Linear Coupled Component Automata for MILP Modeling of Hybrid Systems

Ashish Agarwal[*] Ignacio E. Grossmann[†]

Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

July 18, 2008

Abstract

We first introduce a novel modeling framework, called linear coupled component automata (LCCA), to facilitate the modeling of discrete-continuous dynamical systems with piecewise constant derivatives. Second, we provide a procedure for transforming models in this framework to mixed-integer linear programming (MILP) constraints. Traditionally, such systems have been modeled directly with MILP constraints. We show with an example that our framework significantly simplifies model formulation and allows the complex MILP constraints to be produced systematically.

Key words: hybrid automata; modeling frameworks; optimization; discrete-continuous systems

1 Introduction

Many processes in the chemical industry involve mixed discrete-continuous dynamics, called hybrid dynamics. It is not yet clear what kind of mathematical statements best represent these systems, but frameworks based on hybrid automata are becoming the most widely used. Although these allow modeling hybrid systems, there are few algorithms for optimizing models declared in this form. On the other hand, there are well-developed algorithms for solving mixed-integer linear programs (MILPs), but, as we show with an example, posing an MILP model for a hybrid system can be extremely difficult.

We present a novel framework called linear coupled component automata (LCCA), so named because it supports the modeling of piecewise-linear dynamics in the continuous realm and fairly general discrete dynamics with automata. Several other frameworks already allow modeling such systems in theory, but our goal is to facilitate modeling in practice. LCCA allows expressing constraints in intuitive ways and minimizes the changes required when reconfigurations of the physical process are considered. Next, we describe a procedure for systematically transforming LCCA models to mixed-integer linear programming (MILP) constraints. Presently, MILP models are usually formulated directly, but we show with an example that it is significantly easier to pose the equivalent LCCA model.

One of the earliest efforts to model hybrid systems was by Barton and Pantelides (1994). They augmented the theory of differential algebraic equations (DAEs) with an index set, which served as the possible discrete modes of a system. Each index was associated with a different DAE. Their focus was on developing numerical procedures for handling the switching between DAEs. For instance, they consider how to accurately detect the time at which a switching condition is satisfied, called event detection. Although theoretically

[*]Current address: Department of Molecular Biophysics and Biochemistry, Yale University, New Haven, CT 06520, USA

[†]Corresponding author. Tel.: +1-412-268-2230; fax: +1-412-268-7139. *E-mail addresses:* ashish.agarwal@yale.edu (A. Agarwal), grossmann@cmu.edu (I.E. Grossmann).

as expressive as automata, their mechanism for expressing discrete dynamics is not very flexible. All discrete states must be represented within a single index set, and statements on the indices themselves are not supported. The mixed logical dynamical system defined in Bemporad and Morari (1999) also supports hybrid systems by allowing difference equations that include 0-1 binary variables to represent logical conditions. Westerweele et al. (1999) support hierarchical modelling, an important feature we have not yet considered.

The frameworks coming into most common use are based on a combination of discrete automata and differential equations. Automata are an elegant method for describing discrete dynamics, and differential equations are of course the standard for continuous dynamics. Their combination creates a system more expressive than either alone. Variations, usually in the generality of continuous dynamics allowed, lead to specific frameworks.

Alur and Dill (1994) introduced timed automata, which allow variables measuring elapsed time. Within each discrete mode, the differential equations are of the form $\mathrm{d}x/\mathrm{d}t = 1$, and values can be reset to 0 at event points. Cassez and Larsen (2000) consider an extension allowing the derivative of a variable to be 0 or 1, and call this a stopwatch automaton. Alur et al. (1995) define the linear hybrid system, which allows the rate of change to be any constant. These seemingly minor variations can significantly affect some kinds of analysis. For example, the reachability problem for linear hybrid systems is undecidable but can be solved for timed automata. Finally, these fall under the class of systems allowing general continuous dynamics, generically called hybrid automata. The linear hybrid system in Alur et al. (1995) is actually defined as a restriction of this more general system. See also Henzinger (1996) and Nicollin et al. (1991).

Optimization of hybrid systems has received little attention relative to control (Cury et al.; 1998; Tomlin et al.; 1998) and verification (Chutinan and Krogh; 2003) problems. The approaches that have been taken can be divided into those that employ mathematical programming (MP) and those that define algorithms directly on the hybrid systems model.

Asarin and Maler (1999) show how to design an optimal controller for timed automata, and Abdeddaim and Maler (2001) model job-shop scheduling problems with acyclic timed automata. These have been traditionally modeled in the MILP framework, but these works define optimization methods independent of MILP algorithms. As we do, they also make the case that hybrid automata models allow modeling these problems more naturally than MILP. In Abdeddaim and Maler (2002), they extend their work to preemptive job-shop scheduling, which require modeling with the richer stopwatch automata.

Algorithms developed directly on timed or stopwatch automata can be efficient because the structures of these special classes of systems can be exploited. On the other hand, extensions to the modeling class require new algorithms to be developed. A broader class of systems can be modeled in the MILP framework, and there is an extensive body of literature and commercial software for optimizing problems posed this way. Bixby (2002) reviews the impressive advances in this area, and Kallrath (2000) discusses the role of mathematical programming in the chemical process industry.

In this work, we take the second approach to optimizing hybrid systems, namely to transform the hybrid systems models into MILP models. Stursberg et al. (2002) have also taken this approach. They show how an optimal control problem on their hybrid automata (HA) formulation can be posed as a mathematical program. Our transformation procedure differs firstly because it operates on the alternative HA style framework we provide. Secondly, we focus on the mechanics of performing the transformation. This makes it more applicable to models as written in practice, not just canonical forms.

Lee et al. (2004) have also employed MP to optimize hybrid systems. Their focus is largely on numerical matters. For example, results, such as sensitivity analysis, on purely continuous systems are complicated by the addition of switching. They show how to compute sensitivities for hybrid systems, allowing efficient application of gradient based algorithms. In contrast, we are focused on the transformation as a symbolic procedure, as opposed to the numerical properties of the resulting formulation.

A somewhat similar distinction exists with the work of Heemels et al. (2001). They show that several hybrid systems modeling frameworks are equivalent. However, this does not provide a systematic procedure for translating models in one of those frameworks to another. Also, they consider only the discrete time case. We provide an exact reformulation for continuous time models (at the expense of restricting continuous dynamics to be piecewise linear). Torrisi et al. (2000) describe a software implementation for transforming hybrid systems models into mixed-integer constraints, which is also for discrete time models.

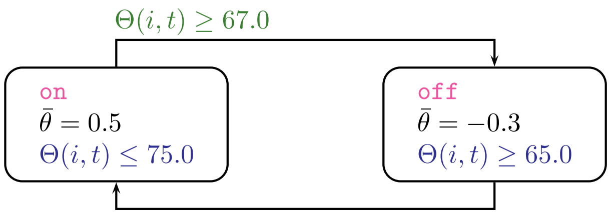
2 Framework for Modeling Hybrid Systems

The modeling framework we will introduce is called linear coupled component automata (LCCA). We demonstrate the main ideas with an example before providing the formal definition. Consider that we have the following conceptual description of a system:

> A thermostat can either be on or off. When it is on, temperature increases at a rate of 0.5 °F/min, and when it is off temperature decreases at rate 0.3 °F/min. The thermostat should not be on if the temperature exceeds 75.0 °F, and when turned on, the temperature should be allowed to rise to at least 67 °F. Finally, it should not be off if the temperature is below 65 °F.

Now, our goal is to produce a formal representation of this system. For optimization purposes, we would normally consider formulating MILP constraints directly. Instead, we provide the following description, which we will see later as being a formal LCCA model.

$$n = 10 \tag{1a}$$



(1b)

$$\frac{\mathrm{d}\Theta\left(i,t\right)}{\mathrm{d}t} = \bar{\theta}\left(Q\left(i\right)\right) \qquad \forall\left(i,t\right) \in \mathbb{T} \tag{1c}$$

The constraint $n = 10$ bounds the number of event switches to at most 10. This is required to bound the feasible space. Equation (1b) depicts a component automata, the main construct of our novel framework. It represents the thermostat, which has two modes `on` and `off`. When on, it sets the temperature's rate of change $\bar{\theta}$ to 0.5 and requires the temperature Θ to remain below 75.0. The guard on the transition allows switching to off only if the temperature is greater than 67.0. When off, $\bar{\theta}$ is set to -0.3, and so we see that $\bar{\theta}$ is a function of the mode the automaton is in, given by $Q(i)$. The differential equation describes the overall rate of change of the temperature, where the value on the right-hand-side will switch discretely. Figure 1 shows a feasible trajectory for the temperature and discrete modes of the thermostat.

Importantly, $\bar{\theta}$ is local to each automaton, each of which describes a single process. If another heater was also present, another component automaton would be used to represent it. It would provide its own contribution to the temperature's rate of change $\bar{\theta}'$, and the differential equation would be $\mathrm{d}\Theta\left(i,t\right)/\mathrm{d}t = \bar{\theta}\left(Q\left(i\right)\right) + \bar{\theta}'\left(Q'\left(i\right)\right)$. The LCCA framework enforces this separation of variables to promote modular modeling. Modifying, deleting, or adding a process is guaranteed to affect only a single automaton.

In the following sections, we provide formal definitions for the constructs used in this example. Firstly we define a timeline and then provide several constraint forms on continuous and discrete variables. Finally, these are used to define the full LCCA framework.

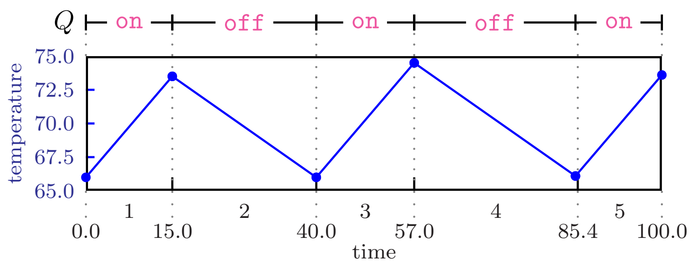


Figure 1: Feasible trajectory for thermostat example.

2.1 Hybrid Timeline

For continuous systems, the timeline is simply an interval of the real numbers, but a different model of time is needed for hybrid systems. Discrete dynamics occur instantaneously and the timeline must allow specification of two values at certain time points, called event points.

Let $\mathbb{N} = \{1, \ldots, n\}$ for some constant n. Lygeros et al. (1999) define a hybrid timeline[1], depicted in Figure 2, as an ordered sequence of intervals $\mathcal{T} = \{[t_i^s, t_i^e]\}_{i \in \mathbb{N}}$ such that

- $t_i^s \leq t_i^e$ for $i \in \mathbb{N}$, and
- $t_i^e = t_{i+1}^s$ for $i \in \mathbb{N} \setminus \{n\}$.

The interpretation is that all discrete variables are constant during each interval; only continuous variables evolve within intervals. Discrete variables change their values at the boundaries between intervals, the event points. Let $\Delta t_i = t_i^e - t_i^s$ denote the length of interval i.

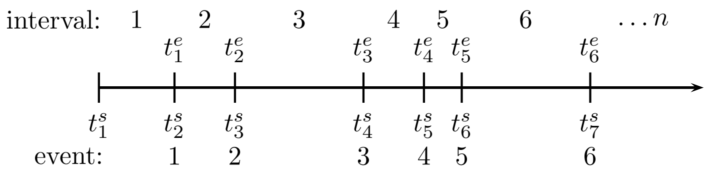


Figure 2: Hybrid timeline.

Despite this alternative timeline for hybrid systems, the definition of dynamic variables in the existing literature (e.g. Aubin et al.; 2002) has followed that of purely continuous systems, i.e. a dynamic variable is defined on the domain $\mathbb{R}$. This complicates the mathematics because a variable can take two values at the event points, requiring them to be treated as more general mappings than functions. However, a functional form can be maintained if the above timeline definition is slightly rearranged. We define the hybrid timeline

$$\mathbb{T} = \{(i, t) : t \in [t_i^s, t_i^e] \in \mathcal{T}\}\,.$$

With this definition, a single time point is a pair (i, t). We strictly maintain this interpretation; any reference to t by itself is considered incomplete.

There is a total order relation $\preceq$ on $\mathbb{T}$, as expected of a timeline. The time point (i, t) precedes or is equal to (i', t'), denoted $(i, t) \preceq (i', t')$, if and only if $i \leq i'$ and $t \leq t'$. If $(i, t) \npreceq (i', t')$, then it must be that $(i', t') \preceq (i, t)$. If both $(i, t) \preceq (i', t')$ and $(i', t') \preceq (i, t)$, then $(i, t) = (i', t')$.

The set $\mathbb{N} \setminus \{n\} = \{1, 2, \ldots, n-1\}$ can be interpreted as the set of event points. Our convention is that the i^{th} event point occurs at the *end* of interval i. Event point i coincides

[1]We have restricted their definition to the $n < \infty$ case.

with two time points: (i, t_i^e) and $\left(i+1, t_{i+1}^s\right)$. Only the integer component of time changes when an event occurs. The real component at both times is $t = t_i^e = t_{i+1}^s$.

Events are said to occur instantaneously, but this is only with respect to the real component of time. The integer component of time does progress. The concept of instantaneous has been captured by a finite time change in our definition. Precisely, an event occurs over zero units of real time and one unit of integer time. As a consequence, events do not occur at the initial $(1, t_1^s)$ and final (n, t_n^e) time points.

In summary, a hybrid timeline can be discrete or continuous and can either restrict event times to fixed points or not. Discretization will however not affect us because we consider only differential equations simple enough to be symbolically integrated.

2.2 Constraints

Constraints on real and discrete variables will be needed in the overall modeling framework. Here, we discuss the meaning of continuous and discrete dynamic variables and define some notation. The following font conventions will be used:

- plain font will represent a single variable, e.g. i, t, X, Q
- bold font will represent a set of variables, e.g. $\mathbf{X}$, $\mathbf{Q}$
- blackboard bold font will represent a space of values, e.g. $\mathbb{N}$, $\mathbb{R}$, $\mathbb{T}$, $\mathbb{Q}$
- calligraphic font will represent more complex mathematical constructs, e.g. $\mathcal{L}$.

A real valued dynamic variable is a function from the timeline to the reals, $X : \mathbb{T} \to \mathbb{R}$. Our modeling framework allows use of equations and inequalities: $=$, $\leq$, and $\geq$. Strict inequalities are not allowed because they are also not allowed in MILPs, to which we wish to convert our models.

Continuous variables change infinitesimally in an infinitesimal amount of time, but their values can also jump instantaneously at an event point. Consider an event occurring from time (i, t) to $(i+1, t)$. The instantaneous change can be defined by an algebraic equation, e.g. $X(i+1, t) = X(i, t) + 1$ would increment the value of X by 1 at event i.

In purely continuous systems, it is customary to refer to "a set of constraints". What is meant by this is a conjunction of constraints. For example,

$$\begin{aligned} &X(i+1, t) = X(i, t) + 1 \quad \wedge \\ &Y(i, t) = X(i, t) \end{aligned}$$

is usually considered two constraints and the conjunction symbol $\wedge$ is not shown. We will often call this a single constraint (which happens to be a conjunction of two other constraints).

Discrete variables come in various forms. MILP allows integer variables (Kallrath; 2000), and GDP includes Boolean logic (Raman and Grossmann; 1994). We consider variables taking values from a finite set, e.g. $\mathbb{Q} = \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$, which are often called set valued or finite domain variables. `A`, `B`, and `C` are finite domain constants, just as 1, 2, and 3 are integer constants.

Given a finite set $\mathbb{Q}$, we can now define a dynamic finite domain variable $Q : \mathbb{T} \to \mathbb{Q}$. However, by definition, discrete variables do not evolve within an interval. The value of $Q(i, t)$ depends only on the interval number i, making the t superfluous. It suffices to define dynamic discrete variables as functions on the set of intervals, $Q : \mathbb{N} \to \mathbb{Q}$. Equations can be used to restrict the values taken by Q over time, e.g. $Q(2) = \mathtt{B}$ sets Q in the second interval to the value `B`. Also, the values between intervals can be related, e.g. $Q(i) = Q(i+1)$ forces Q's value to remain unchanged from interval i to $i+1$.

Of course, multiple discrete variables might be needed. Superscripts are used to refer to different finite sets, e.g. $\mathbb{Q}^\alpha = \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$ and $\mathbb{Q}^\beta = \{\mathtt{D}, \mathtt{E}\}$. The corresponding dynamic variables are named with the same superscript. So Q^{α} is understood to be a function from $\mathbb{N} \to \mathbb{Q}^{\alpha}$. The equation $Q^{\alpha}(i) = \mathtt{D}$ would be erroneous.

Constraints on reals can only be connected by conjunction, but equations on finite domain variables can be connected with the logical operators of negation $\neg$, disjunction $\vee$, and conjunction $\wedge$. For example, Q^{α} taking the value B requires the use of some resource, and that same resource is required if Q^{β} is equal to D. The constraint

$$\neg\left(\left(Q^{\alpha}(i) = \mathtt{B}\right) \wedge \left(Q^{\beta}(i) = \mathtt{D}\right)\right)$$

assures that these values are not taken at the same time. Another example is

$$\left(Q^{\alpha}(i) = \mathtt{B}\right) \vee \left(Q^{\alpha}(i) = \mathtt{C}\right)$$

which requires Q^{α} to take either the value B or C in interval i.

The modeling framework defined in the next section allows constraints in the above forms but requires various restrictions on which variables can be used and the time points at which the variables can be evaluated. Some notation will allow stating these restrictions compactly. Let $\mathcal{L}$ denote the set of all constraints in the forms discussed above, and let $\mathbf{X}$ be a set of dynamic real variables and $\mathbf{Q}$ a set of dynamic finite domain variables. Also, let $\mathbf{X}(i,t)$ and $\mathbf{Q}(i)$ mean each of the variables in the sets are evaluated at time point (i,t). Finally, $\mathcal{L}(\mathbf{X}(i,t))$ is the set of constraints involving only variables in $\mathbf{X}$ evaluated at time (i,t), and similarly for $\mathcal{L}(\mathbf{Q}(i))$.

2.3 Linear Coupled Component Automata

With preliminary concepts in place, we can now define the LCCA framework. Conceptually, the system consists of a set of component automata, so called because they are components of an overall system. Each automaton has a set of discrete modes associated with it, and a dynamic finite domain variable specifies the mode the automaton is in over time. There are overall real valued system variables, and each automaton specifies its contribution to how these variables evolve. Automata cannot directly specify the rate of change of a system variable, but the values of system variables can prohibit or require discrete transitions in the automata. These are natural restrictions that support modular modeling, as we will show with an example.

A dynamical system consists of a timeline, variables, and a specification of how these variables evolve. Precisely, we define a linear coupled component automata model as the 5-tuple

$$(n, G_t, \mathbf{X}, Aut, G_V) \qquad \text{(LCCA)}$$

where n is an integer specifying the number timeline intervals, G_t is a constraint on the timeline variables, $\mathbf{X}$ is a set of dynamic real variables, Aut is a set of component automata, and G_V is a constraint coupling the component automata.

The integer n specifies the number of intervals in the timeline. Given n we take the timeline to be $\mathcal{T} = \{[t_i^s, t_i^e]\}_{i\in\mathbb{N}}$, where $\mathbb{N} = \{1,\ldots,n\}$. We can also refer to the timeline in the form $\mathbb{T}$, which is defined in terms of $\mathcal{T}$ in section 2.1. Let $\mathbf{t} = \cup_{i\in\mathbb{N}} \{t_i^s, t_i^e\}$ be the set of timeline variables.

G_t is an element of $\mathcal{L}(\mathbf{t})$, allowing constraints on the timeline variables. For example, interval lengths could be fixed to a constant by requiring $t_i^e - t_i^s = k$ for all i. For optimization purposes, an upper bound on the time horizon will be required. This can be given by the constraint $t_n^e \leq T^{\max}$.

$\mathbf{X}$ is the desired set of continuous system variables. These variables can be used in the subsequent constructs Aut and G_V.

Each automaton $a \in Aut$ is itself a 5-tuple of the form

$$(\mathbb{Q}, \bar{\mathbf{x}}, \hat{\mathbf{x}}, F, Arc) \qquad \text{(component automaton)}$$

where:

- $\mathbb{Q}$ gives the discrete modes of the automaton, and we let $Q : \mathbb{N} \to \mathbb{Q}$ give the discrete mode in each interval.

- $\bar{\mathbf{x}}$ is a set of given rates, the values of which can vary by mode. Each $\bar{x} \in \bar{\mathbf{x}}$ is a function from $\mathbb{Q} \to \mathbb{R}$. These will be used in differential equations governing the continuous evolution of $\mathbf{X}$.

- $\hat{\mathbf{x}}$ is a set of jump variables, whose values are dependent on the event number. Each $\hat{x} \in \hat{\mathbf{x}}$ is a function from $\mathbb{N} \backslash \{n\} \to \mathbb{R}$. These will be used in algebraic equations governing the discrete evolution of $\mathbf{X}$.

- $F : \mathbb{Q} \to \mathcal{L}(\mathbf{X}(i,t))$ is an invariant, a constraint that must be satisfied during continuous evolution. The constraint can depend on the mode of the automaton. Formally, the condition is enforced as $F(Q(i))$ for all $(i,t) \in \mathbb{T}$. At time (i,t), the automaton's mode is $Q(i)$. Thus, $F(Q(i))$ gives the desired constraint for the active mode.

- $Arc \subseteq \mathbb{Q} \times \mathbb{Q}$ is a set of transitions. Associated with each transition $(q, q') \in Arc$ is a *guard* $\gamma_{(q,q')} \in \mathcal{L}(\mathbf{X}(i,t))$ and a *reset* $\rho_{(q,q')} \in \mathcal{L}(\mathbf{X}(i,t) \cup \hat{\mathbf{x}}(i))$. The transition can be made at event point i only if the guard holds at time (i,t), and, if the transition is made, the reset is also enforced. There can exist at most one transition from any q to q'.

  It is required that a dummy transition (q,q) exists from every $q \in \mathbb{Q}$ to itself. On this transition, the guard $\gamma_{(q,q)}$ is set to a trivially satisfied constraint such as $1 = 1$, and the reset $\rho_{(q,q)}$ is $\wedge_{\hat{x} \in \hat{\mathbf{x}}} (\hat{x}(i) = 0)$. Dummy transitions are required because the mathematical model forces all automata to transition at an event point, but this is a superficial physical requirement. The dummy transition enables an automaton to transition in a manner that has no effect.

In the above, an automaton has been generically denoted by the tuple $(\mathbb{Q}, \bar{\mathbf{x}}, \hat{\mathbf{x}}, F, Arc)$. Superscripts are used to refer to multiple automata, e.g. automaton a consists of the elements $(\mathbb{Q}^a, \bar{\mathbf{x}}^a, \hat{\mathbf{x}}^a, F^a, Arc^a)$. The discrete variable associated with this automaton is Q^a. Let $\mathbf{Q} = \cup_{a \in Aut} Q^a$ be the set of discrete variables in an LCCA model, just as $\mathbf{X}$ is the set of continuous variables.

Variables $\bar{\mathbf{x}}^\alpha$ and $\hat{\mathbf{x}}^\alpha$ are local to automaton α. Another automaton β cannot make any reference to these variables, only to its own $\bar{\mathbf{x}}^\beta$ and $\hat{\mathbf{x}}^\beta$. Separating the variable name space requires modeling to be done in a modular fashion. In contrast to existing hybrid automata modeling frameworks, differential equations are not associated with the discrete modes of our component automata. They specify only rates $\bar{\mathbf{x}}$. Similarly, resets do not directly dictate discrete changes to continuous variables; they dictate only the values of $\hat{\mathbf{x}}$. Component automata are so named because they do not represent a dynamical system on their own; they are used as components of the overall LCCA system.

The last element of the modeling framework is G_V. This is where the dynamical equations are specified. All local and system variables can be used. G_V can include several types of constraints:

- differential equations of the form

$$\frac{\mathrm{d}X(i,t)}{\mathrm{d}t} = \sum_{a \in Aut} \bar{x}^a(Q^a(i)) + k \qquad \forall (i,t) \in \mathbb{T} \tag{2}$$

  where k is some constant. Thus, the overall rate of change of X is potentially dependent on the active modes of all automata,

- discontinuity equations of the form

$$X\left(i+1, t_{i+1}^s\right) = X\left(i, t_i^e\right) + \sum_{a \in Aut} \hat{x}^a(i) + k \qquad \forall i \in \mathbb{N} \backslash \{n\} \tag{3}$$

where k is some constant. Discontinuity equations are the discrete analog of differential equations. A flow rate $\bar{x}$ specifies an infinitesimal change in an infinitesimal amount of time. Similarly, a jump variable $\hat{x}$ specifies a finite change over an instantaneous step in time,

- finite domain constraints in $\mathcal{L}(\mathbf{Q}(i))$ or real constraints in $\mathcal{L}(\mathbf{X}(i,t))$. These are constraints that must always hold, independent of the active modes of automata, and
- initial conditions from $\mathcal{L}(\mathbf{Q}(i))$ or $\mathcal{L}(\mathbf{X}(i,t))$, where (i,t) and i are specific time values. For example, $X(1, t_1^s) = 0.0$.

Mathematical notation has been used thus far, but automata are customarily depicted graphically, e.g. see equation (1b). A box is drawn for each discrete mode, and the name of the mode is written within the box. This is followed by the values of variables $\bar{\mathbf{x}}$ and invariant F for that mode. Next, for each transition $(q, q') \in Arc$, an arrow is drawn from box q to q'. The guard $\gamma_{(q,q')}$ is written near the tail of the arrow and reset $\rho_{(q,q')}$ near the head. Dummy transitions are not shown because their definition is fixed. LCCA models are thus provided with a mixture of graphical and textual declarations.

3 Optimizing Hybrid Systems

Dynamic systems represented in the LCCA framework can have multiple feasible trajectories. Optimization would allow the system to be operated along its most profitable path. In this section, we first state what an optimization problem on an LCCA model is, and then define a procedure for transforming this problem to a mixed-integer linear program (MILP).

The transformation will serve two purposes. It is a systematic procedure for generating an MILP, which connects our modeling framework to existing algorithms. But also, it demonstrates how the framework we propose facilitates modeling. As we proceed through the transformation from LCCA to MILP, it will be clear that corresponding declarations become more cumbersome.

3.1 Optimization Problems

In an LCCA, there is freedom to choose the length of time spent in each discrete mode, various discrete transitions are possible, and values of jump variables are flexible. These choices lead to various trajectories, and we let Ξ denote the set of all feasible trajectories for a given model. See Agarwal (2006, Section 2.3) for a formal definition.

An objective Ω is a metric on the space Ξ. An optimization problem seeks the trajectory $\xi \in \Xi$ such that Ω is minimized (or maximized) and is denoted

$$\min_{\xi \in \Xi} \Omega.$$

The objective function can involve any of the real valued variables in the LCCA model, which are the timeline variables $\mathbf{t}$ and the dynamic continuous variables $\mathbf{X}$. For example, if X represents cost, one may wish to minimize its final value. The objective is

$$\Omega = X(n, t_n^e). \tag{4}$$

Or often one is concerned with a time average performance criterion,

$$\begin{aligned}\Omega &= \frac{1}{(t_n^e - t_1^s)} \int_{t_1^s}^{t_n^e} X(i,t)\mathrm{d}t \\ &= \frac{1}{2(t_n^e - t_1^s)} \sum_{i \in \mathbb{N}} (X(t_i^s) + X(t_i^e)) \Delta t_i \end{aligned} \tag{5}$$

where the integral calculates trapezoidal areas because continuous variables evolve piecewise linearly. Finally, in makespan minimization problems, the value of time is itself the objective,

$$\Omega = t_n^e. \tag{6}$$

Such a problem would normally define the notion of a completed job, e.g. producing a certain amount of material. The problem then is to complete the job as fast as possible.

Our general definition of a hybrid timeline allows systems to evolve indefinitely. For optimization purposes, it is necessary to bound the timeline with respect to both the integer and real components of time. The final time t_n^e, called the time horizon, must be bounded with a constraint of the form $t_n^e \leq T^{\max}$ in G_t. The definition of LCCA already requires specification of the number of intervals n, but this is less restrictive than it appears. If all transitions into a certain interval are dummy transitions, there has been effectively no change in the physical behavior of the system. The value of n is thus only an upper bound on the *effective* number of intervals. Nonetheless selecting n can be difficult. A larger value increases the likelihood of finding better solutions but a smaller value makes the problem more tractable.

The optimization problem we wish to solve is $\min_{\xi \in \Xi} \Omega$. An MILP is of the form $\min_{x \in P} \Omega'$, where x is a vector of real and integer variables and P is a region defined by a system of mixed-integer linear inequalities. Our goal is to convert the former into the latter. The objective function Ω must be converted into a form Ω' allowed in MILP, and an LCCA model must be converted into linear inequalities.

3.2 Constraint Conversions

Constraints in the LCCA framework contain several features not allowed in MILP constraints: they are quantified over infinite sets, they employ variable arguments, and finite domain variables are used. In this section, methods for eliminating these features without altering the meaning of the constraints are presented. These will be used in the next section, which discusses a transformation for the full modeling framework.

First, we state a theorem needed a few times in the subsequent discussion.

Theorem 1. *Let $g(i)$ and $f(i)$ be two constraints involving an index i ranging over a set $\mathbb{S}$. Assume $g(i)$ holds for exactly one i, i.e. $\veebar_{i \in \mathbb{S}} g(i)$ is true. Then, the conjunction of implications*

$$\bigwedge_{i \in \mathbb{S}} [g(i) \Rightarrow f(i)] \tag{conj-impl}$$

is equivalent to the disjunction of conjunctions

$$\bigvee_{i \in \mathbb{S}} [g(i) \wedge f(i)]. \tag{disj-conj}$$

The proof is provided in the appendix.

3.2.1 Eliminating Infinite Quantifiers

Several constraints are required to hold for all time points $(i, t) \in \mathbb{T}$, which is not allowed in MILP. Since all constraints in LCCA are linear or piecewise linear, it is possible to consider a finite quantification such that, if a constraint holds over it, it must hold over the infinite set.

First, consider the differential equations, which must be of the form (2). The derivative is not continuous over changes in i, but integration over each interval can be considered. We have

$$\int_{t_i^s}^{t} \mathrm{d}X(i,t) = \int_{t_i^s}^{t} \left[\sum_{a \in Aut} \bar{x}^a (Q^a(i)) + k \right] \mathrm{d}t \qquad \forall (i,t) \in \mathbb{T}, \tag{7}$$

which gives

$$X(i,t) = X(i,t_i^s) + \left[\sum_{a \in Aut} \bar{x}^a(Q^a(i)) + k\right](t - t_i^s) \qquad \forall (i,t) \in \mathbb{T}. \tag{8}$$

The quantification $\forall (i,t) \in \mathbb{T}$ can be rewritten as $\forall i \in \mathbb{N}, \forall t \in [t_i^s, t_i^e]$. For a fixed i, the above equation is linear in t. Thus, it suffices to consider only $t = t_i^e$, giving

$$X(i,t_i^e) = X(i,t_i^s) + \left[\sum_{a \in Aut} \bar{x}^a(Q^a(i)) + k\right]\Delta t_i \qquad \forall i \in \mathbb{N}, \tag{9}$$

where $\Delta t_i = t_i^e - t_i^s$. In other words, the value of X is determined only at time points t_i^s and t_i^e. Given these, $X(i,t)$ for any $t \in [t_i^s, t_i^e]$ can be determined because X evolves linearly.

Infinite quantifications also occur in invariants F and certain constraints of G_V. In both cases, the constraints are required to be from $\mathcal{L}(\mathbf{X}(i,t))$. By definition of $\mathcal{L}$, these are linear and the situation is similar to equation 8. Each constraint involving $X(i,t)$ can be written for $X(i,t_i^s)$ and $X(i,t_i^e)$ and then enforced over the finite set of intervals. For example, if X represents mass, we might have the constraint

$$X(i,t) \geq 0 \qquad \forall (i,t) \in \mathbb{T}, \tag{10}$$

which requires mass to be non-negative at all times. This can be replaced with

$$X(i,t_i^s) \geq 0 \qquad \forall i \in \mathbb{N} \tag{11a}$$
$$X(i,t_i^e) \geq 0 \qquad \forall i \in \mathbb{N}, \tag{11b}$$

which requires mass to be non-negative only at the beginning and end of every interval. Since it varies linearly within an interval, it is guaranteed to be non-negative at all points in between.

3.2.2 Eliminating Variable Arguments

Both dynamic variables and parameters are, in various places, evaluated with variable arguments. For example, $X(i,t_i^s)$ is evaluated at the two arguments i and t_i^s. Argument i can be interpreted as an index; it is not an unknown. However, the second argument t_i^s is a variable of unknown value because event points are not fixed. Continuous dynamic variables arise in this way in several locations: differential equations after integration (9), discontinuity equations (3), and the initial conditions that are allowed in G_V.

Automata specify a set of flow rates $\bar{\mathbf{x}}$. Each flow rate $\bar{x} \in \bar{\mathbf{x}}$ is a parameter, i.e. $\bar{x}(q)$ is a known constant given as part of the automaton's declaration. However, in equation (9), flow rates are used in the form $\bar{x}(Q(i))$, where the argument $Q(i)$ is an unknown. Again, there is a term with a variable argument.

Mixed-integer programs do not allow variable arguments of any form. We present methods for eliminating them in both forms encountered: $X(i,t_i^s)$ or $X(i,t_i^e)$, and $\bar{x}(Q(i))$.

$X(i,t_i^s)$ can be replaced with $X^s(i)$ wherever it occurs. This is possible with the recognition that $X(i,t_i^s)$ depends ultimately on just i because its second argument t_i^s is itself fully determined by i. Identically, all occurrences of $X(i,t_i^e)$ are replaced with $X^e(i)$. Instead of a single variable X evaluated at two time points (for every i), we have two variables X^s and X^e (for every i).

Replacing $\bar{x}(Q(i))$ is motivated by the same insight. The value of $\bar{x}(Q(i))$ is ultimately dependent on just i. Let us imagine another variable $\bar{w}(i)$ such that $\bar{w}(i) = \bar{x}(Q(i))$ for all i. The goal now is to satisfy this equation without resorting to any use of variable arguments. One way to accomplish this is to require

$$\bigwedge_{q \in \mathbb{Q}} [Q(i) = q \Rightarrow \bar{w}(i) = \bar{x}(q)] \qquad \forall i \in \mathbb{N}. \tag{12}$$

This constraint considers every mode. If the automaton is in mode q, then $\bar{w}(i)$ is set equal to the parameter $\bar{x}(q)$.

This is a conjunction of implications as in Theorem 1 if we recognize q as i, $\mathbb{Q}$ as $\mathbb{S}$, $Q(i) = q$ as g_i, and $\bar{w}(i) = \bar{x}(q)$ as f_i. The theorem can be applied if $Q(i) = q$ holds for exactly one q (for each i). This of course is true because an automaton can only be in a single mode at a time. So the above can be reformulated into the disjunctive constraint

$$\bigvee_{q \in \mathbb{Q}} \begin{bmatrix} Q(i) = q \\ \bar{w}(i) = \bar{x}(q) \end{bmatrix} \qquad \forall i \in \mathbb{N}. \tag{13}$$

In general, a new variable $\bar{w}^a$ will be needed for each $\bar{x}^a \in \bar{\mathbf{x}}^a$ of every automaton.

Let us implement these replacements into equation (9), which involves both forms of variable arguments. It becomes the two constraints

$$X^e(i) = X^s(i) + \sum_{a \in Aut} \left(\bar{w}^a(i) + k\right) \Delta t_i \qquad \forall i \in \mathbb{N} \tag{14}$$

$$\bigvee_{q \in \mathbb{Q}^a} \begin{bmatrix} Q^a(i) = q \\ \bar{w}^a(i) = \bar{x}^a(q) \end{bmatrix} \qquad \forall a \in Aut, \forall i \in \mathbb{N}. \tag{15}$$

Unfortunately, the first equation contains a bilinearity $\bar{w}^a(i)\,\Delta t_i$.

A slight modification to the procedure allows producing a linear equation. Instead of letting $\bar{w}(i) = \bar{x}(Q(i))$, require $\bar{w}(i) = \bar{x}(Q(i))\,\Delta t_i$. Now, we can write

$$X^e(i) = X^s(i) + \sum_{a \in Aut} \left(\bar{w}^a(i) + k\Delta t_i\right) \qquad \forall i \in \mathbb{N} \tag{16}$$

$$\bigvee_{q \in \mathbb{Q}^a} \begin{bmatrix} Q^a(i) = q \\ \bar{w}^a(i) = \bar{x}^a(q)\,\Delta t_i \end{bmatrix} \qquad \forall a \in Aut, \forall i \in \mathbb{N}. \tag{17}$$

Instead of having to multiply Δt_i with $\bar{w}^a(i)$, it is multiplied by $\bar{x}^a(q)$ within the disjunction. This is a parameter, and so the equation is still linear.

3.2.3 Converting Finite Domains to Booleans

By definition of $\mathcal{L}$, finite domain constraints are of the form $Q(i) = q$ or $Q(i) = Q(i+1)$, and these can be connected by the logical operators $\neg$, $\vee$, and $\wedge$. Both equations can be converted into Boolean propositions. For each dynamic finite domain variable Q of type $\mathbb{N} \to \mathbb{Q}$, introduce a Boolean variable Y of type $\mathbb{Q} \times \mathbb{N} \to \{\texttt{true}, \texttt{false}\}$. This allows associating a Boolean with each possible value of the finite domain variable.

The equation $Q(i) = q$ is simply replaced by the Boolean $Y(q, i)$. This substitution alone is not sufficient however. It would be possible for $Y(q, i)$ and $Y(q', i)$ to both be true for distinct q and q'. Translating this solution back into the original model would imply that $Q(i)$ takes two values. Clearly, that cannot be allowed. For every Boolean Y introduced, it is also necessary to include the constraint

$$\underline{\bigvee}_{q \in \mathbb{Q}} Y(q, i) \qquad \forall i \in \mathbb{N} \tag{18}$$

which guarantees that $Y(q, i)$ will be true for exactly one q (for each i).

The equation $Q(i) = Q(i+1)$ effectively says that there exists some q such that the automaton is in mode q for both intervals i and $i+1$. Stated as a formula, we have

$$\exists q \in \mathbb{Q} \text{ s.t. } \left[Q(i) = q \wedge Q(i+1) = q\right], \tag{19}$$

and now the finite domain equations are in the simpler form. The existential quantifier, when quantified over a finite set, is merely an alternative notation for indexed disjunction.

Simply changing this notation and replacing the equations with Booleans gives

$$\bigvee_{q\in\mathbb{Q}} \left[Y\left(q,i\right)\wedge Y\left(q,i+1\right)\right]. \tag{20}$$

3.3 Symmetry Breaking

In the course of generating an MILP, we also add some constraints for efficiency purposes; these do not affect the model. The source of the inefficiency is dummy transitions. They are a mathematical artifact allowing an automaton to transition but in a way that has no physical consequence. Unfortunately, these introduce a redundancy in the set of feasible trajectories because, at some event point, all automata might make a dummy transition, meaning the system has not actually evolved. Also, this could occur at a continuum of time values. An infinite number of mathematically distinct trajectories represent an identical physical solution. Avraam et al. (1998, p. S225) recognized this problem in a related system and proposed a solution which we accommodate to our framework.

A *dummy event point* is one at which all automata make dummy transitions. We require all dummy event points to occur at the end of a trajectory, i.e. if i is a dummy event point, then j is a dummy event point for all $j>i$. Also, the interval length after each dummy event point should be zero, i.e. if i is a dummy event point, then $\Delta t_{i+1}=0.0$. Occurrence of a dummy event point means the effective number of intervals is less than n. The trajectory with dummy event points squeezed at the end of the timeline has many others equivalent to it. Adding the stated constraints makes all but this one infeasible.

From the previous section, we let the Boolean $Y^a\left(q,i\right)$ substitute for the constraint $Q^a\left(i\right)=q$. For convenience, we let $YY^a\left(i\right)$ mean automaton a makes a dummy transition at the i^{th} event point and $YYY\left(i\right)$ mean the i^{th} event point is dummy. These are defined in terms of Y's with the constraints

$$YY^a\left(i\right)\Leftrightarrow \bigvee_{q\in\mathbb{Q}^a}\left[Y^a\left(q,i\right)\wedge Y^a\left(q,i+1\right)\right] \tag{21}$$

$$YYY\left(i\right)\Leftrightarrow \bigwedge_{a\in Aut} YY^a\left(i\right). \tag{22}$$

Now, the two symmetry breaking constraints are

$$YYY\left(i\right)\Rightarrow YYY\left(i+1\right)\qquad \forall i\in\mathbb{N}\backslash\left\{n-1,n\right\} \tag{23a}$$

$$YYY\left(i\right)\Rightarrow\left(\Delta t_{i+1}=0.0\right)\qquad \forall i\in\mathbb{N}\backslash\left\{n\right\}. \tag{23b}$$

The antecedent of both determines if i is a dummy event point by checking if all automata's discrete modes have remained unchanged. If so, the first requires the next event point to also be dummy and the second sets the next interval length to zero. If the antecedent is satisfied for i, the consequent of the first constraint is such that its antecedent will be true for $i+1$. This causes the constraint to be iteratively enforced for all $j>i$. The antecedent might not be satisfied for any i—there might not be any dummy event points—in which case these constraints have no effect.

3.4 Model Transformation

Converting the optimization problem $\min_{\xi\in\Xi}\Omega$ into the MILP $\min_{x\in P}\Omega'$ requires converting the objective and the model. Some objectives, e.g. minimize makespan, are already in an MILP form. Others, e.g. time averaged cost, involve terms with variable arguments. These are easily transformed by substituting variables X^s and X^e as discussed in Section 3.2.2. It remains to convert an LCCA model into MILP constraints.

A few steps are required to transform each of the elements $\left(n,G_t,\mathbf{X},Aut,G_V\right)$ of an LCCA model.

- The timeline is represented with MILP constraints.
- Component automata are reformulated into two disjunctive constraints: one over the discrete modes and the other over the transitions.
- These constraints along with G_V are converted using the methods of the previous section to eliminate infinite quantifiers, remove variable arguments, and transform finite domain logic into Boolean propositions. This will provide a model in GDP form.
- Finally, the GDP model is converted into an MILP using known techniques.

The first element of an LCCA model n is used simply to define the index set $\mathbb{N} = \{1, \ldots, n\}$ in the MILP model. According to the definition of LCCA, n is used to construct a hybrid timeline. The timeline is dictated by variables $\mathbf{t}$, and these are included in the MILP model unaltered. The constraints implicit in the definition of a hybrid timeline

$$t_i^s \leq t_i^e \qquad \forall i \in \mathbb{N} \tag{24a}$$

$$t_i^e = t_{i+1}^s \qquad \forall i \in \mathbb{N} \backslash \{n\} \tag{24b}$$

are included explicitly in the MILP model. Constraint G_t is already in an MILP form and is included unaltered.

Each variable $X \in \mathbf{X}$ is of type $\mathbb{T} \to \mathbb{R}$. Functions whose domains are infinite spaces are not allowed in MILP. Following the methods of Section 3.2.2, each X will be replaced with two variables X^s and X^e of type $\mathbb{N} \to \mathbb{R}$. Such functions can be interpreted simply as indexed variables and are allowed in MP models. Let $\mathbf{X}^s$ and $\mathbf{X}^e$ denote these new sets of variables.

The component automata *Aut* are the most involved constructs of an LCCA model. They can be restated as disjunctive constraints on real and finite domain variables. Recall each automaton in *Aut* is of the form $(\mathbb{Q}, \bar{\mathbf{x}}, \hat{\mathbf{x}}, F, Arc)$. (In this section, we speak generically of any automaton, and so the superscript a is omitted.) The finite domain space is left unaltered but will now be interpreted as an index set. The parameters $\bar{\mathbf{x}}$ and variables $\hat{\mathbf{x}}$ also remain unchanged; they are of a type allowed in MILP. It is F and *Arc* that must be transformed.

The invariant constraint F for each automaton is enforced as

$$F\left(Q\left(i\right)\right) \qquad \forall \left(i, t\right) \in \mathbb{T}, \tag{25}$$

which says there is some constraint associated with each mode, and we apply the constraint for the mode the system is currently in. Instead, consider each possible value of $Q\left(i\right)$ separately. Then, the above can be equivalently stated as

$$Q\left(i\right) = q \Rightarrow F\left(q\right) \qquad \forall q \in \mathbb{Q}, \forall \left(i, t\right) \in \mathbb{T}. \tag{26}$$

If the automaton is in mode q, then the invariant for that mode must be applied. The variable argument has been removed at the expense of introducing a quantifier. A universal quantifier over a finite set can be viewed as a notational variation for indexed conjunction. The above can be rewritten as

$$\bigwedge_{q \in \mathbb{Q}} \left[Q\left(i\right) = q \Rightarrow F\left(q\right)\right] \qquad \forall \left(i, t\right) \in \mathbb{T}. \tag{27}$$

This is a conjunction of implications as in Theorem 1 if we recognize q as i, $\mathbb{Q}$ as $\mathbb{S}$, $Q\left(i\right) = q$ as g_i, and $F\left(q\right)$ as f_i. The theorem allows reformulating the constraint into a *disjunction over modes*

$$\bigvee_{q \in \mathbb{Q}} \begin{bmatrix} Q\left(i\right) = q \\ F\left(q\right) \end{bmatrix} \qquad \forall \left(i, t\right) \in \mathbb{T}. \tag{28}$$

The constraint still involves an infinite quantifier and the disjunct involves a finite domain variable. Application of the constraint conversions discussed in Section 3.2 will produce a disjunction in GDP form.

At event i, a transition can occur from mode q to q' if $(q, q') \in Arc$. In addition, it is required that both the guard $\gamma_{(q,q')}$ and reset $\rho_{(q,q')}$ are enforced. Written as a formula, we can say

$$\bigwedge_{(q,q')\in Arc} \left[(Q(i) = q \wedge Q(i+1) = q') \Rightarrow \gamma_{(q,q')} \wedge \rho_{(q,q')} \right] \qquad \forall i \in \mathbb{N} \backslash \{n\}. \tag{29}$$

Every transition is considered. If the transition taken from interval i to $i+1$ is from mode q to q', then the guard and reset along that transition are enforced.

Again, we have a conjunction of implications as in Theorem 1 if we recognize (q, q') as i, Arc as $\mathbb{S}$, $Q(i) = q \wedge Q(i+1) = q'$ as g_i, and $\gamma_{(q,q')} \wedge \rho_{(q,q')}$ as f_i. The theorem is applicable if $Q(i) = q \wedge Q(i+1) = q'$ is valid only for a unique pair (q, q'). It must hold for at least one pair because a transition must be made at an event. It does not hold for more than one because component automaton, by definition, allow only a single transition between any two modes (with possibly both modes the same). Thus, we can transform the above constraint into a *disjunction over transitions*

$$\bigvee_{(q,q')\in Arc} \left[\begin{array}{c} Q(i) = q \wedge Q(i+1) = q' \\ \gamma_{(q,q')} \wedge \rho_{(q,q')} \end{array} \right] \qquad \forall i \in \mathbb{N} \backslash \{n\}. \tag{30}$$

The number of disjuncts can be reduced by recognizing that $\gamma_{(q,q)}$ and $\rho_{(q,q)}$ are identical for all dummy transitions (q, q). First, let the finite domain constraint be replaced with the Boolean proposition $Y(q,i) \wedge Y(q', i+1)$, and recall equation (21) defined $YY(i)$—the superscript a omitted for now—to mean an automaton makes a dummy transition at event i. Now, the disjuncts can be partitioned to give

$$\left(\bigvee_{\substack{(q,q')\in Arc \\ q\neq q'}} \left[\begin{array}{c} Y(q,i) \wedge Y(q', i+1) \\ \gamma_{(q,q')} \wedge \rho_{(q,q')} \end{array} \right] \right) \vee \left[\begin{array}{c} YY(i) \\ \gamma_{(q,q)} \wedge \rho_{(q,q)} \end{array} \right] \qquad \forall i \in \mathbb{N} \backslash \{n\}. \tag{31}$$

There is now one disjunct instead of $|\mathbb{Q}|$ for the dummy transitions.

The disjunction over modes (28) and the disjunction over transitions (31) can be written for all component automaton $a \in Aut$, replacing F^a and Arc^a for each.

These disjunctions along with G_V still involve infinite quantifiers, variable arguments, and finite domain constraints. Let F', γ', ρ', and G'_V refer to the respective constraints after applying the conversions of Section 3.2. This introduces a new set of variables $\bar{\mathbf{w}}$, when eliminating variable arguments in terms of the form $\bar{x}(Q(i))$. Let $\mathbf{Y}$ be the set of Boolean variables introduced to replace the finite domain variables. In summary, the resulting GDP model is

$$
\begin{aligned}
\min_{\mathbf{t},\mathbf{X}^s,\mathbf{X}^e,\hat{\mathbf{x}},\bar{\mathbf{w}},\mathbf{Y}} \quad & \Omega' \\
\text{s.t.} \quad & t_i^s \le t_i^e && \forall i \in \mathbb{N} \\
& t_i^e = t_{i+1}^s && \forall i \in \mathbb{N} \setminus \{n\} \\
& G_t \\
& \bigvee_{q \in \mathbb{Q}^a} \begin{bmatrix} Y^a(q,i) \\ F'(q) \end{bmatrix} && \forall i \in \mathbb{N}, \forall a \in Aut \\
& \left(\bigvee_{\substack{(q,q') \in Arc \\ q \neq q'}} \begin{bmatrix} Y(q,i) \wedge Y(q', i+1) \\ \gamma'_{(q,q')} \wedge \rho'_{(q,q')} \end{bmatrix} \right) \vee \begin{bmatrix} YY(i) \\ \gamma'_{(q,q)} \wedge \rho'_{(q,q)} \end{bmatrix} && \forall i \in \mathbb{N} \setminus \{n\}, \forall a \in Aut \\
& G'_V \\
& YYY(i) \Rightarrow YYY(i+1) && \forall i \in \mathbb{N} \setminus \{n-1, n\} \\
& YYY(i) \Rightarrow (\Delta t_{i+1} = 0.0) && \forall i \in \mathbb{N} \setminus \{n\}. \qquad \text{(GDP(LCCA))}
\end{aligned}
$$

Each of the elements $(n, G_t, \mathbf{X}, Aut, G_V)$ of an LCCA model are represented in this GDP model; so we call it GDP(LCCA). This intermediate GDP representation is valuable for two reasons. Firstly, it is more natural to reformulate the LCCA model with disjunctive constraints, and it is unlikely that we could have provided MILP constraints directly. Furthermore, Raman and Grossmann (1994) have shown that GDP models often lead to efficient MILP formulations.

Element n is used to define the index set $\mathbb{N} = \{1, \ldots, n\}$, the first two constraints define a timeline, G_t is included unaltered, the two main disjunctions represent the component automata, and G'_V is the result of converting the final component G_V. The symmetry breaking constraints are added for efficiency reasons. Items $\mathbb{Q}^\alpha$, Aut, and Arc are still present but they are now used only as index sets.

Model GDP(LCCA) is nearly in the form defined by Raman and Grossmann (1994), and so their transformation can be mostly applied to produce an MILP. Two main steps are required: the Boolean propositions within the disjunctions, G'_V, and the symmetry breaking constraints are converted into integer constraints, and the disjunctive constraints are transformed using the convex hull method.

An implicit requirement of Raman and Grossmann's (1994) method is that exactly one Boolean variable amongst all in the disjuncts of a disjunction must be true. For example, their method can be applied to convert the disjunction over modes only if $\veebar_{q \in \mathbb{Q}^a} Y^a(q,i)$ holds for all i and a. This is satisfied because the disjunction over modes was obtained by application of Theorem 1 for which this is a precondition. Similarly, $\veebar_{(q,q') \in Arc^a} Z^a(q,q',i)$ is guaranteed; so their method can be applied to the disjunction over transitions also. Finally, G'_V will contain disjunctions of the form (17), and their requirement is satisfied here also.

Model GDP(LCCA) differs from Raman and Grossmann's form in two minor ways. They allow only a single Boolean variable in disjunctive constraints, but the disjunction over transitions includes the Boolean expression $Y^a(q,i) \wedge Y^a(q', i+1)$. This is easily rectified by adding the Boolean constraint

$$
Z^a(q,q',i) \Leftrightarrow (Y^a(q,i) \wedge Y^a(q',i+1)) \qquad \forall i \in \mathbb{N} \setminus \{n\}, \forall a \in Aut, \forall q, q' \in \mathbb{Q}^a \tag{32}
$$

and then replacing the Boolean expression in the disjunct with $Z^a(q,q',i)$. (We can also use Z^a to simplify equation (21) to $YY^a(i) \Leftrightarrow \vee_{q \in \mathbb{Q}^a} Z^a(q,q,i)$.)

The second difference arises in the symmetry breaking constraint $YYY(i) \Rightarrow (\Delta t_{i+1} = 0.0)$. This is equivalent to the disjunction $[\neg YYY(i)] \vee [\Delta t_{i+1} = 0]$, which is not in the form required for Raman and Grossmann's method. However, this constraint is easily seen to be equivalent to the mixed-integer inequalities

$$0.0 \le \Delta t_{i+1} \le T^{\max} \left(1 - yyy\left(i\right)\right). \tag{33}$$

This final step converts model GDP(LCCA) into an MILP.

4 Application: Switched Flow Process

We now demonstrate use of the LCCA framework by modeling a small switched flow process. This model will then be converted to an MILP using our procedure, and some optimization results provided. The following is a conceptual description of the system we wish to model.

> Figure 3 depicts a tank being filled by two hybrid processes, α and β, and being emptied continuously at a rate of $F^{\text{out}} = 1.8$. Initially, the material level in the tank is $M_0 = 20.0$. The tank's maximum capacity $M^{\max} = 150.0$ and the material level should never fall below $M^{\min} = 10.0$.
>
> Process α represents a pump that can be either on or off. When it is on, it feeds material to the tank at rate 2.0. There is also an operating cost of 10.0 per unit time for running the pump. Operational constraints on the pump forbid it from being continuously run longer than 30.0 time units; it must be switched off before this time limit is reached. There are no operating costs while it is off, but it must not be switched on again in less than 2.0 time units and should not remain off for more than 1000.0 time units. When it is switched on again, a startup cost of 50.0 is incurred.
>
> Process β is similar, but it represents a pump that is always on, either at a high or low setting. In the high setting, material flows to the tank at rate 4.0 and the operating cost is 15.0 per time unit. It cannot remain on for more than 40.0 time units. In the low setting, the material flow rate drops to 0.5, and the operating cost is 2.0. It should not remain in the low setting for longer than 1000.0 time units. Once set to low, the pump cannot be switched to the high setting again for at least 3.0 time units, and, when it does, a startup cost of 40.0 is incurred.
>
> We wish to study how the material level changes over time and to understand the cost of running the system for $T^{\max} = 500.0$ time units.

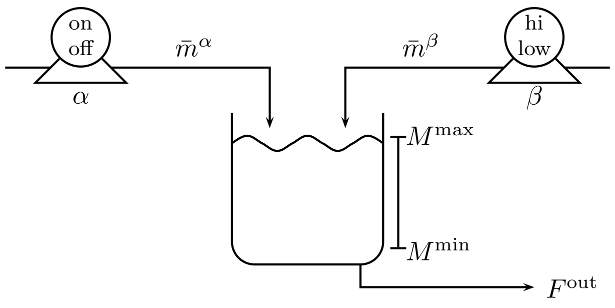


Figure 3: Schematic of switched flow process.

4.1 LCCA Model

We now show that an LCCA model for this system can be formulated with relative ease. First, the following variables are defined:

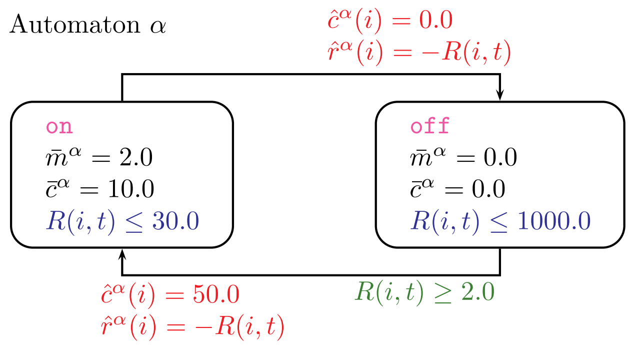
- $M(i,t)$ is material level in tank, and $\bar{m}^a(q)$ is rate at which process a feeds material into tank when in mode q.
- $C(i,t)$ is total incurred cost, $\bar{c}^a(q)$ is operating cost incurred for process a in mode q, and $\hat{c}^a(i)$ is instantaneous cost incurred for process a switching modes at event i.
- R and S stand for the amount of time processes α and β, respectively, have been in their current discrete mode, $\hat{r}^a(i)$ and $\hat{s}^a(i)$ are the instantaneous changes to these values at event i.

Then, the LCCA model is

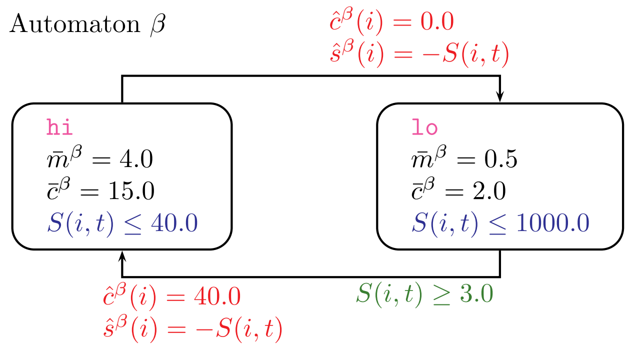
$$n = 10 \tag{34}$$

$$t_n^e = T^{\max} \tag{35a}$$

$$t_1^s = 0.0 \tag{35b}$$



(36a)



(36b)

$$\frac{\mathrm{d}R(i,t)}{\mathrm{d}t} = 1.0 \qquad \forall (i,t) \in \mathbb{T} \tag{37a}$$

$$R(i+1,t) = R(i,t) + \hat{r}^\alpha(i) \qquad \forall i \in \mathbb{N} \setminus \{n\} \tag{37b}$$

$$R(1, t_1^s) = 0.0 \tag{37c}$$

$$\frac{\mathrm{d}S(i,t)}{\mathrm{d}t} = 1.0 \qquad \forall (i,t) \in \mathbb{T} \tag{37d}$$

$$S(i+1,t) = S(i,t) + \hat{s}^\beta(i) \qquad \forall i \in \mathbb{N} \setminus \{n\} \tag{37e}$$

$$S(1, t_1^s) = 0.0 \tag{37f}$$

$$\frac{\mathrm{d}M\left(i,t\right)}{\mathrm{d}t} = \bar{m}^{\alpha}\left(Q^{\alpha}\left(i\right)\right) + \bar{m}^{\beta}\left(Q^{\beta}\left(i\right)\right) - F^{\text{out}} \qquad \forall\left(i,t\right)\in\mathbb{T} \tag{37g}$$

$$M\left(i+1,t\right) = M\left(i,t\right) \qquad \forall i\in\mathbb{N}\setminus\left\{n\right\} \tag{37h}$$

$$M\left(1,t_1^s\right) = M_0 \tag{37i}$$

$$M^{\min} \le M\left(i,t\right) \le M^{\max} \qquad \forall\left(i,t\right)\in\mathbb{T} \tag{37j}$$

$$\frac{\mathrm{d}C\left(i,t\right)}{\mathrm{d}t} = \bar{c}^{\alpha}\left(Q^{\alpha}\left(i\right)\right) + \bar{c}^{\beta}\left(Q^{\beta}\left(i\right)\right) \qquad \forall\left(i,t\right)\in\mathbb{T} \tag{37k}$$

$$C\left(i+1,t\right) = C\left(i,t\right) + \hat{c}^{\alpha}\left(i\right) + \hat{c}^{\beta}\left(i\right) \qquad \forall i\in\mathbb{N}\setminus\left\{n\right\} \tag{37l}$$

$$C\left(1,t_1^s\right) = 0.0 \tag{37m}$$

All components $(n, G_t, \mathbf{X}, Aut, G_V)$ of an LCCA model have been defined above. The first equation sets n to 10. Equations (35) represent constraints on the timeline in the form of G_t. The set of variables $\mathbf{X} = \{R, S, M, C\}$. A component automata is provided for each hybrid process, $Aut = \{\alpha, \beta\}$, and both are declared graphically. The remaining equations constitute constraint G_V. Figure 4 shows a feasible trajectory for this system.

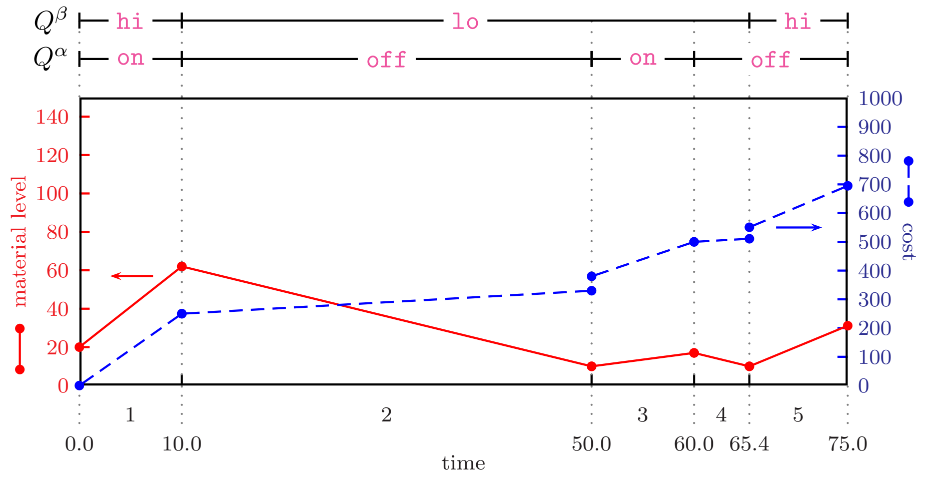


Figure 4: Initial segment of a feasible trajectory for switched flow process.

Equation (37g) states that the material level's rate of change depends on $\bar{m}^{\alpha}$ and $\bar{m}^{\beta}$, and these values themselves depend on the modes of the automata. Consider the trajectory shown in Figure 4 during interval 3. Within this interval, $Q^{\alpha}\left(i\right) = \texttt{on}$ and $Q^{\beta}\left(i\right) = \texttt{lo}$. In these modes, the differential equation for M is

$$\begin{aligned}\frac{\mathrm{d}M\left(i,t\right)}{\mathrm{d}t} &= 2.0 + 0.5 - 1.8\\ &= 0.7\end{aligned}$$

At $t = 60.0$, automaton α transitions to its `off` mode, the equation becomes

$$\begin{aligned}\frac{\mathrm{d}M\left(i,t\right)}{\mathrm{d}t} &= 0.0 + 0.5 - 1.8\\ &= -1.3\end{aligned}$$

The right-hand-side switches between different constants.

The differential equation for cost C is similar, but costs can also be incurred instantaneously by equation (37l). These represent the startup costs. Consider the transition from interval 2 to 3 in Figure 4. Automaton α transitions from `off` to `on`. The reset on this transition includes the equation $\hat{c}^{\alpha}(i) = 50.0$. Automata β remains in its `lo` mode. This is modeled with the dummy transition, which includes the reset $\hat{c}^{\beta}(i) = 0.0$ by definition. Equation (37l) becomes

$$\begin{aligned} C(i+1,t) &= C(i,t) + 50.0 + 0.0 \\ &= C(i,t) + 50.0 \end{aligned}$$

which means the cost gets incremented by 50.0. The data is correctly plotted with two filled in circles at $t = 50.0$. Since this is an event point, there are two time points, $(2, 50.0)$ and $(3, 50.0)$, at this position.

R and S are called clock variables because they keep track of time, as indicated by setting their rates to 1.0. Equation (37b) increments the value of clock R by $\hat{r}^{\alpha}(i)$ at every event. On both transitions of α, the reset is $\hat{r}^{\alpha}(i) = -R(i,t)$, but this is simply the negation of the clock value at the beginning of the event. In other words, the clock gets reset to 0.0. Dummy transitions are not shown, but their definition is fixed such that $\hat{r}^{\alpha}(i) = 0.0$ on those transitions. Clocks are not reset when an automaton transitions from some mode back to itself.

The guard on the transition from `off` to `on` is $R(i,t) \geq 2.0$. Upon entry into the `off` mode, clock R is set to 0.0 and evolves at rate 1.0 while in this mode. Thus, this guard states that the automaton must remain in the `off` mode for at least 2.0 time units. Finally, there are several invariants regarding the maximum time that each automaton can remain in its discrete modes, and these are satisfied in the trajectory shown.

4.2 GDP Model

Now, using the transformation methods discussed in Section 3, we present a GDP model that is equivalent to the LCCA model of the previous section. Several new variables are required; the naming convention follows that in the description of the general procedure. The GDP model is

$$t_i^s \leq t_i^e \qquad \forall i \in \mathbb{N} \tag{38a}$$

$$t_i^e = t_{i+1}^s \qquad \forall i \in \mathbb{N} \setminus \{n\} \tag{38b}$$

$$\Delta t_i = t_i^e - t_i^s \qquad \forall i \in \mathbb{N} \tag{38c}$$

$$t_n^e = T^{\max} \tag{39a}$$

$$t_1^s = 0.0 \tag{39b}$$

$$\begin{bmatrix} Y^{\alpha}(\texttt{on}, i) \\ R^s(i) \leq 30.0 \\ R^e(i) \leq 30.0 \end{bmatrix} \vee \begin{bmatrix} Y^{\alpha}(\texttt{off}, i) \\ R^s(i) \leq 1000.0 \\ R^e(i) \leq 1000.0 \end{bmatrix} \qquad \forall i \in \mathbb{N} \tag{40a}$$

$$\begin{bmatrix} Z^{\alpha}(\texttt{on}, \texttt{off}, i) \\ \hat{c}^{\alpha}(i) = 0.0 \\ \hat{r}^{\alpha}(i) = -R^e(i) \end{bmatrix} \vee \begin{bmatrix} Z^{\alpha}(\texttt{off}, \texttt{on}, i) \\ R^e(i) \geq 2.0 \\ \hat{c}^{\alpha}(i) = 50.0 \\ \hat{r}^{\alpha}(i) = -R^e(i) \end{bmatrix} \vee \begin{bmatrix} YY^{\alpha}(i) \\ \hat{c}^{\alpha}(i) = 0.0 \\ \hat{r}^{\alpha}(i) = 0.0 \end{bmatrix} \qquad \forall i \in \mathbb{N} \setminus \{n\} \tag{40b}$$

$$\begin{bmatrix} Y^{\beta}\left(\texttt{hi}, i\right) \\ S^{s}\left(i\right) \leq 40.0 \\ S^{e}\left(i\right) \leq 40.0 \end{bmatrix} \vee \begin{bmatrix} Y^{\beta}\left(\texttt{lo}, i\right) \\ S^{s}\left(i\right) \leq 1000.0 \\ S^{e}\left(i\right) \leq 1000.0 \end{bmatrix} \qquad \forall i \in \mathbb{N} \tag{40c}$$

$$\begin{bmatrix} Z^{\beta}\left(\texttt{hi}, \texttt{lo}, i\right) \\ \hat{c}^{\beta}\left(i\right) = 0.0 \\ \hat{s}^{\beta}\left(i\right) = -S^{e}\left(i\right) \end{bmatrix} \vee \begin{bmatrix} Z^{\beta}\left(\texttt{lo}, \texttt{hi}, i\right) \\ S^{e}\left(i\right) \geq 3.0 \\ \hat{c}^{\beta}\left(i\right) = 40.0 \\ \hat{s}^{\beta}\left(i\right) = -S^{e}\left(i\right) \end{bmatrix} \vee \begin{bmatrix} YY^{\beta}\left(i\right) \\ \hat{c}^{\beta}\left(i\right) = 0.0 \\ \hat{s}^{\beta}\left(i\right) = 0.0 \end{bmatrix} \qquad \forall i \in \mathbb{N}\setminus\left\{n\right\} \tag{40d}$$

$$R^{e}\left(i\right) = R^{s}\left(i\right) + \Delta t_i \qquad \forall i \in \mathbb{N} \tag{41a}$$

$$R^{s}\left(i+1\right) = R^{e}\left(i\right) + \hat{r}^{\alpha}\left(i\right) \qquad \forall i \in \mathbb{N}\setminus\left\{n\right\} \tag{41b}$$

$$R^{s}\left(1\right) = 0.0 \tag{41c}$$

$$S^{e}\left(i\right) = S^{s}\left(i\right) + \Delta t_i \qquad \forall i \in \mathbb{N} \tag{41d}$$

$$S^{s}\left(i+1\right) = S^{e}\left(i\right) + \hat{s}^{\beta}\left(i\right) \qquad \forall i \in \mathbb{N}\setminus\left\{n\right\} \tag{41e}$$

$$S^{s}\left(1\right) = 0.0 \tag{41f}$$

$$M^{e}\left(i\right) = M^{s}\left(i\right) + \bar{w}^{m,\alpha}\left(i\right) + \bar{w}^{m,\beta}\left(i\right) - F^{\text{out}}\Delta t_i \qquad \forall i \in \mathbb{N} \tag{41g}$$

$$M^{s}\left(i+1\right) = M^{e}\left(i\right) \qquad \forall i \in \mathbb{N}\setminus\left\{n\right\} \tag{41h}$$

$$M^{s}\left(1\right) = M_0 \tag{41i}$$

$$M^{\min} \leq M^{s}\left(i\right) \leq M^{\max} \qquad \forall i \in \mathbb{N} \tag{41j}$$

$$M^{\min} \leq M^{e}\left(i\right) \leq M^{\max} \qquad \forall i \in \mathbb{N} \tag{41k}$$

$$C^{e}\left(i\right) = C^{s}\left(i\right) + \bar{w}^{c,\alpha}\left(i\right) + \bar{w}^{c,\beta}\left(i\right) \qquad \forall i \in \mathbb{N} \tag{41l}$$

$$C^{s}\left(i+1\right) = C^{e}\left(i\right) + \hat{c}^{\alpha}\left(i\right) + \hat{c}^{\beta}\left(i\right) \qquad \forall i \in \mathbb{N}\setminus\left\{n\right\} \tag{41m}$$

$$C^{s}\left(1\right) = 0.0 \tag{41n}$$

$$\bigvee_{q \in \mathbb{Q}^a} \begin{bmatrix} Y^{a}\left(q, i\right) \\ \bar{w}^{m,a}\left(i\right) = \bar{m}^{a}\left(q\right)\Delta t_i \\ \bar{w}^{c,a}\left(i\right) = \bar{c}^{a}\left(q\right)\Delta t_i \end{bmatrix} \qquad \forall i \in \mathbb{N}, \forall a \in Aut \tag{41o}$$

$$\underline{\bigvee}_{q \in \mathbb{Q}^a} Y^{a}\left(q, i\right) \qquad \forall i \in \mathbb{N}, \forall a \in Aut \tag{42}$$

$$YYY(i) \Rightarrow YYY\left(i+1\right) \qquad \forall i \in \mathbb{N}\setminus\left\{n-1, n\right\} \tag{43a}$$

$$YYY(i) \Rightarrow \left(\Delta t_{i+1} = 0.0\right) \qquad \forall i \in \mathbb{N}\setminus\left\{n\right\} \tag{43b}$$

$$YY^{a}\left(i\right) \Leftrightarrow \bigvee_{q \in \mathbb{Q}^a} Z^{a}\left(q, q, i\right) \qquad \forall i \in \mathbb{N}\setminus\left\{n\right\}, \forall a \in Aut \tag{44a}$$

$$YYY\left(i\right) \Leftrightarrow \bigwedge_{a \in Aut} YY^{a}\left(i\right) \qquad \forall i \in \mathbb{N}\setminus\left\{n\right\} \tag{44b}$$

$$Z^{a}\left(q, q', i\right) \Leftrightarrow \left(Y^{a}\left(q, i\right) \wedge Y^{a}\left(q', i+1\right)\right) \qquad \forall i \in \mathbb{N}\setminus\left\{n\right\}, \forall a \in Aut, \forall q, q' \in \mathbb{Q}^a \tag{44c}$$

The first set of constraints involve the timeline variables. Then, there are two disjunctions for each of the two automata. Following these, there is a set of equations for each of the continuous variables R, S, M, and C. Finally, there is a disjunction needed for the auxiliary variable $\bar{w}$, and the symmetry breaking constraints.

This GDP model is rather more complex than the corresponding LCCA model. It would have been difficult to think of this model directly. It requires several variables, e.g. $\bar{w}^{m,\alpha}$, unrelated to the physical conception of the system. Various constraints, such as evolution of mass, have to be defined in terms of these auxiliary variables, making the constraints themselves less intuitive.

4.3 MILP Model

Finally, we convert the GDP model to an MILP model. This requires introduction of many new variables. For conversion of disjunctive constraints, our notational convention employs superscripts of the form 1, 2, or 11, 21. For example, if there is a disjunction with two disjuncts, two disaggregated variables will be needed for each variable. For the variable X, we introduce X^1 and X^2 as its disaggregated variables. If the same variable X is also used in another disjunction, a separate set of disaggregated variables will be needed for it. In that case, double superscripts are used, e.g. X^{21} is the disaggregated variable for the 2nd disjunct of the 1st disjunction. By convention, Boolean variables are capitalized and their corresponding binary $\{0,1\}$ variables are in lower case.

The full MILP model for the example is

$$t_i^s \leq t_i^e \qquad \forall i \in \mathbb{N} \tag{45a}$$
$$t_i^e = t_{i+1}^s \qquad \forall i \in \mathbb{N} \backslash \{n\} \tag{45b}$$
$$\Delta t_i = t_i^e - t_i^s \qquad \forall i \in \mathbb{N} \tag{45c}$$

$$t_n^e = T^{\max} \tag{46a}$$
$$t_1^s = 0.0 \tag{46b}$$

$$\begin{bmatrix} 0.0 \leq R^{s,1}(i) \leq 30.0 y^{\alpha}(\texttt{on}, i) \\ 0.0 \leq R^{e,11}(i) \leq 30.0 y^{\alpha}(\texttt{on}, i) \end{bmatrix} \qquad \forall i \in \mathbb{N} \tag{47a}$$
$$\begin{bmatrix} 0.0 \leq R^{s,2}(i) \leq 1000.0 y^{\alpha}(\texttt{off}, i) \\ 0.0 \leq R^{e,21}(i) \leq 1000.0 y^{\alpha}(\texttt{off}, i) \end{bmatrix} \qquad \forall i \in \mathbb{N} \tag{47b}$$
$$R^s(i) = R^{s,1}(i) + R^{s,2}(i) \qquad \forall i \in \mathbb{N} \tag{47c}$$
$$R^e(i) = R^{e,11}(i) + R^{e,21}(i) \qquad \forall i \in \mathbb{N} \tag{47d}$$

$$\begin{bmatrix} \hat{c}^{\alpha,1}(i) = 0.0 \\ \hat{r}^{\alpha,1}(i) = -R^{e,12}(i) \\ 0.0 \leq R^{e,12}(i) \leq 30.0 z^{\alpha}(\texttt{on}, \texttt{off}, i) \end{bmatrix} \qquad \forall i \in \mathbb{N} \backslash \{n\} \tag{47e}$$
$$\begin{bmatrix} 2.0 z^{\alpha}(\texttt{off}, \texttt{on}, i) \leq R^{e,22}(i) \leq 1000.0 z^{\alpha}(\texttt{off}, \texttt{on}, i) \\ \hat{c}^{\alpha,2}(i) = 50.0 z^{\alpha}(\texttt{off}, \texttt{on}, i) \\ \hat{r}^{\alpha,2}(i) = -R^{e,22}(i) \end{bmatrix} \qquad \forall i \in \mathbb{N} \backslash \{n\} \tag{47f}$$
$$\begin{bmatrix} \hat{c}^{\alpha,3}(i) = 0.0 \\ \hat{r}^{\alpha,3}(i) = 0.0 \\ 0.0 \leq R^{e,32}(i) \leq 1000.0 yy^{\alpha}(i) \end{bmatrix} \qquad \forall i \in \mathbb{N} \backslash \{n\} \tag{47g}$$
$$\hat{c}^{\alpha}(i) = \hat{c}^{\alpha,1}(i) + \hat{c}^{\alpha,2}(i) + \hat{c}^{\alpha,3}(i) \qquad \forall i \in \mathbb{N} \backslash \{n\} \tag{47h}$$
$$\hat{r}^{\alpha}(i) = \hat{r}^{\alpha,1}(i) + \hat{r}^{\alpha,2}(i) + \hat{r}^{\alpha,3}(i) \qquad \forall i \in \mathbb{N} \backslash \{n\} \tag{47i}$$
$$R^e(i) = R^{e,12}(i) + R^{e,22}(i) + R^{e,32}(i) \qquad \forall i \in \mathbb{N} \backslash \{n\} \tag{47j}$$

$$\begin{bmatrix} 0.0 \leq S^{s,1}(i) \leq 40.0 y^{\beta}(\texttt{hi}, i) \\ 0.0 \leq S^{e,11}(i) \leq 40.0 y^{\beta}(\texttt{hi}, i) \end{bmatrix} \quad \forall i \in \mathbb{N} \tag{47k}$$

$$\begin{bmatrix} 0.0 \leq S^{s,2}(i) \leq 1000.0 y^{\beta}(\texttt{lo}, i) \\ 0.0 \leq S^{e,21}(i) \leq 1000.0 y^{\beta}(\texttt{lo}, i) \end{bmatrix} \quad \forall i \in \mathbb{N} \tag{47l}$$

$$S^{s}(i) = S^{s,1}(i) + S^{s,2}(i) \quad \forall i \in \mathbb{N} \tag{47m}$$

$$S^{e}(i) = S^{e,11}(i) + S^{e,21}(i) \quad \forall i \in \mathbb{N} \tag{47n}$$

$$\begin{bmatrix} \hat{c}^{\beta,1}(i) = 0.0 \\ \hat{s}^{\beta,1}(i) = -S^{e,12}(i) \\ 0.0 \leq S^{e,12}(i) \leq 40.0 z^{\beta}(\texttt{hi}, \texttt{lo}, i) \end{bmatrix} \quad \forall i \in \mathbb{N} \setminus \{n\} \tag{47o}$$

$$\begin{bmatrix} 3.0 z^{\beta}(\texttt{lo}, \texttt{hi}, i) \leq S^{e,22}(i) \leq 1000.0 z^{\beta}(\texttt{lo}, \texttt{hi}, i) \\ \hat{c}^{\beta,2}(i) = 40.0 z^{\beta}(\texttt{lo}, \texttt{hi}, i) \\ \hat{s}^{\beta,2}(i) = -S^{e,22}(i) \end{bmatrix} \quad \forall i \in \mathbb{N} \setminus \{n\} \tag{47p}$$

$$\begin{bmatrix} \hat{c}^{\beta,3}(i) = 0.0 \\ \hat{s}^{\beta,3}(i) = 0.0 \\ 0.0 \leq S^{e,32}(i) \leq 1000.0 yy^{\beta}(i) \end{bmatrix} \quad \forall i \in \mathbb{N} \setminus \{n\} \tag{47q}$$

$$\hat{c}^{\beta}(i) = \hat{c}^{\beta,1}(i) + \hat{c}^{\beta,2}(i) + \hat{c}^{\beta,3}(i) \quad \forall i \in \mathbb{N} \setminus \{n\} \tag{47r}$$

$$\hat{s}^{\beta}(i) = \hat{s}^{\beta,1}(i) + \hat{s}^{\beta,2}(i) + \hat{s}^{\beta,3}(i) \quad \forall i \in \mathbb{N} \setminus \{n\} \tag{47s}$$

$$S^{e}(i) = S^{e,12}(i) + S^{e,22}(i) + S^{e,32}(i) \quad \forall i \in \mathbb{N} \setminus \{n\} \tag{47t}$$

$$R^{e}(i) = R^{s}(i) + \Delta t_i \quad \forall i \in \mathbb{N} \tag{48a}$$

$$R^{s}(i+1) = R^{e}(i) + \hat{r}^{\alpha}(i) \quad \forall i \in \mathbb{N} \setminus \{n\} \tag{48b}$$

$$R^{s}(1) = 0.0 \tag{48c}$$

$$S^{e}(i) = S^{s}(i) + \Delta t_i \quad \forall i \in \mathbb{N} \tag{48d}$$

$$S^{s}(i+1) = S^{e}(i) + \hat{s}^{\beta}(i) \quad \forall i \in \mathbb{N} \setminus \{n\} \tag{48e}$$

$$S^{s}(1) = 0.0 \tag{48f}$$

$$M^{e}(i) = M^{s}(i) + \bar{w}^{m,\alpha}(i) + \bar{w}^{m,\beta}(i) - F^{\text{out}} \Delta t_i \quad \forall i \in \mathbb{N} \tag{48g}$$

$$M^{s}(i+1) = M^{e}(i) \quad \forall i \in \mathbb{N} \setminus \{n\} \tag{48h}$$

$$M^{s}(1) = M_0 \tag{48i}$$

$$M^{\min} \leq M^{s}(i) \leq M^{\max} \quad \forall i \in \mathbb{N} \tag{48j}$$

$$M^{\min} \leq M^{e}(i) \leq M^{\max} \quad \forall i \in \mathbb{N} \tag{48k}$$

$$C^{e}(i) = C^{s}(i) + \bar{w}^{c,\alpha}(i) + \bar{w}^{c,\beta}(i) \quad \forall i \in \mathbb{N} \tag{48l}$$

$$C^{s}(i+1) = C^{e}(i) + \hat{c}^{\alpha}(i) + \hat{c}^{\beta}(i) \quad \forall i \in \mathbb{N} \setminus \{n\} \tag{48m}$$

$$C^{s}(1) = 0.0 \tag{48n}$$

$$\left[\begin{array}{c} w^{m,a,q}(i) = \bar{m}^a(q)\,\Delta t_i^{1,q,a} \\ w^{c,a,q}(i) = \bar{c}^a(q)\,\Delta t_i^{1,q,a} \\ 0.0 \le \Delta t_i^{1,q,a} \le T^{\max} y^a(q,i) \end{array}\right] \qquad \forall i \in \mathbb{N}, \forall a \in Aut, \forall q \in \mathbb{Q}^a \tag{48o}$$

$$\bar{w}^{m,a}(i) = \sum_{q \in \mathbb{Q}^a} w^{m,a,q}(i) \qquad \forall i \in \mathbb{N}, \forall a \in Aut \tag{48p}$$

$$\bar{w}^{c,a}(i) = \sum_{q \in \mathbb{Q}^a} w^{c,a,q}(i) \qquad \forall i \in \mathbb{N}, \forall a \in Aut \tag{48q}$$

$$\Delta t_i = \sum_{q \in \mathbb{Q}^a} \Delta t_i^{1,q,a} \qquad \forall i \in \mathbb{N}, \forall a \in Aut \tag{48r}$$

$$\sum_{q \in \mathbb{Q}^a} y^a(q,i) = 1 \qquad \forall i \in \mathbb{N}, \forall a \in Aut \tag{49}$$

$$1 - yyy(i) + yyy(i+1) \ge 1 \qquad \forall i \in \mathbb{N} \backslash \{n-1, n\} \tag{50a}$$

$$0.0 \le \Delta t_{i+1} \le T^{\max}(1 - yyy(i)) \qquad \forall i \in \mathbb{N} \backslash \{n\} \tag{50b}$$

$$\left.\begin{array}{r} 1 - yy^a(i) + \sum_{q \in \mathbb{Q}^a} z^a(q,q,i) \ge 1 \\ 1 - z^a(q,q,i) + yy^a(i) \ge 1 \qquad \forall q \in \mathbb{Q}^a \end{array}\right\} \qquad \forall i \in \mathbb{N} \backslash \{n\}, \forall a \in Aut \tag{51a}$$

$$\left.\begin{array}{r} yy^a(i) + (1 - yyy(i)) \ge 1 \qquad \forall a \in Aut \\ \left(\sum_{a \in Aut} (1 - yy^a(i))\right) + yyy(i) \ge 1 \end{array}\right\} \qquad \forall i \in \mathbb{N} \backslash \{n\} \tag{51b}$$

$$\left.\begin{array}{r} 1 - z^a(q,q',i) + y^a(q,i) \ge 1 \\ 1 - z^a(q,q',i) + y^a(q',i+1) \ge 1 \\ 1 - y^a(q,i) + 1 - y^a(q',i+1) + z^a(q,q',i) \ge 1 \end{array}\right\} \qquad \forall i \in \mathbb{N} \backslash \{n\}, \forall a \in Aut, \forall q,q' \in \mathbb{Q}^a \tag{51c}$$

We have now expressed the optimization problem of interest as an MILP and thus can apply any of several well-developed algorithms to solve it. The above MILP is finally expressed in the GAMS language, distribution 22.0, and solved with the CPLEX algorithm, version 9.1.

Figure 5 depicts the optimal trajectory when the objective is to minimize cost, $\Omega = C(t_n^e)$. This is easily put into the MILP form $C^e(n)$. With this objective, there is essentially no benefit to running the system. However, we are requiring it to run to the time point $(i,t) = (10, 500.0)$. Ideally, processes α and β could remain in their `off` and `lo` modes, where the operating costs are lower. Process α does so, but β has to switch into its `hi` mode because otherwise the material level would fall below 10.0, which is not allowed. The optimal solution obtained is $C(t_n^e) = 3537.1$.

5 Conclusions

Defining the novel LCCA framework and defining a procedure for converting models in this framework into MILP constraints have been the two main contributions of this paper. Our main example in the previous section demonstrates the value of the novel modeling framework introduced in this work. It is clear that the LCCA model is more intuitive than the GDP model and much simpler than the MILP model. At a glance, this is evident because the MILP model is much longer, but the true difficulty is even greater than this suggests. The constraints of the MILP are much more complex, and even experts would find it difficult to think of them directly.

Our LCCA modeling framework provides several features easing the declaration of discrete-continuous dynamical systems. Most importantly, component automata allow logical conditions involving continuous variables to force or prohibit discrete changes. They support time-based constraints intrinsically. For example, statements of the form "until some condition is satisfied, some event cannot occur" are naturally expressed with guards. For large systems, modular modeling is also crucial. LCCA supports this by requiring each automaton to maintain its own set of variables. Thus, a change to one automaton is unlikely to have any effect on other automata.

Several extensions would be valuable. Non-constant rates would generalize the continuous dynamics allowed, and discretization would still allow reformulation to an MILP. Ideally, the transformation procedure could be automated. The challenge here is to provide an input syntax for LCCA that could be parsed by a computer. The internal representation of both this syntax and the output MILP format would likely require an object-oriented design.

Acknowledgments

Many thanks to Robert Harper (Computer Science Department, Carnegie Mellon University) for valuable feedback on drafts of this paper.

Proof of Theorem 1

Let us first define some terminology. Given an implication $a \Rightarrow b$, a is called the *antecedent* and b the *consequent*. In a conjunctive constraint $a \wedge b$, each of a and b are called *conjuncts*. For indexed conjunction $\wedge_{i \in \mathbb{S}} g(i)$, each $g(i)$ is called a conjunct. Similarly the terms comprising a disjunctive constraint are called *disjuncts*.

The equivalence is proven by demonstrating implication in both directions.

First, we prove the forward implication: conj-impl $\Rightarrow$ disj-conj, i.e. assume conj-impl is true and show disj-conj must be true. An assumption of the theorem is that $g(i)$ holds for exactly one i. Without loss of generality, assume this value is i'. Rewrite conj-impl as

$$\underbrace{[g(i') \Rightarrow f(i')]}_{\text{lconj}} \wedge \underbrace{\bigwedge_{i \in \mathbb{S} \setminus \{i'\}} [g(i) \Rightarrow f(i)]}_{\text{rconj}}.$$

For conj-impl to be true, as is being assumed, both lconj and rconj must be true. First, we state the conditions under which these both hold:

- lconj is true if $f(i')$ is satisfied. This is because, by assumption, the antecedent $g(i')$ of lconj is true, and for the whole implication to be true, the consequent must be true.
- rconj is always true. This follows because we are assuming $g(i')$, and thus $g(i)$ is false for all $i \neq i'$. This means the antecedent $g(i)$ of every conjunct of rconj is false, making every conjunct true irrespective of the consequent $f(i)$ because falsehood can imply anything.

The net result is that $f(i')$ must be true. With this observation, it remains to show that disj-conj must be true. Divide up disj-conj similarly as

$$\underbrace{[g(i') \wedge f(i')]}_{\text{ldisj}} \vee \underbrace{\bigvee_{i \in \mathbb{S} \setminus \{i'\}} [g(i) \wedge f(i)]}_{\text{rdisj}}$$

For disj-conj to be true, either ldisj must be true or rdisj must be true. We show that ldisj is true. By assumption $g(i')$ is true, and we just argued that $f(i')$ must be true. So $[g(i') \wedge f(i')]$ is true.

Now, we prove the implication in the opposite direction: disj-conj $\Rightarrow$ conj-impl, i.e. show that conj-impl must be true under the assumption that disj-conj is true. disj-conj can be true if any one of its disjuncts is true. Let i' be the index of one of the true disjuncts, i.e. $[g(i') \wedge f(i')]$ is true. (In fact, this is the only disjunct that can be true because all others require $g(i)$ to hold for some $i \neq i'$, which violates the assumption of the theorem.) So we know that $f(i')$ must hold. Now, divide up conj-impl as above into lconj and rconj. It remains to show that both lconj and rconj are true. It was assumed that $g(i')$ holds and we just argued that $f(i')$ must hold; thus lconj follows immediately. rconj is vacuously true because $g(i)$ is false for all $i \neq i'$, making each conjunct of rconj true irrespective of $f(i)$.

References

Abdeddaim, Y. and Maler, O. (2001). Job-shop scheduling using timed automata, *in* G. Berry, H. Comon and A. Finkel (eds), *Proceedings of the13th International Conference on Computer Aided Verification, CAV 2001.*, Vol. 2102 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, Germany, pp. 478–492. 2

Abdeddaim, Y. and Maler, O. (2002). Preemptive job-shop scheduling using stopwatch automata, *Tools and Algorithms for the Construction and Analysis of Systems. 8th International Conference, TACAS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002*, Vol. 2280 of *Lecture Notes in Computer Science*, Springer-Verlag, Grenoble, France, pp. 113–126. 2

Agarwal, A. (2006). *Logical Modeling Frameworks for the Optimization of Discrete-Continuous Systems*, PhD thesis, Carnegie Mellon University. 8

Alur, R. and Dill, D. L. (1994). A theory of timed automata, *Theoretical Computer Science* **126**(2): 183–235. 2

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P. H., Nicollin, X., Olivero, A., Sifakis, J. and Yovine, S. (1995). The algorithmic analysis of hybrid systems, *Theoretical Computer Science* **138**(1): 3–34. 2

Asarin, E. and Maler, O. (1999). Optimal control for timed automata, *in* H. F. Chen, D. Z. Cheng and J. F. Zhang (eds), *14th World Congress of the International Federation of Automatic Control*, Vol. 6, Elsevier Science, Beijing, China, pp. 1–6. 2

Aubin, J. P., Lygeros, J., Quincampoix, M., Sastry, S. and Seube, N. (2002). Impulse differential inclusions: A viability approach to hybrid systems, *IEEE Transactions on Automatic Control* **47**(1): 2–20. 4

Avraam, M. P., Shah, N. and Pantelides, C. C. (1998). Modelling and optimisation of general hybrid systems in the continuous time domain, *Computers & Chemical Engineering* **22**: S221–S228. 12

Barton, P. I. and Pantelides, C. C. (1994). Modeling of combined discrete-continuous processes, *AIChE Journal* **40**(6): 966–979. 1

Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints, *Automatica* **35**(3): 407–427. 2

Bixby, R. (2002). Solving real-world linear programs: a decade and more of progress, *Operations Research* **50**(1): 3–15. 2

Cassez, F. and Larsen, K. (2000). The impressive power of stopwatches, *in* C. Palamidessi (ed.), *Proceedings of the 11th International Conference on Concurrency Theory, CONCUR 2000.*, Vol. 1877 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, Germany, pp. 138–152. 2

Chutinan, A. and Krogh, B. H. (2003). Computational techniques for hybrid system verification, *IEEE Transactions on Automatic Control* **48**(1): 64–75. 2

Cury, J. E. R., Krogh, B. H. and Niinomi, T. (1998). Synthesis of supervisory controllers for hybrid systems based on approximating automata, *IEEE Transactions on Automatic Control* **43**(4): 564–568. 2

Heemels, W. P. M. H., De Schutter, B. and Bemporad, A. (2001). On the equivalence of classes of hybrid dynamical models, *Proceedings of the 40th IEEE Conference on Decision and Control*, Vol. 1, IEEE, Orlando, FL, USA, pp. 364–369. 2

Henzinger, T. A. (1996). The theory of hybrid automata, *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, IEEE Comput. Soc. Press, pp. 278–292. 2

Kallrath, J. (2000). Mixed integer optimization in the chemical process industry - experience, potential and future perspectives, *Chemical Engineering Research & Design* **78**(A6): 809–822. 2, 5

Lee, C. K., Singer, A. B. and Barton, P. I. (2004). Global optimization of linear hybrid systems with explicit transitions, *Systems & Control Letters* **51**(5): 363–375. 2

Lygeros, J., Pappas, G. J. and Sastry, S. (1999). An introduction to hybrid system modeling, analysis, and control, *Preprints of the First Nonlinear Control Network Pedagogical School*, Athens, Greece, pp. 307–329. 4

Nicollin, X., Olivero, A., Sifakis, J. and Yovine, S. (1991). An approach to the description and analysis of hybrid systems, *in* R. L. Grossman, A. Nerode, A. P. Ravn and H. Rischel (eds), *Hybrid Systems Conference*, Springer-Verlag, Lyngby, Denmark, pp. 149–178. 2

Raman, R. and Grossmann, I. E. (1994). Modelling and computational techniques for logic based integer programming, *Computers & Chemical Engineering* **18**(7): 563–578. 5, 15

Stursberg, O., Panek, S., Till, J. and Engell, S. (2002). Generation of optimal control policies for systems with switched hybrid dynamics, *Modelling, Analysis, and Design of Hybrid Systems*, Vol. 279 of *Lecture Notes in Control and Information Sciences*, Springer-Verlag, Berlin, pp. 337–352. 2

Tomlin, C., Lygeros, J. and Sastry, S. (1998). Synthesizing controllers for nonlinear hybrid systems, *Hybrid Systems: Computation and Control*, Vol. 1386 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 360–373. 2

Torrisi, F. D., Bemporad, A. and Mignone, D. (2000). Hysdel — a tool for generating hybrid models, *Technical Report AUT00-03*, Automatic Control Lab, ETH. 3

Westerweele, M. R., Preisig, H. A. and Weiss, M. (1999). Concept and design of Modeller, a computer-aided modelling tool, *Computers & Chemical Engineering* **23**: S751–S754. 2

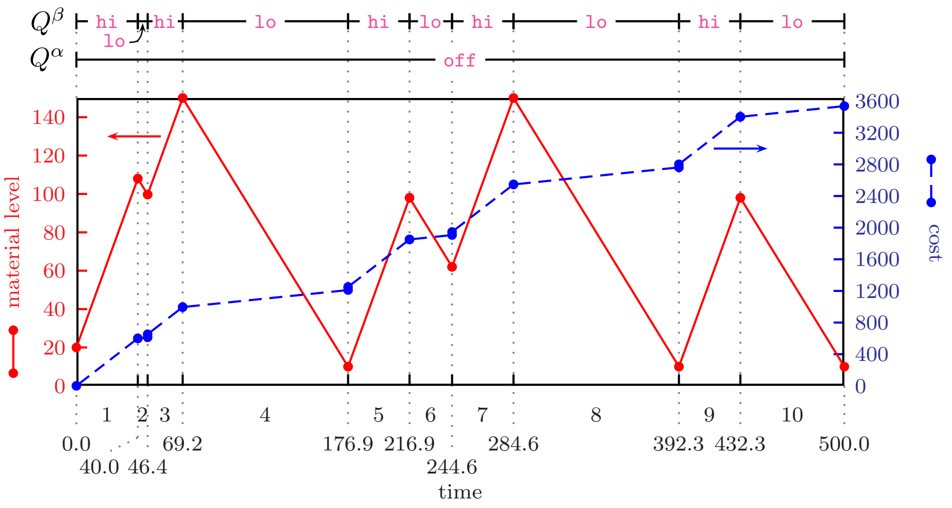


Figure 5: Optimal trajectory for minimizing cost.