reach}}(\tau^*_{q,u} \mid q, u) \neq \emptyset \tag{3.4}$$

and for all $t \in [0, \tau^*_{q,u})$ the collision avoidance condition $\overline{\mathrm{Reach}}(t \mid q, u) \cap (X \setminus S) = \emptyset$ is satisfied. In opposite to (3.1), in (3.4), we only assume that sampling parameter $\tau^*_{q,u}$ is greater than zero, and it is determined for any given $q \in Q, u \in U$. Such a definition does not put any requirements on a choice of time-sampling parameters permitting us to handle fixed ([Nilsson et al., 2016]), multi-scale ([Hsu et al., 2019, Girard et al., 2016]) or adaptive time-samplings described in the previous section in a unified way.

It is clear that if we use Algorithm 3.2 synthesize a controller C for the abstraction $\Sigma^*(Q, U, F^*)$, the safety set Q_S and the information function $I_{EB}(q)$ defined by (3.3), there is no guarantee that C is a safety controller. However, in the next section, we show that C can still be refined towards a continuous-time controller, which solves the original safety problem.

3.3 Controller Refinement for an Arbitrary Abstraction

Let C be a controller given by the Algorithm 3.2 for transition system $\Sigma^* = (Q, U, F^*)$ and a safe set $Q_S = Q \setminus \{q_{us}\}$, while the $I_{EB}(q)$ is defined as in (3.3). For every $q \in Q_A \setminus \mathrm{Dom}(C)$, we define $C^{dur}(q) = \emptyset$, while if $q \in \mathrm{Dom}(C)$ and $u \in C(q)$, then the pair $(u, \tau^*_{u,q}) \in C^{dur}(q)$. Hence, the controller C^{dur} store not only safe inputs, but a real duration of safe transitions. Let us now define a set of border points

$$Q_B = \mathrm{cl}\left(\{x \in \mathbb{R}^{n_x} \mid \exists\, q \in \mathrm{Dom}(C) \text{ such that } x \in q \text{ and } N_A(q) \not\subseteq \mathrm{Dom}(C)\}\right)$$

and a set of all internal points $Q_I = \mathrm{Dom}(C) \setminus Q_B$ correspondingly. We then define a quantizer, associating every border point $x \in Q_B$ with a state of transition system $Q^x(x) = \{q \in \mathrm{Dom}(C) \mid x \in \mathrm{cl}(q)\}$. If there are several states $q \subset Q_B$ such that x belongs to q, let $Q^x(x)$ returns any of them.

Now we are ready to introduce a controller refinement procedure. Let us consider the control input given for all $t \in [t_k, t_{k+1})$ by

$$\begin{cases} u(t,x) \in \mathcal{U} & \text{if } m_k = 0 \\ u(t,x) = u_k, & \text{if } m_k = 1 \end{cases} \tag{3.5}$$

where

$$\begin{cases} m_k(t,x) = 0, & \text{if } x(t_k) \in Q_I \\ m_k(t,x) = 1, & \text{if } x(t_k) \in Q_B \end{cases} \tag{3.6}$$

$$(u_k, \tau_k) \in C^{dur}(Q^x(x(t_k))), \text{ if } m_k(t,x) = 1 \tag{3.7}$$

and the sequence of instants $\{t_k\}$ is given by $t_0 = 0$ and

$$\begin{cases} t_{k+1} = \inf\{t > t_k \mid Q^x(x(t)) \in Q_B\} & \text{if } m_k = 0 \\ t_{k+1} = t_k + \tau_k & \text{if } m_k = 1 \end{cases} \tag{3.8}$$

Theorem 3.3.1. *For a given control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ all trajectories of closed-loop system* (1.1),(3.5)-(3.8) *starting from a $\mathrm{Dom}(C)$ at $t = 0$ stays within the set $S \subseteq Y$ for all $t \in \mathcal{T}$ no matter which disturbance $\mathbf{w} \in \mathcal{L}(\mathcal{T}, \mathcal{W})$ has been applied.*

Proof. For any initial condition $x(0) \in \mathrm{Dom}(C)$ and any disturbance $\mathbf{w} \in \mathcal{L}^\infty(\mathcal{T}, \mathcal{W})$ the closed-loop trajectory $\dot{x}(t) = f(x(t), u(t,x), w(t)),\ t \in T$ can not leave the $\mathrm{Dom}(C)$ without passing through the set Q_B, and for every state in Q_B there is a controller which brings us back to the $\mathrm{Dom}(C) \subseteq S \subseteq Y$ (see Algorithm 3.2). The Zeno behaviour is impossible since being in mode 0 at $t = s \in T$ we can either stay in mode 0 for all $t \in [s, +\infty)$ or switch to mode 1. If we have switched to mode 1 we spent there at least τ_k seconds before switching back to mode 0. □

Let us remark, that while starting inside the set Q_I in the mode 0 we can apply any admissible closed-loop control: e.g. $u(t,x) = v(t,x)$. A closed-loop control $v(t,x)$ is admissible if the solution of $\dot{x}(t) = f(x(t), v(t,x), w(t)),\ t \in T$

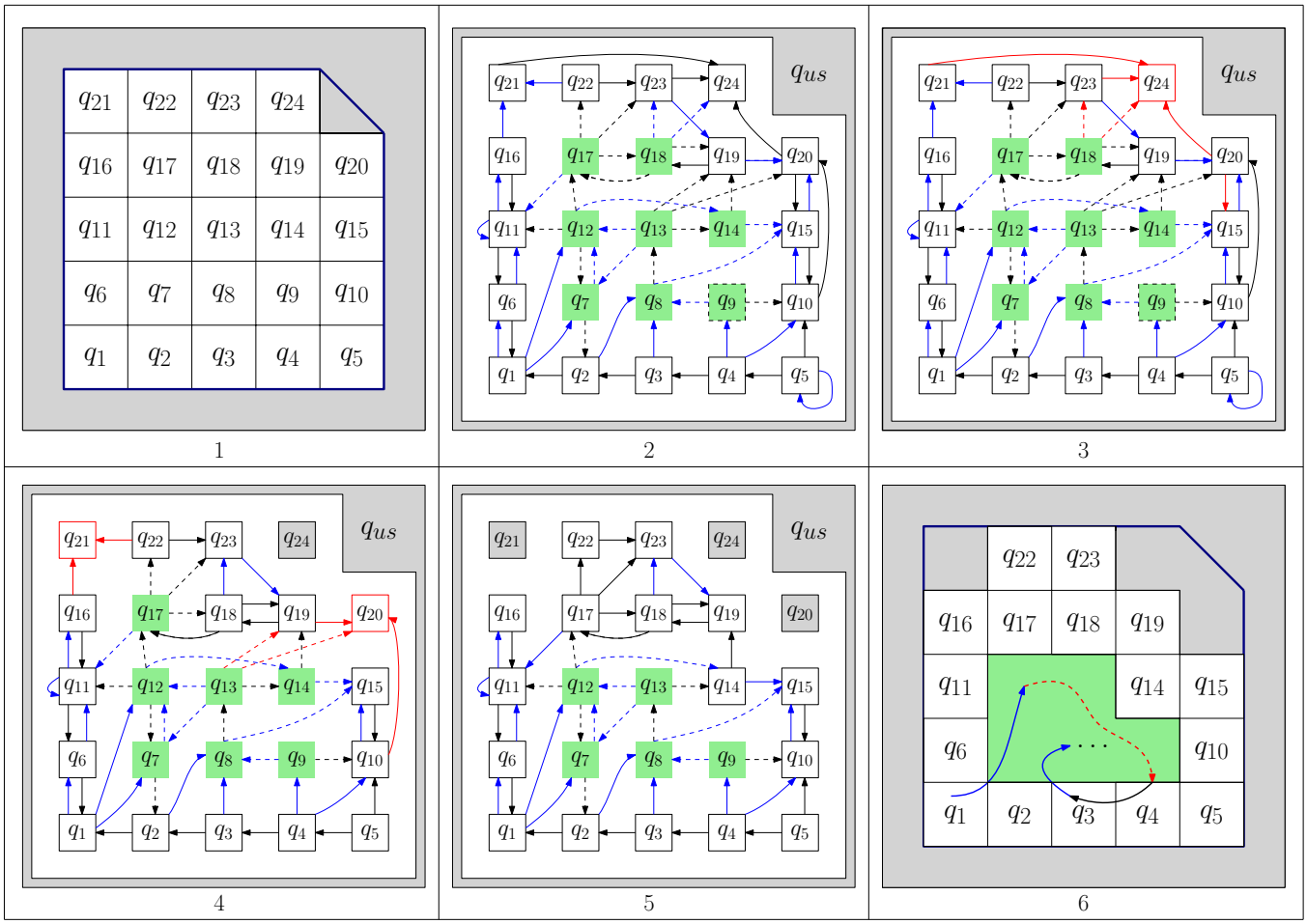


Figure 3.2: Illustration lazy synthesis based on boundary exploration idea. First figure illustrates the discretization of the original state space $\mathbb{R}^2$. The safe set Y is countered with blue; white cells are safe states; the unsafe state is filled with gray. Figures 2-5 illustrate the execution of Algorithm 3.2. States filled with white and green have priority 1; with grey - priority 0. White states belong to Q_{EB}. States contoured with red are blocking. The inputs set $U = \{u_1, u_2\}$. Actions corresponding to control u_1 are black, u_2 are blue. Transitions marked with red are unsafe. In opposite to normal line, the dashed line represents transitions that haven't been computed yet. Last figure illustrates a piece of trajectory of a closed-loop system (1.1),(3.5)-(3.8). White states belong to Q_B; green area corresponds to Q_I. Normal line represents mode 0, dashed line - mode 1.

exists and $v(t, x) \in U$ for all $t \in T$ and $x \in \mathbb{R}^{n_x}$.

Let us now illustrate the ideas discussed in this Chapter. In step 1 in Figure 3.2, we introduce a Cartesian partitioning on $\mathbb{R}^2$ and mark intervals $q_1, \ldots, q_{24}$ as safe abstract states, since they are included in the safe set Y. We then represent the region $\mathbb{R}^2 \setminus (\cup_{i=1}^{24} q_i)$ by an unsafe state q_{us}. On step 2, we initialize q_{us} with priority zero, while all the others with priority 1. Then we compute transitions only for states which are bordered with the unsafe region. Indeed, since all the internal states (marked with green) are not explored in the first iteration of the loop 6-11, there is no need to pre-compute the symbolic model for them. In step 3, the blocking state q_{24} gets priority zero. Hence, we should re-update the set Q_{EB} on step 4 and compute transitions corresponding to $q_{18} \in Q_{EB}$. We then re-explore all states in Q_{EB}. Since by the definition of F_p, we eliminate all transition steering into q_{24}, states

q_{21}, q_{20} become blocking, and we reduce their priority by one point. We then re-update the set Q_{EB} on step 5 and compute transitions originated in $q_{14}, q_{17} \in Q_{EB}$. On step 5, for any state in Q_{EB}, all transitions go either to another state in Q_{EB} or to an internal state, and we exit the loop 6-11. However, if the abstraction is not neighbor-linked, there is no guarantee that controller C synthesized with lines 12-15 is a safety controller for abstraction. Indeed, for internal states, we may have actions over-jumping the "controllable border." Still procedure (3.5)-(3.8) refines C to solve the original problem in the continuous-time domain. For any $x \in Q_B \subseteq \mathrm{Dom}(C)$, there is an abstract input steering the closed-loop trajectory either to Q_I or Q_B.h For any $x \in Q_I$, we can use any control value from $\mathcal{U}$ until the trajectory reaches the border Q_B again. There we should switch the mode. Certainly, the trajectory may stay in Q_I infinitely long, but this behavior is also safe. In step 6, we start from $x \in q_1 \in Q_B$. We apply the available abstract controller u_2, and it can bring us to state $q_{12} \in Q_I$ or $q_6 \in Q_B$. Suppose that we end up on $q_{12} \in Q_I$, then we can apply any controller until the trajectory runs into the state $q_4 \in Q_B$. There we apply abstract input u_1 and in $\tau^*_{q_4,u_1}$ reach $q_3 \in Q_{EB}$, where we use u_2 for $\tau^*_{q_3,u_2}$ seconds and so on.

3.4 Numerical Illustration: Adaptive Cruise Control

As an illustrative example, let us consider the adaptive cruise control problem for two vehicles moving along a straight line ([Darbha, 1997], [Nilsson et al., 2016]). Each vehicle is modeled as a point mass m with velocity changing according to the law

$$\dot{v}_i = \alpha(F_i, v_i) = (F_i - (f_0 + f_1 v_i + f_2 v_i^2))/m, \quad i = 1, 2.$$

In equation above, F_i represents a net action of braking and engine torque applied to the wheels, while the second term $(f_1 + f_2 v_i + f_3 v_i^2)$ describes aerodynamic and rolling resistance effects. The net force F_i is viewed as a control input for the following vehicle and as a disturbance for the lead one. It is assumed that $F_i \in [a, b]$, where $a = -0.3mg$, $b = 0.2mg$, g is a gravitational constant. Such bounds are consistent with non-emergency braking and acceleration.

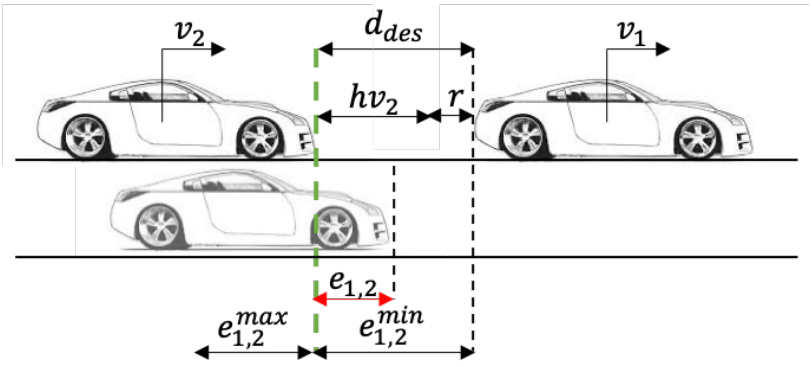


Figure 3.3: Illustration an adaptive cruise control problem configuration with a constant time headway spacing policy.

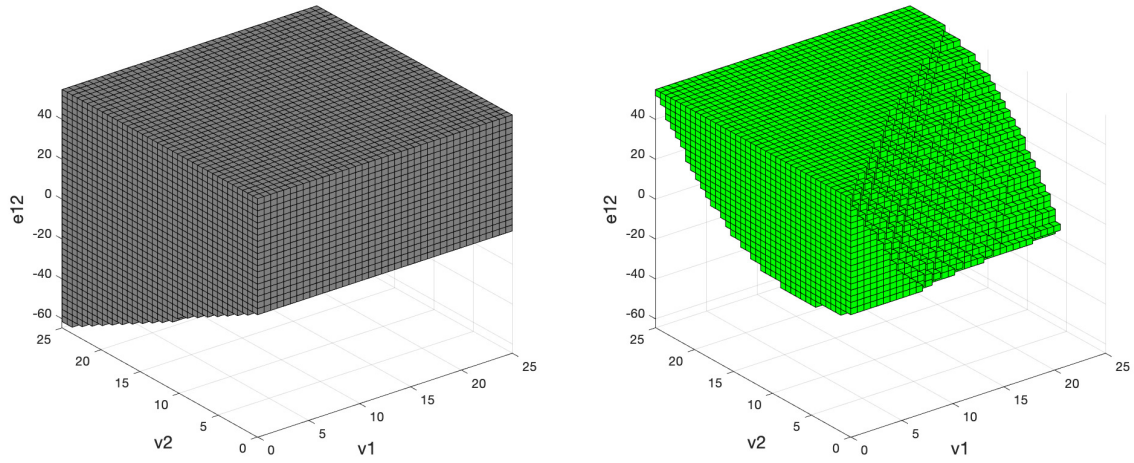


Figure 3.4: Simulation results. Left figure illustrates a safety specification. Right figure represents the domain of the safety controller computed by Algorithm 3.2.

First, we do a feedback linearization of the model for controlled vechile by introducing $F_{2,lin} = \alpha(F_2, v_2)$. Let us assume that the first car doesn't violate speed restrictions $v_1 \in [0, v^{max}]$. Since for any $v_2 \in [0, v^{max}]$, zero belongs to $[\alpha(a, v_2), \alpha(b, v_2)]$ we can always choose a suitable control law to do the same for the second car.

Then, we chose a feedback stabilizer as a control law $F_{2,lin} = u_1 + u_2(v_1 - v_2) + u_3 e_{1,2}$. Here $u_1 \in \mathbb{R}, u_2, u_3 \geq 0$. The deviation $e_{1,2}$ from the desirable distance between the cars changes according to the following equation

$$\dot{e}_{1,2} = (v_1 - v_2) - \dot{d}_{des} \quad \text{where } d_{des} = hv_2 + r,\ h > 0.$$

Hence, the constant time headway spacing policy is considered [Darbha, 1997] (see Figure 3.3 for an illustration).

Finally, the system dynamic is described as follows

$$\begin{aligned} \dot{v}_1 &= (F_1 - f_0 - f_1 v_1 - f_2 v_1^2)/m \\ \dot{v}_2 &= \beta(u_1 + u_2(v_1 - v_2) + u_3 e_{1,2}, v_2) \\ \dot{e}_{12} &= v_1 - v_2 - h\dot{v}_2 \end{aligned} \tag{3.9}$$

where

$$\beta(z, v) = \begin{cases} z, \text{ if } v \in (0, v^{max}) \\ \max(z, 0), \text{ if } v = 0 \\ \min(z, 0), \text{ if } v = v^{max} \end{cases}$$

Varying the new control parameters u_1, u_2, u_3 we want to keep the error $e_{1,2}$ in a range $[-hv_2 - r, e_{1,2}^{max}]$, while ensuring that $u_1 + u_2(v_1 - v_2) + u_3 e_{1,2} \in [\alpha(a, v_2), \alpha(b, v_2)]$. To construct a symbolic model we introduce on a set

Table 3.1: Vehicle and safety parameters. The values are taken from [Darbha, 1997, Saoud et al., 2020].

Parameter	Value	Unit	Parameter	Value	Unit
m	1370	Kg	v^{max}	25	m/s
f_0	51.0709	N	h	2	s
f_1	0.3494	Ns/m	r	2.5	m
f_2	0.4161	Ns^2/m^2	$e_{1,2}^{max}$	55	m

$Y = [0, v^{max}] \times [0, v^{max}] \times [-hv^{max} - r, e_{1,2}^{max}]$ a uniform Cartesian partition. The number of intervals in every direction is described by a given parameter n_x^p. Intervals on the border are flat, while internal intervals whether semi-closed from the right side or open if they are next to the right border. For example, in first direction we have $\{0\}, (0, \eta], (\eta, 2\eta], \dots, (v^{max} - \eta, v^{max}), \{v^{max}\}$, where $\eta = v^{max}/n_x^p(1)$. For every state belonging to a safe set Y, we chose n_u^p different admissible inputs $u = [u_1, u_2, u_3]$, ensuring that $F_{2,lin} \in [\alpha(a, v_2), \alpha(b, v_2)]$, and apply them τ, $\tau/2, \tau/4$ or $\tau/8$ seconds.

To compute the symbolic model we use interval over-approximations instead of reachable sets. Let us use notation $x = (v_1, v_2, e_{12})$. The system (3.9) is a mixed-monotone (see Appendix A) and hence for any given state $q = [\underline{x}^q, \overline{x}^q]$, any given constant control function $\mathbf{u}\colon [0, t] \to u$ the following inclusion holds for a reachable set $\mathrm{Reach}(t \mid q, u)$ robust to all admissible disturbances $\mathbf{F_1} \in \mathcal{L}(\mathcal{T}, [a, b])$

$$\mathrm{Reach}(t \mid q, u) \subseteq [\underline{x_g}(t \mid [\underline{x}^q, \overline{x}^q], \mathbf{u}, \mathbf{F_1}), \overline{x_g}(t \mid [\overline{x}^q, \underline{x}^q], \mathbf{u}, \mathbf{F_1})].$$

In the equation above, trajectories $\underline{x_g}(t \mid [\underline{x}^q, \overline{x}^q], \mathbf{u}, \mathbf{F_1}), \overline{x_g}(t \mid [\overline{x}^q, \underline{x}^0], \mathbf{u}, \mathbf{F_1})$ are correspondingly first three and second three components of the solution of the following system

$$\begin{aligned} \dot{\overline{x}}_i &= g_i(\zeta_i(\overline{x}, \underline{x}), [u, u], [b, a]), \quad \overline{x}_i(0) = \overline{x}_i^q \\ \dot{\underline{x}}_i &= g_i(\zeta_i(\underline{x}, \overline{x}), [u, u], [a, b]), \quad \underline{x}_i(0) = \underline{x}_i^q \end{aligned} \qquad \text{where } (\zeta_i(y, z))_j = \begin{cases} y_j, j < 3 \\ z_j, j > 3,\ j \neq 3 + i \\ y_i, j = 3 + i \end{cases} \quad i = \overline{1,3}, j = \overline{1,6} \tag{3.10}$$

Here the tight decomposition function $g : \mathbb{R}^6 \times (u, u) \times [a, b]^2 \to \mathbb{R}^3$ is defined as follows,

$$\begin{aligned} g_1([y; z], [u; u], [F_1^y; F_1^z]) &= (F_1^y - f_0 - f_1 v_1^z - f_2 (v_1^z)^2)/m \\ g_2([y; z], [u; u], [F_1^y; F_1^z]) &= \beta(u_1 + u_2(v_1^y - v_2^y) + u_3 e_{1,2}^y, v_2^y) \\ g_3([y; z], [u; u], [F_1^y; F_1^z]) &= \begin{cases} (1 - h * u_2)(v_1^y - v_2^z) - h u_3 e_{1,2}^y - h u_1, & \text{if } 1 - h u_2 \geq 0 \\ (1 - h * u_2)(v_1^z - v_2^y) - h u_3 e_{1,2}^y - h u_1, & \text{otherwise} \end{cases} \end{aligned}$$

where $y = [v_1^y, v_2^y, e_{1,2}^y],\ z = [v_1^z, v_2^z, e_{1,2}^z]$. Hence, to find a reachable set over approximation it is enough to solve 6 differential equations (3.10). Let is remark that collision avoidance condition $\mathrm{Reach}(t \mid q, u) \cap (\mathbb{R}^3 \setminus Y) = \emptyset$ can be easily verified, since as soon as one of the corners $\underline{x}$ or $\overline{x}$ violates the safety restrictions we can interrupt ODE

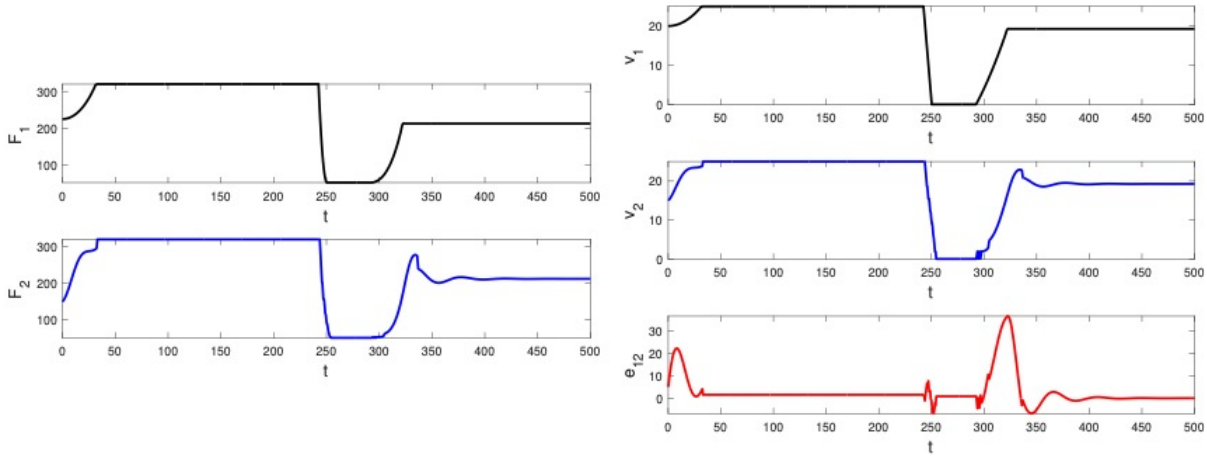


Figure 3.5: Simulation results. Left figures represent a disturbance realization F_1 and a safety control F_2. Right figures represent a corresponding closed-loop trajectory of the dynamic system (3.9), starting at $[20, 15, 5]$

solver and mark the transition as unsafe.

Setting $n_x^p = [41, 41, 41],\ n_u^p = 18,\ \tau = 1$ we use the Algorithm 1 and Algorithm 2 to compute a controllers C_1 and C_2 for the abstraction. In our particular example, we got that the controllable domains $\mathrm{Dom}(C_1)$ and $\mathrm{Dom}(C_2)$ coincide. So, both of them are represented on Figure 3.4 (right), while Figure 3.4 (left) illustrate the safe specification in the abstract domain. However, our approach, with run-time equals to 70.3 min, is 2.58 times faster than the classical one since it explores 19574 less states. In the Figure 3.5, a closed-loop trajectory simulated $500s$ for a given disturbance is also shown. The closed-loop trajectory satisfies the safety restriction and shows a nice behavior in terms of stability. The values of parameters shown in Table 3.1 are taken from [Saoud et al., 2020]. In the following, the implementations has been done in MATLAB, Processor Intel Core i7-8700, 3.20 Hg, RAM 16 GB.

3.5 Conclusion

In this Chapter, we introduced a novel lazy synthesis algorithm for a finite transition system with a safety specification. The main idea was to iteratively explore only the boundary states of the controllable domain, supposing that internal states are safely controllable a priory. The worst-case complexity of the proposed algorithm coincides with the complexity of the classical brute-force exploration ([Tabuada, 2009]), but our approach is more efficient in practice. Indeed, if a controllable domain for the abstract controller is non-empty, then we have a computational gain since we don't explore internal states. Moreover, real-time implementation of a closed-loop controller for the original continuous system is more memory efficient since the information for the internal states is not stored. To illustrate the benefits of the proposed approach, we considered an adaptive cruise control problem with a constant headway spacing policy.

Chapter 4

Lazy Safety Controller Synthesis with Multiscale Adaptive-Sampling Abstractions

As a finite abstraction for an infinite system, the symbolic model is commonly obtained by discretizing time, state, and input spaces. The smaller the sampling parameters are, the better abstractions mimic the behavior of the concrete system. Consequently, with finer symbolic models, there is more likely a solution for the abstract control problem when there is a solution for the original one. However, dealing with accurate abstractions costs a price. First, increasing the number of states and inputs in the symbolic model increases the number of transitions to be computed. Second, the complexity of discrete synthesis algorithms essentially depends on the size of symbolic models. Third, abstraction-based controllers are implemented as look-up tables, and the memory requirements for storing them are too high when there are too many elements in the abstraction. In this Chapter, we trade-off between accuracy and efficiency and focus on approaches iteratively refining symbolic models when synthesis with coarser abstractions is failed [Gol et al., 2013, Girard et al., 2016, Hsu et al., 2019, Li and Liu, 2018b].

To refine an abstraction, one can decrease the duration of transitions [Gonzalez et al., 2010] or precise the space partitioning [Gol et al., 2013, Li and Liu, 2018b, Nilson et al., 2017] or both [Girard et al., 2016, Hsu et al., 2019]. In the last case, one may reduce time and state discretization parameters synchronously, supposing that the duration of transitions, which start from the initial states with the same size, coincide [Hsu et al., 2018]. However, there is a common trend in the community [Hsu et al., 2019, Girard et al., 2016, Nilson et al., 2017] to permit for the same initial state transitions with different duration in order to improve the flexibility of abstractions. Following this trend, we consider in this Chapter abstractions allowing multiple transition duration. For such abstractions, it makes sense to synthesize the maximal inputs-lazy safety controller prioritizing those transitions which are

longer [Girard et al., 2016, Nilson et al., 2017]. What is about state-space discretization let us refine abstraction locally [Girard et al., 2016, Hsu et al., 2019, Nilson et al., 2017, Li and Liu, 2018a] since it is more efficient than global refinement [Hsu et al., 2018]. Indeed, usually, one should be accurate only next to the obstacles or in the regions where the system dynamics change too fast. To react even faster to the local speed of the dynamical system, we propose to constrain the duration of the transitions by a state neighborhood that should contain the reachable set, as opposed to a predetermined duration [Gol et al., 2013, Girard et al., 2016, Hsu et al., 2019, Li and Liu, 2018b]. Hence, we again use adaptive time sampling described in the previous Chapter. However, this Chapter uses neighborhoods of different sizes to implement the multiple time duration idea. In addition to abstraction refining, we propose to explore the symbolic model forwardly and thus restrict the controller synthesis computations to states that are reachable from the initial set only. This idea was proposed in ([Girard et al., 2016]), but their approach is restricted to deterministic abstractions. The algorithm presented in this Chapter applies to non-deterministic symbolic models as well.

This Chapter again provides an abstraction-based synthesis approach to solving the Problem 1.1.1. However, unlike the previous Chapter, the initial set X_0 is known, and multi-scale abstractions are considered. In section 4.1, we construct a multilayered abstraction with multi-scale adaptive time sampling. In section 4.2.1, we introduce a partial order on the input space, prioritizing transitions with longer duration. We also enrich the input space by an auxiliary input up, allowing switching from a finer level of the abstraction to a coarser one and give this input the highest priority. We then provide an efficient algorithm for computing the maximal input-state safety controller based on two ideas. To explore inputs with a lower priority only if we failed to find a safe input with higher priority and do not deal with non-reachable from initial set states. The proposed algorithm is implemented as a simple combination of Algorithm 2.3 and Algorithm 2.4. In subsection 4.2.1 we show that a multilayered and an adaptive grid are equivalent from the perspective of control synthesis purposes, but the adaptive grid is less time/memory consumpting. Section 4.3 refines safety controller computed for the abstraction to the continuous-time safety controller, solving the Problem 1.1.1. In section 4.4, we illustrate the benefits of the proposed approach synthesizing a safety controller maintaining the temperature on the building in a comfortable range for a person.

4.1 Multilayered Abstractions with Multi-scale Adaptive-Time Sampling

In Problem 1.1.1, we aim to synthesize a controller for a continuous-time control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ that maintains all closed-loop trajectories starting in the initial set $X_0 \subset Y$ within a safety set $Y \subset \mathbb{R}^{n_x}$. Let us also remind that the dynamic of $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ is given by eq.(1.1):

$$\dot{x}(t) = f(x(t), u(t), w(t)), \quad t \in \mathcal{T},\ u(t) \in \mathcal{U} \subset \mathbb{R}^{n_u},\ w(t) \in \mathcal{W} \subset \mathbb{R}^{n_w},$$

where $\mathcal{U}, \mathcal{W}$ are compact sets. Let us suppose for this Chapter that the set Y is a compact as well. To solve the Problem 1.1.1 we use abstraction-based synthesis techniques, and this section is devoted to multilayered abstraction with multi-scale time sampling for continuous dynamical systems. We first abstract the state space of the concrete system with several uniform Cartesian partitions embedded one into another. Then for every abstract state, we allow the same abstract controllers. To define a transition relation, we use adaptive time sampling techniques introduced in the previous chapter: the duration of the transitions is constrained by state intervals that should contain the reachable set.

Let us start the construction of a symbolic model for the original plant $\Sigma_f = (T, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ by discretizing the state space $\mathbb{R}^{n_x}$. Let $X = [\underline{x}, \overline{x})$ be an n-dimensional interval including a safe set Y. Given a number $L > 0$, and a state space sampling parameter $m_x > 0,\ m_x \in \mathbb{N}^{n_x}$ we introduce for every $l = 1, \ldots, L$ a uniform partition

$$Q_l = \{q \in 2^X \mid \exists z \in \mathbb{Z}^n \text{ s.t. } q = [\underline{x} + z * \eta_l, \underline{x} + (z+1) * \eta_l)\} \tag{4.1}$$

where $\eta_l = (\overline{x} - \underline{x})/(2^{l-1} m_x)$ and we associate every state $x \in X$ with a unique cell $q \in Q_l$ such that $x \in q$. We also define for every $q \in Q_l$, the center of the interval $q_c = \underline{x} + (z + \frac{1}{2}) * \eta_l$. Depending on the context, we regard an element $q \in Q_l$ either as an atomic symbol, representing an infinite number of states from X, or as a subset of X. It is obvious from the definition that for any $q_i, q_j \in Q_l,\ q_i \neq q_j$ the intersection $q_i \cap q_j = \emptyset$ and that $X = \cup_{q \in Q_l} q$.

Lemma 4.1.1. *For any $l = \{1, \ldots, L-1\}$ and for any $q \in Q_l$ there exists a unique set a states $\{q_i\}_{i=1}^{2^n}, q_i \in Q_{l+1}$ such that $q = \cup_{i=1}^{2^n} q_i$.*

Regarding the safety specification, an element $q \in Q_l$ is a safe state if and only if $q \subseteq Y$, an initial state if and only if $q \cap X_0 \neq \emptyset$. Let us denote by $Q_{l,S}$ a set of all safe states in Q_l, and by $Q_{l,init}$ a set of all initial states. We then combine all these partitions into a multilayered grid $Q = \cup_{l=1}^{L} Q_l$ and use it as a set of states for the multilayered abstraction. Obviously, the layer Q_1 is the coarsest layer of Q and Q_L is the finest one (see eq. (4.1)). From Lemma 4.1.1 it also follows that the partitionings $Q_1, \ldots, Q_L$ are embedded one into another.

In the next step, a control set $\mathcal{U}$ is approximated by its finite subset $U_\mu \subseteq \mathcal{U}$. For every $q \in Q_l, u_\mu \in U_\mu$ we consider the set of all reachable states at the time $t \in T$:

$$\mathrm{Reach}(t \mid q, u_\mu) = \{x \in \mathbb{R}^{n_x} \mid \exists\, x(0) \in q \text{ and } \exists\, \mathbf{w} \in \mathcal{L}^\infty([0,t], \mathcal{W}) \text{ such that } x_f(t \mid x(0), \mathbf{u}, \mathbf{w}) = x\},$$

corresponding to an initial set q, a constant control function $\mathbf{u} : [0,t] \to u_\mu$ and all admissible disturbances $\mathbf{w}$. Since the exact computation of reach set is rarely possible, an over-approximation $\overline{\mathrm{Reach}}(t \mid q, u_\mu)$ is commonly used to built symbolic models. Let us remark that we use the same inputs for all states for simplicity, and extension towards the case when for every state $q \in Q_l$ its proper set of abstract inputs $U_\mu(q)$ is defined is always possible.

In this Chapter, a symbolic model with multi-scale time sampling is considered, which means that for every given

state $q \in Q$ and control $u_\mu \in U_\mu$ we construct n_p transitions with different duration. Let us also introduce an input up, which allows switching from the current layer to the previous (coarser) one and define a function $l : Q \to \{1, \dots, L\}$ returning for any $q \in Q$ the index of the layer to which q belongs. We now ready to define a transition relation F for every available input in the set $U = (U_\mu \times \{1, \dots, n_p\}) \cup \{up\}$.

- for any $q' \in Q \setminus Q_1$ the transition $(q', up, q) \in F$ if and only if $q \in Q_{l(q)-1}$ and $q' \subset q$. See Fig.4.1 (left).
- for any $q \in Q$ and any $(u_\mu, j) \in U_\mu \times \{1, \dots, n_p\}$ the transition $(q, (u_\mu, j), q') \in F$ if and only if

$$q' \in Q_L,\ q' \cap \overline{\mathrm{Reach}}(\overline{\tau}_q^{u_\mu, j} \mid q, u_\mu) \neq \emptyset, \tag{4.2}$$

  and the condition of a collision avoidance

$$\overline{\mathrm{Reach}}(t \mid q, u_\mu) \cap (X \setminus Y) = \emptyset, \quad t \in [0, \overline{\tau}_q^{u_\mu, j}] \tag{4.3}$$

  is satisfied. Here $\overline{\tau}_q^{u_\mu, j} = \min(\tau_l, \tau_q^{u_\mu, j} - \varepsilon)$, where τ_l is a given parameter which determines the maximal evolution time allowed at this layer, while $\tau_q^{u_\mu, j}$ is a moment in time given by

$$\tau_q^{u_\mu, j} = \inf_{t \in [0, +\infty)} \left\{ \overline{\mathrm{Reach}}(t \mid q, u_\mu) \not\subset \left[q_c - (j + 1/2) * \eta_l, q_c + (j + 1/2) * \eta_l \right) \right\} \tag{4.4}$$

  when the over-approximation of a reachable set leaves the interval with a radius $(j + 1/2)\eta_l$ and a center in q_c. We chose $\varepsilon < \tau_q^{u_\mu, j}$ arbitrary small to stop evolution just before leaving, while τ_l should be big enough since it serves only to manage situations where a solution is stuck within the box. In the Fig.4.1 there is illustration of the idea.

Such a definition of a transition duration using adaptive time-sampling is a contribution of this manuscript. Instead of using prescribed time sampling parameters ([Girard et al., 2016], [Hsu et al., 2019]), this approach allows us to control, approximately, where symbolic transitions finish and better analyze the behavior of the system. At the same time, the collision avoidance condition enables to eliminate solutions, which start and end in the safe set but passes some unsafe regions during the evolution. However, since the intersection of the reachable set with the grid is checked at every instant of time, the authors recommend using simple interval over-approximations [Meyer et al., 2021]), trading accuracy for simplicity of implementation. We also finish any non-up transition at the finest layer to be more flexible while moving close to the obstacles.

Hence, in this section, we constructed a multilayered and multi-scale symbolic model $\Sigma = (Q, U, F)$ which mimics the dynamic of the original system. In the next section, we will synthesize safety controller for the abstraction, while supposing that specification is given by the safe set $Q_S = \cup_{l=1}^{L} Q_{l,S}$ and the initial set $Q_{init} = Q_{L,init}$ is also known.

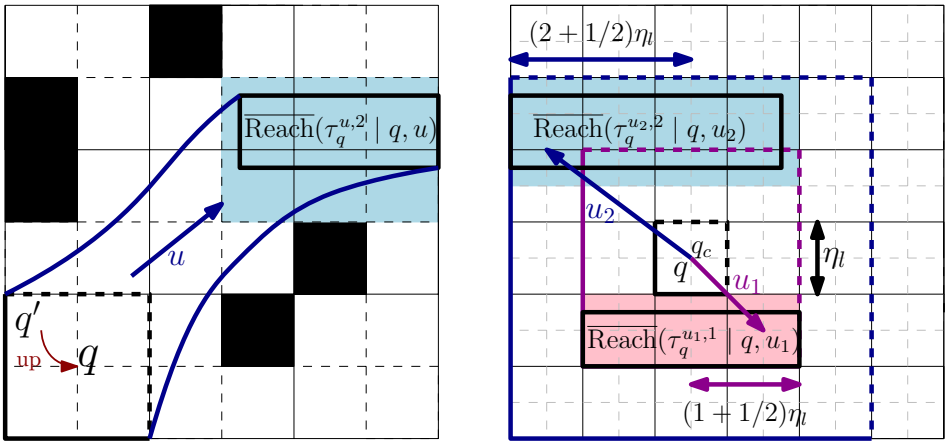


Figure 4.1: Illustration of transition relation on a 2-layered grid. Left figure illustrates (4.2) with a transition $F(q', up) = q$, and (4.3) with a transition $F(q, u)$. Unsafe states are marked with black. Right figure illustrates adaptive time-sampling techniques (see (4.4)). Transitions with two different duration are shown: for a control u_1 we stop before leaving an interval with a radius 1, while for a control u_2 – an interval with a radius 2.

4.2 Synthesis of Maximal Input-State Lazy Safety Controller

The less restrictive safety controller one can provide for $\Sigma = (Q, U, F)$ is the maximal safety controller $\bar{C}$. Still, its computation is too labor-intensive for many real-world problems. Indeed, according to the classical synthesis approach [Tabuada, 2009] one should first pre-compute the whole symbolic model and then iteratively remove all unsafe transitions from $\Sigma = (Q, U, F)$ until the abstraction stops changing. However, the idea of working with a multi-scale abstraction is to explore (and to compute) for a state q transitions with a shorter duration only if q is uncontrollable with longer duration actions. We also do not want to compute abstraction for states embedded in q if it is controllable. Moreover, since Q_{init} is known, it is reasonable to provide a controller only for states that are reachable from the initial set. Let us relax the maximality requirement for a desirable controller. Let the input set U be equipped with the following partial order $\preceq_U$

- for all $(u_\mu, j_1), (u_\mu, j_2) \in U_\mu \times \{1, \dots, n_p\}$ we say $(u_\mu, j_1) \prec_U (u'_\mu, j_2)$ if and only if $j_1 < j_2$.
- for any $(u_\mu, j) \in U_\mu \times \{1, \dots, n_p\}$ the following holds $(u_\mu, j) \prec_U up$.

I.e., transitions with a longer duration are preferred to transitions with a shorter one, and control up has the highest priority. We then aim to compute for $\Sigma = (Q, U, F)$ the maximal lazy safety (MLS) controller C^*. Where the term maximal refers to the fact that all safety controllable initial states should be in $\text{Dom}(C^*)$, and if the controller enables an input, it should also enable all inputs which have the same priority and preserve safety. The term lazy refers to the fact that when several inputs can preserve safety, the controller should enable only inputs with the highest priority. Intuitively, C^* can be obtained from $\bar{C}$ by keeping for every state only those enabled actions, which have a higher priority, and by removing the states that are not reachable from initial states. Certainly, it is not the best way to find C^*. In [Girard et al., 2016] the authors propose a more efficient synthesis procedure, but their result is

restricted to the class of deterministic transition systems. Algorithm 4.2 provided below overcomes this limitation.

Algorithm 4.1: ClassicalSynthesisMSC(Σ, Q_S)

Input: $\Sigma = (Q, U, F)$ and a safe set Q_S
Output: Maximal Safety Controller C

begin
  for $q \in Q_S$ **do**
    $p(q) := 1$;
  for $q \in Q \setminus Q_S$ **do**
    $p(q) := 0$;
  repeat
    $Q_E := \{q \in Q \mid p(q) \neq 0\}$;
    $B := \mathrm{Block}_{F_p}(Q_E)$;
    for $q \in B$ **do**
      $p(q) := p(q) - 1$;
  until $B = \emptyset$;
  for $q \in Q_E$ **do**
    $C(q) := \mathrm{Enab}_{F_p}(q)$;
  for $q \in Q \setminus Q_E$ **do**
    $C(q) := \emptyset$;
  return C;

Algorithm 4.2: MLSC(Σ, Q_S, Q_{init})

Input: $\Sigma = (Q, U, F)$, a safe set Q_S, an initial set Q_{init}
Output: MSLS controller C.

begin
  for $q \in Q_S$ **do**
    $p(q) := N$;
  for $q \in Q \setminus Q_S$ **do**
    $p(q) := 0$;
  repeat
    $Q_E^R := \mathrm{Reach}_{F_{p,\preceq_U}}(\{q \in Q_{init} \mid p(q) \neq 0\})$;
    $B := \mathrm{Block}_{F_{p,\preceq_U}}(Q_E^R)$;
    for $q \in B$ **do**
      $p(q) := p(q) - 1$;
  until $B = \emptyset$;
  for $q \in Q_E^R$ **do**
    $C(q) := \mathrm{Enab}_{F_{p,\preceq_U}}(q)$;
  for $q \in Q \setminus Q_E^R$ **do**
    $C(q) := \emptyset$;
  return C;

Theorem 4.2.1. *For a given transition system $\Sigma = (Q, U, F)$, a safe set Q_S and an initial set Q_I, let C^* be a controller computed by Algorithm 4.2. Then, C^* is a safety controller, satisfying the following properties:*

1. $Q_{init} \cap \mathrm{Cont}(\Sigma, Q_S) \subseteq \mathrm{Dom}(C^*)$;
2. $\mathrm{Dom}(C^*) \subseteq \mathrm{Reach}_{F_{C^*}}(Q_{init} \cap \mathrm{Dom}(C^*))$;
3. *for all states* $q \in \mathrm{Dom}(C^*)$:
   (a) *if* $u \in C^*(q)$ *then for all* $u' \in \mathrm{Enab}_F(q)$ *such that* $u' \simeq u$, *it holds that* $u' \in C^*(q)$ *if and only if* $F(q, u') \subseteq \mathrm{Cont}(\Sigma, Q_S)$;
   (b) *if* $u \in C^*(q)$, *then for all* $u' \in \mathrm{Enab}_F(q)$ *with* $u \prec u'$, *it holds that* $F(q, u') \cap \mathrm{Cont}(\Sigma, Q_S) \neq F(q, u')$.

Proof. Indeed, Algorithm 4.2 is a simple merge of Algorithm 2.3 and Algorithm 2.4. Consequently, the controller synthesised by Algorithm 4.2 inherits properties from MILS and MSLS controllers. □

Similar to Figures 2.3 and 2.4 from Chapter 2, Figure 4.2 illustrates the execution of Algorithm 4.2 for a simple transition system: the input set U consist of 4 actions and inputs u_1, u_2 are more prioritised than u_3, u_4, while the initial set is given by $Q_{init} = \{q_1, q_2, q_3, q_4\}$. However, let us highlight the difference between the classical synthesis approach (Algorithm 4.1) and lazy synthesis approach (Algorithm 4.2) while dealing with multilayered and multi-scale symbolic model $\Sigma = (Q, U, F)$. So, Algorithm 4.1 coincides with Algorithm 2.1 and hence it returns the maximal safety controller. The function $p\colon Q \to \{0, 1\}$ indicates whether a state $q \in Q$ is controllable ($p(q) = 1$) or not ($p(q) = 0$) and the reduced transition relation F_p is defined as follows: $(q, u, q') \in F_p$ if and only if $p(q) \neq 0$ and for all

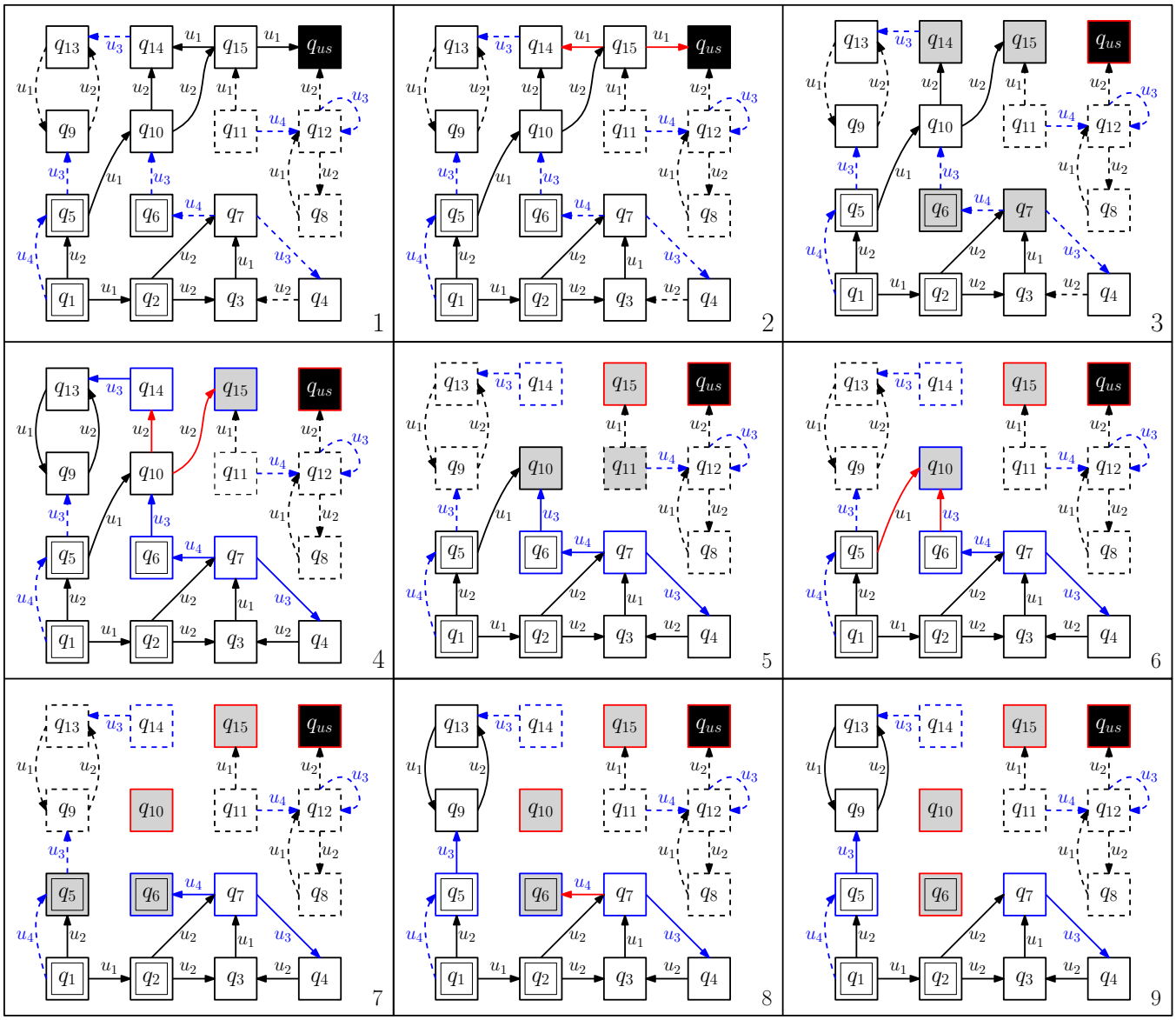


Figure 4.2: Illustration of Algorithm 4.2 execution. Unsafe actions are red. Blocking states are grey. Controls u_1, u_2 have priority 2, u_3, u_4 have priority 1. If a state is contoured with black, it has priority 2, and the actions with priority 2 are enabled for this state. If with blue, it has priority 1, and the inputs with priority 1 are available. If with red, then its priority is 0. We enable lower priority transitions for a state only if it is uncontrollable with higher priority actions. Enable at the current iteration transition marked with a normal line, while action marked with a dashed line are non-available. Reachable from the initial set states are countered with a normal line, non-reachable with a dashed line. The actions marked with a dashed line in step 9 are not included in the maximal input lazy safety controller.

$q'' \in F(q,u)$ the equality $p(q'') \neq 0$ is satisfied. The same time in Algorithm 4.2 the function $p\colon Q \to \{0,\dots,n_p+1\}$ determines the priority of a state $q \in Q$. The reduced transition relation $F_{p,\preceq_U}$ is defined as follows $(q,u,q') \in F_{p,\preceq_U}$ if and only if $p(q) \in \{1,\dots,n_p+1\}$, $u \in U_{p(q)}$, $(q,u,q') \in F$ and for all $q'' \in F(q,u)$ the equality $p(q'') \neq 0$ is satisfied. Here $U_j \subset U,\ j = 1,\dots,n_p+1$ are classes of equivalency with respect to the partial order $\preceq_U$, i.e. $U_{n_p+1} = \{up\}$, $U_j = \cup_{u \in U_\mu}(u,j),\ j = 1,\dots n_p$. Thus, unlike Algorithm 4.1 we enable shorter duration actions for a state, only if it is uncontrollable (blocking) by longer duration inputs. Moreover, at every iteration of the loop 6-11, Algorithm 4.2

explores only states which are reachable from Q_{init} with respect to the transition relation $F_{p,\preceq_U}$ (line 7). Since $Q_{init} = Q_{L,init}$ and input up has the highest priority the later means that we explore $q' \subset q$ only if q is uncontrollable. In other words if q eventually gets the priority 0 and $l(q) > 1$ it is replaced by 2^n states $q_i \in Q_{l(q)+1}$ such that $q = \cup_{i=1}^{2^n} q_i$. Thus, the introduction of an artificial input up allows us to manipulate a multilayered grid and transitions duration using one general framework. Let us remark that using Algorithm 4.2 it makes sense to calculate the symbolic model on-the-fly remaining the unexplored part of the abstraction unconstructed.

4.2.1 From a Multilayered to an Adaptive Grid

However, the abstraction considered in the previous section has two significant disadvantages. First, when we run Algorithm 4.2, we have to store a multilayered grid Q, while it is better to work with an adaptive grid consisting of cells of different sizes. Second, for every fixed distribution of priorities of states $p\colon Q \to \{0, \dots, n_p + 1\}$ a reduced transition relation $F_{p,\preceq_U}$ includes a lot of auxiliary transitions which serves only for switching from a finer layer to a coarser one. At the same time, we prefer to keep in memory only transitions directly related to the dynamic of the original system.

For a given p let us choose as an adaptive grid $Q_p^A = Q_{up}^p(Q_L)$, where $Q_{up}^p\colon Q_L \to Q$ is defined as follows

$$Q_{up}^p(q) = \{q' \in Q \mid F_{p,\preceq_U}(q', up) = \emptyset \text{ and } \exists\{q_i\}_{i=0}^N, q_i \in Q, i = \overline{0,N} \text{ such that } q_0 = q, q' = q_N, q_{j+1} = F_{p,\preceq_U}(q_j, up), j = \overline{0, N-1}\}$$

and its extension for all $Q \subseteq Q_L$ is $Q_{up}^p(Q) = \cup_{q\in Q} Q_{up}^p(q)$. So, a state of multilayered grid Q is included in Q_p^A if and only if it is reachable from the finest layer only with up transitions, and there is no possibility to go higher for this state. Let us remark that for any $q_i, q_j \in Q_p^A$, $q_i \neq q_j$ the following $q_i \cap q_j = \emptyset$ is satisfied, moreover $X = \cup_{q\in Q_p^A} q$.

For the associated transition relation let us introduce $F_p^A \subseteq Q_p^A \times U \times Q_p^A$ such that $(q, u, q') \in F_p^A$ if and only if $q' \in Q_{up}^p(F_{p,\preceq_U}(q, u))$. Its definition is correct because for any $q \in Q_p^A, u \in U$ the set $F_{p,\preceq_U}(q, u)$ either empty, or included in Q_L. Let us remark that for any given $p\colon Q \to \{0, \dots, n_p + 1\}$, any $q \in Q_p^A$ the set $F_p^A(q, up) = \emptyset$, i.e., in fact, $F_p^A \subseteq Q_p^A \times U \setminus \{up\} \times Q_p^A$. However we keep the redundant input up to have the same input set U since we want to show that we can operate with a grid Q_p^A and a transition relation F_p^A, instead of Q and $F_{p,\preceq_U}$ while executing the Algorithm 4.2.

Proposition 4.2.1. *For any $q \in Q$, $q \in Q_E^{A,R}$, where $Q_E^{A,R} = \mathrm{Reach}_{F_p^A}(Q_{up}^p(\{q \in Q_{init} \mid p(q) \neq 0\}))$ if and only if $q \in Q_E^R$, where $Q_E^R = \mathrm{Reach}_{F_{p,\preceq_U}}(\{q \in Q_{init} \mid p(q) \neq 0\})$ and $F_{p,\preceq_U}(q, up) = \emptyset$.*

Proof. Remembering that any non-up transition always finishes at the lowest layer, while input up strictly prioritized that all the others, we have from the definition of $\mathrm{Reach}_{F_{p,\preceq_U}}$, that $q \in R$, $F_{p,\preceq_U}(q, up) = \emptyset$ is equivalent to the

existence of a trajectory

$$q_{0,0} \xrightarrow[F_{p,\preceq_U}]{up} q_{0,1} \xrightarrow[F_{p,\preceq_U}]{up} \dots \xrightarrow[F_{p,\preceq_U}]{up} q_{0,n_0} \xrightarrow[F_{p,\preceq_U}]{u_1} q_{1,0} \xrightarrow[F_{p,\preceq_U}]{up} q_{1,1} \xrightarrow[F_{p,\preceq_U}]{up} \dots \xrightarrow[F_{p,\preceq_U}]{up} q_{1,n_1} \xrightarrow[F_{p,\preceq_U}]{u_2} \dots$$
$$\dots \xrightarrow[F_{p,\preceq_U}]{u_m} q_{m,0} \xrightarrow[F_{p,\preceq_U}]{up} q_{m,1} \xrightarrow{up} \dots \xrightarrow[F_{p,\preceq_U}]{up} q_{m,n_m} = q$$

such that $q_{0,0} \in \{q \in Q_{init} \mid p(q) \neq 0\}$, $q_{i,0} \in Q_L$ for all $i = 0, \dots, m$ and $q_{j,n_j} \in Q_p^A$ for all $j = 0, \dots, m$. This is true (see the the definition of $Q_{up}^p(q)$) if and only if there exist a trajectory

$$q_{0,n_0} \xrightarrow[F_p^A]{u_1} q_{1,n_1} \xrightarrow[F_p^A]{u_2} \dots\dots \xrightarrow[F_p^A]{u_m} q_{m,n_m} = q$$

such that $q_{0,n_0} \in Q_{up}^p(\{q \in Q_{init} \mid p(q) \neq 0\})$ and $q_{j,n_j} \in Q_p^A$ for all $j = 0, \dots, m$. The last is equivalent to the fact $q \in Q_E^{A,R}$. □

Proposition 4.2.2. *Let us run Algorithm 4.2 for the transition system $\Sigma = (Q, U, F)$. Then at every iteration of loop 6-11 the following $\mathrm{Block}_{F_{p,\preceq_U}}(Q_E^R) = \mathrm{Block}_{F_p^A}(Q_E^{A,R})$ is satisfied.*

Proof. So, $q \in \mathrm{Block}_{F_{p,\preceq_U}}(Q_E^R)$ if and only if $q \in Q_E^R$ and $F_{p,\preceq_U}(q,u) = \emptyset$ for any $u \in U$, which is equivalent to $q \in Q_E^{A,R}$ and $Q_{up}^p(F_{p,\preceq_U}(q,u)) = \emptyset$ for any $u \in U$. The last is true if and only if $q \in \mathrm{Block}_{F_p^A}(Q_E^{A,R})$. □

So, we can reinitialize the priority of states p (line 11), using only knowledge about Q_p^A and F_p^A. Now we explain how to update the adaptive grid Q_p^A and a transition relation F_p^A, while changing p during the evolution of Algorithm 4.2.

After execution of lines 2-5, a state in Q is reachable from the finest layer if and only if it is safe or belonging to Q_L. Hence, if we start from the coarsest layer Q_1 and recursively split all unsafe states into 2^n pieces until they do not belong to the finest layer Q_L, we finally get a grid Q_p^A, corresponding to the distribution of priority of states p just before the loop 6-11 execution. Since with every iteration of the loop only states included in Q_p^A change their priorities (see Proposition 4.2.2), we can also update our adaptive grid Q_p^A without direct usage of the transition relation $F_{p,\preceq_U}$. Indeed, if a state $q \in Q \setminus Q_L$ gets a priority 0, then, from Lemma 4.1.1 and definition of up transition, it follows that we should replace it by 2^n states $q_i' \in Q_{l(q)+1}$, such that $q = \cup_{i=1}^{2^n} q_i'$.

There also exists a way to compute the transition relation F_p^A, using only the current version of Q_p^A.

Proposition 4.2.3. *For any $q \in Q_p^A, u \in U \setminus \{up\}$ a state $q' \in Q_p^A$ belongs to a set $F_p^A(q,u)$ if and only if the intersection $q' \cap \overline{\mathrm{Reach}}(\overline{\tau}_q^{u_\mu,j} \mid q, u_\mu) \neq \emptyset$, and the condition of a collision avoidance $\overline{\mathrm{Reach}}(t \mid q, u_\mu) \cap (X \setminus Y) = \emptyset$ is satisfied for all $t \in [0, \overline{\tau}_q^{u_\mu,j}]$.*

Proof. Let $q' \in F_p^A(q,u)$. From the definition of F_p^A, this is true if and only if $q' \in Q_{up}^p(F_{p,\preceq_U}(q,u))$. From the definition of Q_{up}^p last is equivalent to existence $q'' \in Q_L$ such that $q' \in Q_{up}^p(q'')$ and $q'' \in F_{p,\preceq_U}(q,u)$. This is satisfied if and

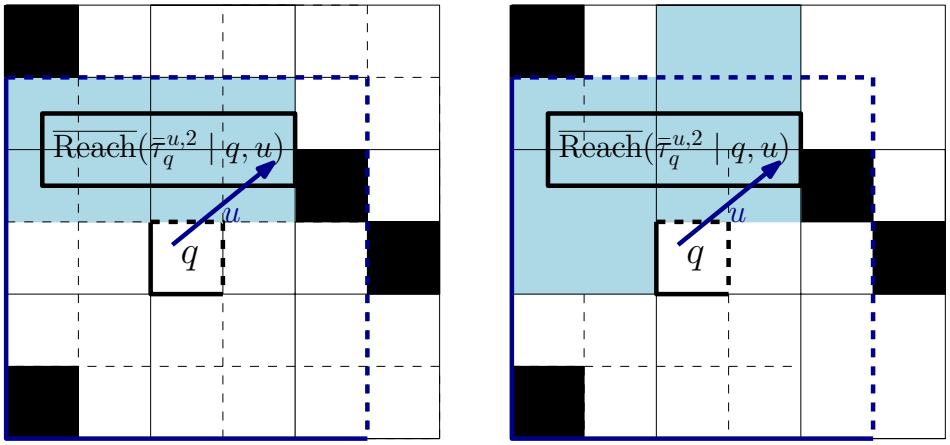


Figure 4.3: Illustration of the difference between multilayered and adaptive grid. Left figure shows the successors $F(q,u)$ on 2-layered grid, while the right figure on the corresponding adaptive grid.

only if there exists $q'' \in Q_L$ such that $q' \in Q^p_{up}(q'')$, $q'' \cap \overline{\mathrm{Reach}}(\overline{\tau}_q^{u_\mu,j} \mid q,u_\mu) \neq \emptyset$ and the condition of a collision avoidance $\overline{\mathrm{Reach}}(t \mid q,u_\mu) \cap (X \setminus Y) = \emptyset$ is not violated for all $t \in [0, \overline{\tau}_q^{u_\mu,j}]$. From Lemma 4.1.1 and definition of up transition the last is equivalent to $q' \in Q^A_p$, $q' \cap \overline{\mathrm{Reach}}(\overline{\tau}_q^{u_\mu,j} \mid q,u_\mu) \neq \emptyset$, and the condition of a collision avoidance $\overline{\mathrm{Reach}}(t \mid q,u_\mu) \cap (X \setminus Y) = \emptyset$ is satisfied for all $t \in [0, \overline{\tau}_q^{u_\mu,j}]$. That ends the proof. □

We illustrate the difference between a multilayered grid and an adaptive grid in Fig. 4.3. There, unsafe states are marked with black and successors filled with blue color. The left figure shows two uniform Cartesian partitioning embedded one into another. For a coarser partitioning, we use a normal line; for a finer — a dashed line. The right figure illustrates the corresponding adaptive grid.

Finally, we show that we can fully simulate the main part of Algorithm 4.2, using only adaptive grid Q^A_p and a transition relation F^A_p. It is also important to mention that the abstraction is not required to be pre-computed but constructed on-the-fly. Let us write, $C_*(q) := \mathrm{Enab}_{F^A_p}(q)$ for all $q \in R^*$ and empty otherwise, for a controller initialization part (lines 12-15). It is obvious, that $C_*(q)$ is a safety controller for a transition system $\Sigma_* = (Q_*, U, F_*)$, where a state space Q_* and a transition relation F_* are correspondingly Q^A_p and F^A_p after we exit the loop. Moreover, since for any $p\colon Q \to \{0,\dots,n_p+1\}$, any $q \in Q^A_p$ the set $F^A_p(q,up) = \emptyset$, the input up never appears in a final safety controller C_*. So, we can just skip it from the earlier beginning by initializing in line 3 every safe state with priority n_p, instead of n_p+1.

4.3 Controller Refinement for an Abstraction with Adaptive Grid

In the previous section, we computed a safety controller $C_*(q)$ for a transition system $\Sigma_* = (Q_*, U_\mu \times \{1,\dots,n_p\}, F_*)$, which is an abstraction for the continuous time $\Sigma_f = (T, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$. Let us now provide a safety controller for the original plant.

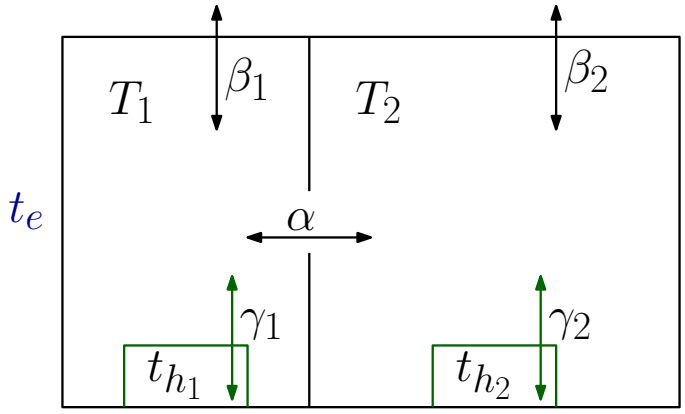


Figure 4.4: Illustration of the considered configuration for temperature regulation problem.

First of all we introduce a controller $C_*^{dur} : Q_* \to U_\mu \times T$, such that for every $q \in Q_* \setminus \mathrm{Dom}(C_*)$ we say $C_*^{dur}(q) = \emptyset$, while if $q \in \mathrm{Dom}(C_*)$ and $(u_\mu, j) \in \mathrm{Enab}_{F_*}(q)$, then the pair $(u_\mu, \overline{\tau}_q^{u_\mu, j}) \in C_*^{dur}(q)$. Hence, a controller C_*^{dur} store real durations of safe transitions, instead of the sizes of the boxes.

Now we are ready to refine the controller C_*^{dur} to a safety controller u for the original continuous system. Let us consider the control input given for all $t \in [t_k, t_{k+1})$ by

$$u(t, x) = u_k \text{ where } (u_k, \tau_k) \in C_*^{dur}(Q^x(x(t_k))) \tag{4.5}$$

and the sequence of instants $\{t_k\}$ is given by

$$t_0 = 0 \text{ and } t_{k+1} = t_k + \tau_k \tag{4.6}$$

Theorem 4.3.1. *For a given control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ all trajectories of closed-loop system* (1.1),(4.5)*, and* (4.6) *starting from a $\mathrm{Dom}(C)$ at $t = 0$ stays within the set $S \subseteq Y$ for all $t \in \mathcal{T}$ no matter which disturbance $\mathbf{w} \in \mathcal{L}(\mathcal{T}, \mathcal{W})$ has been applied.*

The statement of the theorem directly follows from the discussion above. We only remark here that the computation of the controller for the abstraction is usually implemented off-line and then stored in a control device memory, while a controller for the continuous system is calculated online using this pre-computed information.

4.4 Numerical Illustration: Temperature Regulation in the Building

This section we consider a problem of temperature regulation in two rooms building [Girard et al., 2016]. Let each room is equipped with a heater and let T_i is a temperature in the room i, $i = 1, 2$. The evolution of the temperatures

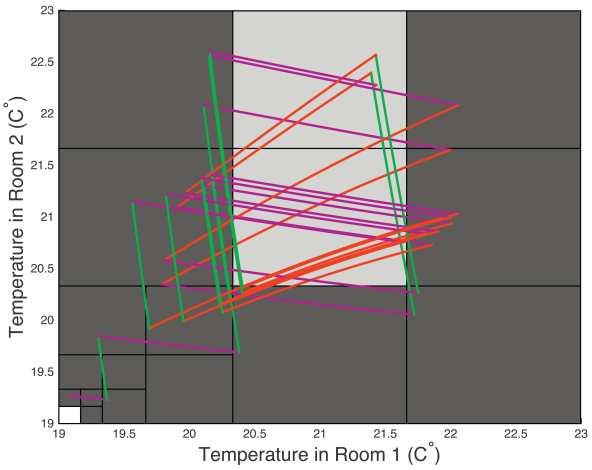


Figure 4.5: Simulation results. A grey area corresponds to the controllable domain: light states are controllable by the actions with priority 1; dark states - with priority 2. The white area is uncontrollable. Concerning the closed-loop trajectory, orange color corresponds to a control $\{0,0\}$, green - to a control $\{0,1\}$, violet - to a control $\{1,0\}$.

could be described by the following system of the differential equations

$$\begin{aligned}\dot{T}_1 &= \alpha(T_2 - T_1) + \beta_1(t_e - T_1) + \gamma_1(t_{h_1} - T_1)u_1 \\ \dot{T}_2 &= \alpha(T_1 - T_2) + \beta_2(t_e - T_2) + \gamma_2(t_{h_2} - T_2)u_2\end{aligned} \tag{4.7}$$

Here t_e is temperature of external environment of the building, t_{h_1}, t_{h_2} are temperatures of the heaters, α is the conduction factor between rooms, β_1, β_2 are conduction factors between external environment and the first room and the second room correspondingly, γ_1, γ_2 are conduction factor between heater and rooms. We illustrate the considered configuration in the Figure 4.4. Control parameter u_i equals 1 if the room i is heated and 0 otherwise. Temperature t_e is considered as a bounded disturbance.

We run our simulation for the following set of parameters

$$\alpha = 1/2 * 10^{-4}\, W/J,\ \beta_1 = 1/6 * 10^{-4}\, W/J, \beta_2 = 1/11 * 10^{-4}\, W/J,$$
$$\gamma_1 = 1.5 * 10^{-4}\, W/J,\ \gamma_2 = 1.5 * 10^{-4}\, W/J,\ t_{h_1} = 30\, C^\circ,\ t_{h_1} = 40\, C^\circ.$$

A safety specification were given by an initial set $X_0 = [19, 23] \times [19, 23]$, safe set $Y = [19, 23] \times [19, 23]$, and a disturbance $t_e \in [-10, 10]$. We also suppose that at given instant at most one heater is switched on, i.e. a control set $\mathcal{U} = \{\{0,0\},\{0,1\},\{1,0\}\}$. For the abstraction, we chose $L = 4$, $n_x = [4;4]$, $U_\mu = \mathcal{U}$, and $n_p = 2$. Let us introduce the notation A state vector $x = [T_1; T_2]$. To construct the over-approximations of reachable states we benefit from the fact the system (4.7) is monotone, and, hence, for any given state $q = [\underline{x}^q, \overline{x}^q]$, any given constant control function $\mathbf{u}\colon [0,t] \to (u_1, u_2), (u_1, u_2) \in U_\mu$ the reachable set robust to any admissible disturbances $\mathrm{Reach}(t \mid$

Table 4.1: Runtime comparison when varying the number of states.

Grid	Number of states	Execution time	Controllability Ratio
Adaptive grid	18	7 s	98%
Coarsest grid	9	5 s	89%
Finest grid	625	50 s	98%

$q, u) \subseteq [x_f(t \mid \underline{x}^q, \mathbf{u}, \mathbf{t_e^{min}}), x_f(t \mid \overline{x}^q, \mathbf{u}, \mathbf{t_e^{max}})]$, where $x_f(t \mid \underline{x}^q, \mathbf{u}, \mathbf{t_e^{min}}), x_f(t \mid \overline{x}^q, \mathbf{u}, \mathbf{t_e^{max}})$ are trajectories of the system (4.7), corresponding to initial values $\underline{x}^q, \overline{x}^q$, control function $\mathbf{u}$, and constant disturbances $\mathbf{t_e^{min}} \colon [0, t] \to t_e^{min}$, $\mathbf{t_e^{max}} \colon [0, t] \to t_e^{max}$.

In Figure 4.5 the results of simulation are provident. We use a dark grey and a light grey for states, which are controllable with a $(u_\mu, 1),\ u_\mu \in U_\mu$ and with a $(u_\mu, 2),\ u_\mu \in U_\mu$ correspondingly, while white region is uncontrollable. The closed loop trajectory were simulate for 24 hours, supposing that external temperature is varying between $t_e^{min} = -10C^\circ$ and $t_e^{max} = 10C^\circ$ and initial point $x_0 = [19.084; 19.27]$. The orange color correspond to a control $\{0, 0\}$, green to a control $\{0, 1\}$, violet to a control $\{1, 0\}$. To evaluate efficiency, we also run a simulation for two extreme cases: $L = 1, n_x = [4; 4]$ and $L = 1, n_x = [25; 25]$, which correspond to a coarsest grid and to a finest grid of considered four-layered adaptive grid. One can see the comparison of the results in Table 4.1, where the controllability ratio is a capacity ratio $\mathrm{Dom}(C^*)$ to Q_S. The controllable sets coincides for the adaptive grid and the finest grid, but we get a noticeable time and memory gain. The implementations has been done in C++, processor Intel Core i7-8700, 2.5 Hg, RAM 16 GB.

4.5 Conclusion

In this Chapter, we present an abstraction-based approach to safety controller synthesis for continuous-time nonlinear systems. To reduce the computational burden associated with symbolic control approaches, we develop a lazy controller synthesis algorithm, which uses the incremental forward exploration of the symbolic dynamics and thus allows us to restrict the controller synthesis computations to reachable states only. We propose to use this algorithm with a novel type of multilayered, multi-scale abstractions, which use adaptive sampling of time. Instead of using transitions of predetermined duration, the duration of the transitions is constrained by state intervals that must contain the reachable set, thus enabling a better control of the symbolic transitions. We also prioritize transitions with longer duration and explore lower priority actions only if we can guarantee safety using a higher priority inputs. In simple words, the proposed lazy algorithm is an alternation procedure between a forward exploration of state space and a backward correction of the obtained transition system in order to satisfy the safety requirements. At the beginning of every iteration, we update for every state a set of enabled inputs. The set of enabled actions depends on the priority of the state. Then we try to find a safety controller. If a state is uncontrollable we either reduce duration of enabled transitions or split a state in several pieces. And then start exploration again. We provide a

simple example to illustrate the benefits of the approach.

Chapter 5

Efficient Controller Synthesis for Monotone Dynamical Systems and Directed Safety Specifications

This Chapter is devoted to monotone control systems, i.e., systems whose trajectories preserve some partial orderings on their state and input spaces [Angeli and Sontag, 2003, Hirsch and Smith, 2004, Kamke, 1932, Müller, 1927, Smith, 1995]. On the one hand, our interest is motivated by many practical applications in which such systems appear. For example, traffic networks [Kim et al., 2017, Coogan et al., 2016], biological networks [Sontag, 2007, Angeli and Sontag, 2003], blood glucose control [Gillis et al., 2007], power systems [Zonetti et al., 2019], vehicle platoons [Nilsson et al., 2016] and temperature regulation systems [Meyer et al., 2013]. On the other hand, monotone control systems are a subclass of nonlinear control systems relatively easy to analyze, and many notable results were derived for them last decades. Extending pioneer works of [Kamke, 1932, Müller, 1927, Krasnoselskii, 1968], Hirsch and Smith contribute to stability theory for continuous-time monotone systems and provide several verification criteria for a system to be monotone [Smith, 1988, Smith, 1995, Hirsch and Smith, 2004]. Angeli and Sontag extend their results to systems with inputs [Angeli and Sontag, 2003]. The monotonicity property also simplifies significally the reachability analysis of the non-linear system [Meyer et al., 2021, Ramdani et al., 2010, Sinyakov and Girard, 2020b]. Concerning to invariance for monotone systems, relying on ideas of [Abate et al., 2009], the authors in [Meyer et al., 2017] propose a necessary and sufficient condition for an interval to be controlled invariant. However, their result is restricted to monotone systems where every input affects no more than one state variable. We overcome this limitation by using abstraction-based techniques, which allow us to compute a non-convex safely controllable set for a general monotone dynamical system.

However, we focus on a particular class of safety specifications represented by directed (lower-closed or upper-

closed) sets or their intersection. Our choice is inspired by the paper [Kim et al., 2017], where efficient sparse abstractions are proposed for monotone dynamical systems and directed specifications. Going one step further, we benefit from a particular structure and synthesize the maximal safety controller lazily.

In spirit, the closest works in the literature is [Sadraddini and Belta, 2016]. In [Sadraddini and Belta, 2016], the authors introduce a notion of s-sequence to characterize a controlled invariant of the system. This notion is relatively close to the notion of basis, which we use to represent directed sets in our work. However, looking at the computation of controlled invariants as an optimization problem [Sadraddini and Belta, 2016] does not guarantee the maximality of the obtained control invariant set. They also propose to use a simple open-loop control policy to keep a trajectory within a safe set. Although such an approach is memory efficient, it is hardly possible to use their controller as a start point for more general tasks (for example, obstacle-avoided reachability specification [Reissig et al., 2017] or optimal control problems with hard constraints).

The Chapter is organized as follows. Section 5.1 introduces input-state and state monotone dynamical systems. It also provides an illustrative example emphasizing the difference between these two classes of systems. We then adapt results from [Angeli and Sontag, 2003] to characterize the considered objects. Subsection 5.1 is devoted to directed safety specifications. Section 5.2 defines lower and upper (input-)state monotone transition systems, provides a tool for their characterization, and determines the procedure, preserving the monotonicity property of the original plant while creating a symbolic abstraction. Section 5.3 presents lazy algorithms to compute the maximal safety controller for monotone transition systems and directed safety specification and their intersection.

5.1 Monotone Dynamical Systems and Directed Safety Specifications

In this section, we consider monotone control systems with disturbances [Meyer et al., 2017], but in contrast to the existing terminology, we distinguish the systems whose trajectories preserve a partial order on the state-space from those that also preserve the monotonicity of the input space. We adapt the results [Angeli and Sontag, 2003] to provide criteria for a dynamical system to be state monotone or input-state monotone and illustrate the gap between these two classes of systems with a simple example. We then follow the paper [Kim et al., 2017] and focus on directed safety specifications.

Input-State and State Monotone Dynamical Systems and Their Characterization

Let us start with introducing the notions of state monotone and input-state monotone dynamical systems and with Figure 5.1 illustrating the given definitions.

Definition 5.1.1. *Let $\mathcal{S}$ be a compact subset of a state space $\mathbb{R}^s$ and let $\mathcal{S}$ be equipped with a partial order $\preceq_{\mathcal{S}}$. Then for two functions $\mathbf{s}, \mathbf{s}' \in \mathcal{L}^\infty(\mathcal{T}, \mathcal{S})$ we say, that $\mathbf{s} \preceq_{\mathcal{S},\mathcal{T}} \mathbf{s}'$ if the following holds $s(t) \preceq_{\mathcal{S}} s'(t)$ for all $t \in \mathcal{T}$.*

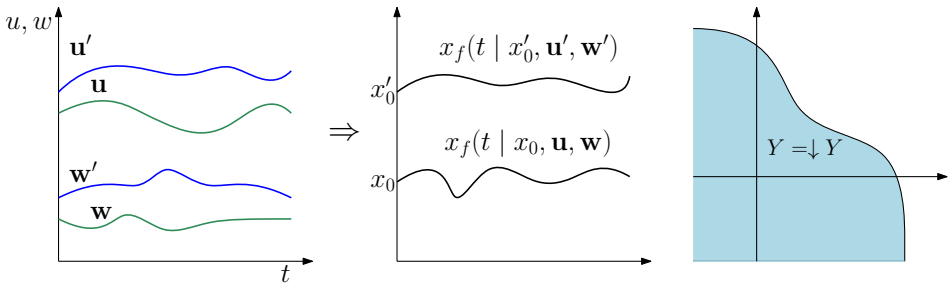


Figure 5.1: Illustration of Definitions 5.1.3 and 5.1.5. The left figures show two trajectories of an input-state monotone dynamical system. The right figure provides an example of a lower-closed safety set. In all figures, a natural component-wise order on $\mathbb{R}^2$ is considered for both the input and the state spaces.

Definition 5.1.2. *Consider a control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$, where $\mathbb{R}^{n_x}$ and $\mathcal{W}$ are equipped with partial orders $\preceq_{\mathbb{R}^{n_x}}$ and $\preceq_{\mathcal{W}}$ correspondingly. The system $\Sigma = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ is said to be state monotone (SM) with respect to the partial orderings $\preceq_{\mathbb{R}^{n_x}}, \preceq_{\mathcal{W}}$ if for all $x_0, x_0' \in \mathbb{R}^{n_x}$, for all $\mathbf{u} \in \mathcal{L}^\infty(\mathcal{T}, \mathcal{U})$ and for all $\mathbf{w}, \mathbf{w}' \in \mathcal{L}^\infty(\mathcal{T}, \mathcal{W})$ the following implication holds $x_0 \preceq_{\mathbb{R}^{n_x}} x_0', \mathbf{w} \preceq_{\mathcal{W},\mathcal{T}} \mathbf{w}' \Rightarrow x_f(t, x_0, \mathbf{u}, \mathbf{w}) \preceq_{\mathbb{R}^{n_x}} x_f(t, x_0, \mathbf{u}, \mathbf{w}'),\ t \in \mathcal{T}$.*

Definition 5.1.3. *Consider a control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$, where $\mathbb{R}^{n_x}$, $\mathcal{U}$, and $\mathcal{W}$ are equipped with partial orders $\preceq_{\mathbb{R}^{n_x}}, \preceq_{\mathcal{U}}$ and $\preceq_{\mathcal{W}}$ correspondingly. The system $\Sigma = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ is input-state monotone (ISM) with respect to the partial orderings $\preceq_{\mathbb{R}^{n_x}}, \preceq_{\mathcal{U}}$, and $\preceq_{\mathcal{W}}$ if for all $x_0, x_0' \in \mathbb{R}^{n_x}$, $\mathbf{u}, \mathbf{u}' \in \mathcal{L}^\infty(\mathcal{T}, \mathcal{U})$, and $\mathbf{w}, \mathbf{w}' \in \mathcal{L}^\infty(\mathcal{T}, \mathcal{W})$ such that $x_0 \preceq_{\mathbb{R}^{n_x}} x_0', \mathbf{u} \preceq_{\mathcal{U},\mathcal{T}} \mathbf{u}'$, and $\mathbf{w} \preceq_{\mathcal{W},\mathcal{T}} \mathbf{w}'$ the following holds $x_f(t, x_0, \mathbf{u}, \mathbf{w}) \preceq_{\mathbb{R}^{n_x}} x_f(t, x_0, \mathbf{u}, \mathbf{w}'),\ t \in \mathcal{T}$.*

It follows straightforwardly from the definitions above that any ISM control system is a SM control system. Moreover, any SM system can be seen as ISM, with a partial order defined as $u \preceq_{\mathcal{U}} u' \Leftrightarrow u = u'$. However, this trivial order does not have any practical interest, and while speaking about ISM transition systems, we assume that there are at least two elements in $u, u' \in \mathcal{U}$, such that $u \prec_{\mathcal{U}} u'$.

In this manuscript, we focus on a subclass of (input-)state monotone systems, which is called (input-) state cooperative systems. Similar to definitions in [Angeli and Sontag, 2003] we say that an (input-)state monotone system $\Sigma = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ is a (input-)state cooperative system if partial orders on $\mathbb{R}^{n_x}, \mathcal{U} \subseteq \mathbb{R}^{n_u}, \mathcal{W} \subseteq \mathbb{R}^{n_w}$ are induced by the positive orthants $\mathbb{R}^{n_x}_+, \mathbb{R}^{n_u}_+, \mathbb{R}^{n_w}_+$ correspondingly. Then, adapting the Kamke-Müller condition [Kamke, 1932] we characterize (input-) state cooperative systems, as follows.

Theorem 5.1.1. *The system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ with locally Lipschitz vector field f is input-state cooperative if and only if for all $i = 1, \dots, n_x$ for all $x \preceq_{\mathbb{R}^{n_x}} x$, such that $x_i = x_i'$, for all $u \preceq_{\mathcal{U}} u', w \preceq_{\mathcal{W}} w'$ the following inequality is satisfied $f_i(x, u, w) \leq f_i(x', u', w')$, where partial orders $\preceq_{\mathbb{R}^{n_x}}, \preceq_U, \preceq_W$ are induced by $\mathbb{R}^{n_x}_+, \mathbb{R}^{n_u}_+, \mathbb{R}^{n_w}_+$ correspondingly.*

Corollary 5.1.1. *The system $\Sigma = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ with locally Lipschitz vector field f is state cooperative if and only if for all $i = 1, \dots, n_x$ for all $x \preceq_{\mathbb{R}^{n_x}} x$, such that $x_i = x_i'$, for all $u \in \mathcal{U}$ and for all $w \preceq_{\mathcal{W}} w'$ the following inequality is satisfied $f_i(x, u, w) \leq f_i(x', u, w')$, where partial orders $\preceq_{\mathbb{R}^{n_x}}, \preceq_W$ are induced by $\mathbb{R}^{n_x}_+, \mathbb{R}^{n_w}_+$ correspondingly.*

We are now ready to provide an example illustrating the difference between SM and ISM systems.

Example 5.1.1. *Let us consider a control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^2, \mathcal{U}, \mathcal{W}, f)$, where $\mathcal{U} = \{1, 2\}$ and $\mathcal{W} = \emptyset$. Hence, there is no disturbance in the system's dynamic $\dot{x} = f(x, u)$, where $f(x, u)$ defined as follows*

$$f(x,u) = \begin{cases} A_1 x \ \text{ if } \ u = 1 \\ A_2 x \ \text{ if } \ u = 2 \end{cases} \quad \text{where } A_1 = \begin{pmatrix} 0.1 & 0.9 \\ 3 & 0.7 \end{pmatrix} \text{ and } A_2 = \begin{pmatrix} 0.2 & 2 \\ 0.1 & 0.7 \end{pmatrix}$$

Using the Corollary 5.1.1 we can conclude that the transition system presented above is state cooperative, while it is not input-state cooperative for any non-trivial partial order on U.

Another well-known in the literature criterion is formulated in terms of partial derivatives of the vector field.

Theorem 5.1.2. *The system $\Sigma = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ with continuously differentiable vector field f is input-state cooperative if and only if for all $x \in \mathbb{R}^{n_x}, u \in \mathcal{U}, w \in \mathcal{W}$, for all $i, j = 1, \dots, n_x,\ j \neq i,\ k = 1 \dots n_u, l = 1 \dots n_w$ the following inequalities are satisfied*

$$\frac{\partial f_i}{\partial x_j}(x,u,w) \geq 0,\ \frac{\partial f_i}{\partial u_k}(x,u,w) \geq 0,\ \frac{\partial f_i}{\partial w_l}(x,u,w) \geq 0.$$

Corollary 5.1.2. *The system $\Sigma = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ with continuously differentiable vector field f is input-state cooperative if and only if for all $x \in \mathbb{R}^{n_x}, w \in \mathcal{W}$, for all $i, j = 1, \dots, n_x,\ j \neq i, l = 1 \dots n_w$ the following inequalities are satisfied*

$$\frac{\partial f_i}{\partial x_j}(x,u,w) \geq 0,\ \frac{\partial f_i}{\partial w_l}(x,u,w) \geq 0.$$

Directed Safety Specifications

In this section, we introduce the notion of directed safety specifications, i.e., specifications determined by lower-closed or upper-closed sets.

Definition 5.1.4. *Let $\mathcal{L}$ be a partially ordered set, a lower closure of a set $A \subseteq \mathcal{L}$ is a set $\downarrow A = \bigcup_{a \in A} \downarrow a$, where $\downarrow a = \{q \in \mathcal{L} \mid q \preceq_{\mathcal{L}} a\}$, while an upper closure is a set $\uparrow A = \bigcup_{a \in A} \uparrow a$, where $\uparrow a = \{q \in \mathcal{L} \mid a \preceq_{\mathcal{L}} q\}$.*

Definition 5.1.5. *Given a partially ordered set, we say that a subset $A \subseteq \mathcal{L}$ is lower-closed (respectively upper-closed) if $\downarrow A = A$ (respectively $\uparrow A = A$).*

We illustrate the Definition 5.1.5 with a lower-closed set Y in Figure 5.1. The following result shows that union (an intersection) of a lower-closed (upper-closed) sets is lower-closed (upper-closed).

Proposition 5.1.1 ([Reddy, 1995])**.** *Let $\mathcal{L}$ be a partially ordered set, $A_i \subseteq \mathcal{L}$ be finite subsets satisfying $A_i = \downarrow A_i$ for all $i \in \{1, \dots, p\}, p \in \mathbb{N}_{\geq 1}$. Then $\downarrow (\cup_{i=1}^{p} A_i) = \cup_{i=1}^{p} A_i$ and $\downarrow (\cap_{i=1}^{p} A_i) = \cap_{i=1}^{p} A_i$. Analogously, let $A_i \subseteq \mathcal{L}$ be finite subsets satisfying $A_i = \uparrow A_i$ for all $i \in \{1, \dots, p\}, p \in \mathbb{N}_{\geq 1}$. Then $\uparrow (\cup_{i=1}^{p} A_i) = \cup_{i=1}^{p} A_i$ and $\uparrow (\cap_{i=1}^{p} A_i) = \cap_{i=1}^{p} A_i$.*

5.2 Monotone Abstractions for Monotone Dynamical Systems

In this section, we first define the class of monotone dynamical systems. We then present different types of abstractions, namely box and sparse abstractions. Finally, we show under which assumptions these abstractions preserve the monotonicity of the original plant.

5.2.1 Monotone Transition Systems

Let us now introduce the class of monotone transition systems that preserve partial order on input and state spaces.

Definition 5.2.1. *Consider a transition system $\Sigma = (Q, U, F)$ where the set of states Q is equipped with a partial order $\preceq_Q$. The transition system Σ is said to be:*

- *Lower state monotone (LSM) if for all $q_1, q_2 \in Q$ such that $q_1 \preceq_Q q_2$, for all $u \in \mathrm{Enab}_F(q_2)$ it follows, that $u \in \mathrm{Enab}_F(q_1)$ and for any $q_1' \in F(q_1, u)$, there is $q_2' \in F(q_2, u)$, such that $q_1' \preceq_Q q_2'$;*
- *Upper state monotone (USM) if for all $q_1, q_2 \in Q$ such that $q_1 \preceq_Q q_2$, for all $u \in \mathrm{Enab}_F(q_1)$ it follows, that $u \in \mathrm{Enab}_F(q_2)$ and for any $q_2' \in F(q_2, u)$, there is $q_1' \in F(q_1, u)$, such that $q_1' \preceq_Q q_2'$.*

The transition system Σ is said to be state monotone (SM) if it is both LSM and USM.

Let us remark that the concepts of USM and LSM coincide when the transition system Σ is deterministic.

Definition 5.2.2. *Consider a transition system $\Sigma = (Q, U, F)$ where the set of states Q and the set of inputs U are equipped with partial orders $\preceq_Q$, $\preceq_U$, respectively. The transition system Σ is said to be:*

- *Lower input-state monotone (LISM) if for all $q_1, q_2 \in Q$, such that $q_1 \preceq_Q q_2$, for all $u_2 \in \mathrm{Enab}_F(q_2)$ all inputs $u_1 \in U$, such that $u_2 \succeq_U u_1$ belong to $\mathrm{Enab}_F(q_1)$, and for all $q_1' \in F(q_1, u_1)$, there is $q_2' \in F(q_2, u_2)$, such that $q_1' \preceq_Q q_2'$;*
- *Upper input-state monotone (UISM) if for all $q_1, q_2 \in Q$, such that $q_1 \preceq_Q q_2$, for all $u_1 \in \mathrm{Enab}_F(q_1)$ all inputs $u_2 \in U$, such that $u_2 \succeq_U u_1$ belong to $\mathrm{Enab}_F(q_2)$, and for all $q_2' \in F(q_2, u_2)$, there is $q_1' \in F(q_1, u_1)$ such that $q_1' \preceq_Q q_2'$.*

The transition system Σ is said to be input-state monotone (ISM) if it is both LISM and UISM.

Similar to the case of SM transition systems, the concepts of UISM and LISM coincide when the transition system Σ is deterministic.

Characterization of Monotone Transition Systems

In this section, we provide a criterion allowing us to verify if a transition system is a lower or upper (input-)state monotone. Let us characterize lower (input-)state monotone transition systems only while keeping in mind that

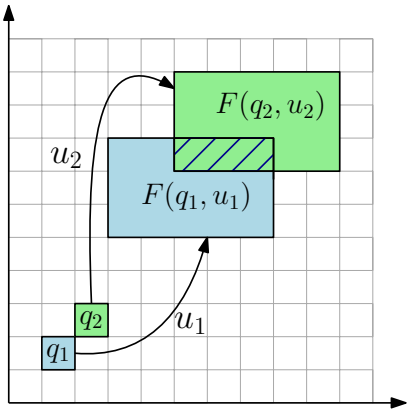


Figure 5.2: Illustration of Theorem 5.2.1. Given two states $q_1 \preceq_Q q_2$ and two inputs $u_1 \preceq_U u_2$, if the transition system is LISM then we have that $F(q_1, u_1) \subseteq\downarrow F(q_2, u_2)$.

analogous results hold for upper (input-)state monotone transition systems as well. We start with an auxiliary lemma.

Lemma 5.2.1. *Let $\mathcal{L}$ be a partially ordered set and $A, B \subseteq \mathcal{L}$. The set A is included in the lower closure of the set B (i.e. $A \subseteq\downarrow B$) if and only if for any $a \in A$, there exists $b \in B$ such that $a \preceq_Q b$.*

The proof follows immediately from the fact that for any set $B \subseteq \mathcal{L}$ we have $\downarrow B = \{q \in \mathcal{L} \mid \exists\, b \in B \text{ s.t. } q \preceq_Q b\}$.

Theorem 5.2.1. *For a transition system $\Sigma = (Q, U, F)$ the following statements are equivalent:*

(i) Σ is a LISM transition system;

(ii) for all $q_1, q_2 \in Q$, for all $u_2 \in \mathrm{Enab}_F(q_2), u_1 \in U$ if $q_1 \preceq_Q q_2$ and $u_1 \preceq_U u_2$, then $u_1 \in \mathrm{Enab}_F(q_1)$ and $F(q_1, u_1) \subseteq\downarrow F(q_2, u_2)$;

Proof. Let $q_1, q_2 \in Q$, for all $u_2 \in \mathrm{Enab}_F(q_2), u_1 \in U$, with $q_1 \preceq_Q q_2$ and $u_1 \preceq_U u_2$. Then from Lemma 5.2.1 $u_1 \in \mathrm{Enab}_F(q_1)$ and $F(q_1, u_1) \subseteq\downarrow F(q_2, u_2)$ if and only if $u_1 \in \mathrm{Enab}_F(q_1)$ and for any $q_1' \in F(q_1, u_1)$, there exists $q_2' \in F(q_2, u_2)$ with $q_1' \preceq_Q q_2'$. Hence, $(i) \Leftrightarrow (ii)$. □

A graphical representation of the conditions in Theorem 5.2.1 is provided in Figure 5.2. We then have the following corollary for LSM transition systems.

Corollary 5.2.1. *For a system $\Sigma = (Q, U, F)$ the following statements are equivalent:*

(i) Σ is a LSM transition system;

(ii) for all $q_1, q_2 \in Q$, for all $u \in \mathrm{Enab}_F(q_2)$ if $q_1 \preceq_Q q_2$, then $u \in \mathrm{Enab}_F(q_1)$ and $F(q_1, u) \subseteq\downarrow F(q_2, u)$;

5.2.2 Box abstraction

In this part, we discuss a special class of symbolic models $\Sigma^B = (Q, U, F)$, preserving the monotonicity property of the original (input-) state monotone system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$.

First of all, we over-approximate the compact set $\mathcal{W}$ by a finite union of multi-dimensional intervals $\bigcup_{m=1}^{M}[w_1^m, w_2^m]$. Then, we introduce a partitioning Q on $\mathbb{R}^{n_x}$. We say that a state $q \in Q$ belongs to a safe set $Q_S \subset Q$ if and only if $q \subset Y$, otherwise the state q is unsafe, i.e., $q \in Q_{US} = Q \setminus Q_S$. We assume that with introduced partitioning, the set Q_S is a finite set. Moreover, it is supposed that the partitioning Q is grid-aligned, i.e. each element $q \in Q$ is described as a multi-dimensional semi-intervals $q = [x_1^q, x_2^q)$. Choosing the set of admissible abstract inputs, as follows $U = \{u_1, \dots, u_m\} \subseteq \mathcal{U}$ we then use reachability analysis techniques to compute the transition relation F.

Remark 5.2.1. *Consider a (input-)state monotone control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ and let $\mathcal{W} \subseteq \bigcup_{m=1}^{M}[w_1^m, w_2^m]$. Then from Definitions 5.1.2,5.1.3 for the reachable set corresponding to an initial set $q = [x_1^q, x_2^q)$, and a constant control function $\mathbf{u} : \mathcal{T} \to u, u \in U$ the following is satisfied*

$$\mathrm{Reach}(t \mid q, u) \subseteq \cup_{m=1}^{M}[x_f(t \mid x_1^q, \mathbf{u}, \mathbf{w_1^m}), x_f(t \mid x_2^q, \mathbf{u}, \mathbf{w_2^m})] \quad \textit{for all } t \in \mathcal{T},$$

where for all $m = 1 \dots M$ functions $\mathbf{w_1^m}, \mathbf{w_2^m}$ are constant disturbances with values w_1^m, w_2^m correspondingly.

Let for all $q \in Q_S$, for all $u \in U$ transition $(q, u, q') \in F$ if and only if $q \in Q_S$, $u \in U$, $q' \in Q$, the intersection

$$q' \cap \cup_{m=1}^{M}[x_f(\tau \mid x_1^q, \mathbf{u}, \mathbf{w_1^m}), x_f(\tau \mid x_2^q, \mathbf{u}, \mathbf{w_2^m})] \neq \emptyset, \tag{5.1}$$

and for all $t \in [0, \tau]$ the condition of a collision avoidance

$$\cup_{m=1}^{M}\,[x_f(t \mid x_1^q, \mathbf{u}, \mathbf{w_1^m}), x_f(t \mid x_2^q, \mathbf{u}, \mathbf{w_2^m})] \cap (\mathbb{R}^{n_x} \setminus Y) = \emptyset \tag{5.2}$$

is satisfied. Where $\mathbf{u}$ is a constant control function with value u, functions $\mathbf{w_1^m}, \mathbf{w_2^m}$ are constant disturbances with values w_1^m, w_2^m correspondingly, τ is a fixed time sampling parameter. We illustrate the construction of the transitions for the box abstraction in Figure 5.4. Taking into account the remark 5.2.1, we may conclude that the constructed abstraction $\Sigma^B = (Q, U, F)$, is related to the original (input-)state monotone system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ with feed-back refinement relation [Reissig et al., 2017], and a safety controller synthesised for $\Sigma^B = (Q, U, F)$, can be refined to solve the Problem 1.1.1. However, to increase the efficiency of the synthesis procedure, we are interested in preserving monotonicity of the original system while constructing $\Sigma^B = (Q, U, F)$. Let us make some additional assumptions about partitioning Q_S.

Assumption 5.2.1. *Let for all $q, q' \in Q$ if there exists $(x, x') \in q \times q'$ satisfying $x \preceq_{\mathbb{R}^{n_x}} x'$, then $x_2^q \preceq_{\mathbb{R}^{n_x}} x_2^{q'}$.*

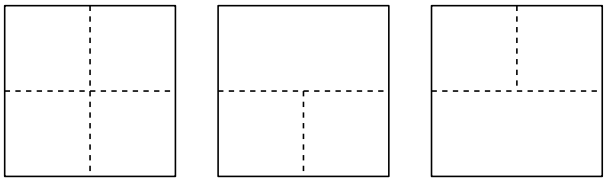


Figure 5.3: The first two partitions satisfy Assumption 5.2.1, while the third partition does not satisfy Assumption 5.2.1. The first and third partitions satisfy Assumption 5.2.2, while the second partition does not satisfy Assumption 5.2.2. The state-space is equipped with the component-wise partial order $\preceq$ defined on $\mathbb{R}^2$.

Assumption 5.2.2. *Let for all $q, q' \in Q$ if there exists $(x, x') \in q \times q'$ satisfying $x \preceq_{\mathbb{R}^{n_x}} x'$, then $x_1^q \preceq_{\mathbb{R}^{n_x}} x_1^{q'}$.*

Intuitively, Assumption 5.2.1 (respectively, Assumption 5.2.2) reflects the fact that the partitioning should preserve the lower (respectively, upper) monotonicity property from continuous to symbolic states. Fig. 5.3 shows examples of partitions satisfying Assumptions 5.2.1 and 5.2.2. Keeping in mind Assumptions 5.2.1 and 5.2.2, let us define a partial order on the abstract set of states$\preceq_Q$ as follows:

- If Assumption 5.2.1 is satisfied, then for all $q_1, q_2 \in Q$, $q_1 \preceq_Q q_2$ if and only if $x_2^{q_1} \preceq_{\mathbb{R}^{n_x}} x_2^{q_2}$;
- If Assumption 5.2.2 is satisfied, then for all $q_1, q_2 \in Q$, $q_1 \preceq_Q q_2$ if and only if $x_1^{q_1} \preceq_{\mathbb{R}^{n_x}} x_1^{q_2}$;
- If both Assumptions 5.2.1 and 5.2.2 are satisfied, then for all $q_1, q_2 \in Q$, $q_1 \preceq_Q q_2$ if and only if $x_1^{q_1} \preceq_{\mathbb{R}^{n_x}} x_1^{q_2}$ and $x_2^{q_1} \preceq_{\mathbb{R}^{n_x}} x_2^{q_2}$.

In the following result, we show that, under Assumptions 5.2.1 and 5.2.2, monotonicity of the original system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ is preserved when constructing its symbolic box abstraction $\Sigma^B = (Q, U, F)$.

Proposition 5.2.1. *Let us consider control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$. If $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ is an input state cooperative system and if its symbolic box abstraction $\Sigma^B = (Q, U, F)$ satisfies Assumption 5.2.1, (respectively, Assumption 5.2.2), then $\Sigma^B = (Q, U, F)$ is a LISM (respectively, UISM) transition system. Moreover, when both Assumptions 5.2.1 and 5.2.2 are satisfied, then $\Sigma^B = (Q, U, F)$ is an ISM transition system*

Proof. We only provide a proof for the case of LISM, the cases of UISM and ISM can be derived similarly. We should prove that for all $q_1, q_2 \in Q$ and for all $u_1, u_2 \in U$, if $q_1 \preceq_Q q_2$ and $u_1 \preceq_U u_2$, the following is satisfied: for all $q_1' \in F(q_1, u_1)$ there is $q_2' \in F(q_2, u_2)$, such that $q_1' \preceq_Q q_2'$. Let $q_1, q_2 \in Q$ and $u_1, u_2 \in U$, such that $q_1 \preceq_Q q_2$ and $u_1 \preceq_U u_2$. Under Assumption 5.2.1, the last is equivalent to $x_2^{q_1} \preceq_{\mathbb{R}_x^n} x_2^{q_2}$ and $u_1 \preceq_U u_2$. Let $q_1' \in F(q_1, u_1)$, from the monotonicity of $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ and the construction of $\Sigma^B = (Q, U, F)$, we have the existence of $m \in \{1, \ldots, M\}$ such that $x_f(\tau \mid x_2^{q_1}, \mathbf{u_1}, \mathbf{w_2^m}) \in q_1'$. Let $q_2' \in Q$ is such that $x_f(\tau \mid x_2^{q_2}, \mathbf{u_2}, \mathbf{w_2^m}) \in q_2'$. Then $q_2' \in F(q_2, u_2)$. Here functions $\mathbf{u_1}$, $\mathbf{u_2}$, $\mathbf{w_1^m}, \mathbf{w_2^m}$ are constant functions with values w_1^m, w_2^m correspondingly Moreover, there are $\underline{x} = x_f(\tau \mid x_2^{q_1}, \mathbf{u_1}, \mathbf{w_2^m})$ and $\overline{x} = x_f(\tau \mid x_2^{q_2}, \mathbf{u_2}, \mathbf{w_2^m})$, such that $\underline{x} \in q_1', \overline{x} \in q_2'$ and since $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ is an input-state monotone control system, one has $\underline{x} \preceq_{\mathbb{R}^{n_x}} \overline{x}$. Hence, from Assumption 5.2.1, $x_2^{q_1'} \preceq_{\mathbb{R}^{n_x}} x_2^{q_2'}$. The last is equivalent to $q_1' \preceq_Q q_2'$. □

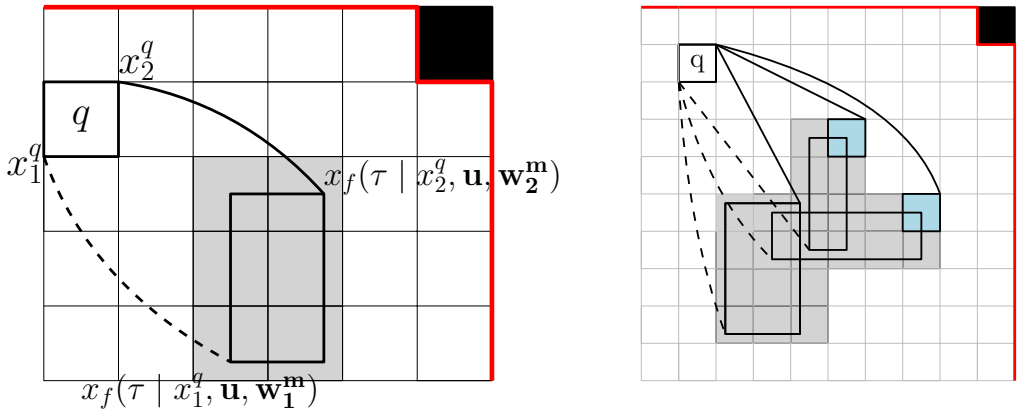


Figure 5.4: Illustration of difference between box and sparse abstractions. Left: A box abstraction for a fixed disturbance interval $[\mathbf{w_1^m}, \mathbf{w_2^m}]$, $m \in \{1, \ldots, M\}$. Right: box (grey) and sparse (blue) abstractions for a monotone system with a lower-closed safety specification for $M = 3$.

We then have the following corollary for SM systems.

Corollary 5.2.2. *Let us consider control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$. If $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ is a state cooperative system and if its symbolic box abstraction $\Sigma^B = (Q, U, F)$ satisfies Assumption 5.2.1, (respectively, Assumption 5.2.2), then $\Sigma^B = (Q, U, F)$ is a LSM (respectively, USM) transition system. Moreover, when both Assumptions 5.2.1 and 5.2.2 are satisfied, then $\Sigma^B = (Q, U, F)$ is an ISM transition system*

Hence, the monotonicity property is preserved when going from the original system to its symbolic abstraction. As soon as Assumption 5.2.1 or Assumption 5.2.2) or both is satisfied. Moreover, if the Q_S is lower-closed (upper-closed) with respect two the partial order $\preceq_{\mathbb{R}^{n_x}}$, and Assumption 5.2.1 (respectively, Assumption 5.2.2) holds, then the set Q_S is lower (respectively, upper) closed with respect to the partial order $\preceq_{Q_S}$. Here again, we look at the set Q_S either as a subset of $\mathbb{R}^{n_x}$, or as a collection of abstracts states. Let us remark that Q_S can be lower-closed with respect two the partial order $\preceq_{\mathbb{R}^{n_x}}$ and be finite only if there is $q \in Q_S$ such that $q = (-\infty, x_2^q)$.

5.2.3 Sparse Abstractions for (Input-)State Monotone Control Systems

In this section we introduce sparse abstractions for $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$. We then show that when we are dealing with directed specifications, sparse abstractions are equivalent to box abstraction $\Sigma^B = (Q, U, F)$ from the perspectives of synthesis. However, sparse abstractions are more efficient since they are less non-deterministic. To define a sparse abstraction, let us introduce the operators $\max$ and $\min$ for a finite partially ordered set.

Definition 5.2.3. *Let $\mathcal{L}$ be a partially ordered set and $A \subseteq \mathcal{L}$ is a finite subset. The set of minimal elements of A is defined as $\min(A) = \{q \in A \mid \forall q_1 \in A,\, q \preceq_{\mathcal{L}} q_1$ or $(q, q_1) \in Inc_{\mathcal{L}}\}$. Similarly, the set of maximal elements of A is defined as $\max(A) = \{q \in A \mid \forall q_1 \in A,\, q \succeq_{\mathcal{L}} q_1$ or $(q, q_1) \in Inc_{\mathcal{L}}\}$.*

For a control (input-)state monotone control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ an upper-sparse abstraction is defined as $\Sigma^{US} = (Q, U, F^{US})$ where sets Q, U are inherited from $\Sigma^B = (Q, U, F)$ and the transition relation is defined

for $q \in Q_S$, $u \in U$ as $F^{US}(q,u) = \max(F(q,u))$. Right sub-figure of Figure 5.4 illustrates the difference of an upper sparse abstraction $\Sigma^{US} = (Q,U,F^{US})$ and box abstraction $\Sigma^B = (Q,U,F)$. While for a state q the set of successors $F(q,u)$ include all grey boxes, the set $F^{US}(q,u)$ consist of two blue ones. Hence the abstraction $\Sigma^{US} = (Q,U,F^{US})$ is less non-deterministic than $\Sigma^B = (Q,U,F)$. Similarly, a lower-sparse abstraction is defined as $\Sigma^{LS}(\Sigma) = (Q,U,F^{LS})$ where Q,U are inherited from $\Sigma^B(\Sigma)$ and the transition relation is defined for $q \in Q$, $u \in U$ as $F^{US}(q,u) = \min(F(q,u))$.

Remark 5.2.2. *Let us remark that the transition relation of the upper sparse abstraction $\Sigma^{US} = (Q,U,F^{US})$ can be equivalently defined as follows: for all $q \in Q_S, u \in U$, $q' \in F^{US}$ if and only $q' \in \max(\cup_{m=1}^{M} q^m)$, where q^m is such that $x_f(\tau \mid x_2^q, \mathbf{u}, \mathbf{w_2^m}) \in q^m$ (see Figure 5.4 for an illustration).*

Although our definition of upper-sparse abstraction is slightly different from one proposed in [Kim et al., 2017], the transition system $\Sigma^{US} = (Q,U,F^{US})$ is still related to $\Sigma^B = (Q,U,F)$ with an upper alternating simulation relation [Kim et al., 2017].

We first show how to preserve the monotonicity when constructing upper or lower-sparse abstractions.

Proposition 5.2.2. *Let the box abstraction $\Sigma^B = (Q,U,F)$ of the control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ be LISM, then its upper-sparse abstraction $\Sigma^{US} = (Q,U,F^{US})$ is also LISM. Similarly, let the box abstraction $\Sigma^B = (Q,U,F)$ of the discrete-time control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ is UISM, then its lower-sparse abstraction $\Sigma^{US} = (Q,U,F^{US})$ is also UISM.*

Proof. We only provide a proof for the case of LISM, the case of UISM can be derived similarly. Consider $q_1, q_2 \in Q$, $u_1, u_2 \in U$ such that $q_1 \preceq_Q q_2$, $u_1 \preceq_U u_2$ and let $q_1' \in F^{US}(q_1,u_1) = \max(F(q_1,u_1))$, then $q_1' \in F(q_1,u_1)$. From Proposition 5.2.1, $\Sigma^B = (Q,U,F)$ is input-state monotone, then there exists $q_2' \in F(q_2,u_2)$ such that $q_1' \preceq_Q q_2'$. Coupling the last with the fact that there exists $\overline{q}_2' \in \max(F(q_2,u_2))$ such that $q_2' \preceq_Q \overline{q}_2'$ (see Definition 5.2.3), we finally get that $\Sigma^{US} = (Q,U,F^{US})$ is ISM. □

We then have the following corollary for SM systems.

Corollary 5.2.3. *Let the box abstraction $\Sigma^B = (Q,U,F)$ of the control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ be LSM, then the upper-sparse abstraction $\Sigma^{US} = (Q,U,F^{US})$ is also LSM. Similarly, let the box abstraction $\Sigma^B = (Q,U,F)$ of the control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ is USM, then its lower-sparse abstraction $\Sigma^{US} = (Q,U,F^{US})$ is also USM.*

Let us now show the equivalence between box and upper-sparse (respectively, lower-sparse) abstractions while synthesizing the maximal safety controller for lower-closed (respectively, upper-closed) safety specifications.

Proposition 5.2.3. *Consider the ISM control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$ If Q_S is a lower-closed safety specification, then the maximal safety controller $\bar{C}_B$ for the box abstraction $\Sigma^B = (Q,U,F)$ and the safety specification Q_S coincides with the maximal safety controller $\bar{C}_{US}$ for the upper-sparse abstraction $\Sigma^{US} = (Q,U,F^{US})$ and the safety*

specification Q_S. Similarly, if Q_S is an upper-closed safety specification, then the maximal safety controller $\bar{C}_B$ for the box abstraction $\Sigma^B = (Q, U, F)$ and the safety specification Q_S coincides with the maximal safety controller $\bar{C}_{LS}$ for the lower-sparse abstraction $\Sigma^{LS} = (Q, U, F^{LS})$ and the safety specification Q_S.

Proof. We only provide a proof for the case of lower-closed safety specifications, the case of upper-closed safety specifications can be derived similarly. Let us show that $\bar{C}_B$ is a safety controller for $\Sigma^{US} = (Q, U, F^{US})$. Indeed, since $\bar{C}_B$ is a safety controller for Σ^B it is obvious that $\mathrm{Dom}(\bar{C}_B) \subseteq Q_S$. Let $q \in \mathrm{Dom}(\bar{C}_B)$, then for all $u \in \bar{C}_B(q)$ the following is satisfied $F^{US}(q, u) \subseteq F(q, u) \subseteq \mathrm{Dom}(\bar{C}_B)$. Hence, $\bar{C}_B$ is a safety controller for $\Sigma^{US} = (Q, U, F^{US})$ and safe set Q_S. Then, from the definition of maximal safety controller one has $\bar{C}_B(q) \subseteq \bar{C}_{US}(q)$ for all $q \in Q$.

Let us show that $\bar{C}_{US}$ is a safety controller for $\Sigma^B = (Q, U, F)$. Again, it is obvious, that $\mathrm{Dom}(\bar{C}_{US}) \subset Q_S$. Let $q \in \mathrm{Dom}(\bar{C}_{US}), u \in \bar{C}_{US}(q)$ and let $q' \in F(q, u)$. Then there exists $\overline{q}' \in F^{US}(q, u) = \max(F(q, u))$ such that $q' \preceq_Q \overline{q}'$. Since $F^{US}(q, u) \subseteq \mathrm{Dom}(\bar{C}_{US})$ we have that $\overline{q}' \in \mathrm{Dom}(\bar{C}_{US})$. Moreover, since the safe set Q_S is lower-closed, we have from Proposition 5.3.1 that $\mathrm{Dom}(\bar{C}_{US})$ is lower-closed and one we get that $q' \in \mathrm{Dom}(\bar{C}_{US})$. Hence, $\bar{C}_{US}$ is a safety controller for $\Sigma^B = (Q, U, F)$ and from the maximality of the controller $\bar{C}_B$, one has $\bar{C}_{US}(q) \subseteq \bar{C}_B(q)$ for all $q \in Q$. □

We then have the following corollary for SM systems.

Corollary 5.2.4. *Consider the sate cooperative control system $\Sigma_f = (\mathcal{T}, \mathbb{R}^{n_x}, \mathcal{U}, \mathcal{W}, f)$. If Q_S is a lower-closed safety specification, then the maximal safety controller $\bar{C}_B$ for the box abstraction $\Sigma^B = (Q, U, F)$ and the safety specification Q_S coincides with the maximal safety controller $\bar{C}_{US}$ for the upper-sparse abstraction $\Sigma^{US} = (Q, U, F^{US})$ and the safety specification Q_S. Similarly, if Q_S is an upper-closed safety specification, then the maximal safety controller $\bar{C}_B$ for the box abstraction $\Sigma^B = (Q, U, F)$ and the safety specification Q_S coincides with the maximal safety controller $\bar{C}_{LS}$ for the lower-sparse abstraction $\Sigma^{LS} = (Q, U, F^{LS})$ and the safety specification Q_S.*

Intuitively, the result of Proposition 5.2.3 shows that we can use sparse abstractions instead of box abstraction while working with (input-) state monotone systems and directed specifications.

5.3 Maximal Safety Controller for Monotone Transition Systems and Directed Specifications

Considering a system $\Sigma = (Q, U, F)$ and a safety specification $Q_S \subseteq Q$, we solve, in this section, a synthesis problem that consists in determining a controller, which keeps the trajectories of the system inside a safe set Q_S.

Algorithm 5.1: ClassicalSynthesisMSC(Σ, Q_S)

Input: $\Sigma = (Q, U, F)$ and a safe set Q_S
Output: Maximal Safety Controller C

begin
  for $q \in Q_S$ **do**
    $p(q) := 1$;
  for $q \in Q \setminus Q_S$ **do**
    $p(q) := 0$;
  repeat
    $Q_E := \{q \in Q \mid p(q) \neq 0\}$;
    $B := \mathrm{Block}_{F_p}(Q_E)$;
    for $q \in B$ **do**
      $p(q) := p(q) - 1$;
  until $B = \emptyset$;
  for $q \in Q_E$ **do**
    $C(q) := \mathrm{Enab}_{F_p}(q)$;
  for $q \in Q \setminus Q_E$ **do**
    $C(q) := \emptyset$;
  return C;

Algorithm 5.2: LazySynthesisMSC(Σ, Q_S, I_{EB})

Input: $\Sigma = (Q, U, F)$, a safe set Q_S, an indicator I_{EB}
Output: Maximal Safety Controller C

begin
  for $q \in Q_S$ **do**
    $p(q) := 1$;
  for $q \in Q \setminus Q_S$ **do**
    $p(q) := 0$;
  repeat
    $Q_{EB} := \{q \in Q \mid p(q) \neq 0 \text{ and } I_{EB}(q) = 1\}$;
    $B := \mathrm{Block}_{F_p}(Q_{EB})$;
    for $q \in B$ **do**
      $p(q) := p(q) - 1$;
  until $B = \emptyset$;
  for $q \in \{q \in Q \mid p(q) \neq 0\}$ **do**
    $C(q) := \mathrm{Enab}_{F_p}(q)$;
  for $q \in Q \setminus \{q \in Q \mid p(q) \neq 0\}$ **do**
    $C(q) := \emptyset$;
  return C;

We focus on the case when $\Sigma = (Q, U, F)$ is a lower or upper (input-)state monotone, and a safety specification is directed. To synthesize the maximal safety controller [Tabuada, 2009]one can use the classical synthesis Algorithm 5.1 (repeats Algorithm 2.1 from Chapter 2), where the function $p\colon Q \to \{0, 1\}$ indicates whether a state $q \in Q$ is controllable ($p(q) = 1$) or not ($p(q) = 0$) and the reduced transition relation F_p is defined as follows: $(q, u, q') \in F_p$ if and only if $p(q) \neq 0$ and for all $q'' \in F(q, u)$ the equality $p(q'') \neq 0$ is satisfied. However, the classical approach is too source-demanding for many real-world problem. This section provides a more efficient lazy synthesis approach, benefiting from the particular structure of $\Sigma = (Q, U, F)$ and Q_S.

5.3.1 Lower-closed and Upper-closed Sets

Let us use operators $\max$ and $\min$ (see Definition 5.2.3) to introduce the notion of the basis for finite lower-closed and upper-closed sets [Finkel and Schnoebelen, 2001].

Definition 5.3.1. *Let $\mathcal{L}$ be a finite partially ordered set. Let $A \subseteq \mathcal{L}$ be a lower-closed (upper-closed) set. A set $B = \{s_1, \ldots, s_N\} \subseteq A$ is said to be the basis of A, denoted $B = Bas(A)$, if $B = \max(A)$ ($B = \min(A)$).*

The basis of a lower-closed (upper-closed) set can be used for its simpler representation. Indeed, let A is lower-closed (upper-closed) set and let $B = \mathrm{Bas}(A)$, then $A = \bigcup_{i=1,\ldots,N} \downarrow a_i$ ($A = \bigcup_{i=1,\ldots,N} \uparrow a_i$) and for all $a_i, a_j \in B$, if $a_i \neq a_j$ then $(a_i, a_j) \in \mathrm{Inc}_{\mathcal{L}}$. The existence and uniqueness of a finite basis for a finite lower-closed (upper-closed) set follow from the fact that the relation $\preceq_{\mathcal{L}}$ is a well-quasi-order [Higman, 1952]. An illustration of the concept of basis is given in Figure 5.5.

In the following two subsections, we consider a safety control problem for (input-)state monotone transitions system with directed safety specifications. We show that in a particular case when the transition system is lower

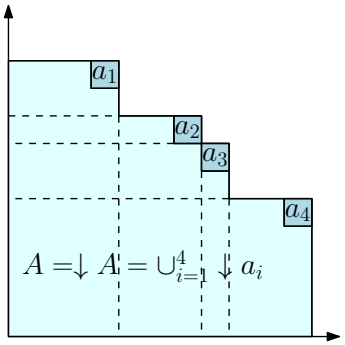


Figure 5.5: Illustration of Definition 5.3.1. A lower-closed set A and its basis $\mathrm{Bas}(A) = \{a_1, a_2, a_3, a_4\}$. The state-space is equipped with the component-wise partial order $\preceq$ defined on $\mathbb{R}^2$ and we have: $A = \downarrow A = \downarrow \mathrm{Bas}(A)$.

(input-)state monotone and the safe set is lower-closed, one can compute the maximal safety controller lazily. Analogous results hold for upper (input-)state monotone transition systems with upper-closed specifications, but we leave them unproven, taking into account to the similarity of the statements.

5.3.2 Lazy Synthesis for State Monotone Transition Systems

In this section, we aim to synthesize the maximal safety controller $\bar{C}$ for a LSM transition system $\Sigma = (Q, U, F)$ and a lower-closed safety specifications $Q_S \subseteq Q$. Looking for a lazy approach, we split the synthesis procedure into two steps. First, benefiting from the ordered structure of the state space, we efficiently compute the domain of the maximal safety controller. And then, we calculate the controller itself using the notion of u-predecessor of a set. Unlike the classical synthesis, where one explores all the states in Q_S and all the inputs U, in our approach, all inputs are explored, but not necessarily all the states.

Domain of Maximal Safety Controller

Let us start with a characterization of the domain of the maximal safety controller for LSM transition systems and lower-closed safety specifications.

Lemma 5.3.1. *Let us run Algorithm 5.1 for a LSM transition system $\Sigma = (Q, U, F)$ and lower-closed safety specification $Q_S \subseteq Q$, then at every iteration of loop 6-11 the set Q_E is lower-closed.*

Proof. Indeed, at the first iteration the set Q_E coincides with Q_S and, consequently, is lower closed. Assuming that the set Q_E^i is lower closed let us show that the set Q_E^{i+1} is lower-closed as well (here Q_E^i is a set Q_E at ith iteration of the loop 6-11). Since set Q_E^{i+1} always included in its lower closure we only need to show that $\downarrow Q_E^{i+1} \subseteq Q_E^{i+1}$. Suppose the opposite. Let there is $q \in \downarrow Q_E^{i+1}$ such that $q \notin Q_E^{i+1}$. On the one hand, since $q \in \downarrow Q_E^{i+1}$ there exists $q' \in Q_E^{i+1}$ such that $q \in \downarrow q'$. From lines 6-11 of Algorithm 5.1 we have that $Q_E^{i+1} = Q_E^i \setminus \mathrm{Block}_{F_p^i}(Q_E^i)$. Consequently, $q' \in Q_E^{i+1}$ is equivalent to $q' \in Q_E^i$ and $q' \notin \mathrm{Block}_{F_p^i}(Q_E^i)$. From the definition of reduced transition relation F_p^i, the

Algorithm 5.3: LazySynthesisDomainMSC(Σ, Q_S, I_{EB})

Input: $\Sigma = (Q, U, F)$, a safe set Q_S, an indicator I_{EB}
Output: Domain of MS controller C

```
1  begin
2      for q ∈ Q_S do
3          p(q) := 1;
4      for q ∈ Q \ Q_S do
5          p(q) := 0;
6      repeat
7          Q_EB := {q ∈ Q | p(q) ≠ 0 and I_EB(q) = 1};
8          B := Block_{F_p}(Q_EB);
9          for q ∈ B do
10             p(q) := p(q) − 1;
11     until B = ∅;
12     return {q ∈ Q | p(q) ≠ 0};
```

latter is true if and only if $p^i(q') = 1$ and there is $u \in U$, such that $F(q', u) \neq \emptyset$ and for all $q'' \in F(q', u)$ the following hold $p(q'') = 1$, i.e. in other words, $F(q', u) \subseteq Q_E^i$.

On the other hand, $q \notin Q_E^{i+1}$ if and only if either $p^i(q) = 0$ or $p^i(q) = 1$, but $q \in \mathrm{Block}_{F_p^i}(Q_E^i)$. Let $p^i(q) = 0$ then $q \notin Q_E^i$, the same time $q \in{\downarrow} q'$ and we have shown that $q' \in Q_E^i$. Thus, we got a contradiction to the fact that Q_E^i is lower closed. Let $p^i(q) = 1$ and $q \in \mathrm{Block}_{F_p^i}(Q_E^i)$. However, it also can not be true, since there is $u \in U$ such that $F(q', u) \neq \emptyset$, $q \succeq_Q q'$ and $\Sigma = (Q, U, F)$ is lower state monotone the set $F(q, u)$ is also non empty, moreover from Corollary 5.2.1 we have $F(q, u) \subseteq{\downarrow} F(q', u) \subseteq Q_E^i$. □

Corollary 5.3.1. *The domain of the maximal safety controller for a LSM transition system $\Sigma = (Q, U, F)$ and lower-closed safety specification $Q_S \subseteq Q$ is a lower-closed set.*

So, at every iteration of loop 6-11, the set Q_E is lower-closed, and from the first sight, $\mathrm{Bas}(Q_E)$ is a nice candidate for being the essential basis of the set Q_E (see Definition 2.3.1), but it is not. Indeed, we only can guarantee that if for all $q \in \mathrm{Bas}(Q_E)$ there exits $u \in U$ such that $F(q, u) \subseteq Q_E$ then for all $q \in Q_E$ there exits $u \in U$ such that $F(q, u) \subseteq Q_E$. The last statement is a consequence of the definition of a lower-closed set and the third item of Corollary 5.2.1. Since $\mathrm{Bas}(Q_E)$ is not the essential basis of the set Q_E Algorithm 5.2 (repeats Algorithm 2.2) does not return the maximal safety controller. Still, the following result shows that it returns a controller which domain coincides with $\mathrm{Dom}(\bar{C})$.

Theorem 5.3.1. *Let us run Algorithm 5.3 for a LSM transition system $\Sigma = (Q, U, F)$, lower-closed safety specification $Q_S \subseteq Q$ and the information function*

$$I_{EB}(q) = \begin{cases} 0 & \text{if } q \notin Bas(Q_E) \\ 1 & \text{if } q \in Bas(Q_E) \end{cases} \quad \text{where} \quad Q_E = \{q \in Q \mid p(q) \neq 0\} \tag{5.3}$$

Then Algorithm 5.3 returns the domain of the maximal safety controller.

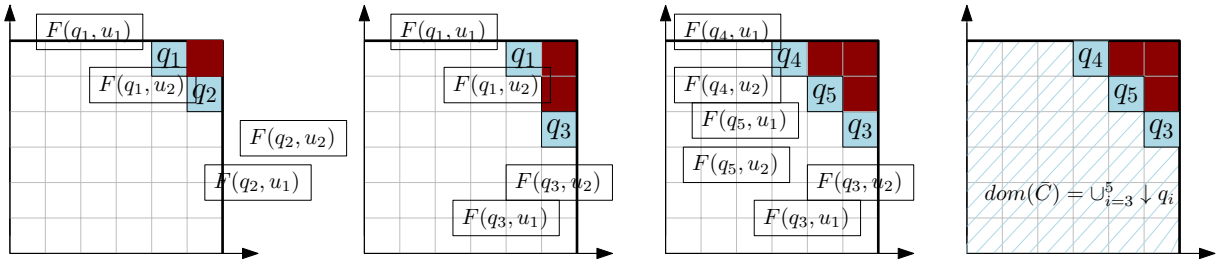


Figure 5.6: Illustration of Algorithm 5.3. If $p(q) = 1$, then q either blue or white. Red states are uncontrollable (i.e. $p(q) = 0$). Blue states are the basis of a set Q_E. The input set $U = \{u_1, u_2\}$.

Proof. Let us first remark that Algorithm 5.3 repeats Algorithm 5.2 until the line 12 and then instead of initializing a controller C it simply returns the domain of C. Thus, it is enough to prove that $\mathrm{Dom}(C) = \mathrm{Dom}(\bar{C})$, where C is a controller synthesised by Algorithm 5.2 for a LSM transition system $\Sigma = (Q, U, F)$, lower-closed safety specification $Q_S \subseteq Q$ and the information function defined by (5.3). And $\bar{C}$ is the maximal safety controller for or the transition system $\Sigma = (Q, U, F)$, and safety specification $Q_S \subseteq Q$. Since we know that $\bar{C}$ can be calculated with Algorithm 2.1, let us run Algorithm 2.1 in parallel to Algorithm 5.2. Let the sets $Q^i_{E,1}, Q^i_{E,2}$ contain all states from Q such that $p^i_1(q) \neq 0, p^i_2(q) \neq 0$ at the beginning of ith iteration of the loop 6-11, of Algorithm 2.1 and 5.2 correspondingly. Obviously, $Q^1_{E,1} = Q^1_{E,2}$. Let $Q^1_{E,i} \subseteq Q^2_{E,i}$ then $B^i_1 \subseteq B^i_2$ and consequently $Q^{i+1}_{E,1} \subseteq Q^{i+1}_{E,2}$. Hence, by induction we get that $\mathrm{Dom}(\bar{C}) \subseteq \mathrm{Dom}(C)$.

To show that $\mathrm{Dom}(C) \subseteq \mathrm{Dom}(\bar{C})$ let us first prove that $\mathrm{Dom}(C)$ is lower-closed. At the beginning of the loop 6-11 of Algorithm 5.2 the set $Q_E = \{q \in Q \mid p(q) \neq 0\}$ coincides with Q_S, and, hence, Q^1_E is lower-closed. Let at the ith iteration of the loop the set $Q^i_E = \{q \in Q \mid p^i(q) \neq 0\}$ is lower-closed. Since with the information function defined by (5.3) we reduce priorities for some states from $\mathrm{Bas}(Q^i_E)$ the set Q^{i+1}_E is also lower-closed. Thus, $\mathrm{Dom}(C) = \downarrow \mathrm{Dom}(C)$. Taking into account that $\Sigma = (Q, U, F)$ is LSM, we can define a new controller $\bar{C}$ for Σ as follows: $C^*(q) = \emptyset$ for all $q \in Q \setminus (\mathrm{Dom}(C))$, while for all $q \in \mathrm{Dom}(C)$ we set $C^*(q) = \cup_{\{q' \in \mathrm{Bas}(\mathrm{Dom}(C)) | q \in \downarrow q'\}} C(q')$. Indeed, if an input is enabled at a state $q' \in Q$ it is also enabled for all $q \in \downarrow q'$ and the definition of C^* is correct. Since $\mathrm{Dom}(C^*) = \mathrm{Dom}(C)$ to end the proof it is enough to show that C^* is a safety controller for the transition system $\Sigma = (Q, U, F)$, and safety specification $Q_S \subseteq Q$. Indeed, $\mathrm{Dom}(C^*) = \mathrm{Dom}(C) \subseteq Q_S$. Moreover, for all $q \in \mathrm{Dom}(C^*)$ for all $u \in C^*(q)$ there exist $q' \in \mathrm{Bas}(\mathrm{Dom}(C^*))$ such that $q \in \downarrow q'$, $F(q', u) \neq \emptyset$ and $F(q', u) \subseteq \mathrm{Dom}(C) = \mathrm{Dom}(C^*)$. From Corollary 5.2.1 we have that $F(q, u) \subseteq \downarrow F(q', u)$. The same time $\downarrow F(q', u) \subset \mathrm{Dom}(C*)$ since $\mathrm{Dom}(C^*)$ is lower-closed. So, C^* is a safety controller and $\mathrm{Dom}(C^*) \subseteq \mathrm{Dom}(\bar{C})$. □

Hence, to compute the domain of the maximal safety controller for LSM transition system and lower-closed safety specification, it is enough to explore only the basis of Q_E. Figure 5.6 illustrates execution of Algorithm 5.3 for a simple transition system. As soon as every state in Q_{EB} is non-blocking with respect to the reduced transition relation F_p, i.e. for every $q \in Q_{EB} = \mathrm{Bas}(Q_E)$ there exists $u \in U$ such that $F(q, u) \subseteq Q_E = \downarrow Q_{EB}$ we may conclude

Algorithm 5.4: $\mathbf{Pre}(A, u, B)$

Input: A LSM transition system $\Sigma = (Q, U, F)$, lower-closed sets $A, B \subseteq Q$, and an input $u \subseteq U$.
Output: $\mathrm{Bas}(\mathrm{Pre}(A, u, B))$

```
begin
    Q_U := ∅; Q_C := ∅;
    repeat
        Q_E := Bas(A \ Q_U) \ (Bas(A \ Q_U) ∩ Q_C);
        Q_C := Q_C ∪ {q ∈ Q_E | F(q, u) ⊆ B};
        Q_U := Q_U ∪ (Q_E \ {q ∈ Q_E | F(q, u) ⊆ B});
    until Q_E ≠ ∅;
    return Q_C;
```

that all states in Q_E are safely controllable.

Maximal Safety Controller

Once $\mathrm{Dom}(\bar{C})$ controller is obtained, we can use the notion of u-predecessors of a set to compute the maximal safety controller $\bar{C}$.

Definition 5.3.2. *For a transition system $\Sigma = (Q, U, F)$, sets $A, B \subseteq Q$ and input $u \in U$, we define the set $Pre(A, u, B)$ of u-predecessors of a set B included in a set A, as follows $Pre(A, u, B) = \{q \in A \mid F(q, u) \subseteq B\}$*

Let us now enumerate all inputs in U and introduce the sets $Z_i = \mathrm{Pre}(\mathrm{Dom}(\bar{C}), u_i, \mathrm{Dom}(\bar{C}))$ for all $u_i \in U$, $i = 1, \ldots, \mathrm{card}(U)$, where $\mathrm{card}(U)$ denotes the cardinality of the finite set U. Then the following result holds.

Theorem 5.3.2. *Let $\bar{C}$ be the maximal safety controller for a LSM transition system $\Sigma = (Q, U, F)$ and lower-closed safety specification $Q_S \subseteq Q$. Then for all $u_i \in U,\ i = 1, \ldots, card(U)$ the input $u_i \in \bar{C}(q)$ if and only if $q \in Z_i$.*

Proof. Indeed, the fact that $u_i \in \bar{C}(q)$ is equivalent to $q \in \mathrm{Dom}(\bar{C})$ and $F(q, u_i) \in \mathrm{Dom}(\bar{C})$, which is equivalent to $q \in \mathrm{Pre}(\mathrm{Dom}(\bar{C}), u_i, \mathrm{Dom}(\bar{C})) = Z_i$. □

In the classical approach, to find $\mathrm{Pre}(A, u, B)$ we should check for every state $q \in A$ if the set $F(q, u) \subseteq B$. However, the following result provides guidelines towards a more efficient computation of the $\mathrm{Pre}(A, u, B)$ in case when $\Sigma = (Q, U, F)$ is a SM transition system, and sets A, B are lower-closed.

Theorem 5.3.3. *Let $\Sigma = (Q, U, F)$ be a SM transition system, and sets A, B are lower-closed. Then the set $Pre(A, u, B)$ is also lower-closed. Moreover, for any lower-closed set $D \subseteq A$ the following holds: $D \subseteq Pre(A, u, B)$ if and only if for all $q \in Bas(D)$ the set $F(q, u) \subseteq B$.*

Proof. Indeed, if $q \in \mathrm{Pre}(A, u, B)$ then $q \in A$ and $F(q, u) \subseteq B$. Let $q' \leq_Q q$. Since A is a lower-closed set, then $q' \in A$. Moreover, we have from Corollary 5.2.1 that $F(q', u) \subseteq\downarrow F(q, u) \subseteq\downarrow B = B$. Hence, $q' \in \mathrm{Pre}(A, u, B)$ and $\mathrm{Pre}(A, u, B) =\downarrow \mathrm{Pre}(A, u, B)$.

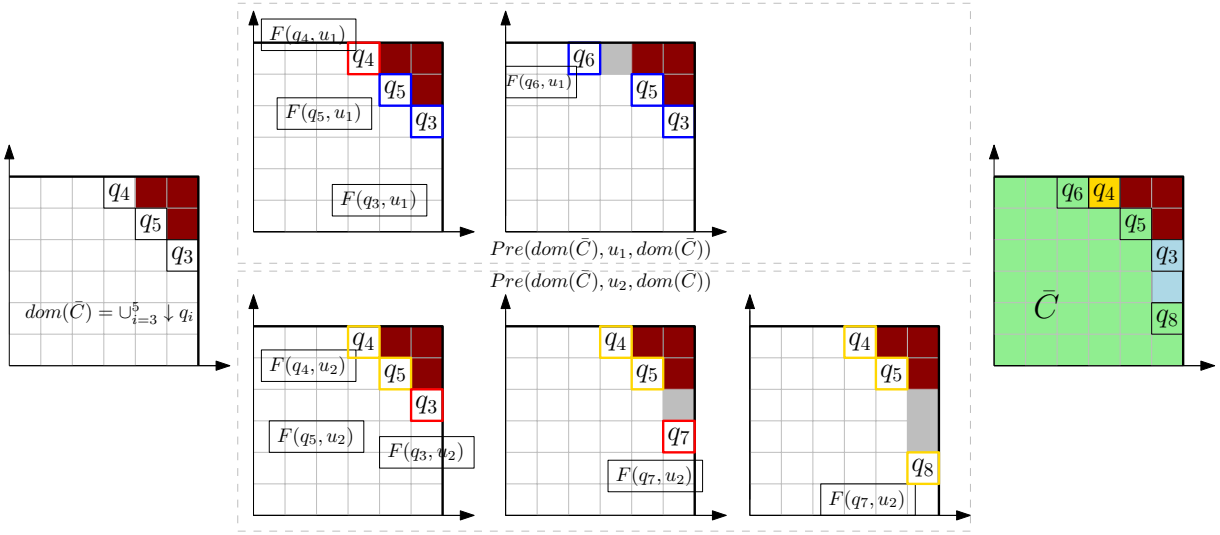


Figure 5.7: Illustration of lazy synthesis of the maximal safety controller for a LSM transition system with lower-closed specification. Red state are uncontrollable. White states belong to $\mathrm{Dom}(\bar{C})$. The left figure represents the $\mathrm{Dom}(\bar{C})$ and its basis $\{q_3, q_4, q_5\}$. Figures in the middle illustrate computation of the sets $Pre(\mathrm{Dom}(\bar{C}, u_1, \mathrm{Dom}(\bar{C})))$ and $Pre(\mathrm{Dom}(\bar{C}, u_2, \mathrm{Dom}(\bar{C})))$ by Algorithm 5.4. States filled with grey belong to Q_U. States contoured with colors belong to $Bas(\mathrm{Dom}(C) \setminus Q_U)$. States contoured with blue and yellow are controllable by inputs u_1 and u_2 correspondingly (i.e. belong to Q_C). The right figure illustrates the result of Theorem 5.3.2. States filled with yellow are controllable by u_2, with blue - by u_1, with green - by both u_1 and u_2.

Let now $D \subseteq A$, assuming that $D \subseteq \mathrm{Pre}(A, u, B)$ we immediately have from the Definition 5.3.2 that $F(q, u) \subseteq B$ for all $q \in \mathrm{Bas}(D)$ since $\mathrm{Bas}(D) \subseteq D \subseteq \mathrm{Pre}(A, u, B)$. Now let us prove the second implication. Let for all $q \in \mathrm{Bas}(D)$ the set $F(q, u) \subseteq B$. First, we have that $D \subseteq A$. For $q \in D$, there exists $q' \in \mathrm{Bas}(D)$ such that $q \leq_Q q'$. Hence, $F(q', u) \subseteq B$. Since $q \leq_Q q'$, we have from Corollary 5.2.1 that $F(q, u) \subseteq\downarrow F(q', u) \subseteq\downarrow B = B$. Hence, $D \subseteq \mathrm{Pre}(A, u, B)$. □

Corollary 5.3.2. *Let $\bar{C}$ be the maximal safety controller for a LSM transition system $\Sigma = (Q, U, F)$ and lower-closed safety specification $Q_S \subseteq Q$. Then for all $q_1, q_2 \in Q$, if $q_1 \leq_Q q_2$ then $\bar{C}(q_2) \subseteq \bar{C}(q_1)$.*

Intuitively, the set $\mathrm{Pre}(A, u, B)$ can be seen as the maximal by inclusion lower-closed subset C of B such that for all $q \in \mathrm{Bas}(C)$ the set $F(q, u) \subseteq B$. Hence, to compute we can use the following procedure. Initializing a set Q_E with a set $\mathrm{Bas}(A)$ at line 4 of Algorithm 5.4, we first explore every element of the Q_E. Those $q \in Q_E$ which are controllable (i.e., there exist an input $u \in U$ such that $F(q, u) \subseteq B$) we accumulate in a set Q_C, all the others in a set Q_U. Then, at the next iteration of loop 3-7, we shrink out all uncontrollable states from a set A and explore the basis of what is left. However, for states belonging to $\mathrm{Bas}(A \setminus Q_U) \cap Q_C$ we already know that they are controllable, so we focus on $q \in Q_E = \mathrm{Bas}(A \setminus Q_U) \setminus (\mathrm{Bas}(A \setminus Q_U) \cap Q_C)$. Then there are two options. If Q_E is non-empty, we should re-update the sets Q_U and Q_C (lines 5 and 6) and go for another loop iteration. While if Q_E is empty, we can stop the procedure. Indeed $Q_E = \emptyset$ if either all states in A are uncontrollable, or we find a set of controllable states Q_C, such that for any two states $q, q' \in Q_C$ are incomparable with respect to the order $\preceq_Q$. Moreover, for all states in A

greater than any state from Q_C were already explored and marked with uncontrollable. Hence, the set $\mathrm{Pre}(A, u, B)$ coincides with the set $\downarrow Q_C$ and $\mathrm{Bas}(\mathrm{Pre}(A, u, B)) = Q_C$.

In Algorithm 5.4, we leave all elements, which are smaller than the basis of $\mathrm{Pre}(A, u, B)$, unexplored. So, we go through all states in A only if $\mathrm{Pre}(A, u, B)$ is empty. This laziness gives us a gain in efficiency while computing the maximal safety controller. Indeed, in classical synthesis approach where one explores all the states in Q_S and all the inputs U, in our approach, all inputs are explored, but not necessarily all the states. Figure 5.7 illustrates the lazy synthesis of maximal safety controller for a simple transition system, such that it input set $U = \{u_1, u_2\}$. Given $\mathrm{Dom}(\bar{C})$ by Algorithm 5.3 we then compute two sets $Z_1 = Pre(\mathrm{Dom}(\bar{C}), u_1, \mathrm{Dom}(\bar{C}))$ and $Z_2 = Pre(\mathrm{Dom}(\bar{C}), u_2, \mathrm{Dom}(\bar{C}))$ using the Algorithm 5.4. At every iteration of the loop $3-7$ we check if a state $q \in Q_E$ is controllable with u_i. If yes, we put q into Q_C, if no - into Q_U. We then re-update Q_E, as follows $Q_E = \mathrm{Bas}(\mathrm{Dom}(C) \setminus Q_U) \setminus (\mathrm{Bas}(\mathrm{Dom}(C) \setminus Q_U) \cap Q_C)$. The right figure illustrates the result of Theorem 5.3.2, i.e. $u_i \in \bar{C}(q)$ if and only if $q \in Z_i$.

5.3.3 Lazy Synthesis for Input-State Monotone Transition Systems

In this part, we propose a lazy safety synthesis algorithm that exploits ordering not only on the state, but also on the input space. The synthesis of the maximal safety controller is done in two steps. First, we use only inputs with lower priorities to compute the maximal safety controller's domain $\mathrm{Dom}(\bar{C})$. Then we synthesize the maximal controller by exploiting the inputs priorities, making it possible to compute the maximal safety controller without exploring all the inputs. We start by providing some characterizations of the maximal safety controller for LISM transition systems.

Proposition 5.3.1. *Consider a LISM transition system $\Sigma = (Q, U, F)$. Let $\bar{C}$ be the maximal safety controller enforcing the lower-closed safety specification $Q_S \subseteq Q$. The following properties hold:*

(i) $\mathrm{Dom}(\bar{C})$ is lower-closed with respect to the partial order $\preceq_Q$;

(ii) for all $q_1, q_2 \in Q$, if $q_1 \preceq_Q q_2$ then $\bar{C}(q_2) \subseteq \bar{C}(q_1)$;

(iii) for all $q \in Q$, $\bar{C}(q)$ is a lower-closed set with respect to the partial order $\preceq_U$;

Proof. $(i), (ii)$ The result follows immediately from Corollary 5.3.1 and Corollary 5.3.2 and the fact that any LISM transition system is a LSM transition system.

(iii) Let $q \in Q$, $u \in \bar{C}(q)$ and $u' \in\downarrow u$. We have that $F(q, u') \subseteq\downarrow F(q, u) \subseteq\downarrow \mathrm{Dom}(\bar{C}) = \mathrm{Dom}(\bar{C})$, where the first inclusion comes from the fact that Σ is a LISM transition system, the second inclusion comes from the fact that $\bar{C}$ is a safety controller and the last equality comes from the lower-closedness of $\mathrm{Dom}(\bar{C})$. Hence, we have $F(q, u') \subseteq \mathrm{Dom}(\bar{C})$. Then, by maximality of $\bar{C}$, $u' \in \bar{C}(q)$. □

Domain of the controller

Let us define the set $U_{min} = \min(U)$ with respect to partial order $\preceq_U$ on the input set U. Then the following is true.

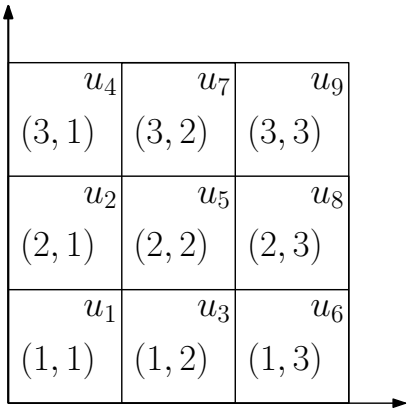


Figure 5.8: Illustration of the reordering $U = \{u_1, \dots, u_9\}$ of the input set $U = \{1,2,3\}^2$ with respect to the component-wise partial order $\preceq$ defined on $\mathbb{R}^2$.

Theorem 5.3.4. *Let $\bar{C}$ be the maximal safety controller for an LISM transition system $\Sigma = (Q, U, F)$ and lower-closed safety specification $Q_S \subseteq Q$. Let $\bar{C}_r$ be the maximal safety controller for the transition system $\Sigma_r = (Q, U_{min}, F)$ and safety specification Q_S. Then we have $\mathrm{Dom}(\bar{C}) = \mathrm{Dom}(\bar{C}_r)$.*

Proof. Let us define the controller C_r of the reduced transition system Σ_r and the safe set Q_S as follows: for $q \in Q$, $C_r(q) = \bar{C}(q) \cap U_{min}$. First let us prove that $\mathrm{Dom}(C_r) = \mathrm{Dom}(\bar{C})$. The inclusion $\mathrm{Dom}(C_r) \subseteq \mathrm{Dom}(\bar{C})$ follows immediately from the construction of the controller C_r. Now let $q \in \mathrm{Dom}(\bar{C})$ and let $u \in \bar{C}(q)$. From (iii) in Proposition 5.3.1 we have that $\downarrow u \subseteq \bar{C}(q)$, then there exists $u' \in U_{min}$ such that $u' \in \bar{C}(q)$. Then, $u' \in C_r(q)$. Hence, $q \in \mathrm{Dom}(C_r)$ and $\mathrm{Dom}(C_r) = \mathrm{Dom}(\bar{C})$ and C_r is a safety controller for Σ_r with a specification Q_S. Now let us prove that for all $q \in Q$, $C_r(q) = C_r^*(q)$. The first inclusion $C_r(q) \subseteq C_r^*(q)$ follows from maximality of the controller C_r^*. For the second inclusion, we have from maximality of $\bar{C}$ and since $U_{min} \subseteq U$ that $C_r^*(q) \subseteq \bar{C}(q)$ for all $q \in Q$. Moreover, by construction of C_r^*, we have that $C_r^*(q) \subseteq U_{min}$ for all $q \in Q$. Then, $C_r(q) = C_r^*(q)$ for all $q \in Q$. Since $\mathrm{Dom}(C_r) = \mathrm{Dom}(\bar{C})$, we have that $\mathrm{Dom}(C_r^*) = \mathrm{Dom}(\bar{C})$. □

The previous result states that for computation of the domain of the maximal safety controller $\mathrm{Dom}(\bar{C})$, it is sufficient to use inputs with lower priorities.

Maximal safety controller

It is always possible to reorder the elements of the input set $U = \{u_1, \dots, u_N\}$ as follows: for all $1 \leq j \leq i \leq N$ we suppose that either $u_j \preceq_U u_i$ or $(u_i, u_j) \in \mathrm{Inc}_U$ (see Fig. 5.8 for the illustration). In this section, we exploit such an order to make synthesis for LISM transition systems more efficient. Algorithm 5.5 is based on the fact, that if a state is uncontrollable with an input $u \in U$, it is also uncontrollable with all the inputs $u' \in U$ such that $u' >_U u$, so if we failed to control a state with u, we do not need to explore $u' >_U u$ for it. Let us now formally prove the result.

Algorithm 5.5: Maximal Safety Controller

Input: LISM transition system $\Sigma = (Q, U, F)$, the domain of maximal safety controller $\mathrm{Dom}(\bar{C})$.
Output: Controller C.

```
1  begin
2      for s ∈ Q do
3          C(s) := ∅;
4      for i = 1 : N do
5          if u_i ∈ U_min then
6              S_i = Dom(C̄);
7              K_i = Pre(S_i, u_i, Dom(C̄));
8          else
9              S_i = ∩_{u_j ≺_U u_i} K_j;
10             K_i = Pre(S_i, u_i, Dom(C̄));
11     for s ∈ K_i do
12         C(s) := C(s) ∪ {u_i};
13     return C;
```

Lemma 5.3.2. *At every iteration of the loop 4-11 of the Algorithm 5.5 the set* $K_i = Pre(S_i, u_i, \mathrm{Dom}(\bar{C}))$ *coincides with the set* $Z_i = Pre(\mathrm{Dom}(\bar{C}), u_i, \mathrm{Dom}(\bar{C}))$.

Proof. To prove the result, we proceed by induction. For all i such that $u_i \in U_{min}$ the statement is obvious and we have the base. Let i be such that $u_i \notin U_{min}$. Suppose that for all $j < i$, $Z_j = K_j$, and let us prove that $Z_i = K_i$. Indeed, since for all $j < i$, $K_j = Z_j$ we have that $K_j \subseteq \mathrm{Dom}(\bar{C})$ for all $j < i$, and as a consequence $S_i \subseteq \mathrm{Dom}(\bar{C})$ (see line 8). From where $K_i \subseteq Z_i$ is immediately satisfied. At the same time, for all $q \in Z_i$ we have that $q \in \mathrm{Dom}(\bar{C})$ and $F(q, u_i) \subseteq \mathrm{Dom}(\bar{C})$. Then from Proposition 5.3.1 we have that $q \in \mathrm{Dom}(\bar{C})$ and for all $u_j \prec_U u_i$, $F(q, u_j) \subseteq \mathrm{Dom}(\bar{C})$. Consequently $q \in \mathrm{Pre}(\mathrm{Dom}(\bar{C}), u_j, \mathrm{Dom}(\bar{C})) = Z_j = K_j$ for all j such that $u_j \prec_U u_i$. Then from the line 9 of Algorithm 5.5, we have that $q \in S_i$ and, as a consequence, from line 10, $q \in K_i$. □

Theorem 5.3.5. *Let* $\Sigma = (Q, U, F)$ *be a LISM transition system and let* $\mathrm{Dom}(\bar{C})$ *be a domain of the maximal safety controller* $\bar{C}$ *for a lower-closed safety specification* $Q_S \subseteq Q$*. Algorithm 5.5 returns the maximal safety controller* $\bar{C}$*.*

Proof. The statement follows immediately from Lemma 5.3.2, Theorem 5.3.2 and the fact that $u_i \in C(s)$ if and only if $s \in K_i$ (see lines 11-12 of the Algorithm 5.5). □

Remark 5.3.1. *In the case of a total order*[1] *on the input set* U*, the set* $K_i \subseteq K_j$ *for all* $N \geq j \geq i \geq 1$ *(see Fig. 5.9).*

For LISM transition systems and directed safety specifications, one can use the monotonicity property to represent the maximal safety controller $\bar{C}$ efficiently. Indeed, we have from Proposition 5.3.1 that for a controllable state $q \in \mathrm{Dom}(\bar{C})$, $\bar{C}(q)$ is a lower-closed set. Hence, for the state q, instead of storing all the admissible control inputs $u \in \bar{C}(q)$, one can only store the elements of the basis of $\bar{C}(q)$. The latter significantly reduces the amount of

[1] A binary relation $\preceq_{\mathcal{L}} \subseteq \mathcal{L} \times \mathcal{L}$ is a total order if it is a partial order and for all $l_1, l_2 \in \mathcal{L}$ we have either $l_1 \preceq_{\mathcal{L}} l_2$ or $l_2 \preceq_{\mathcal{L}} l_1$.

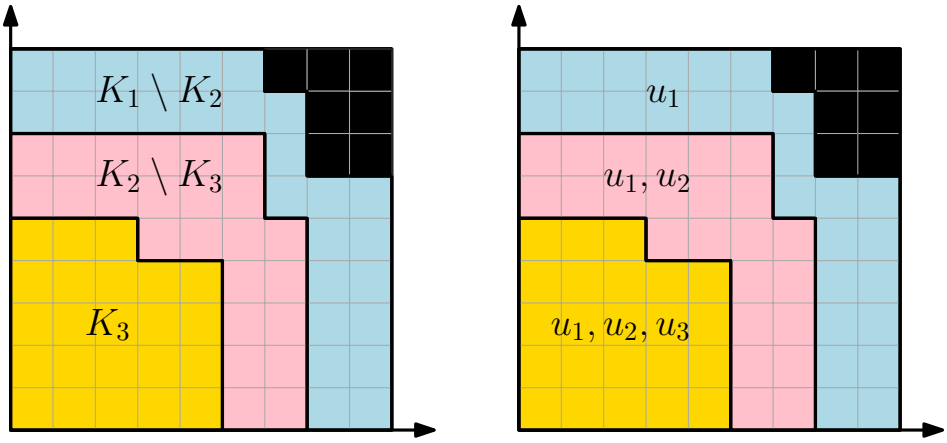


Figure 5.9: Illustration of the maximal safety controller $\bar{C}$ for the case of a total order on the input set. $U = \{u_1, u_2, u_3\}$ with $u_1 \preceq_U u_2 \preceq_U u_3$.

memory required to store the controller. Hence, instead of providing the maximal safety controller $\bar{C}$, in practice, we provide the maximal input-lazy controller. However, we can reconstruct MS controller from MILS controller without any computations in a particular case of LISM systems.

5.3.4 Controller Synthesis for Intersections of Directed Safety Specifications

In the previous sections, we have presented lazy synthesis algorithms to deal with lower (input-)state monotone transitions systems and lower-closed safety specifications. Following the duality between upper and lower-closed sets, similar results for upper-closed safety specifications can be obtained using the same approaches.

This part aims to deal with more complex specifications described as intersections of upper and lower-closed sets. Let us mention that from Proposition 5.1.1, the intersection of lower (respectively upper) closed sets is again a lower (respectively upper) closed set. Hence, one can use the same approaches presented in the previous sections to deal with unions of lower (respectively upper) closed specifications.

In this section, we focus on the synthesis of controllers for the intersection of an upper and lower-closed set, which is a natural specification appearing in many applications such as vehicle platoons [Nilsson et al., 2016], microgrids [Zonetti et al., 2019], traffic networks [Coogan et al., 2016] and temperature regulation [Meyer et al., 2013]. The considered setup is the following. Given a (input-)state monotone transition system $\Sigma = (Q, U, F)$, a lower-closed set $Q_L \subseteq Q$ and an upper-closed set $Q_U \subseteq Q$ with $Q_L \cap Q_U \neq \emptyset$, let us consider the problem of synthesizing the maximal safety controller for the transition system Σ and safety specification $Q_{LU} = Q_L \cap Q_U$. We first have the following preliminary result

Lemma 5.3.3. *Consider the (input-)state monotone transition $\Sigma = (Q, U, F)$, the lower-closed $Q_L \subseteq Q$ and the upper-closed set $Q_U \subseteq Q$ with $Q_L \cap Q_U \neq \emptyset$. Let us define the following safety controllers:*

- *$\bar{C}_{LU}$ is the maximal safety controller for the transition system Σ and the safety specification $Q_{LU} = Q_L \cap Q_U$;*

- *$\bar{C}_L$ is the maximal safety controller for the transition system Σ and the lower-closed safety specification Q^L;*
- *$\bar{C}_U$ is the maximal safety controller for the transition system Σ and the upper-closed safety specification Q^U.*

Then, for all $q \in Q$, $\bar{C}_{LU}(q) \subseteq \bar{C}_L(q) \cap \bar{C}_U(q)$.

Proof. Since $Q_{LU} \subseteq Q_L$, we have that $\bar{C}_{LU}(q) \subseteq \bar{C}_L(q)$ for all $q \in Q$. Similarly, using the fact that $Q_{LU} \subseteq Q_U$ we have that $\bar{C}_{LU}(q) \subseteq \bar{C}_U(q)$ for all $q \in Q$. Hence, we have that $\bar{C}_{LU}(q) \subseteq \bar{C}_L(q) \cap \bar{C}_U(q)$ for all $q \in Q$. □

The idea of this section is to incrementally synthesize the controller $\bar{C}_{LU}$ by proceeding in two steps: 1) synthesize the controllers $\bar{C}_U$ and $\bar{C}_L$; 2) synthesize the maximal safety controller for the transition system Σ and safety specification $\mathrm{Dom}(\bar{C}_L) \cap \mathrm{Dom}(\bar{C}_U)$, where for each state $q \in \mathrm{Dom}(\bar{C}_L) \cap \mathrm{Dom}(\bar{C}_U)$, we explore only the inputs $u \in \bar{C}_L(q) \cap \bar{C}_U(q)$. The completeness of the proposed incremental synthesis concerning the classical synthesis is shown in the following result.

Proposition 5.3.2. *Under the preliminaries of Lemma 5.3.3, let us define the set $Z = \mathrm{Dom}(\bar{C}_L) \cap \mathrm{Dom}(\bar{C}_U)$. Let $\bar{C}_Z$ be the maximal safety controller for the transition system Σ and safety specifications Z. Then, for all $q \in Q$, $\bar{C}_{LU}(q) = \bar{C}_Z(q)$.*

Proof. Using the fact that $Z = \mathrm{Dom}(\bar{C}_L) \cap \mathrm{Dom}(\bar{C}_U) \subseteq Q_{LU} = Q_L \cap Q_U$, it follows that $\bar{C}_Z(q) \subseteq \bar{C}_{LU}(q)$ for all $q \in Q$. On the other hand, we have from Lemma 5.3.3 that $\mathrm{Dom}(\bar{C}_{LU}) \subseteq \mathrm{Dom}(\bar{C}_L) \cap \mathrm{Dom}(\bar{C}_U) = Z$. Hence, $\bar{C}_{LU}(q) = \bar{C}_Z(q)$ for all $q \in Q$, which ends the proof. □

Certainly, the attentive reader may remark that execution of both steps of the synthesis procedure proposed above is equivalent to the execution of the most classical Algorithm 2.1 when we apply it for a pre-computed transition system. However, let us remind that in the first step, we can use more efficient sparse abstractions. And since the symbolic model is calculated on-the-fly the latter speeds up the computations. Moreover, in some particular cases, one can get the maximal safety controller $\bar{C}_{LU}$ directly from the controllers $\bar{C}_L$ and $\bar{C}_U$, without running the second step at all.

Proposition 5.3.3. *Under the preliminaries of Lemma 5.3.3, if for all $q \in \mathrm{Dom}(\bar{C}_L) \cap \mathrm{Dom}(\bar{C}_U)$, $\bar{C}_L(q) \cap \bar{C}_U(q) \neq \emptyset$, then for all $q \in Q$, $\bar{C}_{LU}(q) = \bar{C}_L(q) \cap \bar{C}_U(q)$.*

Proof. The first inclusion follows from Lemma 5.3.3. To deal with the second inclusion, let us show that the controller $C_{LU} = \bar{C}_L \cap \bar{C}_U$ is a safety controller for the transition system Σ and safety specification Q_{LU}. First, we have that $\mathrm{Dom}(C_{LU}) \subseteq \mathrm{Dom}(\bar{C}_L) \cap \mathrm{Dom}(\bar{C}_U) \subseteq Q_L \cap Q_U = Q_{LU}$. Hence, the first condition of Definition 2.2.1 is satisfied. Now let $q \in \mathrm{Dom}(C_{LU})$ and $u \in C_{LU}(q) = \bar{C}_L(q) \cap \bar{C}_U(q)$, the existence of such u is guaranteed by the fact that $\bar{C}_L(q) \cap \bar{C}_U(q) \neq \emptyset$. Since $\bar{C}_L$ and $\bar{C}_U$ are safety controllers, we have that $F(q,u) \subseteq \mathrm{Dom}(\bar{C}_L) \cap \mathrm{Dom}(\bar{C}_U) = \mathrm{Dom}(C_{LU})$. Hence, the second condition of Definition 2.2.1 is satisfied. Then, C_{LU} is a safety controller for the transition system Σ and safety specification Q_{LU}. Hence, from maximality of the controller $\bar{C}_{LU}$, we have that $C_{LU}(q) = \bar{C}_L(q) \cap \bar{C}_U(q) \subseteq \bar{C}_{LU}(q)$, for all $q \in Q$, which ends the proof. □

5.4 Numerical Illustration: Adaptive Cruise Control

In this Chapter we again consider two vehicles moving along a straight road. Each vehicle is modeled as a point mass m with velocity changing according to the law

$$m\dot{v} = \alpha(F, v) = F - (f_0 + f_1 v + f_2 v^2). \tag{5.4}$$

In the equation above, F represents a net action of braking and engine torque applied to the wheels, while the second term $f_0 + f_1 v + f_2 v^2$ describes aerodynamic and rolling resistance effects. In simulations

$$m = 1370\,Kg, \quad f_0 = 51.0709\,N, \quad f_1 = 0.3494\,Ns/m \quad f_2 = 0.4161\,Ns^2/m^2.$$

The net force F is viewed as a control input u for the follower vehicle and a disturbance for the lead one. It is assumed to be bounded by $F_{min} = -0.3mg \leq F \leq 0.2mg = F_{max}$, where g is a gravitational constant. Such a bound is consistent with non-emergency braking and acceleration. We slightly adapt equation (5.4) to prohibit a back motion for the follower:

$$m\dot{v} = \begin{cases} \alpha(u, v) & \text{if} \quad v > 0 \\ \max(u - f_0, 0) & \text{if} \quad v = 0 \end{cases} \tag{5.5}$$

For the leader we assume that its velocity w remains in a range $[0, w_{max}]$:

$$m\dot{w} = \begin{cases} \alpha(a, w) & \text{if} \quad 0 < w < w_{max} \\ \max(\alpha(a, w), 0) & \text{if} \quad w = 0 \\ \min(\alpha(a, w), 0) & \text{if} \quad w = w_{max} \end{cases} \tag{5.6}$$

Combining (5.5) and (5.6) with the equation

$$\dot{d} = w - v, \tag{5.7}$$

which describes the distance between two vehicles, we obtain the final model of the system.

5.4.1 Control Objective and Numerical Results

Lower-closed Safety Specification

We start with the objective to synthesize a controller for the follower vehicle, to keep its velocity below $v_{\max} = 30\,m/s$ and to guarantee that the relative distance between the leader and the follower remains larger than $d_{\min} = 10\,m$, while assuming that the leader car acts as a disturbance $a \in [F_{min}, F_{max}]$ and $w_{max} = 30\,m/s$. The following change of coordinates: $h = -d$, $z = -w$ transform the system (5.5)-(5.7) into a monotone one. Moreover, after this change

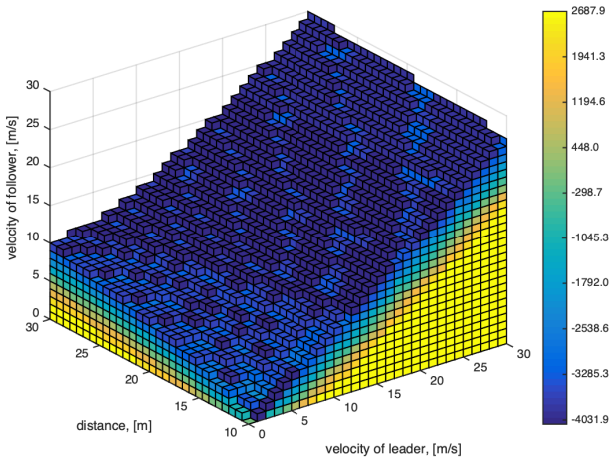


Figure 5.10: Simulation results. Maximal safety controller $\bar{C}$ for a lower-closed safety specification.

of coordinates, we get a lower-closed safety specification.

Setting a time step $\tau = 0.8\,s$, we generate a discrete-time model corresponding to the continuous-time system (5.5)-(5.7). We also introduce a Cartesian partition on the set X, such that a safe set $Y \subset X$, and input space $\mathcal{U}$ with $n_x = (31, 31, 31)$ and $n_u = 50$ as state and input discretization parameters correspondingly. We then construct a sparse symbolic model and use Algorithm 5.5 to synthesize the maximal safety controller $\bar{C}$ for the abstraction. Let us remind that if for a state $q \in X$ an input $u \in \bar{C}(x)$, then all inputs satisfying $u' \preceq_U u$ are enabled by $\bar{C}$ for this state. Moreover, in our example, we have total order on the input space U. Hence, for each state, it is enough to store only the maximal safe input to reconstruct the whole maximal safety controller (i.e. in fact we synthesize and store maximal input-lazy safety controller). Consequently, the amount of memory required for the controller implementation is significantly reduced. We represent the obtained maximal safety controller $\bar{C}$ in Figure 5.10. The blue color of the color bar corresponds to the minimal input $F_{\min} = -4031.9\ mKg/s^2$ and the yellow color corresponds to the maximal input $F_{\max} = 2687.9\ mKg/s^2$. For a given state $x = (w, d, v)$, Figure 5.10 shows the maximal allowed control input. Let us remark that to refine the abstract controller for the original system we can either use the classical refinement scheme (Section 1.3) or even consider a closed-loop controller $u\colon X \to \mathcal{U}$ ensuring more difficult specifications [Darbha, 1997] as soon as $u(x) \preceq_{\mathcal{U}} \bar{C}(q)$, where q is a state such that $x \in q$.

To evaluate the performance of our approach, we compare it with the classical safety synthesis algorithm while exploring different scenarios: we compare our approach with the box abstractions [Tabuada, 2009] and the sparse abstractions [Kim et al., 2017]. We also provide a comparison with lazy synthesis procedure, based on three valued abstractions [Hussien and Tabuada, 2018]. In [Hussien and Tabuada, 2018], once a safe input is found for a state, the other controls are not explored for this state. The authors start with an empty transition relation and iteratively add new transitions essential for the synthesis purpose. At every iteration, the precomputed part of the symbolic model is explored. Then, one randomly chooses $N_{3v} \in \mathbb{N}_{>0}$ states from uncontrollable states, where N_{3v} is a parameter

Table 5.1: Runtime comparison when varying the number of states. T^s_{lm}, T_{cl}, T^s_{cl} and T^s_{3v} are the running time of Algorithm 5.5, the classical fixed point algorithm with box abstractions [Tabuada, 2009] and with sparse abstractions [Kim et al., 2017] and the lazy algorithm from [Hussien and Tabuada, 2018], respectively.

n_x	T^s_{lm}	T_{cl}/T^s_{lm}	T^s_{cl}/T^s_{lm}	T^s_{3v}/T^s_{lm}
(15,15,15)	2.96 s	31.22	15.52	11.18
(31,31,31)	18.16 s	53.35	23.19	14.92
(63,63,63)	118.67 s	85.21	30.85	16.25

Table 5.2: Memory comparison when varying the number of states. M^s_{lm}, M_{cl}, M^s_{cl} and M^s_{3v} are the required memory to implement the controller resulting from Algorithm 5.5, the classical fixed point algorithm with box abstractions [Tabuada, 2009], the classical fixed point algorithm with sparse abstractions [Kim et al., 2017] and the lazy algorithm from [Hussien and Tabuada, 2018], respectively.

n_x	$M^s_{lm} = M^s_{3v}$	$M_{cl} = M^s_{cl}$	M_{cl}/M_{lm}
(15,15,15)	26.4 KB	499.9 KB	18.93
(31,31,31)	232.7 KB	4729 KB	20.32
(63,63,63)	1953.5 KB	41469.3 KB	21.22

given by user. For each of chosen states, one randomly picks an unexplored input and adds the corresponding tradition to the abstraction. The algorithm returns only the domain of the maximal safety controller and not the whole maximal safety controller. Indeed, for every state, only one safe input is returned. Moreover, since the usage of sparse abstractions speed-up synthesis process we also use them when implementing the approach from [Hussien and Tabuada, 2018]. The evaluation is based on two criteria, the computation time and the memory required to implement the maximal safety controller. We explore two different scenarios. In the first case, we vary the state-space discretization parameter n_x while keeping the input discretization parameter as a constant $n_u = 10$. The results of run time and memory comparison are represented in Table 5.1 and Table 5.2. In the second case, we set $n_x = (31, 31, 31)$ and vary the input discretization parameter n_u. The computational results are given in Table 5.3 and Table 5.4.

In Tables 5.1, 5.3, time T^s_{lm} is a running time of Algorithm 5.5. Time T_{cl} is a running time of the classical fixed point algorithm when box abstractions are used [Tabuada, 2009]. Time T^s_{cl} is a running time of the classical fixed point algorithm when sparse symbolic models are used [Kim et al., 2017]. Time T^s_{3v} is a running time of the lazy algorithm from [Hussien and Tabuada, 2018] implemented for sparse abstractions. For the parameter N_{3v} the value $round(0.6 * n_x(1) * n_x(2) * n_x(3))$ have been chosen. We also store the amount of memory needed for controllers implementation in variables M^s_{lm}, M_{cl}, M_{cl} and M^s_{3v} correspondingly. Let us remind that for ISM with directed specifications the controllers synthesised for sparse abstractions and box abstractions coincide, hence $M_{cl} = M^s_{cl}$ (column 3 in Tables 3,5). However, it is obvious from columns 3 and 4 of Tables 2,4 that sparse abstractions-based synthesis is much faster, so we repeated the result from [Kim et al., 2017]. Since our controller and 3-valued abstractions-based controller store just one safe input for every state, their memory requirements also coincide (column 2 in Tables 3,5). However, we store the maximal safe input for every state. We can easily reconstruct the maximal safety controller without any computations. In contrast, the three-valued abstractions-based

Table 5.3: Runtime comparison when varying the number of inputs. T^s_{lm}, T_{cl}, T^s_{cl} and T^s_{3v} are the running time of Algorithm 5.5, the classical fixed point algorithm with box abstractions [Tabuada, 2009] and with sparse abstractions [Kim et al., 2017] and the lazy algorithm from [Hussien and Tabuada, 2018], respectively.

n_u	T^s_{lm}	T_{cl}/T^s_{lm}	T^s_{cl}/T^s_{lm}	T^s_{3v}/T^s_{lm}
10	18.19 s	54.01	23.42	15.06
20	20.56 s	95.08	41.08	25.96
40	25.26 s	154.03	66.67	43.25

Table 5.4: Memory comparison when varying the number of states. M^s_{lm}, M_{cl}, M^s_{cl} and M^s_{3v} are the required memory to implement the controller resulting from Algorithm 5.5, the classical fixed point algorithm with box abstractions [Tabuada, 2009], the classical fixed point algorithm with sparse abstractions [Kim et al., 2017] and the lazy algorithm from [Hussien and Tabuada, 2018], respectively.

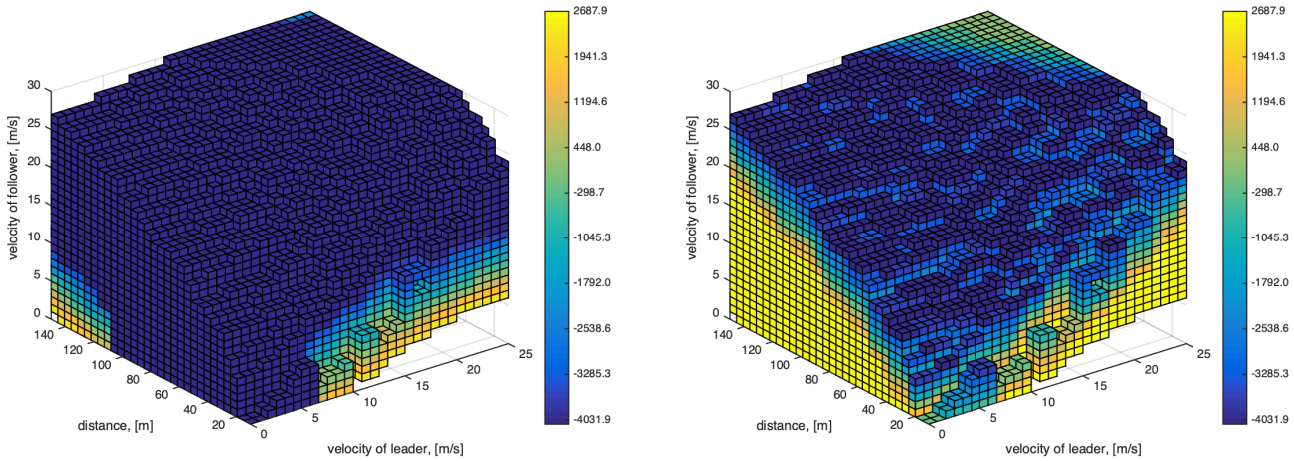
n_u	$M^s_{lm} = M^s_{3v}$	$M_{cl} = M^s_{cl}$	M_{cl}/M_{lm}
10	232.7 KB	4729 KB	20.32
20	232.7 KB	6200.1 KB	26.64
40	232.7 KB	9141.8 KB	39.29

controller store just a random safe input, and to get the maximal safety controller, one should check for all the other inputs if they allow to remain in the controllable domain. The latter is not efficient when the set of discrete inputs is large. The numerical results highlight the practical speedups and memory efficiency that can be attained using the lazy approach while ensuring completeness with respect to the classical safety algorithm.

Intersection of lower and upper-closed safety specifications

Let us consider another control objective to illustrate the results of Section 5.3.4. We want to synthesise a controller for the follower vehicle, which takes its values from $[F_{min}, F_{max}]$, to keep the velocity of the follower below $v_{\max}$ and guarantees that the relative distance between the leader and the follower remains larger than $d_{\min} = 10\,m$ and smaller than $d_{\max} = 150\,m$, while assuming that the leader vehicle acts as a disturbance $d \in [0.65 * F_{min}, 0.65 * F_{max}]$ and $w_{max} = 25\,m/s$. In this case, we can use the approach proposed in Proposition 5.3.2 to speed up the computation by a factor of 3.7 in comparison to the classical approach, or by 5.1 if me combine the the incremental synthesis approach with ideas from Chapter 2. Time gain is a result of using a sparse abstraction, instead of a more common box abstraction, to find $\bar{C}_L$ and $\bar{C}_U$ and then use them as a warm point for the following computations. Since our abstraction is input-state monotone and we have the total order on set of inputs, it is sufficient for every state x to store only safe actions $u^s_{min}(x),\ u^s_{max}(x)$ with minimal and maximal value, because all inputs in a range $[u^s_{min}(x), u^s_{max}(x)]$ are admissible. See the maximal controller in the Figure 5.11.

If we set up the parameter d_{max} as $200\,m$ the assumption of Proposition 5.3.3 is satisfied and $\bar{C}_{LU}(x) = \bar{C}_L(x) \cap \bar{C}_U(x)$. In this case we can fully used the benefits of the Algorithm 5.5, and we are 45.7 times faster than the classical approach. All the implementations has been done in MATLAB, Processor Intel Core i7-8700, 3.20 Hg, RAM 16 GB.



(a) Minimal value

(b) Maximal value

Figure 5.11: Simulation results. Maximal safety controller $\bar{C}$ for an intersection of lower-closed and upper-closed safety specification.

5.5 Conclusion

In this Chapter, we provide a lazy control synthesis algorithm for monotone transition systems and directed safety specifications. Two classes of monotone transition systems are presented: state monotone transition systems and input-state monotone transition systems. For the first class of systems, a partial order is defined only on the state space. For the second, the input space is ordered as well. The introduced lazy synthesis approach benefits from the ordered structure of the state (input) space and the fact that directed safety specifications are considered. To enrich the class of the considered specifications, we also present an incremental controller synthesis framework, which allows us to deal with intersections of upper and lower-closed safety requirements. We then compare the proposed approach with the classical safety synthesis algorithm and illustrate the advantages, in terms of run-time and memory efficiency, on an adaptive cruise control problem.

Chapter 6

Conclusion and Future Work

A tendency to automate most life processes makes cyber-physical systems (CPS) are irreplaceable in modern society. However, their design is still challenging, and one of the major issues is how to guarantee that the behavior of CPS satisfies the safety constraints. Indeed, a sound synthesis of safety controllers requires model-based approaches. At the same time, the heterogeneous nature of models motivates the usage of abstraction-based techniques, which unfortunately suffer from poor scalability. In this manuscript, we improved the efficiency of abstraction-based approaches by developing algorithms that synthesize the controllers for symbolic models lazily. The proposed algorithms outperform the classical synthesis procedure [Tabuada, 2009] by avoiding computations, which are non-essential for synthesis goals.

Chapter 2 introduced three lazy synthesis approaches applicable for a general non-deterministic transition system $\Sigma = (Q, U, F)$. Algorithm 2.2 avoids exploration of states and transitions which are safely controllable a priory. It returns the maximal safety controller but requires external knowledge about abstractions, which is not always available. Algorithm 2.3 assumes that the input set U is equipped with a partial order and then explores the lower-priority actions only if the problem is unsolvable with higher-priority actions. It returns the maximal input-lazy safety controller, whose domain coincides with the maximal controllable set, but only inputs with the highest priority are enabled among several inputs preserving safety. Algorithm 2.4 assumes that the initial set $Q_{init} \subset Q$ is known and avoids exploration of non-reachable part of abstraction. It returns the maximal state-lazy safety controller, whose domain includes a safely controllable state if and only if it is reachable from Q_{init}. Using lazy algorithms, we can synthesize the abstract controller more efficiently than with the classical Algorithm 2.1. Moreover lazy synthesis allows us to calculate only the part of the abstraction which is involved in the synthesis procedure. Thus, the performance of symbolic model-based approaches is improved.

In Chapter 3 we used abstraction-based techniques to synthesize a safety controller for a general continuous-time control system. The main idea was is to create a neighbor-linked abstraction, where only transitions between neighboring states are allowed. To enforce the desired structure of the symbolic model, we introduced a novel

adaptive time-sampling technique. Unlike existing approaches, where transitions duration was determined with a given parameter, we proposed interrupting every transition before leaving the abstract state neighborhood. We then use Algorithm 2.2 to synthesize the maximal safety controller lazily by iteratively exploring only those states that border the uncontrollable domain. We also showed that one could restrict computations to the boundary states, even when abstraction is not neighbor-linked. Indeed, with a novel control refinement scheme that interrupts the closed-loop trajectory of the original system when it reaches a boundary state of the controllable domain, we can guarantee that the closed-loop trajectory of the original system does not overjump the boundary. And since for every boundary state, we have an abstract input that pushes the system back towards the interior, a safety behavior is ensured. Though the gain efficiency of the proposed lazy synthesis approach was not overwhelming, the idea to explore only boundary states still deserves to be part of others lazy synthesis approaches, for example, when we are dealing with multi-scale abstractions as in Chapter 4.

Chapter 4 showed how to reduce the size of the symbolic models by using abstractions with adaptive grids. The main idea is to start with a coarse partitioning on the state space and then locally refine it in uncontrollable areas. We also used multi-scale time-sampling prioritizing transitions with a longer duration. We then benefit from a partial order on the input set U and the fact that the initial set $Q_{init} \subset Q$ is known to synthesize the maximal lazy safety controller C^*. The term maximal comes from the fact that all safety controllable initial states are in $\mathrm{Dom}(C^*)$, and if the controller enables an input, it also enables all inputs which have the same priority and preserve safety. The term lazy refers to the fact that the controller enables only the most prioritized inputs among actions that maintain safety. Hence, C^* represents a trade-off between maximal permissiveness and efficiency. Such a controller was first considered [Girard et al., 2016], but the synthesis procedure was restricted to a deterministic transition system. We merged Algorithm 2.3 and Algorithm 2.4 to provide a procedure applicable for non-deterministic abstractions as well. Usage of multi-scale abstractions with an adaptive grid showed a good performance since it allows the construction of accurate symbolic models with less computational effort. However, manipulation with adaptive grids requires more complex data structures (graphs, binary decision diagrams etc.) than simple matrices. The latter motivated us to use C++ instead of Matlab for the implementations.

Chapter 5 was devoted to monotone control systems and specifications given by directed sets. We first considered two classes of control systems: state monotone or input-state monotone dynamical systems. We then showed that an abstraction inherits monotonicity only if the state-space partitioning satisfies some restrictions. We distinguished lower and upper (input-)state monotone transition systems and lower and upper closed safety specifications but focused on lower ones due to symmetry of results. A finite lower-closed set can be represented by its basis. And we proved that to compute the controllable domain, it is enough to explore only basis states. Moreover, it is sufficient to use inputs with lower priorities for lower input-state monotone transition systems. We then provided two algorithms for lazily computing the maximal safety controller for lower (input-)state monotone transition systems. These algorithms benefit from the fact that for a lower state-monotone system, if a state q is controllable with an

input u, then all states belonging to $\downarrow q$ are controllable with an input u. Moreover, for a lower input-state monotone transition system, if for a state q an input u is safe, then all actions $u' \succ u$ are safe. We then enriched the class of considered specifications by intersections of lower-closed and upper-closed sets. The numerical implementations of Chapter 5 emphasised the significant improvement in efficiency comparing to the classical algorithms. Certainly, at first glance, it may seem that the results are too limited by a specific class of systems and specifications, but recent research showed that such a problem statement do arrive in challenging practical applications [Smith et al., 2022].

6.1 Future Research

What priorities on inputs should be like?

In Chapter 3 we introduced a partial order on input space by prioritising actions with a longer duration. In Chapter 5 the partial order was naturally inherited from the dynamic of the original plant. However, we find inspiring the ideas to explore the most promising actions [Hart et al., 1968, Rungger and Stursberg, 2012] first or favor least-violating [Girard and Eqtami, 2021, Tůmová et al., 2012] actions. Moreover, one may be interested in prioritizing the actions based on requirements of an additional specification. Another interesting trend in symbolic model control is prioritize actions which reduce non-determinism.

Adaptive time sampling technique as a way to enforce the desired structure

Abstractions computed with pre-fixed time-sampling are hard to be analyzed since in general, they don't have any particular structure. The same time the results of Chapters 3 and 5 showed that geometrical patterns are the way to ease computations. Moreover playing with transition duration one can create lattice graphs [Pivtoraiko et al., 2009], which were advocated when dealing with systems invariant with respect to geometrical shift, like kinematics models in robotics applications.

Another issue is that for the moment adaptive time-sampling is restricted to a continuous-time system. However, it is interesting to investigate the applicability of such a technique when dealing with discrete-time systems: what the relation between time and space-sampling parameters should be like.

Monotone dynamical systems and beyond...

In Chapter 5 we adopted the results designed for directed safety specifications to their intersection. However, the incremental synthesis approach of Section 5.3.4 still includes a classical synthesis procedure as the last step. At the same time there exists a criterion for an interval to be the controlled interval when monotone dynamical systems are considered [Meyer et al., 2013]. Moreover, the result from [Meyer et al., 2013] allows us to synthesize a safety controller. We believe that in combination with the ideas of this manuscript, the safety specification can be extended

towards the union of several intervals. Another drawback of [Meyer et al., 2013] to be overcome, is an assumption that every control component can affect no more than one state variable.

One of the practical interests to investigate monotone dynamical systems is the fact that their analysis is the first step towards the analysis of mixed-monotone systems, which are a very general class of systems [Yang et al., 2019]. Non-linear systems are often considered through the prism of hybridization or linearisation in order to benefit from linear analysis techniques. However, we believe that addressing monotone patterns in dynamic of non-linear systems through decomposition functions and then benefit from the results for monotone systems is a promising direction of the research.

One important step towards more general specifications

This thesis was dedicated to safety specifications. The motivation was to develop efficient synthesis algorithms for non-deterministic abstractions. The interest for a non-deterministic symbolic model is justified by a limited class of systems [Girard et al., 2016] and specifications [Kim et al., 2017] for which a deterministic abstraction can be constructed. However, when addressing other control problems, such as, for example, a reach-avoid specification, the non-determinism in the abstraction may ruin everything, making the synthesis unsuccessful even when the controller exists for the original plant. Indeed, if the reachable sets are growing with time, the error propagates too fast (see Figure 6.1 (left)). Hence, it is hardly possible that we can target a small final set. Even in some particular cases, the wrapping effect can be compensated by fine discretisation [Hsu et al., 2019] this results in very intensive computations. The demand to enforce determinism raised a wave of research in coupling abstraction-based methods with local feedback controllers ensuring local stability properties of the closed-loop systems [Tabuada, 2008, Sinyakov and Girard, 2020b]. And this is a problem to pay attention for.

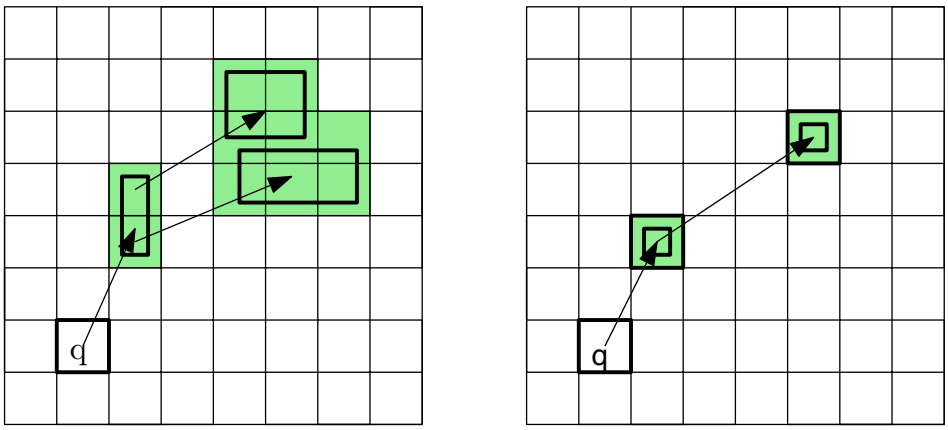


Figure 6.1: Left figures illustrates construction of a non-deterministic abstraction. Right figures illustrates a deterministic one.

Appendix A

Reachability Analysis for Mixed-monotone Dynamic Systems

The results of this Appendix were originally presented in [Yang et al., 2019] and provided here to ease the reading of the main part of the manuscript.

Consider a nonlinear system of the following type:

$$\dot{x}(t) = f(t, x(t), u(t), w(t)), \quad t \in [0, T] \tag{A.1}$$

Here $x \in \mathbb{R}^{n_x}$ is a state, $u \in U = [\underline{u}, \overline{u}] \subset \mathbb{R}^{n_u}$ is a control, $w \in W = [\underline{w}, \overline{w}]$ is a disturbance. The set of admissible open-loop controls is $\mathcal{U}(t, \tau) = \mathcal{L}([t, \tau], U)$. The set of admissible realization of disturbance is $\mathcal{W}(t, \tau) = \mathcal{L}([t, \tau], W)$. The notation $\mathcal{L}([t, \tau], S)$ is used for the space of all measurable on $[t, \tau]$ functions $\mathbf{s}$, such that $s(p) \in S,\ p \in [t, \tau]$ almost everywhere.

Let $x_f(t \mid x(0), \hat{\mathbf{u}}, \hat{\mathbf{w}})$ denote a trajectory of the system (1.1) satisfying the initial condition $x(0) = x$ and corresponding to a given control $\hat{\mathbf{u}}$ and a disturbance $\hat{\mathbf{w}}$. Finally, let $\mathrm{Reach}(t \mid, X^0, \hat{\mathbf{u}})$ denote the robust reachable set, corresponding to a given control $\hat{\mathbf{u}}$, i.e.

$$\mathrm{Reach}(t \mid, X^0, \hat{\mathbf{u}}) = \{x \in \mathbb{R}^{n_x} \mid \exists\, x^0 \in X^0, \exists\, \mathbf{w} \in \mathcal{W}(0, t), \text{ such that } x_f(t \mid x^0, \hat{\mathbf{u}}, \mathbf{w}) = x\}.$$

Computation of the exact reachable set for nonlinear dynamic systems is an intractable problem in the general case, and, in practice, their over-approximations are sought after.

Definition A.0.1. *Function $g : T \times \mathbb{R}^{2n_x} \times U^2 \times W^2 \to \mathbb{R}^{n_x}$ is called a decomposition function for a function $f : T \times \mathbb{R}^{n_x} \times U \times W \to \mathbb{R}^{n_x}$ if*

- $g(t,[x;x],[u;u],[w;w]) = f(t,x,u,w)$;
- $g_i(t,[x;y],[u^1;u^2],[w^1;w^2])$ *is non-decreasing in* x_j*, non-increasing in* y_j *when* $i \neq j$*.*
- $g_i(t,[x;y],[u^1;u^2],[w^1;w^2]))$ *is non-decreasing in* u^1 *and* w^1*, non-increasing in* u^2 *and* w^2*.*

We call f a mixed monotone function and system (1.1) a mixed monotone system if there exits such decomposition function g.

Definition A.0.2. *Let f be a mixed monotone function and g be a decomposition of f . Decomposition function g is called tight if for all* $i \in \{1,\dots,n_x\}$*, for all* $t \in [0,T]$*, for all* $x^1, x^2 \in \mathbb{R}^{n_x}$*, for all* $u^1, u^2 \in U$*, and all* $w^1, w^2 \in W$ *such that* $x^1 \leq x^2, u^1 \leq u^2, w^1 \leq w^2$*, it follows that*

$$g_i(t,[x^2;x^1],[u^1;u^2],[w^2;w^1]) = \max_{x\in[x^1,x^2]} \min_{u\in[u^1;u^2]} \max_{w\in[w^1,w^2]} f_i(t,x,u,w)$$
$$g_i(t,[x^1;x^2],[u^2;u^1],[w^1;w^2]) = \min_{x\in[x^1,x^2]} \max_{u\in[u^1;u^2]} \min_{w\in[w^1,w^2]} f_i(t,x,u,w)$$

Let us also introduce a set of notation: $\overline{\theta} = [\overline{u}, \underline{u}]$, $\underline{\theta} = [\underline{u}, \overline{u}]$, $\overline{\omega} = [\overline{w}, \underline{w}]$, $\underline{\omega} = [\underline{w}, \overline{w}]$. And define function ζ_i that map R^{2n_x} into itself:

$$(\zeta_i(x,y))_j = \begin{cases} x_j, j < n_x \\ y_j, j > n_x,\ j \neq n_x + i \\ x_i, j = n_x + i \end{cases}$$

Supposing that the initial set $X^0 = [\underline{x}^0, \overline{x}^0]$ is an interval, consider a system of equations $(i = 1,\dots,n_x)$

$$\begin{aligned} \dot{\overline{x}}_i &= g_i(t, \zeta_i(\overline{x},\underline{x}), [\hat{u},\hat{u}], \overline{\omega}), \quad \overline{x}_i(0) = \overline{x}_i^0 \\ \dot{\underline{x}}_i &= g_i(t, \zeta_i(\underline{x},\overline{x}), [\hat{u},\hat{u}], \underline{\omega}), \quad \underline{x}_i(0) = \underline{x}_i^0 \end{aligned} \tag{A.2}$$

Denoting the components of its solution as $\underline{\mathbf{x}}(t;[\underline{x}^0,\overline{x}^0],\hat{u})$ and $\overline{\mathbf{x}}(t;[\overline{x}^0,\underline{x}^0],\hat{u})$, we have that

$$\overline{\mathrm{Reach}}(t\mid, X^0, \hat{\mathbf{u}}) = [\underline{\mathbf{x}}(t;[\underline{x}^0,\overline{x}^0],\hat{u}), \overline{\mathbf{x}}(t;[\overline{x}^0,\underline{x}^0],\hat{u})] \tag{A.3}$$

is an over-approximation of the forward reachable set $\mathrm{Reach}(t\mid, X^0, \hat{\mathbf{u}})$.

Bibliography

[Abate et al., 2009] Abate, A., Tiwari, A., and Sastry, S. (2009). Box invariance for biologically-inspired dynamical systems. *Automatica*, 45:1601–1610.

[Alam et al., 2014] Alam, A., Gattami, A., Johansson, K. H., and Tomlin, C. J. (2014). Guaranteeing safety for heavy duty vehicle platooning: Safe set computations and experimental evaluations. *Control Engineering Practice*, 24:33–41.

[Ames et al., 2017] Ames, A. D., Xu, X., Grizzle, J. W., and Tabuada, P. (2017). Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876.

[Anevlavix and Tabuada, 2019] Anevlavix, T. and Tabuada, P. (December 2019). Computing controlled invariant sets in two moves. *Proceedings of the IEEE Conference on Decision and Control*, pages 6249–6254.

[Angeli, 2002] Angeli, D. (2002). A lyapunov approach to incremental stability properties. *IEEE Transactions on Automatic Control*, 47(3):410–421.

[Angeli and Sontag, 2003] Angeli, D. and Sontag, E. (2003). Monotone control systems. *IEEE Transactions on Automatic Control*, 48.

[Asarin et al., 2000] Asarin, E., Bournez, O., Dang, T., and Maler, O. (2000). Approximate reachability analysis of piecewise-linear dynamical systems. In Lynch, N. and Krogh, B. H., editors, *Hybrid Systems: Computation and Control*, pages 20–31, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Aubin and Frankowska, 1991] Aubin, J. and Frankowska, H. (1991). *Viability Kernel of Control Systems*, volume 9. Birkhäuser, Boston, MA.

[Belta et al., 2017] Belta, C., Yordanov, B., and Gol, E. A. (2017). *Formal Methods for Discrete-Time Dynamical Systems*. Springer.

[Blanchini, 1999] Blanchini, F. (1999). Set invariance in control. *Automatica*, 35(11):1747–1767.

[Bloem et al., 2012] Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., and Saar, Y. (2012). Synthesis of reactive(1) designs. *Journal of Computer and System Sciences*, 78(3):911–938. In Commemoration of Amir Pnueli.

[Cassandras and Lafortune, 2009] Cassandras, C. G. and Lafortune, S. (2009). *Introduction to Discrete Event Systems, 2nd ed.* Springer.

[Chen et al., 2015] Chen, M., Hu, Q., Mackin, C., Fisac, J. F., and Tomlin, C. J. (December 2015). Safe platooning of unmanned aerial vehicles via reachability. *Proceedings of the IEEE Conference on Decision and Control*, 54rd IEEE Conference on Decision and Control:4695–4701.

[Chen et al., 2012] Chen, X., Ábrahám, E., and Sankaranarayanan, S. (2012). Taylor model flowpipe construction for non-linear hybrid systems. *2012 IEEE 33rd Real-Time Systems Symposium*, pages 183–192.

[Chutinan and Krogh, 2003] Chutinan, A. and Krogh, B. (2003). Computational techniques for hybrid system verification. *IEEE Transactions on Automatic Control*, 48(1):64–75.

[Coogan and Arcak, 2015] Coogan, S. and Arcak, M. (2015). Efficient finite abstraction of mixed monotone systems. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC '15, page 58–67, New York, NY, USA. Association for Computing Machinery.

[Coogan et al., 2016] Coogan, S., Gol, E. A., Arcak, M., and Belta, C. (2016). Traffic network control from temporal logic specifications. *IEEE Transactions on Control of Network Systems*, 3(2):162–172.

[Cormen et al., 2001] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms, 2nd ed.* MIT Press.

[Dang et al., 2010] Dang, T., Maler, O., and Testylier, R. (2010). Accurate hybridization of nonlinear systems. In *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '10, page 11–20, New York, NY, USA. Association for Computing Machinery.

[Darbha, 1997] Darbha, S. (1997). String stability of interconnected systems: An application to platooning in automated highway systems. *Transportation Research Part A: Policy and Practice*, 31(1):65.

[Derler et al., 2012] Derler, P., Lee, E. A., and Sangiovanni Vincentelli, A. (2012). Modeling cyber–physical systems. *Proceedings of the IEEE*, 100(1):13–28.

[Fainekos et al., 2009] Fainekos, G. E., Girard, A., Kress-Gazit, H., and Pappas, G. J. (2009). Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352.

[Fan et al., 2016] Fan, C., Kapinski, J., Jin, X., and Mitra, S. (2016). Locally optimal reach set over-approximation for nonlinear systems. In *2016 International Conference on Embedded Software (EMSOFT)*, pages 1–10.

[Finkel and Schnoebelen, 2001] Finkel, A. and Schnoebelen, P. (2001). Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92.

[Finucane et al., 2010] Finucane, C., Jing, G., and Kress-Gazit, H. (2010). Ltlmop: Experimenting with language, temporal logic and robot control. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan*, pages 1988–1993. IEEE.

[Fu et al., 2013] Fu, J., Shah, S., and Tanner, H. G. (2013). Hierarchical control via approximate simulation and feedback linearization. In *2013 American Control Conference*, pages 1816–1821.

[Gillis et al., 2007] Gillis, R., Palerm, C. C., Zisser, H., Jovanovič, L., Seborg, D. E., and Doyle, F. J. (2007). Glucose estimation and prediction through meal responses using ambulatory subject data for advisory mode model predictive control. *Journal of Diabets Science and Technology*, 1(6):825—-833.

[Girard, 2005] Girard, A. (2005). Reachability of uncertain linear systems using zonotopes. *Hybrid Systems: Computation and Control. HSCC 2005. Lecture Notes in Computer Science*, 3414:291—-305.

[Girard, 2010] Girard, A. (2010). Controller synthesis for safety and reachability via approximate bisimulation. *Automatica*, 48.

[Girard and Eqtami, 2021] Girard, A. and Eqtami, A. (2021). Least-violating symbolic controller synthesis for safety, reachability and attractivity specifications. *Automatica*, 127:109543.

[Girard et al., 2016] Girard, A., Gössler, G., and Moueli, S. (2016). Safety controller synthesis for incrementally stable switched systems using multiscale symbolic models. *IEEE Transactions on Automatic Control*, 61(6):1537–1549.

[Girard and Guernic, 2008] Girard, A. and Guernic, C. L. (2008). Efficient reachability analysis for linear systems using support functions. *IFAC Proceedings Volumes*, 41(2):8966–8971. 17th IFAC World Congress.

[Girard and Pappas, 2005] Girard, A. and Pappas, G. (2005). Approximate bisimulations for constrained linear systems. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 4700–4705.

[Girard and Pappas, 2007] Girard, A. and Pappas, G. (2007). Hierarchical control using approximate simulation relations. In *Proceedings of the IEEE Conference on Decision and Control*, pages 264 – 269.

[Gol et al., 2013] Gol, E. A., Lazar, M., and Belta, C. (2013). Language-guided controller synthesis for linear systems. *IEEE Transactions on Automatic Control*, 59(5):1163–1176.

[Gonzalez et al., 2010] Gonzalez, H., Vasudevan, R., Kamgarpour, M., Sastry, S. S., Bajcsy, R., and Tomlin, C. J. (2010). A descent algorithm for the optimal control of constrained nonlinear switched dynamical systems. In *HSCC'10 - Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, pages 51–60.

[Gruber et al., 2017] Gruber, F., Kim, E. S., and Arcak, M. (2017). Sparsity-Sensitive Finite Abstraction. In *IEEE Conference on Decision and Control (CDC)*, pages 2366–2371.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

[Henzinger, 2000] Henzinger, T. A. (2000). *The Theory of Hybrid Automata*, pages 265–292. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Higman, 1952] Higman, G. (1952). Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 3(1):326–336.

[Hirsch and Smith, 2004] Hirsch, M. and Smith, H. (2004). Monotone dynamical systems. *Handbook of Differential Equations: Ordinary Differential Equations*, 2.

[Hovorka et al., 2004] Hovorka, R., Canonico, V., Chassin, L., Haueter, U., Massi-Benedetti, M., Federici, M., Pieber, T., Schaller, H., Schaupp, L., Vering, T., and Wilinska, M. (2004). Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes. *Physiological measurement*, 25:905–20.

[Hsu et al., 2018] Hsu, K., Majumdar, R., Mallik, K., and Schmuck, A.-K. (2018). Multi-layered abstraction-based controller synthesis for continuous-time systems. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (Part of CPS Week)*, HSCC '18, page 120–129, New York, NY, USA. Association for Computing Machinery.

[Hsu et al., 2019] Hsu, K., Majumdar, R., Mallik, K., and Schmuck, A.-K. (2019). Lazy abstraction-based controller synthesis. In Chen, Y.-F., Cheng, C.-H., and Esparza, J., editors, *Automated Technology for Verification and Analysis*, pages 23–47, Cham. Springer International Publishing.

[Hussien and Tabuada, 2018] Hussien, O. and Tabuada, P. (2018). Lazy controller synthesis using three-valued abstractions for safety and reachability specifications. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 3567–3572.

[Jaulin et al., 2001] Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. (2001). *Applied Interval Analysis.With Examples in Parameter and State Estimation, Robust Control and Robotics.* Springer-Verlag London.

[Kader et al., 2019] Kader, A., Saoud, A., and Girard, A. (2019). Safety controller design for incrementally stable switched systems using event-based symbolic models. In *European Control Conference*, pages 1269–1274.

[Kamke, 1932] Kamke, E. (1932). Zur Theorie der Systeme gewöhnlicher Differentialgleichungen. *Acta Mathematica II*, 1(58):57–85.

[Khlebnikov et al., 2011] Khlebnikov, M. V., Polyak, B. T., and Kuntsevich, V. M. (2011). Optimization of linear systems subject to bounded exogenous disturbances: The invariant ellipsoid technique. *Automation and Remote Control*, 72(11):2227–2275.

[Kim et al., 2015] Kim, E., Arcak, M., and Seshia, S. (2015). Compositional controller synthesis for vehicular traffic networks. In *IEEE Conference on Decision and Control*, pages 6165–6171.

[Kim et al., 2017] Kim, E. S., Arcak, M., and Seshia, S. A. (2017). Symbolic control design for monotone systems with directed specifications. *Automatica*, 83:10–19.

[Kim and Kumar, 2012] Kim, K.-D. and Kumar, P. (2012). Cyber–physical systems: A perspective at the centennial. *Proceedings of the IEEE*, 100:1287–1308.

[Kloetzer and Belta, 2008] Kloetzer, M. and Belta, C. (2008). A fully automated framework for control of linear systems from temporal logic specifications. *Automatic Control, IEEE Transactions on*, 53:287 – 297.

[Koenig and Likhachev, 2005] Koenig, S. and Likhachev, M. (2005). Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363.

[Kolmanovsky and Gilbert, 1998] Kolmanovsky, I. and Gilbert, E. G. (1998). Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical Problems in Engineering*, 4:317–367.

[Krasnoselskii, 1968] Krasnoselskii, M. A. (1968). American mathematical society. *The operator of translation along the trajectories of differential equations*, 1(19):281–292.

[Kurzhanski and Varaiya, 2014] Kurzhanski, A. B. and Varaiya, P. (2014). *Dynamics and Control of Trajectory Tubes. Theory and Computation*. Birkhäuser.

[Kushner et al., 2019] Kushner, T., Bequette, B. W., Cameron, F., Forlenza, G., Maahs, D., and Sankaranarayanan, S. (August 2019). Models, devices, properties, and verification of artificial pancreas systems. *Automated Reasoning for Systems Biology and Medicine*, pages 825—-833.

[Lafferriere et al., 2001] Lafferriere, G., Pappas, G. J., and Yovine, S. (2001). Symbolic reachability computation for families of linear vector fields. *Journal of Symbolic Computation*, 32(3):231–253.

[Li and Liu, 2018a] Li, Y. and Liu, J. (2018a). Invariance control synthesis for switched nonlinear systems: An interval analysis approach. *IEEE Transactions on Automatic Control*, 63(7):2206–2211.

[Li and Liu, 2018b] Li, Y. and Liu, J. (2018b). Rocs: A robustly complete control synthesis tool for nonlinear dynamical systems. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (Part of CPS Week)*, HSCC '18, page 130–135.

[Liu et al., 2013] Liu, J., Ozay, N., Topcu, U., and Murray, R. M. (2013). Synthesis of reactive switching protocols from temporal logic specifications. *IEEE Transactions on Automatic Control*, 58(7):1771–1785.

[Liu et al., 2012] Liu, J., Topcu, U., Ozay, N., and Murray, R. M. (2012). Reactive controllers for differentially flat systems with temporal logic constraints. In *Proceedings of the IEEE Conference on Decision and Control*, pages 7664–7670.

[Liu et al., 2014] Liu, Q., Wang, Z., He, X., and Zhou, D. (2014). A survey of event-based strategies on control and estimation. *Systems Science & Control Engineering*, 2(1):90–97.

[Lygeros et al., 2003] Lygeros, J., Johansson, K., Simic, S., Zhang, J., and Sastry, S. (2003). Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48(1):2–17.

[Maidens and Arcak, 2014] Maidens, J. and Arcak, M. (2014). Reachability analysis of nonlinear systems using matrix measures. *IEEE Transactions on Automatic Control*, 60(1):265–270.

[Mazo et al., 2010] Mazo, M., Davitian, A., and Tabuada, P. (2010). Pessoa: A tool for embedded controller synthesis. In Touili, T., Cook, B., and Jackson, P., editors, *Computer Aided Verification*, pages 566–569, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Meyer et al., 2018] Meyer, P.-J., Coogan, S., and Arcak, M. (2018). Sampled-data reachability analysis using sensitivity and mixed-monotonicity. *IEEE Control Systems Letters*, PP.

[Meyer et al., 2019] Meyer, P.-J., Devonport, A., and Arcak, M. (2019). Tira: Toolbox for interval reachability analysis. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, page 224–229, New York, NY, USA. Association for Computing Machinery.

[Meyer et al., 2021] Meyer, P.-J., Devonport, A., and Arcak, M. (2021). *Interval Reachability Analysis.Bounding Trajectories of Uncertain Systems with Boxes for Control and Verification*. SpringerBriefs in Control, Automation and Robotics.

[Meyer et al., 2017] Meyer, P.-J., Girard, A., and Witrant, E. (2017). Compositional abstraction and safety synthesis using overlapping symbolic models. *IEEE Transactions on Automatic Control*, 63(6):1835–1841.

[Meyer et al., 2013] Meyer, P.-J., Girard, A., and Witrant, E. (December 2013). Controllability and invariance of monotone systems for robust ventilation automation in buildings. *Proceedings of the IEEE Conference on Decision and Control*, pages 1289–1294.

[Milner, 1989] Milner, R. (1989). *Communication and Concurrency*. Prentice-Hall, Inc., USA.

[Mitchell et al., 2005] Mitchell, I., Bayen, A., and Tomlin, C. (2005). A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *Automatic Control, IEEE Transactions on*, 50:947 – 957.

[Moor and Raisch, 2002] Moor, T. and Raisch, J. (2002). Abstraction based supervisory controller synthesis for high order monotone continuous systems. In Engell, S., Frehse, G., and Schnieder, E., editors, *Modelling, Analysis, and Design of Hybrid Systems*, pages 247–265, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Mouelhi et al., 2013] Mouelhi, S., Girard, A., and Gössler, G. (2013). Cosyma: A tool for controller synthesis using multi-scale abstractions. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*, HSCC '13, page 83–88. Association for Computing Machinery.

[Müller, 1927] Müller, M. (1927). Über das Fundamentaltheorem in der Theorie der gewöhnlichen Differentialgleichungen. *Mathematische Zeitschrift*, 1(26):619–645.

[Nilson et al., 2017] Nilson, P., Ozay, N., and Liu, J. (2017). Augmented finite transition systems as abstractions for control synthesis. *Discrete Event Dynamic Systems*, 27(3):301–340.

[Nilsson et al., 2016] Nilsson, P., Hussien, O., Balkan, A., Chen, Y., Ames, A. D., Grizzle, J. W., Ozay, N., Peng, H., and Tabuada, P. (2016). Correct-by-Construction Adaptive Cruise Control: Two Approaches. *IEEE Transactions on Control Systems Technology*, 24(4):1294–1307.

[Papachristodoulou and Prajna, 2005] Papachristodoulou, A. and Prajna, S. (2005). A tutorial on sum of squares techniques for systems analysis. *Proceedings of the American Control Conference*, 4(July 2005):2686–2700.

[Pappas, 2003] Pappas, G. (2003). Bisimilar linear systems. *Automatica*, 39:2035–2047.

[Park, 1981] Park, D. (1981). Concurrency and automata on infinite sequences. In Deussen, P., editor, *Theoretical Computer Science*. Springer Berlin Heidelberg.

[Peng and Li, 2018] Peng, C. and Li, F. (2018). A survey on recent advances in event-triggered communication and control. *Information Sciences*, 457-458:113–125.

[Pivtoraiko et al., 2009] Pivtoraiko, M., Knepper, R., and Kelly, A. (2009). Differentially constrained mobile robot motion planning in state lattices. journal of field robotics (jfr), 26(3), 308-333. *J. Field Robotics*, 26:308–333.

[Pola et al., 2010] Pola, G., Pepe, P., Di Benedetto, M. D., and Tabuada, P. (2010). Symbolic models for nonlinear time-delay systems using approximate bisimulations. *Systems and Control Letters*, 59(6):365–373.

[Rajkumar et al., 2010] Rajkumar, R., Lee, I., Sha, L. R., and Stankovic, J. (2010). Cyber-physical systems: The next computing revolution. In *Proceedings of the 47th Design Automation Conference, DAC '10*, pages 731–736.

[Ramdani et al., 2010] Ramdani, N., Meslem, N., and Candau, Y. (2010). Computing reachable sets for uncertain nonlinear monotone systems. *Nonlinear Analysis: Hybrid Systems*, 4:263–278.

[Reddy, 1995] Reddy, U. S. (1995). Topology and stone duality for domains.

[Reissig et al., 2017] Reissig, G., Weber, A., and Rungger, M. (2017). Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, 62(4):1781—-1796.

[Rungger et al., 2013] Rungger, M., Mazo, M., and Tabuada, P. (2013). Specification-guided controller synthesis for linear systems and safe linear-time temporal logic. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*, HSCC '13, page 333–342, New York, NY, USA. Association for Computing Machinery.

[Rungger and Stursberg, 2012] Rungger, M. and Stursberg, O. (2012). On-the-fly model abstraction for controller synthesis. *Proceedings of the American Control Conference*, pages 2645–2650.

[Rungger and Zamani, 2016] Rungger, M. and Zamani, M. (2016). Scots: A tool for the synthesis of symbolic controllers. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, HSCC '16, page 99–104, New York, NY, USA. Association for Computing Machinery.

[Sadraddini and Belta, 2016] Sadraddini, S. and Belta, C. (2016). Safety control of monotone systems with bounded uncertainties. In *IEEE Conference on Decision and Control*, pages 4874–4879.

[Saoud, 2019] Saoud, A. (2019). Compositional and efficient controller synthesis for cyber-physical systems.

[Saoud and Girard, 2018] Saoud, A. and Girard, A. (2018). Optimal multirate sampling in symbolic models for incrementally stable switched systems. *Automatica*, 98:58–65.

[Saoud et al., 2020] Saoud, A., Girard, A., and Fribourg, L. (2020). Contract-based design of symbolic controllers for safety in distributed multiperiodic sampled-data systems. *IEEE Transactions on Automatic Control*. To appear.

[Saoud et al., 2020] Saoud, A., Jagtap, P., Zamani, M., and Girard, A. (2020). Compositional abstraction-based synthesis for interconnected systems: An approximate composition approach. *arXiv preprint arXiv:2002.02014*.

[Scott and Barton, 2013] Scott, J. and Barton, P. (2013). Bounds on the reachable sets of nonlinear control systems. *Automatica*, 49:93–100.

[Shen and Scott, 2017] Shen, K. and Scott, J. K. (2017). Rapid and accurate reachability analysis for nonlinear dynamic systems by exploiting model redundancy. *Computers and Chemical Engineering*, 106:596–608. ESCAPE-26.

[Sinyakov and Girard, 2020a] Sinyakov, V. and Girard, A. (2020a). Abstraction of monotone systems based on feedback controllers. *IFAC-PapersOnLine*, 53(2):1819–1824. 21st IFAC World Congress.

[Sinyakov and Girard, 2021] Sinyakov, V. and Girard, A. (2021). Formal controller synthesis from specifications given by discrete-time hybrid automata. *Automatica*, 131:109768.

[Sinyakov and Girard, 2020b] Sinyakov, V. and Girard, A. (December 2020b). Formal Synthesis from Control Programs. *Proceedings of the IEEE Conference on Decision and Control*, pages 2138–2145.

[Smith, 1995] Smith, H. (1995). *Monotone Dynamical Systems: An Introduction to the Theory of Competitive and Cooperative Systems: An Introduction to the Theory of Competitive and Cooperative Systems*, volume 41 of *Mathematical surveys and monographs*. American Mathematical Society.

[Smith, 1988] Smith, H. L. (1988). Systems of ordinary differential equations which generate an order preserving flow. a survey of results. *SIAM Review*, 1(30):87–113.

[Smith et al., 2022] Smith, S. W., Saoud, A., and Arcak, M. (2022). Monotonicity-based symbolic control for safety in driving scenarios. *IEEE Control Systems Letters*, 6:830–835.

[Sontag, 2007] Sontag, E. D. (2007). Monotone and near-monotone biochemical networks. *Systems and Synthetic Biology*, 1:59–87.

[Subbotin, 1995] Subbotin, A. I. (1995). *Generalized Solutions of First Order PDEs: The Dynamical Optimization Perspective*. Birkhäuser Basel.

[Swikir and Zamani, 2019] Swikir, A. and Zamani, M. (2019). Compositional synthesis of finite abstractions for networks of systems: A small-gain approach. *Automatica*, 107:551–561.

[Tabuada, 2008] Tabuada, P. (2008). An approximate simulation approach to symbolic control. *IEEE Transactions on Automatic Control*, 53(6):1406–1418.

[Tabuada, 2009] Tabuada, P. (2009). *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media.

[Tabuada and Pappas, 2006] Tabuada, P. and Pappas, G. J. (2006). Linear time logic control of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 51(12):1862–1877.

[Tajvar et al., 2020] Tajvar, P., Barbosa, F. S., and Tumova, J. (2020). Safe Motion Planning for an Uncertain Non-Holonomic System with Temporal Logic Specification. *IEEE International Conference on Automation Science and Engineering*, August:349–354.

[Tan and Packard, 2008] Tan, W. and Packard, A. (2008). Stability region analysis using polynomial and composite polynomial lyapunov functions and sum-of-squares programming. *IEEE Transactions on Automatic Control*, 53(2):565–571.

[Thavlov and Bindner, 2015] Thavlov, A. and Bindner, H. W. (2015). A heat dynamic model for intelligent heating of buildings. *International Journal of Green Energy*, 12(3):240–247.

[Tripakis and Altisen, 1999] Tripakis, S. and Altisen, K. (1999). On-the-fly controller synthesis for discrete and dense-time systems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1708, pages 233–252.

[Trodden, 2016] Trodden, P. (2016). A One-Step Approach to Computing a Polytopic Robust Positively Invariant Set. *IEEE Transactions on Automatic Control*, 61(12):4100–4105.

[Tůmová et al., 2012] Tůmová, J., Hall, G. C., Karaman, S., Frazzoli, E., and Rus, D. (2012). Least-violating control strategy synthesis with safety rules. In *HSCC 2013 - Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control, Part of CPSWeek 2013*, volume 1.

[Weiss et al., 2018] Weiss, A., Kalabić, U. V., and Cairano, S. D. (2018). Station keeping and momentum management of low-thrust satellites using mpc. *Aerospace Science and Technology*, 76:229–241.

[Wongpiromsarn et al., 2009] Wongpiromsarn, T., Topcu, U., and Murray, R. M. (2009). Receding horizon temporal logic planning for dynamical systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 5997–6004.

[Wongpiromsarn et al., 2011] Wongpiromsarn, T., Topcu, U., Ozay, N., Xu, H., and Murray, R. M. (2011). Tulip: A software toolbox for receding horizon temporal logic planning. In *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control*, HSCC '11, page 313–314.

[Yang et al., 2019] Yang, L., Mickelin, O., and Ozay, N. (2019). On sufficient conditions for mixed monotonicity. *IEEE Transactions on Automatic Control*, 64(12):5080–5085.

[Yordanov et al., 2012] Yordanov, B., Tumova, J., Cerna, I., Barnat, J., and Belta, C. (2012). Temporal logic control of discrete-time piecewise affine systems. *IEEE Transactions on Automatic Control*, 57:1491–1504.

[Zamani et al., 2011] Zamani, M., Pola, G., Mazo, M., and Tabuada, P. (2011). Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control*, 57(7):1804–1809.

[Zonetti et al., 2019] Zonetti, D., Saoud, A., Girard, A., and Fribourg, L. (2019). A symbolic approach to voltage stability and power sharing in time-varying DC microgrids. In *ECC 2019 - European control conference*, Naples, Italy. European Control Conference 2019.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Synthèse efficace des contrôleurs de sécurité à l'aide de modèles symboliques et d'algorithmes paresseux

Mots clés: Synthèse de contrôleur, Modèles symboliques, Méthodes formelles

Résumé: Cette thèse porte sur le développement d'approches efficaces de synthèse de contrôleurs basées sur l'abstraction pour les systèmes cyber-physiques (CPS). Alors que les méthodes basées sur l'abstraction pour la conception de CPS ont fait l'objet de recherches intensives au cours des dernières décennies, l'évolutivité de ces techniques reste un problème. Cette thèse se concentre sur le développement d'algorithmes de synthèse paresseuse pour les spécifications de sécurité. Les spécifications de sécurité consistent à maintenir la trajectoire du système à l'intérieur d'un ensemble sûr donné. Cette spécification est de la plus haute importance dans de nombreux problèmes d'ingénierie, souvent prioritaires par rapport à d'autres exigences de performance. Les approches paresseuses surpassent l'algorithme de synthèse classique [Tabuada, 2009] en évitant les calculs, qui ne sont pas essentiels pour les objectifs de synthèse. Le chapitre 1 motive la thèse et présente l'état de l'art. Le chapitre 2 structure les approches de synthèse paresseuses existantes et met l'accent sur trois sources d'efficacité : les informations sur les états contrôlables a priori, les priorités sur les entrées et les états non accessibles à partir de l'ensemble initial. Le chapitre 3 propose un algorithme, qui explore itérativement les états à la frontière du domaine contrôlable tout en évitant l'exploration des états internes, en supposant qu'ils sont contrôlables en toute sécurité a priori. Un contrôleur de sécurité en boucle fermée pour le problème d'origine est alors défini comme suit : nous utilisons le contrôleur abstrait pour repousser le système d'un état limite vers l'intérieur, tandis que pour les états internes, toute entrée admissible est valide. Le chapitre 4 présente un algorithme qui restreint les calculs de synthèse du contrôleur aux seuls états atteignables tout en privilégiant les transitions de plus longue durée. Le système original est abstrait par un modèle symbolique avec une grille adaptative. De plus, un nouveau type d'échantillonnage temporel est également envisagé. Au lieu d'utiliser des transitions de durée prédéterminée, la durée des transitions est contrainte par des intervalles d'état qui doivent contenir l'ensemble accessible. Le chapitre 5 est consacré aux systèmes de transition monotones. L'approche de synthèse paresseuse introduite bénéficie d'une propriété monotone des systèmes de transition et de la structure ordonnée de l'espace d'état (d'entrée), et du fait que des spécifications de sécurité dirigées sont prises en compte. La classe de spécifications considérée s'enrichit alors d'intersections d'exigences de sécurité supérieures et inférieures fermées. Le chapitre 6 conclut la discussion et soulève de nouvelles questions pour les recherches futures.

Title: Efficient Synthesis of Safety Controllers using Symbolic Models and Lazy Algorithms

Keywords: Controller synthesis, Formal methods, Symbolic models

Abstract: This thesis focuses on the development of efficient abstraction-based controller synthesis approaches for cyber-physical systems (CPS). While abstraction-based methods for CPS design have been the subject of intensive research over the last decades, the scalability of these techniques remains an issue. This thesis focus on developing lazy synthesis algorithms for safety specifications. Safety specifications consist in maintaining the trajectory of the system inside a given safe set. This specification is of the utmost importance in many engineering problems, often prioritized over other performance requirements. Lazy approaches outperform the classical synthesis algorithm [Tabuada, 2009] by avoiding computations, which are non-essential for synthesis goals. Chapter 1 motivates the thesis and discusses the state of the art. Chapter 2 structures the existing lazy synthesis approaches and emphasizes three sources of efficiency: information about a priori controllable states, priorities on inputs, and non-reachable from initial set states. Chapter 3 proposes an algorithm, which iteratively explores states on the boundary of controllable domain while avoiding exploration of internal states, supposing that they are safely controllable a priory. A closed-loop safety controller for the original problem is then defined as follows: we use the abstract controller to push the system from a boundary state back towards the interior, while for inner states, any admissible input is valid. Chapter 4 presents an algorithm that restricts the controller synthesis computations to reachable states only while prioritizing longer-duration transitions. The original system is abstracted by a symbolic model with an adaptive grid. Moreover, a novel type of time sampling is also considered. Instead of using transitions of predetermined duration, the duration of the transitions is constrained by state intervals that must contain the reachable set. Chapter 5 is dedicated to monotone transition systems. The introduced lazy synthesis approach benefits from a monotone property of transition systems and the ordered structure of the state (input) space, and the fact that directed safety specifications are considered. The considered class of specifications is then enriched by intersections of upper and lower-closed safety requirements. Chapter 6 concludes the discussion and raises new issues for future research.

Université Paris-Saclay

Espace Technologique / Immeuble Discovery

Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France