



Extended Formulations for Control Languages Defined by Finite-State Automata

Christoph Buchheim ¹ and Maximilian Merkert ²

¹ TU Dortmund, Fakultät für Mathematik, Vogelpothsweg 87, 44227 Dortmund, Germany

² TU Braunschweig, Institute for Mathematical Optimization, Universitätsplatz 2, 38106 Braunschweig, Germany

April 30, 2024

Abstract

Many discrete optimal control problems feature combinatorial constraints on the possible switching patterns, a common example being minimum dwell-time constraints. After discretizing to a finite time grid, for these and many similar types of constraints, it is possible to give a description of the convex hull of feasible (finite-dimensional) binary controls via extended formulations. In this work, we aim to transfer a large class of such descriptions to function space, by determining extended formulations for the closed convex hull of feasible control functions. Such an infinite-dimensional grid-independent convex-hull description can serve as a recipe for the practitioner for obtaining tight formulations for any discretization. Our result applies to a large class of constraints that follow a certain modular principle, described by a generalization of finite-state automata to infinite-dimensional input data, which we specifically define for this purpose.

Keywords: Extended Formulations, Optimal Control, Convex Hull, Finite Automaton, Mixed-Integer Programming

Mathematics Subject Classification: 49M25, 68Q45, 90C11

1 Introduction

Optimal control problems with discrete controls have recently become an increasing focus of research. Such problems arise when searching for an optimal switching pattern over time, e.g., when shifting gears in a vehicle [RZSB21], navigating traffic at traffic-light controlled intersections [LMS⁺21] or when operating a gas network [HLM⁺17]. In many applications, such problems contain ordinary or partial differential equations. However, in the context of discrete controls, the set of feasible controls alone is a very interesting object and therefore worth studying. In particular, this set is usually non-convex, so that a tight description of its convex hull is necessary to obtain dual bounds and hence to compute globally optimal solutions, e.g., by a branch-and-bound

approach [BGM24b, BGM24a]. Moreover, also heuristic methods such as combinatorial integral approximation [SJK11] benefit from strong continuous relaxations.

Combinatorial constraints often arise in hybrid dynamical systems, i.e., systems which are governed by continuous dynamics as well as discrete events such as switches triggered by an optimizer; see for example [Bra05] for an introduction to hybrid systems, or the book [VDSS07] for more details and many examples. A more recent survey on hybrid and switched systems can be found in [ZA15]. The description of such systems naturally involves automata, and concepts such as timed automata [AD94] and hybrid automata [Hen96] have been widely used for modeling time-continuous dynamical systems with discrete state changes. While a huge variety of conditions is conceivable that one could call “combinatorial”, certain specially-structured conditions are considered frequently and across different application domains, such as dwell-time constraints on the switchings in switched systems [OWD16, Hey17, GHPS21, ZRS21].

In practice, optimal control problems have to be discretized before being solved (unless a discrete-time model was used in the first place), resulting in a mixed-integer optimization problem in finite-dimensional space. This problem can then be tackled by a variety of methods, including branch-and-cut approaches, for which one may want to strengthen the formulation by applying polyhedral results. A complete linear description of the convex hull for the polytope of 0-1 vectors satisfying dwell-time constraints, often called min-up/min-down constraints in this discrete context, is known in the original space [LLM04]. However, its size is exponential.

In discrete optimization, a well-known tool for obtaining *compact and tight* convex-hull descriptions are extended formulations. Such formulations consist of a feasible set in a higher-dimensional space, as well as a linear projection to the original space yielding exactly the desired convex hull; the size of an extended formulation is its number of inequalities. Many examples and construction techniques for extended formulation can be found in [CCZ13]. Indeed, a compact extended formulation, i.e., an extended formulation of a size that is polynomial in the original dimension, exists for the min-up/min-down polytope [RT05]. Moreover, it has been shown that the convex hull of all words in a regular binary language, i.e., a set of 0-1 strings that can be recognized by a finite-state automaton (FSA), can be described by a compact extended formulation, the size of which is linear in the number of states and the number of discretization points [FP15]. This flow-based formulation is essentially an extension of the Carr-Konjevod construction for the parity polytope [CK05], and the min-up/min-down polytope is in fact a special case.

Recently, it has been shown that the feasible sets of some typical classes of control problems admit extended formulations even in function space [Buc24], i.e., formulations consisting of a polynomial number of control variables and a polynomial number of constraints with linear operators in these controls; for a formal definition of a compact extended formulation in this setting, see [Buc24, Def. 3.1]. In particular, this applies to the case of dwell-time constraints. In this work, our aim is to devise extended formulations for a much larger class of problems.

From a complexity point of view, regular languages are one of the most accessible and well-studied type of languages. They can be modelled by finite-state automata: Given a discrete control as input, consisting of a finite list of letters, the automaton starts in one specified state, out of finitely many given states. The automaton reads one letter after the other and thereby moves from state to state. The next state is always chosen from a given list that depends on the last letter read and on the current state. This list may be empty, in which case the automaton is stuck. The input

is feasible if and only if it can be read completely without getting stuck and if the automaton is in a so-called accepting state after the entire input is read.

By introducing an appropriate continuous version of an FSA, we will devise an extended formulation for controls in function space that is analogous to the discrete flow-based formulation mentioned above. In other words, instead of first discretizing and then building the extended formulation, we propose to first build the extended formulation and then discretize it. This has several advantages. First, the model in function space gives rise to discretized models for arbitrary grids. In particular, it avoids artifacts that may appear only for specific discretizations. Second, we will see by an example that the number of states that are needed to model a discretized language may depend on the number of grid cells, which is clearly not desirable. On the contrary, the automaton in function space has a fixed number of states and thus ensures that an extended formulation for discretizations scales well with the grid size.

This paper is structured as follows. In Section 2 we introduce relevant notions and results from functional analysis. In Section 3 we present an automaton model that accepts continuous controls rather than discrete sequences of letters. For the controls characterized via this new class of automata, we derive extended formulations in function space in Section 4. The limits of the concepts will be investigated in Section 5, where we give a necessary condition in the style of a pumping lemma for our class of automata. Finally, Section 6 summarizes the findings and gives an outlook on possible directions for further research.

2 Functions of Bounded Variation

We first recall some basic facts concerning the function spaces considered in the following sections, focussing in particular on functions of bounded variation. We will limit ourselves to essential definitions and observations. For more details, we refer the reader to the monographs [AFP00] and [ABM14] or to the recent paper [Buc24].

Given some open set $\Omega \subseteq \mathbb{R}$, we start with the well-known reflexive Banach space $L^2(\Omega)$, which contains all equivalence classes of measurable functions $u: \Omega \rightarrow \mathbb{R}$ such that $|u|^2$ is Lebesgue integrable. Here, two functions are considered equivalent if they agree outside a null set. The space $L^2(\Omega)$ is equipped with the norm

$$\|u\|_{L^2(\Omega)} := \left(\int_{\Omega} |u(t)|^2 dt \right)^{1/2}.$$

We say that u_n converges strongly to u in $L^2(\Omega)$ if $\|u - u_n\|_{L^2(\Omega)} \rightarrow 0$ for $n \rightarrow \infty$. In this case, we write $u_n \rightarrow u$.

Throughout this paper, we will deal with the subspace of $L^2(\Omega)$ consisting of functions of bounded variation. For a precise definition, consider the seminorm on $L^2(\Omega)$ given by

$$|u|_{\text{BV}(\Omega)} := \sup_{\substack{\varphi \in C_c^\infty(\Omega) \\ \|\varphi\|_\infty \leq 1}} \int_{\Omega} u(t) \varphi'(t) dt,$$

where $\mathcal{C}_c^\infty(\Omega)$ denotes the set of all smooth functions $\varphi: \Omega \rightarrow \mathbb{R}$ with compact support. We then define

$$\text{BV}(\Omega) := \{u \in L^2(\Omega) \mid |u|_{\text{BV}(\Omega)} < \infty\}.$$

The distributional derivative Du of a function $u \in \text{BV}(\Omega)$ can be defined as the unique finite signed regular Borel measure μ satisfying

$$Du(\varphi) = \int_{\Omega} \varphi(t) \, d\mu \quad \forall \varphi \in \mathcal{C}_c^\infty(\Omega).$$

In our extended formulation, we will use constraints of the form $Du \geq 0$, meaning that

$$Du(\varphi) \geq 0 \quad \forall \varphi \in \mathcal{C}_c^\infty(\Omega), \varphi \geq 0.$$

The latter condition implies that the function u is monotonously increasing (outside a null set).

In the following, the set Ω will always describe the time horizon, where usually $\Omega = (0, T)$ for some $T \in \mathbb{R}_+$. For any subset $\Sigma \subseteq \mathbb{R}^n$, we now introduce the notation

$$\text{BV}(T, \Sigma) := \{u \in \text{BV}((0, T))^n \mid u \in \Sigma \text{ a.e. in } (0, T)\},$$

where ‘‘a.e.’’ is short for ‘‘almost everywhere’’ and indicates that the given condition is satisfied outside a null set. We will often refer to elements of $\text{BV}(T, \Sigma)$ as *controls* in the following. Each control $u \in \text{BV}(T, \Sigma)$ thus consists of n components u_i of bounded variation and we define

$$|u|_{\text{BV}} := \sum_{i=1}^n |u_i|_{\text{BV}}.$$

In case Σ is finite, it follows from the definition that all controls $u \in \text{BV}(T, \Sigma)$ are piecewise constant. For our extended models, we will in particular consider controls with $\Sigma = \{0, 1\}$. Such a control corresponds to a binary switch that may change only a finite number of times over the time horizon $(0, T)$. Moreover, we will usually require that u is zero outside the time horizon. To model this, we introduce the notation

$$\text{BV}_0(T, \Sigma) := \{u \in \text{BV}(\mathbb{R})^n \mid u = 0 \text{ a.e. in } (-\infty, 0) \cup (T, \infty), u \in \Sigma \text{ a.e. in } (0, T)\}.$$

Note that a condition of the form ‘‘ $u(0) = 0$ ’’ is not well-defined in L^2 or BV , since $\{0\}$ is a null set. We thus cannot directly fix the control on the boundary of $[0, T]$.

Given a control $u \in \text{BV}_0(T, \{0, 1\})$, it follows from the above definitions that the measure Du is discrete with $Du(t) = 1$ where u jumps from 0 to 1, $Du(t) = -1$ where u jumps from 1 to 0, and $Du(t) = 0$ everywhere else. In particular, we have that $Du(0)$ agrees with the right-sided limit of u in 0 and that $-Du(T)$ agrees with the left-sided limit of u in T .

3 An Automaton Model for Control Languages

In this section, we characterize the classes of controls for which we will devise extended formulations in the subsequent section. To this end, we define a variant of a finite-state automaton that

accepts continuous controls as input, and that recognizes the class of controls considered. Before we do so, we introduce some auxiliary definitions for continuous controls emphasizing the analogy to classical finite-state automata, by adapting standard definitions for letters and words from the discrete context:

Definition 3.1 (length, concatenation, subcontrol). Let $\Sigma \subseteq \mathbb{R}^n$ be given.

- We define the *length* $|u|$ of a control $u \in \text{BV}(T, \Sigma)$ as the length of the interval it is defined on, i.e., $|u| = T$.
- Similar to the concatenation operator for sequences of letters, for functions $u_1 \in \text{BV}(T_1, \Sigma)$ and $u_2 \in \text{BV}(T_2, \Sigma)$, we define the *concatenation* $u_1 \cdot u_2$ by

$$(u_1 \cdot u_2): (0, T_1 + T_2) \rightarrow \Sigma, \quad (u_1 \cdot u_2)(t) = \begin{cases} u_1(t), & \text{for } t \in (0, T_1) \\ u_2(t - T_1), & \text{for } t \in (T_1, T_2) \end{cases}.$$

It is easy to see that $u_1 \cdot u_2 \in \text{BV}(T_1 + T_2, \Sigma)$. We may also omit the “ \cdot ” and simply write $u_1 u_2$ where the meaning is clear from the context.

- Similar to the notion of a substring, we call $u' \in \text{BV}(T_{u'}, \Sigma)$ a *subcontrol* of $u \in \text{BV}(T_u, \Sigma)$ if there exist functions $v \in \text{BV}(T_v, \Sigma)$, $w \in \text{BV}(T_w, \Sigma)$ such that $u = v \cdot u' \cdot w$, allowing $T_v = 0$ or $T_w = 0$. In case $T_v = 0$ or $T_w = 0$, we also call u' *prefix* or *suffix* of u , respectively.
- In analogy to the Kleene star in the discrete setting, we will write $u \in \Sigma^*$ if $u \in \text{BV}(T, \Sigma)$ for some $T \in \mathbb{R}_+$. Moreover, $u \in \Sigma^+$ will be short for $u \in \Sigma^*$ and $|u| > 0$.

Note that in the definition of the concatenation $u_1 \cdot u_2$, it is not necessary that the controls u_1 and u_2 fit together, since $\{T_1\}$ is a null set and the values of u_1 in T_1 and of u_2 in 0 are not well-defined anyway. However, the concatenation may lead to jumps in T_1 .

We are now ready to introduce our automaton model.

Definition 3.2 (finite-state control automaton). A (non-deterministic) *finite-state control automaton* (FSCA) is a quintuple $(Q, \delta, \Sigma, q_0, F)$, where

- Q is a nonempty finite *set of states*,
- $\Sigma \subseteq \mathbb{R}^n$ is the nonempty and bounded *input alphabet*,
- $\delta: Q \times \Sigma^* \rightarrow 2^Q$ is the *transition function*,
- $q_0 \in Q$ is the *initial state*, and
- $F \subseteq Q$ is the set of *accepting states*.

For each state $q \in Q$, an *input dictionary* $D_q = P_q \cup C_q \subseteq \Sigma^*$ is given such that $\delta(q, f) = \emptyset$ for all $f \in \Sigma^* \setminus D_q$, i.e., when being in state q the automaton can only process controls in D_q . The set $\delta(q, f) \subseteq Q$ contains the possible states of the automaton after processing f , starting from state q . Here, $P_q \subseteq \Sigma^+$ is a finite set of non-empty *patterns* f , while C_q consists of all constant

controls $f \equiv \sigma$ for $\sigma \in \Gamma_q$, where $\Gamma_q \subseteq \Sigma$ is assumed to have cardinality at most one. If $\Gamma_q = \emptyset$ for all $q \in Q$, the automaton M is called *simple*. A finite-state control automaton is said to *recognize* a set $\mathcal{L} \subseteq \Sigma^*$ of controls (a *control language*) if for each $u \in \Sigma^*$, we have $u \in \mathcal{L}$ if and only if there exists an *accepting sequence* for u , i.e., a representation $u = f_1 \cdot \dots \cdot f_k$ for some $k \in \mathbb{N}$ and a finite sequence of states $q_0 = r_0, r_1, \dots, r_k$ in Q such that $r_{i+1} \in \delta(r_i, f_{i+1})$ for $i \in \{0, \dots, k-1\}$ and $r_k \in F$.

Note that requiring accepting sequences to be finite is without loss of generality. A formal argument can be found in the proof of Lemma 3.4.

Definition 3.3 (regular control language). A control language \mathcal{L} is called *regular* if there exists a finite-state control automaton recognizing \mathcal{L} .

A finite-state control automaton thus starts in state q_0 and at position zero of the input u . When being in state q and at position t , it either reads a complete pattern $f \in P_q$ at position $[t, t + |f|]$ in u , then changing to any state in $\delta(q, f)$ and moving to position $t + |f|$, or it reads an element of C_q , i.e., a control of arbitrary length that is constantly σ with $\sigma \in \Gamma_q$.

We emphasize that $|\Gamma_q| \leq 1$ is required by definition. Otherwise, the automaton would be allowed to switch between the constants in Γ_q arbitrarily often, which would lead to controls of arbitrarily large variation. More specifically, we can show:

Lemma 3.4. *Let \mathcal{L} be a regular control language. Then each $u \in \mathcal{L}$ has an accepting sequence of length $O(|u|)$. Moreover, for each $u \in \mathcal{L}$ and each subcontrol v of u , we have $|v|_{\text{BV}} \in O(|v|)$.*

Proof. Define $P := \bigcup_{q \in Q} P_q$ and let $\mu_P := \min\{|f| \mid f \in P\} > 0$ be the shortest length of any pattern in the automaton. By definition, every control $u \in \mathcal{L}$ is of the form $u = f_1 \cdot \dots \cdot f_k$, where each f_i either belongs to P or is constant. Moreover, we may assume that at the beginning and between two consecutive patterns from P there is at most one constant control, since $|\Gamma_q| \leq 1$ and two consecutive constant patterns with the same letter can be merged to one constant control. Thus, for all $i = 1, \dots, k-1$, we obtain $|f_i| + |f_{i+1}| \geq \mu_P$. Hence $|u| \geq \lfloor \frac{k}{2} \rfloor \mu_P \geq \frac{k-1}{2} \mu_P$, which implies $k \leq \frac{2}{\mu_P} |u| + 1$.

To show the second assertion, we use that the finitely many patterns in P all have bounded variation by definition, thus $\kappa := \max\{|f|_{\text{BV}} \mid f \in P\}$ is finite. Between f_i and f_{i+1} , each jump is at most $\text{diam}(\Sigma) := \sup_{\sigma_1, \sigma_2 \in \Sigma} \|\sigma_1 - \sigma_2\|_1$, which is finite since Σ is assumed to be bounded. An upper bound on the total variation of any subcontrol v of u is thus

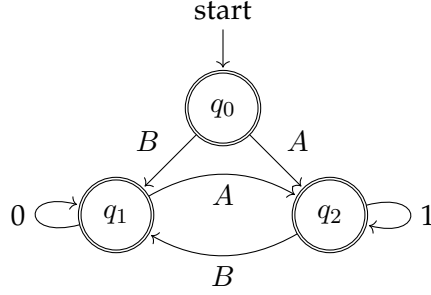
$$\left(\frac{|v|}{\mu_P} + 1\right)(\kappa + \text{diam}(\Sigma)) = \frac{1}{\mu_P}(\kappa + \text{diam}(\Sigma))|v| + \kappa + \text{diam}(\Sigma) \in O(|v|).$$

□

Example 3.5 (Min-up/Min-Down). Consider the language of binary controls satisfying some min-up/min-down constraints, i.e.,

$$\mathcal{L}_{\text{MINUPDOWN}(L, \ell)} := \{u \in \{0, 1\}^* \mid \text{blocks of 1-controls have length at least } L, \\ \text{blocks of 0-controls have length at least } \ell\}$$

for given $L, \ell \in \mathbb{Q}_+$. We can realize $\mathcal{L}_{\text{MINUPDOWN}(L, \ell)}$ by the automaton



where the pattern A is constantly one of length L and B is constantly zero of length ℓ . Accepting states are marked with double circles. Thus $Q = \{q_0, q_1, q_2\}$, $F = \{q_0, q_1, q_2\}$, and $P_{q_0} = \{A, B\}$, $\Gamma_{q_0} = \emptyset$, $P_{q_1} = \{A\}$, $\Gamma_{q_1} = \{0\}$, $P_{q_2} = \{B\}$, $\Gamma_{q_2} = \{1\}$.

Comparing FSCA with classical FSA, we note that FSCA process whole patterns at once. It is also possible – and sometimes done – to define classical FSA that process whole words, i.e., multiple letters, per transition. Conversely, however, there is no version of FSCA that processes atomic parts of input elements, as such a thing does not exist for our setting.

Moreover, FSCA as defined in Definition 3.2 are non-deterministic in the sense that the automaton can end up in different states for the same input control u , depending on how u is decomposed and which transitions are chosen. In order to define a notion of determinism for FSCA, first recall that in a classical deterministic FSA, a single input letter must uniquely determine the next transition. However, in the case of control automata, the input does not consist of a finite string. Even worse, for an input $u \in \Sigma^*$, point-wise evaluations of u are not well-defined. This problem can be resolved by considering right-sided limits $u(t+) := \lim_{\tau \searrow t} u(\tau)$, which are well-defined for functions of bounded variation [AFP00, Theorem 3.28]. We can thus define a deterministic FSCA as follows:

Definition 3.6 (deterministic FSCA). An FSCA $M = (Q, \delta, \Sigma, q_0, F)$ is called *deterministic* if the following conditions hold:

- for all $q \in Q$ and all $f \in \Sigma^*$, we have $|\delta(q, f)| \leq 1$
- for all $q \in Q$ and all $f, f' \in P_q$ with $f \neq f'$, we have $f(0+) \neq f'(0+)$, and
- for all $q \in Q$ and all $f \in P_q$, we have $f(0+) \notin \Gamma_q$.

Definition 3.6 ensures that for a deterministic finite-state control automaton M and input u , there is at most one state M can possibly be in after processing u . More precisely, if the remaining input to be processed is $u' \in \Sigma^*$, then either $u'(0+) = f(0+)$ for exactly one $f \in P_q$, or $u'(0+) \in \Gamma_q$, in which case the input must be processed until the next t is reached with $u'(t+) \neq u'(0+)$, or the automaton cannot proceed at all.

4 Extended Formulations

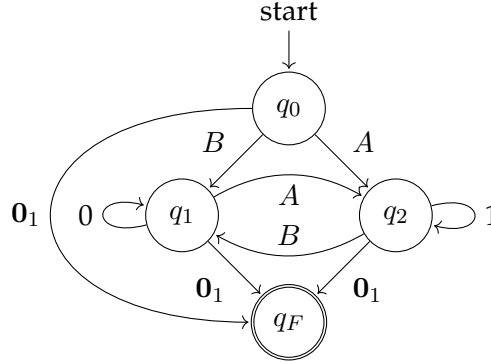
For the following, it will be convenient to interpret a finite-state control automaton $(Q, \delta, \Sigma, q_0, F)$ as a directed multigraph $G = (Q, A)$ with

$$A := \{(q_1, q_2) \mid \exists f \in P_{q_1} : q_2 \in \delta(q_1, f)\} \cup A_0, \quad A_0 := \{(q, q) \mid \Gamma_q \neq \emptyset\}.$$

Where no confusion can arise, we will identify a loop $(q, q) \in A_0$ with the unique letter $\sigma \in \Gamma_q$ and an arc $a = (q_1, q_2) \in A \setminus A_0$ with the corresponding pattern $f \in P_{q_1}$ with $q_2 \in \delta(q_1, f)$. In particular, for $a = (q_1, q_2) \in A \setminus A_0$, we will write $|a|$ for the length of said pattern. Finally, we will denote the outgoing and incoming arcs in $q \in Q$, including the loops, by $\delta^+(q)$ and $\delta^-(q)$, respectively.

To simplify the presentation, we will assume that $F = \{q_F\}$ with $q_F \neq q_0$. This is without loss of generality, since otherwise one can add a new unique accepting state q_F to the automaton and connect all former accepting states to q_F by an arc a of length $|a| = 1$, at the same time increasing the time horizon T by 1.

Example 4.1. The language $\mathcal{L}_{\text{MINUPDOWN}(L, \ell)}$ from Example 3.5 can be realized equivalently by the following automaton with a unique accepting state:



Here $\mathbf{0}_1$ denotes the pattern of length 1 being constantly zero. Obviously, the accepted inputs for the automaton given in Example 3.5 are exactly the accepted inputs for this new automaton shortened by one time unit.

Remark 4.2. By a symmetric construction, one could also allow multiple initial states. In this case, the starting point could be chosen non-deterministically from the set of initial states. While this extension would lead to more compact automata in some cases, we decided to stick to the more common model with a unique initial state in Definition 3.2.

Definition 4.3. Let $M = (Q, \delta, \Sigma, q_0, \{q_F\})$ be a finite-state control automaton with $q_F \neq q_0$. Given any $m \in \mathbb{N}$ with $m_a := m|a|/T \in \mathbb{N}$ for all $a \in A \setminus A_0$, the *time-expanded network* $N(M, m)$ consists of the nodes $Q_m = \{q^{(j)} \mid q \in Q, j \in \{0, \dots, m\}\}$ and the arcs

$$A_m = \{a^{(j)} := (q_1^{(j)}, q_2^{(j+m_a)}) \mid a = (q_1, q_2) \in A, j \in \{0, \dots, m - m_a\}\},$$

where we define $m_a := 1$ for $a \in A_0$. The node $q_0^{(0)}$ has supply 1 in $N(M, m)$, the node $q_F^{(m)}$ has demand 1, and all other nodes have supply and demand zero.

The first step for obtaining an extended formulation in function space is to rewrite the finite-dimensional network flow problem in $N(M, m)$ by means of a variable transformation. Essentially, we sum up the flow on each copy of the original arc at every point in time, instead of starting a new arc at every point in time and keeping its value constant until the flow has crossed it. This will make the model accessible to the desired transfer to infinite dimension, since we then have to deal with an infinite number of time points and hence an infinite number of copies of a given arc. The construction is described by the following two lemmas.

Lemma 4.4. *Given a finite-state control automaton $M = (Q, \delta, \Sigma, q_0, \{q_F\})$ with $q_F \neq q_0$ and $m \in \mathbb{N}$ such that $m_a = m|a|/T \in \mathbb{N}$ for all $a \in A$. Then the set of feasible flows x in $N(M, n)$ is completely described by the linear system*

$$\begin{aligned} x_{a^{(j)}} &\geq 0 && \forall a \in A, j \in \{0, \dots, m - m_a\} \\ \sum_{\substack{a \in \delta^+(q) \\ a \in A_0}} x_{a^{(j)}} + \sum_{\substack{a \in \delta^+(q) \\ a \notin A_0}} \sum_{i=0}^j x_{a^{(i)}} - \sum_{\substack{a \in \delta^-(q) \\ a \notin A_0}} \sum_{i=0}^j x_{a^{(i-m_a)}} &= \bar{b}_{q,j} && \forall q \in Q, j \in \{0, \dots, m\}, \end{aligned}$$

where

$$\bar{b}_{q,j} = \begin{cases} 1 & \text{if } q = q_0 \\ -1 & \text{if } q = q_F \text{ and } j = m \\ 0 & \text{otherwise} \end{cases}$$

and all $x_{a^{(i)}}$ with $i \notin \{0, \dots, m - m_a\}$ are assumed to be zero.

Proof. The classical network flow model for $N(M, m)$ reads

$$\begin{aligned} x_{a^{(j)}} &\geq 0 && \forall a \in A, j \in \{0, \dots, m - m_a\} \\ \sum_{a \in \delta^+(q)} x_{a^{(j)}} - \sum_{a \in \delta^-(q)} x_{a^{(j-m_a)}} &= b_{q,j} && \forall q \in Q, j \in \{0, \dots, m\} \end{aligned}$$

with

$$b_{q,j} = \begin{cases} 1 & \text{if } q = q_0 \text{ and } j = 0 \\ -1 & \text{if } q = q_F \text{ and } j = m \\ 0 & \text{otherwise.} \end{cases}$$

Summing up the flow conservation constraints arc-wise for $i = 0, \dots, j$, for $j = 0, \dots, m$, we obtain the equivalent model

$$\begin{aligned} x_{a^{(j)}} &\geq 0 && \forall a \in A, j \in \{0, \dots, m - m_a\} \\ \sum_{a \in \delta^+(q)} \sum_{i=0}^j x_{a^{(i)}} - \sum_{a \in \delta^-(q)} \sum_{i=0}^j x_{a^{(i-m_a)}} &= \bar{b}_{q,j} && \forall q \in Q, j \in \{0, \dots, m\}. \end{aligned}$$

Considering arcs in A_0 separately, the left-hand side of the second constraint reads

$$\sum_{\substack{a \in \delta^+(q) \\ a \in A_0}} \sum_{i=0}^j (x_{a^{(i)}} - x_{a^{(i-1)}}) + \sum_{\substack{a \in \delta^+(q) \\ a \notin A_0}} \sum_{i=0}^j x_{a^{(i)}} - \sum_{\substack{a \in \delta^-(q) \\ a \notin A_0}} \sum_{i=0}^j x_{a^{(i-m_a)}}.$$

Using $\sum_{i=0}^j (x_{a^{(i)}} - x_{a^{(i-1)}}) = x_{a^{(j)}}$, we obtain the desired model. \square

Lemma 4.5. *Under the assumptions of Lemma 4.4, the feasible flows x in $N(M, m)$ correspond bijectively to vectors $(y_a^{(j)})_{a \in A, j=0, \dots, m-1}$ belonging to the polytope P_m described by*

$$\begin{aligned} \sum_{k=0}^{\infty} y_a^{(j-km_a)} - \sum_{k=0}^{\infty} y_a^{(j-1-km_a)} &\geq 0 \quad \forall a \in A \\ \sum_{\substack{a \in \delta^+(q) \\ a \in A_0}} \sum_{k=0}^{\infty} (y_a^{(j-km_a)} - y_a^{(j-1-km_a)}) + \sum_{\substack{a \in \delta^+(q) \\ a \notin A_0}} \sum_{k=0}^{\infty} y_a^{(j-km_a)} - \sum_{\substack{a \in \delta^-(q) \\ a \notin A_0}} \sum_{k=1}^{\infty} y_a^{(j-km_a)} &= \bar{b}_{q,j} \quad \forall q \in Q, \end{aligned}$$

where all constraints are required for all $j = 0, \dots, m$ and we set $y_a^{(j)} := 0$ for $j \notin \{0, \dots, m-1\}$, so that all appearing sums are finite. The bijection is given by the linear maps

$$y_a^{(j)} := \sum_{i=j-m_a+1}^j x_{a^{(i)}}$$

and

$$x_{a^{(j)}} := \sum_{k=0}^{\infty} (y_a^{(j-km_a)} - y_a^{(j-1-km_a)}).$$

In particular, the polytope P_m is integer.

Proof. We first show that the two mappings are inverse to each other, by induction over j . Indeed, we have

$$\begin{aligned} \sum_{k=0}^{\infty} (y_a^{(j-km_a)} - y_a^{(j-1-km_a)}) &= y_a^{(j)} - y_a^{(j-1)} + \sum_{k=0}^{\infty} (y_a^{(j-m_a-km_a)} - y_a^{(j-1-m_a-km_a)}) \\ &= y_a^{(j)} - y_a^{(j-1)} + x_{a^{(j-m_a)}} \\ &= \sum_{i=j-m_a+1}^j x_{a^{(i)}} - \sum_{i=j-m_a}^{j-1} x_{a^{(i)}} + x_{a^{(j-m_a)}} \\ &= x_{a^{(j)}} - x_{a^{(j-m_a)}} + x_{a^{(j-m_a)}} \\ &= x_{a^{(j)}}, \end{aligned}$$

where the second equation follows from the induction hypothesis. Conversely,

$$\sum_{i=j-m_a+1}^j x_{a^{(i)}} = \sum_{i=j-m_a+1}^j \sum_{k=0}^{\infty} (y_a^{(i-km_a)} - y_a^{(i-1-km_a)}) = y_a^{(j)}.$$

Now the result follows from Lemma 4.4 by substituting $x_{a^{(i)}}$ and $x_{a^{(i-1)}}$ according to the given bijection, using

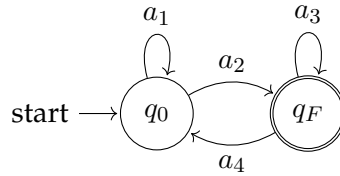
$$\sum_{i=0}^j x_{a^{(i)}} = \sum_{k=0}^{\infty} \sum_{i=0}^j (y_a^{(i-km_a)} - y_a^{(i-1-km_a)}) = \sum_{k=0}^{\infty} y_a^{(j-km_a)}$$

and

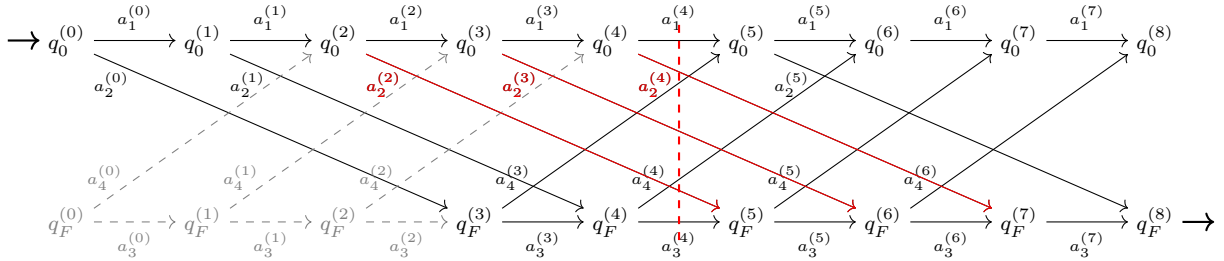
$$\sum_{i=0}^j x_{a^{(i-m_a)}} = \sum_{k=0}^{\infty} \sum_{i=0}^j (y_a^{(i-m_a-km_a)} - y_a^{(i-1-m_a-km_a)}) = \sum_{k=0}^{\infty} y_a^{(j-m_a-km_a)} = \sum_{k=1}^{\infty} y_a^{(j-km_a)}.$$

The integrality of P_m follows from the fact that the classical network flow model is integer [FF62] and that the given bijections preserve integrality. \square

Example 4.6. In order to illustrate the statements of Lemma 4.4 and Lemma 4.5, consider the following finite-state control automaton



with $A_0 = \{a_1, a_3\}$. Let $|a_2| = 6$, $|a_4| = 4$, $T = 16$, and $m = 8$. Then $m_{a_1} = m_{a_3} = 1$, $m_{a_2} = 3$, and $m_{a_4} = 2$. The time-expanded network G_8 is as follows:



The dashed gray arcs are part of G_8 by definition, but cannot be reached by any feasible flow. Considering, e.g., the original arc a_2 at time 4, the bijection described in Lemma 4.5 yields

$$y_{a_2}^{(4)} = x_{a_2^{(2)}} + x_{a_2^{(3)}} + x_{a_2^{(4)}},$$

thus $y_{a_2}^{(4)}$ is the sum of flows on those copies of arc a_2 being “active” between time points 4 and 5; this is marked by the dashed red line and the red arcs in the figure above.

We are now ready to prove our main result. It states that there exists a compact extended formulation in function space for the set of controls (of fixed length T) that are accepted by a given finite-state control automaton.

Theorem 4.7. *Let \mathcal{L} denote a language over Σ and let $M = (Q, \delta, \Sigma, q_0, F)$ be any finite-state control automaton recognizing the language \mathcal{L} . Then for every $T \in \mathbb{Q}_+$ there exists an extended formulation of $\overline{\text{conv}}(\mathcal{L} \cap \text{BV}(T, \Sigma))$ with polynomially many controls and linear constraints.*

Proof. First assume as above that $F = \{q_F\}$ with $q_F \neq q_0$. The extended formulation contains one control $y_a \in \text{BV}_0(T, [0, 1])$ for each $a \in A$, representing the total amount of flow on arc a at a given time point. The first constraint is

$$\sum_{a \in A} y_a = \chi_{[0, T]}, \quad (\text{M0})$$

stating that at each time point in $[0, T)$ the sum of flow over all arcs is one. Next, we require

$$Dz_a \geq 0 \quad \forall a \in A \setminus A_0 \quad (\text{M1})$$

where we define

$$z_a = \sum_{k=0}^{\infty} V_{k|a|}(y_a). \quad (1)$$

Here, the linear map $V_r: \text{BV}_0(T, \mathbb{R}) \rightarrow \text{BV}_0(T, \mathbb{R})$ is defined by $V_r(z)(t) := z(t - r)$ for $r \in \mathbb{R}_+$. Note that the sum in (1) is actually finite, since $V_{k|a|}(y_a) = 0$ on $(-\infty, T)$ for all $k > \lceil T/|a| \rceil$. In particular, we obtain $z_a \in \text{BV}_0(T, \mathbb{R})$ again.

Now assume that some given binary controls $y_a \in \text{BV}_0(T, \{0, 1\})$, $a \in A$, satisfy both (M0) and (M1). By integrality and the bounded variation, there are finitely many time points t_1, \dots, t_r with $0 =: t_0 < t_1 < \dots < t_r < t_{r+1} := T$ where any of the controls y_a changes. By (M0), for each $i = 1, \dots, r + 1$, there exists exactly one $a_i \in A$ such that $y_{a_i} = 1$ on (t_{i-1}, t_i) and $y_a = 0$ on (t_{i-1}, t_i) for $a \in A \setminus \{a_i\}$. We claim that (M1) then implies that $t_i - t_{i-1}$ is an integer multiple of $|a_i|$ whenever $a_i \in A \setminus A_0$. Assume on contrary that $i \in \{1, \dots, r + 1\}$ is minimal with $a_i \in A \setminus A_0$ and $t_i - t_{i-1} \notin |a_i| \cdot \mathbb{N}$. Choose $\ell \in \mathbb{N}_0$ such that $t_i - (\ell + 1)|a_i| < t_{i-1} < t_i - \ell|a_i|$. We have

$$Dz_{a_i}(t_i) = Dy_{a_i}(t_i) + \sum_{k=1}^{\ell} Dy_{a_i}(t_i - k|a_i|) + \sum_{k=\ell+1}^{\infty} Dy_{a_i}(t_i - k|a_i|),$$

where $Dy_{a_i}(t_i) = -1$ and the first sum on the right-hand side is zero as y_{a_i} is constant on (t_{i-1}, t_i) . The second sum on the right-hand side is also zero by the minimality of i . In summary, the right-hand side is -1 , contradicting $Dz_{a_i} \geq 0$ and thus showing our claim.

We next claim that $Dz_a(t) = 1$ if and only if arc a is entered at time t , while $Dz_a(t) = 0$ otherwise. To show this, let $t \in [t_{i-1}, t_i)$. If $a \neq a_i$, we clearly have $Dz_a(t) = 0$, so assume $a = a_i$ and choose $\ell \in \mathbb{N}_0$ minimally such that $t - t_{i-1} \leq \ell|a|$. If $t - t_{i-1} = \ell|a|$, i.e., if arc a is entered at time t , we obtain that $Dz_a(t) = Dz_a(t_{i-1}) = 1$. Otherwise, we have $t - (\ell - 1)|a| \in (t_{i-1}, t_{i-1} + |a|)$ and hence $Dz_a(t) = 0$, since $Dz_a(t) = 1$ would imply $y_a \geq 2$ in $(t, t_{i-1} + |a|)$.

Using the last claim, we can finally model flow conservation by the constraints

$$\sum_{a \in \delta^+(q) \cap A_0} Dy_a + \sum_{a \in \delta^+(q) \setminus A_0} Dz_a - \sum_{a \in \delta^-(q) \setminus A_0} DV_{|a|}(z_a) = \begin{cases} \delta_0 & \text{if } q = q_0 \\ -\delta_T & \text{if } q = q_F \\ 0 & \text{otherwise,} \end{cases}$$

where δ_t denotes the Dirac measure for time point t . Since all functions appearing in the above constraint are zero before 0, we can equivalently require

$$\sum_{a \in \delta^+(q) \cap A_0} y_a + \sum_{a \in \delta^+(q) \setminus A_0} z_a - \sum_{a \in \delta^-(q) \setminus A_0} V_{|a|}(z_a) = \begin{cases} \chi_{[0, \infty)} & \text{if } q = q_0 \\ -\chi_{[T, \infty)} & \text{if } q = q_F \\ 0 & \text{otherwise.} \end{cases} \quad (\text{M2})$$

Summing up the constraints (M2), for all $q \in Q$, we actually obtain (M0). We claim that the desired complete model is given by (M1)–(M2), where we can either add (1) as constraints or substitute z_a in (M1)–(M2). Indeed, the proof so far shows that any integer feasible solution to (M1)–(M2) gives rise to a feasible flow over time through the automaton. Conversely, it is easy to see that a feasible flow defines a feasible solution to (M1)–(M2).

We next show that the formulation is integer, i.e., that the feasible set of the model is the closed convex hull of its integer feasible solutions. For this, we approximate the extended formulation by its discretizations and use Lemma 4.5 above; this proof strategy is very similar to the one applied in [Buc24]. So let $y_a \in \text{BV}_0(T, [0, 1])$, $a \in A$, satisfy the conditions (M1)–(M2). Choose $m \in \mathbb{N}$ such that $m|a|/T \in \mathbb{N}$ for all $a \in A \setminus A_0$. Moreover, given $a \in A$ and $k \in \mathbb{N}$, let $y_a^{(k)} \in [0, 1]^{km}$ be defined as the piecewise average of y_a on the grid with km cells, i.e.,

$$(y_a^{(k)})_i := \frac{km}{T} \int_{(i-1)\frac{T}{km}}^{i\frac{T}{km}} y_a(t) dt,$$

and define $\bar{y}_a^{(k)}$ as the corresponding piecewise constant function on $[0, T)$. Then one can show that $y_a = \lim_{k \rightarrow \infty} \bar{y}_a^{(k)}$ in $L^2_0(0, T)$ for all $a \in A$; see [Buc24, Lemma 2.4]. Now by construction, the vector $(y_a^{(k)})_{a \in A}$ belongs to the polytope P_{km} defined in Lemma 4.5. Indeed, the inequalities in Lemma 4.5 are implied by (M1) using [Buc24, Lemma 2.5], while the equations follows directly from (M2). Since P_{km} has binary vertices by Lemma 4.5, we derive that $(y_a^{(k)})_{a \in A}$ is a convex combination of binary vectors in P_{km} . The latter, interpreted as piecewise constant functions, belong to $\text{BV}_0(T, [0, 1])$ and satisfy (M1)–(M2) when defining z by (1). In particular, defining $\bar{z}_a^{(k)}$ as the corresponding convex combination of the z_a for all $a \in A \setminus A_0$, we have $(y, z) = \lim_{k \rightarrow \infty} (\bar{y}^{(k)}, \bar{z}^{(k)})$ and hence (y, z) belongs to the closed convex hull of integer controls satisfying (M1)–(M2).

We claim that the projection to the original space is now given by the linear map

$$\pi: \text{BV}_0(T, [0, 1])^A \times \text{BV}_0(T, \mathbb{R}_+)^{A \setminus A_0} \rightarrow \text{BV}(T, \Sigma)$$

defined by

$$\pi(y, z)(t) = \sum_{a \in A_0} a y_a(t) + \sum_{a \in A \setminus A_0} \int_{t-|a|}^t a(t-x) dDz_a, \quad (2)$$

where we again identify arcs $(q, q) \in A_0$ and $(q_1, q_2) \in A \setminus A_0$ with the corresponding letters in Γ_q and patterns in P_{q_1} , respectively. Indeed, consider the time points t_1, \dots, t_r defined above and let $t \in (t_{i-1}, t_i)$ for $i \in \{1, \dots, r+1\}$. If the automaton is in a loop $a_i \in A_0$ on (t_{i-1}, t_i) ,

we have $y_a = 1$ on (t_{i-1}, t_i) and the projection correctly yields the constant associated with a_i at time t . Otherwise, if $a_i \in A \setminus A_0$, the model guarantees $Dz_{a_i}(t_{i-1}) = 1$, as argued above. Thus $\int_{t-|a_i|}^t a_i(t-x) dDz_{a_i} = a_i(t-t_{i-1})$ and all other terms are zero.

Finally, if the assumption of a unique accepting state q_F different from q_0 does not hold, we can use the construction described above and obtain an extended formulation for the language \mathcal{L}' consisting of all elements of \mathcal{L} extended by a zero string of length 1. Since restriction to the time horizon $[0, T]$ is a linear projection, the extended formulation for \mathcal{L}' is an extended formulation for \mathcal{L} as well. \square

Note that for a loop $a \in A \setminus A_0$, i.e., an arc $a \in \delta^-(q) \cap \delta^+(q)$ for some $q \in Q$, the two terms corresponding to a in (M2) simplify to $z_a - V_{|a|}(z_a) = y_a$, so that (M2) in fact does not distinguish between loops in A_0 and loops in $A \setminus A_0$. However, the latter must satisfy $Dz_a \geq 0$.

If the given alphabet Σ is finite and hence all patterns $f \in A \setminus A_0$ are piecewise constant, the projection (2) can be rewritten as follows: Let $0 = \tau_f^{(0)} < \tau_f^{(1)} < \dots < \tau_f^{(n_f)} = |a|$ be the switching points of f and let $f \equiv v_f^{(i)}$ on $(\tau_f^{(i-1)}, \tau_f^{(i)})$. Then

$$\begin{aligned} \int_{t-|a|}^t f(t-x) dDz_a &= \sum_{i=1}^{n_f} \int_{t-|a|+\tau_f^{(i-1)}}^{t-|a|+\tau_f^{(i)}} f(t-x) dDz_a \\ &= \sum_{i=1}^{n_f} v_f^{(n_f-i+1)}(z_a(t-|a|+\tau_f^{(i)}) - z_a(t-|a|+\tau_f^{(i-1)})) \end{aligned}$$

and hence the projection simplifies to

$$\pi(y, z) = \sum_{a \in A_0} ay_a + \sum_{f \in A \setminus A_0} \sum_{i=1}^{n_f} v_f^{(n_f-i+1)}(V_{|a|-\tau_f^{(i)}}(z_a) - V_{|a|-\tau_f^{(i-1)}}(z_a)).$$

Example 4.8. As an illustration, consider the FSCA described in Example 4.6, again with $T = 16$. Then the controls in the extended formulation are

$$y_{00}, y_{0F}, y_{F0}, y_{FF} \in \text{BV}_0(16, [0, 1]), \quad z_{0F}, z_{F0} \in \text{BV}_0(16, \mathbb{R}_+)$$

where the subscript xy refers to the arc leading from q_x to q_y . The two sets of controls are connected by the constraints (1),

$$\begin{aligned} z_{0F} &= y_{0F} + V_6(y_{0F}) + V_{12}(y_{0F}) \\ z_{F0} &= y_{F0} + V_4(y_{F0}) + V_8(y_{F0}) + V_{12}(y_{F0}) + V_{16}(y_{F0}). \end{aligned}$$

As in (M1), we need to require $Dz_{0F}, Dz_{F0} \geq 0$. Finally, the flow conservation constraints (M2) read

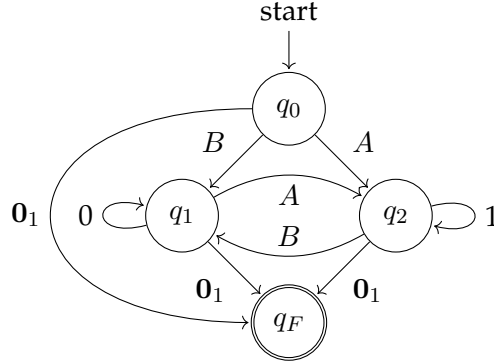
$$\begin{aligned} y_{00} + z_{0F} - V_4(z_{F0}) &= \chi_{[0, \infty)} \\ y_{FF} + z_{F0} - V_6(z_{0F}) &= -\chi_{[16, \infty)}, \end{aligned}$$

which concludes the construction of the extended formulation.

Example 4.9 (Min-up/Min-Down, cont'd). Consider the closed convex hull of the set of binary controls satisfying some min-up/min-down constraints, i.e.,

$$M_{\text{MINUPDOWN}(L,\ell)}^T := \overline{\text{conv}}\{u \in \text{BV}(T, \{0, 1\}) \mid \text{blocks of 1-controls have length at least } L, \\ \text{blocks of 0-controls have length at least } \ell\}.$$

Recall from Example 3.5 (and the modification in Example 4.1) that the language $\mathcal{L}_{\text{MINUPDOWN}(L,\ell)}$ associated with $M_{\text{MINUPDOWN}(L,\ell)}^T$ can be described by the finite-state control automaton



where the pattern A is constantly one of length L , the pattern B is constantly zero of length ℓ , and the pattern 0_1 is constantly zero of length 1. Then the extended formulation constructed in the proof of Theorem 4.7 contains the controls

$$\begin{aligned} y_{01}, y_{02}, y_{0F}, y_{11}, y_{12}, y_{1F}, y_{21}, y_{22}, y_{2F} &\in \text{BV}_0(T+1, [0, 1]) \\ z_{01}, z_{02}, z_{0F}, z_{12}, z_{1F}, z_{21}, z_{2F} &\in \text{BV}_0(T+1, \mathbb{R}_+), \end{aligned}$$

where

$$\left. \begin{aligned} z_{01} &= \sum_{k=0}^{\lfloor T+1/\ell \rfloor} V_{k\ell}(y_{01}) & z_{02} &= \sum_{k=0}^{\lfloor T+1/L \rfloor} V_{kL}(y_{02}) & z_{0F} &= \sum_{k=0}^{\lfloor T+1 \rfloor} V_k(y_{0F}) \\ z_{12} &= \sum_{k=0}^{\lfloor T+1/L \rfloor} V_{kL}(y_{12}) & z_{1F} &= \sum_{k=0}^{\lfloor T+1 \rfloor} V_k(y_{1F}) \\ z_{21} &= \sum_{k=0}^{\lfloor T+1/\ell \rfloor} V_{k\ell}(y_{21}) & z_{2F} &= \sum_{k=0}^{\lfloor T+1 \rfloor} V_k(y_{2F}). \end{aligned} \right\} \quad (3)$$

The extension of the time horizon by one is due to the modification described in Example 4.1. The constraints are

$$Dz_{01}, Dz_{02}, Dz_{0F}, Dz_{12}, Dz_{1F}, Dz_{21}, Dz_{2F} \geq 0$$

and

$$\begin{aligned} z_{01} + z_{02} + z_{0F} &= \chi_{[0,\infty)} \\ y_{11} + z_{12} + z_{1F} - V_\ell(z_{01}) - V_\ell(z_{21}) &= 0 \\ y_{22} + z_{21} + z_{2F} - V_L(z_{02}) - V_L(z_{12}) &= 0 \\ -V_1(z_{0F}) - V_1(z_{1F}) - V_1(z_{2F}) &= -\chi_{[T+1,\infty)}, \end{aligned}$$

while the projection is given by

$$\begin{aligned} u(t) &= y_{22}(t) + \int_{t-L}^t dDz_{02} + \int_{t-L}^t dDz_{12} \\ &= y_{22}(t) + z_{02}(t) - V_L(z_{02})(t) + z_{12}(t) - V_L(z_{12})(t) \end{aligned}$$

restricted to t in $(0, T)$. Clearly, the z -variables can be eliminated using the linear equations (3), so that the resulting extended model only contains the y -variables.

Remark 4.10. Theorem 4.7 thus yields an extended formulation for the closed convex hull of all controls in \mathcal{L} with a fixed length T . In general, the latter controls do *not* agree with the prefixes of length T of controls in \mathcal{L} , since patterns must be fully processed before a new state is reached, and not all states will be accepting states in general. However, by adding arcs of type A_0 to all accepting states, we can also use Theorem 4.7 to model the closed convex hull of all controls in \mathcal{L} with length *at most* T . We can therefore also model the set of controls of length T that are prefixes of controls in \mathcal{L} of length at most $T' > T$, by just cutting at T for the projection. Finally, one can give an estimate for T' such that this set of prefixes actually coincides with the set of prefixes of any control in \mathcal{L} , see Lemma 5.4 in the following section.

5 Limitations

The purpose of this section is to demonstrate the limitations of finite-state control automata and to give tools for showing that a particular control language cannot be regular, i.e., characterized via an FSCA. First of all, we already know from Lemma 3.4 that a language cannot be regular if it allows to switch arbitrarily often on a bounded time horizon. For this reason, we will only consider examples containing a minimum dwell-time in the following.

Classically, non-representability of a language by an FSA can be proven by a pumping lemma. There are different versions; the following one even provides an equivalent characterization for the class of languages that can be recognized by FSA, i.e., the class of regular languages.

Lemma 5.1 (Jaffe's pumping lemma, [Jaf78]). *A (classical) language \mathcal{L} can be recognized by an FSA if and only if there exists a pumping number $n \in \mathbb{N}$ such that for all $y \in \mathcal{L}$ with $|y| = n$ there exists a representation of y as*

$$y = vwx \quad \text{with} \quad |vw| \leq n, |w| > 0$$

such that for all suffixes $z \in \Sigma^$ and for all $i \in \mathbb{N}_0$ it holds*

$$yz \in \mathcal{L} \iff vw^i xz \in \mathcal{L}.$$

A short proof can be found in [Jaf78]. It follows from the proof that we can choose n as the number of states of any deterministic FSA accepting \mathcal{L} . The assumption of determinism is without loss of generality, as deterministic and non-deterministic FSA are capable of recognizing the same class of languages, and there is a standard construction for obtaining a deterministic FSA from a non-deterministic one; see, e.g., [HMU01, Section 2.3.5]. However, the construction increases the number of states exponentially.

In order to extend the main idea of the proof for the “only if” direction in Lemma 5.1 to the infinite-dimensional case, we would need to show equivalence for deterministic and non-deterministic versions of FSCA. It is, however, not clear that such an equivalence holds. Just for the special case of a simple FSCA, we will see the classical construction mentioned above as part of the proof of Lemma 5.5. Even without a determinism argument, we can show the following pumping lemma for FSCA, which is akin to the arguably most classical version of the pumping lemma for regular languages.

Lemma 5.2 (Pumping lemma for FSCA). *Let \mathcal{L} be a regular control language. Then there exist a pumping number $T \in \mathbb{Q}_+$ and a finite set $\mathcal{T} \subset \mathbb{Q}_+$ such that for all $u \in \mathcal{L}$ with $|u| \geq T$ there exists a representation of u as*

$$u = vwx \quad \text{with} \quad |vw| \leq T, |w| > 0$$

with

$$w \text{ is constant or } |w| \in \mathcal{T}$$

such that

$$vw^i x \in \mathcal{L} \quad \forall i \in \mathbb{N}_0.$$

Proof. Let \mathcal{L} be recognized by an FSCA $M = (Q, \delta, \Sigma, q_0, F)$ and define $T := \sum_{q \in Q} \max_{f \in P_q} |f|$. Moreover, let \mathcal{T} consist of all lengths of elementary cycles in the (graph corresponding to the) automaton M when ignoring loops on input from C_q for any $q \in Q$ (i.e., ignoring arcs in A_0 in the graph construction from the beginning of Section 4).

Now consider any $u \in \mathcal{L}$ with $|u| \geq T$ and choose an accepting sequence $u = f_1 \cdot \dots \cdot f_k$ for u as in Definition 3.2. If $f_i \in C_{r_{i-1}}$ for some $i \in \{1, \dots, k\}$ with $\ell := |f_1| + \dots + |f_{i-1}| < T$, we choose $v = f_1 \cdot \dots \cdot f_{i-1}$ and w as the prefix of f_i of length $\min\{|f_i|, T - \ell\}$. Then w is constant with length $|w| > 0$ and $|vw| \leq T$. By extending the constant subcontrol f_i we immediately obtain $vw^i x \in \mathcal{L}$ for all $i \in \mathbb{N}_0$.

We may thus assume that $|f_1| + \dots + |f_j| \geq T$ for some $j \in \{1, \dots, k\}$ such that $f_m \in P_{r_{m-1}}$ for all $m \in \{1, \dots, j\}$. Then there exist $i_1, i_2 \in \{0, \dots, j\}$, $i_1 \neq i_2$, such that $r_{i_1} = r_{i_2}$, i.e., such that M is in the same state q before processing f_{i_1+1} and after processing f_{i_2} , and such that $|f_1| + \dots + |f_{i_2}| \leq T$. We then define $v = f_1, \dots, f_{i_1}$ and $w = f_{i_1+1}, \dots, f_{i_2}$, which obviously satisfies $|vw| \leq T$ as well as $|w| > 0$. Moreover, we have $|w| \in \mathcal{T}$ by construction and it follows from the choice of w that it can be omitted or repeated an arbitrary number of times, since the automaton may always return to state q after reading w again. \square

We illustrate the use of Lemma 5.2 for showing non-representability by an FSCA by the following example.

Example 5.3. Consider the following language of binary controls:

$$\begin{aligned} \mathcal{L} := \{u \in \{0, 1\}^* \mid & \text{blocks of 0-controls have length 1,} \\ & \text{blocks of 1-controls have length at least 1,} \\ & \text{and the lengths of blocks of 1-controls are monotonically increasing}\} \end{aligned}$$

We show that representing \mathcal{L} is not possible within our framework, by using Lemma 5.2. Let us assume there is an FSCA that recognizes \mathcal{L} and let T and \mathcal{T} according to Lemma 5.2 be given.

Let $\lambda > \max(\mathcal{T} \cup \{1\})$. We consider a control $u \in \{0,1\}^*$ of length $|u| > T + \lambda + 1$ that alternates between 0-blocks of length 1 and 1-blocks of length λ , starting with a 1-block. Clearly, we have $u \in \mathcal{L}$, so let $u = vwx$ be a representation of u according to Lemma 5.2. In particular, w is constant or $|w| \in \mathcal{T}$.

If w is constantly 0, we obviously have $vw^i x \notin \mathcal{L}$ for all $i \geq 2$. In case w is constantly 1, $vw^i x$ can have an arbitrarily long 1-block as i increases. Since this 1-block cannot be the last one in u due to $|u| > T + \lambda + 1$ and $|vw| \leq T$, we obtain a contradiction to the monotonicity condition in \mathcal{L} .

If $|w| \in \mathcal{T}$, it follows that $|w| < \lambda$. We have already covered the case in which w is constant above. The remaining possibilities are that w either starts with a 0-block, ends with a 0-block, or has a 0-block in the middle. In any case, the total length of 1-blocks in w is less than λ , and therefore ww contains a 1-block of length $< \lambda$ somewhere in the middle. Since this 1-block is not the first 1-block in $vw^i x$ for $i \geq 2$, as u starts with a 1-block, we have a contradiction to the monotonicity condition in \mathcal{L} , which completes the proof.

Note that in Lemma 5.2 we can replace $|vw| \leq T$ by $|wx| \leq T$ since the proof is completely symmetric in this regard. Using this, we can now give a minimum time horizon we have to look at in order to decide whether any given u is a prefix of a control in a regular control language. The following lemma specifies and proves the respective claim made in Remark 4.10.

Lemma 5.4. *Let \mathcal{L} be a regular control language over Σ and u be a prefix of a control in \mathcal{L} . Then there exists $v \in \Sigma^*$ with $|v| \leq T$ such that $uv \in \mathcal{L}$, where T is a pumping number for \mathcal{L} according to Lemma 5.2.*

Proof. Let \mathcal{L} be a regular control language with pumping number T satisfying Lemma 5.2. Moreover, let u be a prefix of a control in \mathcal{L} , i.e., there exists $z \in \Sigma^*$ with $uz \in \mathcal{L}$. Choose an accepting sequence $uz = f_1 \cdot \dots \cdot f_k$ for uz as in Definition 3.2. Then there is an index $j \in \{1, \dots, k\}$ and a decomposition $f_j = f_{j_1} f_{j_2}$ of f_j into subcontrols $f_{j_1}, f_{j_2} \in \Sigma^*$ such that

$$u = f_1 \cdot \dots \cdot f_{j-1} f_{j_1} \quad \text{and} \quad z = f_{j_2} f_{j+1} \cdot \dots \cdot f_k.$$

If any of $f_{j_2}, f_{j+1}, \dots, f_k$ is in C_q for the respective state q , we may as well omit that subcontrol from z and proceed with the resulting z' . Hence, we can assume without loss of generality that f_i is a pattern in $P_{r_{i-1}}$ for all $i \in \{j, \dots, k\}$. Additionally, let z be the minimum-length control in Σ^* for which $uz \in \mathcal{L}$. Such a minimum exists due to the above and since the overall number of patterns is finite.

For the sake of contradiction, assume $|z| > T$. We invoke Lemma 5.2 (and the modification explained above) for obtaining a representation $uz = vwx$ with $|w| > 0$ and $|wx| \leq T$, such that $vw^i x \in \mathcal{L}$ for all $i \in \mathbb{N}_0$. In particular, we have $vx \in \mathcal{L}$. Due to $|z| > T$ and $|wx| \leq T$, we know that u is still a prefix of the shortened control vx , contradicting the minimality of z . \square

Recall that an easy-to-compute estimate for T can be obtained based on an FSCA for \mathcal{L} , following the proof of Lemma 5.2. Together with Theorem 4.7, we thus obtain a constructive proof for the existence of an extended formulation for the set of all prefixes of controls in \mathcal{L} . E.g., for the min-up/min-down constraints discussed in Example 4.1, we can use $T = L + \ell$.

For the special case of simple FSCA, we can strengthen the final condition in Lemma 5.2 to a statement similar to that in Lemma 5.1:

Lemma 5.5 (A pumping lemma for simple FSCA). *Let \mathcal{L} be a control language that can be recognized by a simple finite-state control automaton. Then there exist a pumping number $T \in \mathbb{Q}_+$ and a finite set $\mathcal{T} \subset \mathbb{Q}_+$ such that for all $u \in \mathcal{L}$ with $|u| \geq T$ there exists a representation of u as*

$$u = vwx \quad \text{with} \quad |vw| \leq T, |w| > 0, \text{ and } |w| \in \mathcal{T}$$

such that for all suffixes $z \in \Sigma^*$ and for all $i \in \mathbb{N}_0$ it holds

$$vwz \in \mathcal{L} \iff vw^i z \in \mathcal{L}.$$

Proof. Let \mathcal{L} be a control language recognized by a simple finite-state control automaton M . Again, we will use the argument that if $|u|$ is sufficiently large, any accepting sequence of M must repeat states before time T . However, for the strengthened version of the last condition we need to show that the set of situations M can possibly be in (including states as well as positions in a pattern that is currently processed) has to repeat. This would follow right away if M was deterministic. And indeed, the idea of the proof is to show that there is a deterministic classical FSA that accepts the same language as M . We split the proof into the following steps:

1. Construct a non-deterministic classical FSA $M' = (Q', \delta', \Sigma', q'_0, F')$ that accepts the same language as M .
2. Construct a deterministic FSA $M'' = (Q'', \delta'', \Sigma'', q''_0, F'')$ that accepts the same language as M' and M .
3. Prove the statement by applying the argumentation from the proof of Lemma 5.2 to M'' .

For the first step, construct a classical FSA $M' = (Q', \delta', \Sigma', q'_0, F')$ from M as follows: Since P_q is a finite set for all $q \in Q$, and all pattern lengths are rational, there exists a number $\gamma \in \mathbb{Q}$ such that $|f|/\gamma \in \mathbb{N}$ for all patterns $f \in P_q$ and $q \in Q$. We split up all patterns $f \in P_q$ into fragments of length γ and define Σ' to be the set of such fragments. Let Q' be composed of Q together with additional intermediate states $q_1^f, \dots, q_{k(f)-1}^f$ at the splitting points of each pattern $f \in P_q$ for some q , where $k(f) := |f|/\gamma$ denotes the number of segments f is split into. Transitions for these intermediate states are defined to allow processing f as in M . More precisely, if $f = a_1 \dots a_{k(f)}$ for $a_1, \dots, a_{k(f)} \in \Sigma'$, we add q_1^f to $\delta'(q, a_1)$ and define

$$\delta'(q_i^f, a_{i+1}) = \{q_{i+1}^f\} \text{ for all } i \in \{1, \dots, k(f) - 2\}, \quad \delta'(q_{k(f)-1}^f, a_{k(f)}) = \delta(q, f).$$

Finally, $q'_0 = q_0$ and $F' = F$ remain unchanged. By construction, Q' and Σ' are finite and M' is in fact a classical FSA that accepts a sequence of patterns if and only if M would do so (when we identify the sequence of symbols in Σ' with their corresponding controls).

In the second step, our goal is a deterministic FSA $M'' = (Q'', \delta'', \Sigma'', q''_0, F'')$ that accepts the same language as M' and M . One could obtain such an FSA by invoking the classical equivalence

result for deterministic and non-deterministic FSA. We give the construction here explicitly for the sake of completeness. The idea is to use subsets of Q' as states and have the deterministic automaton be in state $\{q_1, \dots, q_m\}$ on input u whenever q_1, \dots, q_m are exactly the states the non-deterministic automaton can be in after processing the same input. We thus define

$$Q'' := 2^{Q'}, \Sigma'' = \Sigma', q''_0 = \{q'_0\}, F'' = \{q'' \in 2^{Q'} \mid F' \cap q'' \neq \emptyset\}.$$

Moreover, we define δ'' by

$$\delta''(q'', a) = \bigcup_{q' \in q''} \delta'(q', a)$$

for $(q'', a) \in Q'' \times \Sigma''$.

For the third step, based on M'' we can now give a value for T that is sufficiently large, namely, we define $T := \gamma \cdot |Q''|$. Indeed, let $u \in \mathcal{L}$ be given. As in the proof of Lemma 5.2, we can choose v , w and x with $u = vwx$ and $|vw| \leq T$ such that M'' is in state q'' after processing v as well as after processing vw in the (unique) accepting sequence for u by M'' (where M'' considers u as a sequence of subcontrols of length γ , and rejects u if any such subcontrols are not in Σ'').

Pick v and w such that transitions in M'' follow an elementary cycle in the graph defined by (Q'', δ'') . Since the accepting sequence for u only uses patterns in P_q , $|w|$ is a multiple of γ . Moreover, there are only finitely many possible values for $|w|$ to be included in \mathcal{T} in order to guarantee that the condition $|w| \in \mathcal{T}$ is satisfied.

For the final condition, we make use of the fact that M'' is deterministic, deducing that M'' must be in state q'' after processing vw^i for any $i \in \mathbb{N}_0$. Hence, the set of suffixes z that lead to acceptance by M'' must be the same for all $i \in \mathbb{N}_0$. Since M'' and M recognize the same language, this concludes the proof. \square

We can use Lemma 5.5 for showing non-representability of languages in which no continuous part of any control can be “pumped”. As an example, we consider the set of binary controls satisfying max-up/max-down constraints, i.e.,

$$\mathcal{L}_{\text{MAXUPDOWN}(\bar{L}, \bar{\ell})} := \{u \in \{0, 1\}^* \mid \text{blocks of 1-controls have length at least } \varepsilon \text{ and at most } \bar{L}, \\ \text{blocks of 0-controls have length at least } \varepsilon \text{ and at most } \bar{\ell}\},$$

where $\bar{L}, \bar{\ell} \in \mathbb{Q}_+$ and $\varepsilon \in \mathbb{Q} \cap (0, \min\{\bar{L}, \bar{\ell}\})$. We will show that representing $\mathcal{L}_{\text{MAXUPDOWN}(\bar{L}, \bar{\ell})}$ is not possible with an FSCA.

Proposition 5.6. *There is no FSCA that recognizes $\mathcal{L}_{\text{MAXUPDOWN}(\bar{L}, \bar{\ell})}$.*

Proof. Suppose the language $\mathcal{L} := \mathcal{L}_{\text{MAXUPDOWN}(\bar{L}, \bar{\ell})}$ can be recognized by an FSCA. First, note that for any $u \in \{0, 1\}^*$, we clearly cannot “pump” any constant control w with $|w| > 0$ due to the max-up constraints. We may thus assume that the FSCA accepting \mathcal{L} is simple.

We will complete the proof by showing that \mathcal{L} does not satisfy the properties from Lemma 5.5. To this end, assume the contrary and let T and \mathcal{T} as in Lemma 5.5 be given. We consider a control $u \in \mathcal{L}$ with $|u| \geq T$ that alternates between 0-blocks of length $\bar{\ell}$ and 1-blocks of length $\bar{L} - \beta$,

starting with 0. We choose $\beta \in (0, \bar{L} - \varepsilon)$ in such a way that $k \cdot (\bar{\ell} + \bar{L} - \beta) \notin \mathcal{T}$ for all $k \in \mathbb{N}$, which is possible since \mathcal{T} is finite and in a sufficiently small interval there are only finitely many values for β that do not work. In order for the equivalence

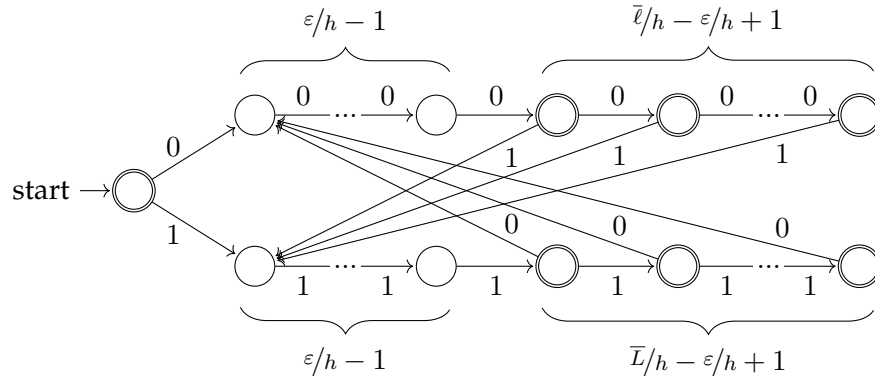
$$vwz \in \mathcal{L} \iff vw^i z \in \mathcal{L}$$

to hold, v and vw have to end with a block of the same length and alphabet symbol (since we can choose $i = 0$). Otherwise, we can find a constant control z that violates the max-up/down constraints for vwz but not vz or vice versa. As a consequence, $|w|$ must be a multiple of $\bar{\ell} + \bar{L} - \beta$, which leads to a contradiction to $k \cdot (\bar{\ell} + \bar{L} - \beta) \notin \mathcal{T}$ for all $k \in \mathbb{N}$. \square

Proposition 5.6 might come as a surprise since the language $\mathcal{L}_{\text{MAXUPDOWN}(\bar{L}, \bar{\ell})}$ is regular for any discretization. Indeed, when discretizing the time horizon into cells of uniform length $h \in \mathbb{Q}_+$, where we assume that $\bar{L}/h, \bar{\ell}/h, \varepsilon/h \in \mathbb{N}$, we obtain languages of the form

$$\begin{aligned} \mathcal{L}_{\text{MAXUPDOWN}(\bar{L}, \bar{\ell})}^h \\ := \bigcup_{n \in \mathbb{N}_0} \{z \in \{0, 1\}^n \mid & \text{blocks of 1-controls have length at least } \varepsilon/h \text{ and at most } \bar{L}/h, \\ & \text{blocks of 0-controls have length at least } \varepsilon/h \text{ and at most } \bar{\ell}/h\}. \end{aligned}$$

More precisely, the elements of the language $\mathcal{L}_{\text{MAXUPDOWN}(\bar{L}, \bar{\ell})}^h$ are in one-to-one correspondence with those controls of $\mathcal{L}_{\text{MAXUPDOWN}(\bar{L}, \bar{\ell})}$ that are constant on each cell $(ih, (i+1)h)$, $i \in \mathbb{N}_0$. Now it is easy to verify that the discretized language $\mathcal{L}_{\text{MAXUPDOWN}(\bar{L}, \bar{\ell})}^h$ can be recognized by the following deterministic FSA with $\bar{L}/h + \bar{\ell}/h + 1$ states:



This automaton uses one state for each integer in $\{1, \dots, \bar{\ell}/h\}$ and $\{1, \dots, \bar{L}/h\}$ to count the number of zeros and ones, respectively, of the current sequence. Since in the control setting the amount of past up- or downtime can not be quantified in discrete amounts, this construction cannot be directly generalized. When we increase the accuracy of the discretization by reducing the step length h , representing the discretization by the above construction will require more and more states – in the order of $1/h$ and thus approaching infinity in the limit.

This example shows that there is a difference between the continuous and discrete versions of a language with respect to whether they can be recognized by our continuous and the classical discrete versions of finite automata, respectively. Note the role of \mathcal{T} in Lemma 5.2 and Lemma 5.5 in capturing this difference. When closely comparing the proofs of these lemmata and the discrete construction above in the limit, the condition on \mathcal{T} essentially prevents that the need for an infinite number of states is shifted to an infinite number of transition patterns for evading a non-representability proof.

Finally, we emphasize that Proposition 5.6 does *not* exclude the existence of an extended formulation for the controls in $\mathcal{L}_{\text{MAXUPDOWN}(\bar{L}, \bar{\ell})}$ of a given length T . It just shows that the construction suggested in Section 4 via finite-state control automata is not applicable to $\mathcal{L}_{\text{MAXUPDOWN}(\bar{L}, \bar{\ell})}$.

6 Conclusion

While research on the closed convex hull of infinite-dimensional feasible sets of optimal control problems with combinatorial constraints has just started recently, this paper gives a construction mechanism for compact extended formulations for a large class of such constraints. For many applications, this opens up the possibility of working with manageable convex-hull formulations of complicating substructures in the space of controls rather than thinking about strong mixed-integer programming formulations only after discretization. While we used a sequence of equidistant discretizations for showing our main result, the infinite-dimensional convex-hull descriptions in principle yield convex-hull descriptions for any discretization grid. For characterizing combinatorial constraints that are accessible for our approach, we used a type of finite-state automaton that accepts continuous controls. The resulting finite-state control automata are new to the best of our knowledge. They are designed to capture the modular structure that is needed for our construction to work, but might still be of independent interest – either in their current form or with modifications to suit other applications. For helping to distinguish representable from non-representable languages, we contribute results in the style of a pumping lemma. As an example application, we considered a language that cannot be recognized by an FSCA although its discrete version can be recognized by a classical FSA for any discretization. So, as in many other cases, there is more to the infinite-dimensional setting than what can be seen from discretizations.

Future research may go further in this direction, possibly discovering a more restrictive pumping lemma that yields an equivalent characterization for regular control languages. Moreover, from a theoretical standpoint, it is an interesting open question whether assuming determinism reduces the representable class of languages. Going in a different direction, one may aim to further generalize the construction from Theorem 4.7. For instance, allowing instantaneous switches (i.e., state switches without processing any input) under certain restrictions would make the concept even more versatile for modeling. However, we do not see a direct way to incorporate such transitions in our proof. Finally, we hope that computational studies will confirm the usefulness of our results in applications.

Acknowledgments

This research was initiated through discussions at the workshop “Mixed-integer Nonlinear Optimization: a hatchery for modern mathematics” held at Mathematisches Forschungsinstitut Oberwolfach (MFO). The first author was partially supported by Deutsche Forschungsgemeinschaft under grant no. BU 2313/7-1.

References

- [ABM14] Hedy Attouch, Giuseppe Buttazzo, and Gérard Michaille. *Variational Analysis in Sobolev and BV Spaces*. Society for Industrial and Applied Mathematics, 2014.
- [AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
- [AFP00] Luigi Ambrosio, Nicola Fusco, and Diego Pallara. *Functions of Bounded Variation and Free Discontinuity Problems*. Oxford Mathematical Monographs, 2000.
- [BGM24a] Christoph Buchheim, Alexandra Grütering, and Christian Meyer. Parabolic optimal control problems with combinatorial switching constraints – Part III: Branch-and-bound algorithm. Technical Report arXiv:2401.10018 [math.OC], 2024.
- [BGM24b] Christoph Buchheim, Alexandra Grütering, and Christian Meyer. Parabolic optimal control problems with combinatorial switching constraints – Part I: Convex relaxations. *SIAM Journal on Optimization*, 34(2):1187–1205, 2024.
- [Bra05] Michael S. Branicky. Introduction to hybrid systems. In *Handbook of networked and embedded control systems*, pages 91–116. Springer, 2005.
- [Buc24] Christoph Buchheim. Compact extended formulations for binary optimal control problems. Technical Report arXiv:2401.03942 [math.OC], 2024.
- [CCZ13] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Extended formulations in combinatorial optimization. *Annals of Operations Research*, 204:97–143, 2013.
- [CK05] Robert D. Carr and Goran Konjevod. *Polyhedral Combinatorics*, pages 2–1–2–46. Springer New York, New York, NY, 2005.
- [FF62] Lester R. Ford and Delbert R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, 1962.
- [FP15] Samuel Fiorini and Kanstantsin Pashkovich. Uncapacitated flow-based extended formulations. *Mathematical Programming*, 153(1):117–131, 2015.
- [GHPS21] Simone Göttlich, Falk M. Hante, Andreas Potschka, and Lars Schewe. Penalty alternating direction methods for mixed-integer optimal control with combinatorial constraints. *Mathematical Programming*, 188(2):599–619, 2021.

- [Hen96] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292, 1996.
- [Hey17] Ali Heydari. Optimal switching with minimum dwell time constraint. *Journal of the Franklin Institute*, 354(11):4498–4518, 2017.
- [HLM⁺17] Falk M. Hante, Günter Leugering, Alexander Martin, Lars Schewe, and Martin Schmidt. *Challenges in Optimal Control Problems for Gas and Fluid Flow in Networks of Pipes and Canals: From Modeling to Industrial Applications*, pages 77–122. Springer Singapore, Singapore, 2017.
- [HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. Introduction to automata theory, languages, and computation. *ACM SIGACT News*, 32(1):60–65, 2001.
- [Jaf78] Jeffrey Jaffe. A necessary and sufficient pumping lemma for regular languages. *ACM SIGACT News*, 10(2):48–49, 1978.
- [LLM04] Jon Lee, Janny Leung, and Francois Margot. Min-up/min-down polytopes. *Discrete Optimization*, 117(1):77–85, 2004.
- [LMS⁺21] Do Duc Le, Maximilian Merkert, Stephan Sorgatz, Mirko Hahn, and Sebastian Sager. Autonomous traffic at intersections: An optimization-based analysis of possible time, energy, and CO2 savings. *Networks, Special Issue: The Future of City Logistics and Urban Mobility*, 79(3):338–363, 2021.
- [OWD16] Chong-Jin Ong, Zheming Wang, and Masood Dehghan. Model predictive control for switching systems with dwell-time restriction. *IEEE Transactions on Automatic Control*, 61(12):4189–4195, 2016.
- [RT05] Deepak Rajan and Samer Takriti. Minimum up/down polytopes of the unit commitment problem with start-up costs. Technical report, IBM Research, 2005.
- [RZSB21] Nicolò Robuschi, Clemens Zeile, Sebastian Sager, and Francesco Braghin. Multi-phase mixed-integer nonlinear optimal control of hybrid electric vehicles. *Automatica*, 123:109325, 2021.
- [SJK11] Sebastian Sager, Michael Jung, and Christian Kirches. Combinatorial integral approximation. *Mathematical Methods of Operations Research*, 73(3):363–380, 2011.
- [VDSS07] Arjan J. Van Der Schaft and Hans Schumacher. *An introduction to hybrid dynamical systems*, volume 251. springer, 2007.
- [ZA15] Feng Zhu and Panos J. Antsaklis. Optimal control of hybrid switched systems: A brief survey. *Discrete Event Dynamic Systems*, 25(3):345–364, 2015.
- [ZRS21] Clemens Zeile, Nicolò Robuschi, and Sebastian Sager. Mixed-integer optimal control under minimum dwell time constraints. *Mathematical Programming*, 188(2):653–694, 2021.