IEEE Control Systems Letters paper presented at
2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Layered Control Systems Operating on Multiple Clocks

Inigo Incer, Noel Csomay-Shanklin, Aaron D. Ames, and Richard M. Murray

***Abstract*— Autonomous systems typically leverage layered control architectures, created by interconnecting components that operate at multiple timescales, i.e., evolve under various clocks. To formalize this typically heuristic procedure, we introduce a new logic, Multiclock Logic (MCL), that can express the requirements of components from the point of view of their local clocks, promoting independent design and component reuse. We then use assume-guarantee contracts expressed in MCL to prove global stability properties of a system using the stability properties of its components. In particular, we consider the classic layered architecture consisting of model predictive control (MPC) layered on top of feedback linearization, and prove overall stability of the systems.**

I. Introduction

Over the years, the field of control theory has developed many tools to design control blocks in isolation—e.g., PID, feedback linearization (FBL), model predictive control (MPC) [1], [3], control Lyapunov functions (CLFs) [2], [12], etc. The design of complex control systems—legged robots, aerial robots, and autonomous vehicles, to name a few—normally involves the combination of various blocks of control functionality. It is often the case that designers working independently on each control block make assumptions on the behaviors of other blocks that are not communicated or explicitly stated—leading to a development process prone to errors. Moreover, control blocks are normally implemented at different loop rates, leading to unaccounted for timing and interfacing issues. The end result is that we interconnect our control blocks *hoping the system will work*.

This letter provides a compositional, specification-based approach to proving control properties of layered systems by using the properties of each layer. We introduce a new logic, Multiclock Logic (MCL), to be able to express properties from the local point of view of a processor running on its own clock—capturing the multi-rate nature of layered control systems. We use MCL to abstract layers into assume-guarantee contracts, which are formal specifications that capture what a component guarantees and requires from its environment to be able to deliver its guarantees [6]. We use the algebra of contracts to compositionally relate the local stability properties of control algorithms with the system-level properties of their interconnection. While multi-rate control architectures were first studied in [10], the problem of compositionally writing and analyzing formal specifications for control systems made from layers that operate on distinct time bases has not been studied.

All authors are with the Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA 91125 (e-mail: inigo@caltech.edu; noelcs@caltech.edu; ames@caltech.edu; murray@cds.caltech.edu).

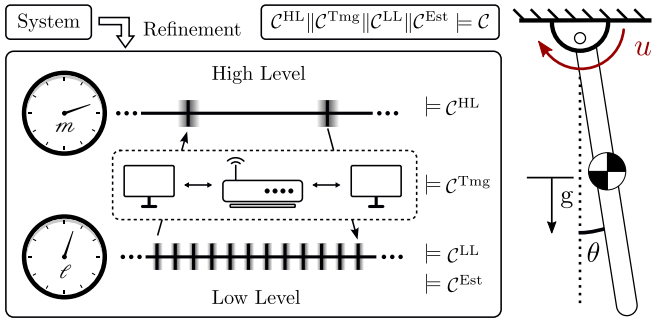

Fig. 1. A typical layered control architecture, whereby MCL contracts enable independently designed controllers to meet system-level specifications.

This letter is organized as follows. Section II introduces our motivating example. Section III presents the syntax and semantics of MCL. Section IV shows how to express contract specifications for a multilayer controller using MCL, and to prove system-level stability properties from the properties of the control layers. An extended version of this letter [7] contains background on behavioral modeling and contracts.

II. Multi-rate system case study

To illustrate our compositional analysis of a layered architecture, we will consider the design of a control strategy for the pendulum shown in Figure 1. Our objective will be for the system to satisfy a top-level stability objective expressed as contract $\mathcal{C}$. We will understand the controller for such as system as an architecture consisting of a high-level and a low-level control layers interconnected by a network. Each control layer will be assumed to be implemented on a processor operating on its own clock, as is common in practice. We will identify contracts $\mathcal{C}^{\mathrm{HL}}$ for the high-level layer, $\mathcal{C}^{\mathrm{LL}}$ for the low-level layer, $\mathcal{C}^{\mathrm{Est}}$ for state estimation, and $\mathcal{C}^{\mathrm{Tmg}}$ for the networking between the layers. As these contracts need to state formal properties over multiple clocks, they will be expressed in MCL in Section IV. The property of these contracts is that their composition is a refinement of the desired stability contract $\mathcal{C}$. This means that we will be free to independently implement these aspects of the functionality, and the procedure will be guaranteed to yield a system that satisfies the system-level objective $\mathcal{C}$. Afterwards, we will implement each of these control blocks independently.

We now consider the details of this case study. The configuration space of the system is given by $\theta \in \mathcal{Q} \triangleq \mathbb{S}^1$, and the associated state by $x = (\theta, \dot{\theta}) \in \mathsf{T}\mathcal{Q}$. We can write the dynamics in control-affine form as $\dot{x} = f(x) + g(x)u$, with control input $u \in \mathbb{R}$, continuously differentiable drift vector $f : \mathsf{T}\mathcal{Q} \to \mathbb{R}^2$, and actuation matrix $g : \mathsf{T}\mathcal{Q} \to \mathbb{R}^2$. We will be concerned with the high level task of reaching

some neighborhood of the goal state $x_g = 0$, the unstable upright equilibrium. Low-level controllers alone struggle to simultaneously enforce state and input constraints while ensuring progress to the goal is being made [3]—this motivates the use of a hierarchy for achieving this control task. Our hierarchy will consist of two layers—a high-level controller and a low-level controller—running on different clocks.

1) Low-level controller design: To actuate the system, we use a feedback controller which is able to track a desired trajectory x_d. We begin by defining an output $y\colon \mathcal{X} \to \mathbb{R}$ as $y(x) = \theta$, whereby we can construct error coordinates $e(x,t) = \begin{bmatrix} y(x) & L_f y(x) \end{bmatrix}^\intercal - \begin{bmatrix} y(x_d(t)) & \dot{y}(x_d(t)) \end{bmatrix}^\intercal$, with $L_f y(x)$ denoting the Lie derivative of y with respect to f. Then, a feedback control law can be created to stabilize the outputs. For example, the control law can be the feedback linearizing controller

$$k_{\text{fbl}}(x,t) = L_g L_f(x)^{-1}(-L_f^2 y(x) + \ddot{y}(x_d(t)) - Ke(x,t)),$$

which exponentially stabilizes the output coordinates for elementwise positive vector $K \in \mathbb{R}^2$. Given bounded additive disturbances to the dynamics, there exists a robust invariant set $\mathcal{E} \subset \mathbb{R}^n$ such that applying k_{fbl} results in bounded tracking error, i.e., $x(t) \in x_d(t) \oplus \mathcal{E}$ [4].

2) High-level controller design: This controller will be concerned with producing the pieces of trajectories x_d which make progress towards the goal x_g for the low-level controller to track. In order to do so, we set up the following MPC program:

$$\begin{aligned} \min_{x_d(t), u_d(t)} \quad & \int_0^T h(x_d(t), u_d(t))dt \qquad (1)\\ \text{s.t.} \quad & \dot{x}_d(t) = f^a(x_d(t)) + g^a(x_d(t))u_d(t),\\ & x_d(0) \in \hat{x} \oplus \mathcal{E}, \qquad x_d(T) = 0,\\ & x(t) \in \mathcal{X}, \qquad u(t) \in \mathcal{U}, \end{aligned}$$

with $\mathcal{X} \subset \mathbb{R}^2$ a state constraint set, $\mathcal{U} \subset \mathbb{R}$ an input constraint set, $h\colon \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{\geq 0}$ a convex stage cost, f^a and g^a linear approximations of the dynamics, $\hat{x} \in \mathcal{X}$ an estimate of the system state, $\mathcal{E} \subset \mathbb{R}^n$ a robust invariant set of the low level controller, and $x(t)$ and $u(t)$ the continuous-time state and input of the low-level system.

Combining MPC with a low-level controller is a popular technique which is extensible to stabilizing complex nonlinear systems [5], [10]. When deploying such an architecture, however, the underlying approximations of convexification and discretization are often not explicitly reasoned about, leaving practitioners hoping rather than knowing that the closed-loop system will work. Common assumptions include: *i*) the estimate of the state $\hat{x}$ is "close enough" to the true state x; *ii*) the MPC runs "fast enough" and produces solutions which are "close enough" to the system dynamics; and *iii*) the low-level controller runs "fast enough" and can track "well enough." The aim of this paper is to construct a pipeline whereby these assumptions are made explicit, with an eye towards automating the system verification process for layered control systems.

III. Multiclock logic

In this section, we introduce Multiclock Logic (MCL), a framework to reason about layered control operating on multiple clocks. First we define the behaviors over which the logic makes predicates, then introduce the logic. MCL will allow us to express properties from the local point of view of each control block in our design without burdening the syntax with synchronization over the multiple clocks present in the system. While some of these properties can be expressed intuitively, MCL is a formal language and so can be manipulated by a machine. This is necessary in order to automate formal system-level design.

1) System-level behaviors: The fundamental notion of system-modeling is the *variable*. We understand a variable as a tuple $(v, \mathcal{X}_v, \mathcal{D}_v)$, where v is the *name* for the variable, $\mathcal{X}_v$ is the *value space* (a topological space where the variable takes values), and $\mathcal{D}_v$ is a totally-ordered Abelian group called the *clock*, giving a notion of a progression of values. Whether a variable is discrete/continuous/constant depends on the clock that we assign to it: $\mathbb{R}/\mathbb{N}/\{\bullet\}$, respectively. We model clocks as totally-ordered Abelian groups because we need order to compare values of a clock and define intervals, and the group structure to add and subtract these values.

Suppose we have a set $\mathfrak{C}$ of clocks and that a set $\mathbf{Vars}$ of variables shares clock $c \in \mathfrak{C}$. The joint behaviors of these variables will be given by maps $c \to \prod_{v \in \mathbf{Vars}} \mathcal{X}_v$, as in the tagged signal model [9]. In order to isolate the value of a variable $u \in \mathbf{Vars}$ from those of the other variables in the same clock, we will use the projection maps that come with the definition of the product, i.e., $\pi_u\colon \prod_{v \in \mathbf{Vars}} \mathcal{X}_v \to \mathcal{X}_u$.

Stating certain predicates may require us to use a clock to read the values of variables belonging to another clock. Suppose $c, d \in \mathfrak{C}$. To read the value of a d-variable in the clock c, we will assume the existence of a map $\tau_c^d\colon c \to d$. The τ maps, which we may call *synchronization maps*, determine the index of the target clock that is used when reading variables from a given clock. E.g, to read the value of a variable x clocked by d from the clock c at time $t \in c$, we would get the value of x at time $\tau_c^d(t)$. The synchronization maps capture the amount of delay incurred by reading data from a clock from which the data does not originate.

Suppose that we build a system that comprises a set of variables $\mathbf{Vars}$ and a set of clocks $\mathfrak{C}$. For each variable v, we let $C(v)$ be the clock corresponding to $v \in \mathbf{Vars}$. The inverse of this map, $C^{-1}(c)$, yields the set of variables corresponding to clock $c \in \mathfrak{C}$.

Definition 1. A system-level behavior is an object $\beta = (b_d)_{d \in \mathfrak{C}} \times \left(\tau_c^d\right)_{c,d \in \mathfrak{C}}$, where the elements $b_d \in \left(d \to \prod_{v \in C^{-1}(d)} \mathcal{X}_v\right)$ carry the behaviors of the variables clocked by d, while the $\tau_c^d \in (c \to d)$ indicate how to read values of d-variables using the clock c.

Components, as usual, are defined as sets of behaviors. The following definition will be useful to give semantics to the eventual modality $\Diamond$ of the logic.

Definition 2. Let $\beta = (b_d)_{d\in\mathfrak{C}} \times \left(\tau_d^{d'}\right)_{d,d'\in\mathfrak{C}}$ be a system behavior. Given $t \in c \in \mathfrak{C}$, we define the (c,t)-execution β_c^t as $\beta_c^t = (b_d)_{d\in\mathfrak{C}} \times \left(\tilde{\tau}_d^{d'}\right)_{d,d'\in\mathfrak{C}}$, where $\tilde{\tau}_d^{d'}(x) = \begin{cases} \tau_c^{d'}(x+t) & d = c \\ \tau_d^{d'}(x) & d \neq c \end{cases}$.

The only difference between β and β_c^t is that in β_c^t the clock c is anticipated by t units.

2) MCL syntax: Assume we have access to a set $\mathbf{Vars}$ of variables, a set $\mathfrak{C}$ of clocks, and a set $\mathfrak{F}$ of formulas of various arities. We let $\mathfrak{P}$ be the set of symbols $d^{d'}$, where d and d' denote clocks. The syntax of MCL is

$$\phi ::= \rho c.\ \Phi \mid \neg\phi \mid \phi \wedge \psi$$
$$\Phi ::= P(\lambda_1(t_1), \ldots, \lambda_n(t_n)) \mid \neg\Phi \mid \Phi \wedge \Psi \mid \Diamond_{[t_1,t_2]}\Phi \mid \Diamond_{t_1}\Phi,$$

where $c \in \mathfrak{C}$, $\lambda_i \in \mathbf{Vars} \cup \mathfrak{P} \cup \mathfrak{C}$, $t_i \in c$, and $P \in \mathfrak{F}$ is an n-ary formula.

We think of the formulas ϕ as the global syntax of MCL, and of the Φ formulas as the local syntax that applies to a clock. The global syntax supports propositional logic. The syntax $\phi = \rho c.\ \Phi$ indicates that a global formula ϕ is created by binding a local formula Φ to the clock c. The local syntax supports propositional logic and the eventual modality. It also supports the enunciation of predicates involving multiple variables evaluated at the indicated times $t_i \in c$. The syntax also supports the enunciation of predicates on clocks $c \in \mathfrak{C}$ and on *clock pairs* $c^d \in \mathfrak{P}$. The use of clocks and clock pairs $c^d \in \mathfrak{P}$ in formulas allows us to express timing constraints in our systems. Observe that the logic leaves abstract the formulas $\mathfrak{F}$ used to define the predicates. The formulas are functions that evaluate to Boolean values.

3) MCL semantics: The semantics of an MCL formula apply to a system behavior. Let $\beta = (b_d)_{d\in\mathfrak{C}} \times \left(\tau_d^{d'}\right)_{d,d'\in\mathfrak{C}}$ be a system behavior. We have the following global semantics:

- $\beta \models \rho c.\ \Phi$ iff $\beta \models_c \Phi$
- $\beta \models \neg\phi$ iff $\beta \not\models \phi$
- $\beta \models \phi \wedge \psi$ iff $\beta \models \phi$ and $\beta \models \psi$

Formulas at the global level are either created by Boolean connectives, or they are local formulas that are assigned to a clock in the system. The symbol "$\models_c$" stands for satisfaction in the local semantics of MCL, defined as follows:

- $\beta \models_c P(\lambda_1(t_1), \ldots, \lambda_n(t_n))$ iff $P\left(\texttt{Interp}(\lambda_i, t_i)\right)_{i=1}^n$, where
$$\texttt{Interp}(\lambda_i, t_i) = \begin{cases} \pi_v \circ b_{C(v)} \circ \tau_c^{C(v)}(t_i) & \lambda_i = v \in \mathbf{Vars} \\ \tau_c^d(t_i) & \lambda_i = d \in \mathfrak{C} \\ \tau_d^{d'} \circ \tau_c^d(t_i) & \lambda_i = d^{d'} \in \mathfrak{P} \text{ and } d \neq c \end{cases}$$
- $\beta \models_c \neg\Phi$ iff $\beta \not\models_c \Phi$
- $\beta \models_c \Phi \wedge \Psi$ iff $\beta \models_c \Phi$ and $\beta \models_c \Psi$
- $\beta \models_c \Diamond_{[t_1,t_2]}\Phi$ iff $\exists t.\ (t_1 \leq t \leq t_2) \wedge (\beta_c^t \models_c \Phi)$
- $\beta \models_c \Diamond_{t_1}\Phi$ iff $\exists t.\ (t_1 \leq t) \wedge (\beta_c^t \models_c \Phi)$

In addition to the formation of local formulas using Boolean connectives and the eventual modality $\Diamond$, MCL local formulas can be formed by making a predicate over a set of symbols λ_i and a set of values t_i of the clock c. In general, the meaning of the formula $P(\lambda_1(t_1), \ldots, \lambda_n(t_n))$ is the evaluation of the n-ary formula P for the values that each symbol λ_i takes when the value of the clock c is $\tau_c^c(t_i)$. For example, the predicate $\rho c.\ \|x(0)\| > 2$ for a variable x means that we will evaluate the value of x when $c = \tau_c^c(0)$. If x is clocked by c, we simply extract the value of x at time $\tau_c^c(0)$; if it is clocked by $d \neq c$, we would return its value at time $\tau_c^d(0)$. We recall that the local semantics can anticipate the value of c by t time units—see Definition 2.

The formula $\rho c.\ d(0) - K > T$ for $d \in \mathfrak{C}$ and constants $T, K \in d$ has the semantics $\tau_c^d(0) - K > T$. This statement is true at ticks of clock c for which the index of clock d that c uses to read the data of d is larger than $K + T$. This type of predicate can be used as a precondition that ensures that the clock d has issued sufficient ticks. The semantics of a formula of the form $\rho c.\ r(0) - d^r(0) < T$ for clocks c, d, r, is $\tau_c^r(0) - \tau_d^r \circ \tau_c^d(0) < T$. It states that the difference in r time units between a tick of c and the tick of d from which c reads the data of d must be less than T. This kind of statement is useful to bound the maximum allowed time for data to have existed in its local clock d before it is read by another clock c.

The notion of satisfaction is extended from system behaviors to components as follows: we say that a component M satisfies an MCL formula ϕ if $\beta \models \phi$ for all $\beta \in M$.

We extend the given syntax and semantics in the standard way to support all Boolean connectives and the modality $\Box$, or "globally." Finally, when stating a formula of the form $\rho c.\ P(\lambda_i(t_i))_i$, we will sometimes omit the t_i arguments. In that case, the parameter should be understood as 0.

IV. Verifying the multi-rate case study

We now continue the analysis of the archetypal layered architecture described in Section II. Our goal is to specify the high-level and low-level control blocks independently and carry out system-level analysis using these specifications. The objective of our system-level analysis is to show that the system reaches and remains inside a neighborhood of the goal state x_g. To do so, we assume we have a cost function $\mathcal{V}$ that maps the state x of the system to a well-ordered set. The value of $\mathcal{V}$ is zero in a neighborhood of x_g. The function $\mathcal{V}$ will be further specified when we consider the implementations of component contracts in Section IV-C. As a well-order does not have infinite descending chains, the fact that the codomain of $\mathcal{V}$ is a well-order means that any process that decreases the cost $\mathcal{V}$ will eventually reach the minimum cost $\mathcal{V}(x) = 0$ in a finite number of iterations.

The system has three clocks: $\mathscr{m} = \mathbb{N}$, which runs the high-level block; $\mathscr{l} = \mathbb{N}$, which runs the low-level and estimation blocks; and $\mathscr{r} = \mathbb{R}_{\geq 0}$, the physical time. The system state is denoted by the variable $(x, \mathcal{X}, \mathscr{r})$, i.e., its behaviors are functions from the physical clock to the state space $\mathcal{X}$. The high-level block outputs a variable $(x_d, \mathcal{X}^{\mathscr{r}}, \mathscr{m})$, which contains the trajectory in the state space that the system is to follow. Observe that the behaviors of this variable are functions from $\mathscr{m}$ to functions from $\mathscr{r}$ to $\mathcal{X}$. Thus, at any tick of the clock $\mathscr{m}$, the high-level controller will provide a function for the low-level controller. This means that x_d

will be doubly-indexed in our formulas. The first index corresponds to the clock evaluation, and the second to the time argument of the trajectory. For example, the ℓ formula $\rho\ell.\ \|x_d(0)(T)\| < K$ evaluates to $\|x_d\left(\tau_\ell^m(0)\right)(T)\| < K$. Finally, an estimation block running on clock ℓ will output a state estimate $(\hat{x}, \mathcal{X}, \ell)$. Our system's objective is $\rho\ell.\ \Diamond\Box\left(\mathcal{V}(x) = 0\right)$, i.e., our cost will stabilize at zero.

A. Specifying the system

We consider the contracts for each layer in the system.

1) High-level controller: We consider the specification of the high-level controller running on a dedicated processor with clock m. The high layer has an input $\hat{x}$, the state estimate coming from another control layer, and a single output x_d, which is a trajectory that the low-level controller has to follow.

On a given tick of the clock m, the high-level block will assume that the state estimate is an accurate approximation of the state. The high-level controller uses this to compute a trajectory in the next tick of m. Finally, the high layer will guarantee that the trajectories it generates either decrease the cost $\mathcal{V}$ or keep it equal to zero. We can represent the high-level controller contract $\mathcal{C}^{\text{HL}} = (A^{\text{HL}}, G^{\text{HL}})$ as follows:

$$
\begin{aligned}
A^{\text{HL}}\colon\ & \phi^{\text{HL}}_{\text{A\_init}} \wedge \phi^{\text{HL}}_{\text{timing}} \wedge \phi^{\text{HL}}_{\text{sensor}} \wedge \phi^{\text{HL}}_{\text{bound\_var}} \\
& \phi^{\text{HL}}_{\text{A\_init}}\colon \rho m.\ \texttt{Close}\left(x, x_i; \delta^{\text{HL}}_{\text{A\_init}}\right) \\
& \phi^{\text{HL}}_{\text{timing}}\colon \rho m.\ \Box\begin{pmatrix}(T^m_{\min} \le r(1) - r \le T^m_{\max}) \wedge \\ (r - \ell^r < T^m_{\text{fresh}})\end{pmatrix} \\
& \phi^{\text{HL}}_{\text{sensor}}\colon \rho m.\ \Box\texttt{Close}\left(\hat{x}, x; \delta^{\text{HL}}_{\text{sensor}}\right) \\
& \phi^{\text{HL}}_{\text{bound\_var}}\colon \rho m.\ \Box\texttt{BoundedVariation}(x; D_x) \\
G^{\text{HL}}\colon\ & \phi^{\text{HL}}_{\text{G\_init}} \wedge \phi^{\text{HL}}_{\text{rsp\_dyn}} \wedge \phi^{\text{HL}}_{\text{dynamics}} \wedge \phi^{\text{HL}}_{\text{traj\_bound\_var}} \wedge \phi^{\text{HL}}_{\text{progress}} \\
& \phi^{\text{HL}}_{\text{G\_init}}\colon \rho m.\ \texttt{Close}\left(x_d(0)(0), x_i; \delta^{\text{HL}}_{\text{G\_init}}\right) \\
& \phi^{\text{HL}}_{\text{rsp\_dyn}}\colon \rho m.\ \Box\texttt{RespectDynamics}(x_d) \\
& \phi^{\text{HL}}_{\text{dynamics}}\colon \rho m.\ \Box\texttt{Close}\left(x_d(0)(T^m_{\text{avg}}), x_d(1)(0); \delta^{\text{HL}}_{\text{dynamics}}\right) \\
& \phi^{\text{HL}}_{\text{traj\_bound\_var}}\colon \rho m.\ \Box\texttt{BoundedVariation}(x_d; D_d) \\
& \phi^{\text{HL}}_{\text{progress}}\colon \rho m.\ \Box\begin{pmatrix}(\mathcal{V}\left(x_d(0)(0)\right) > \mathcal{V}\left(x_d(1)(0)\right)) \vee \\ \Box\left(\mathcal{V}\left(\texttt{Inflate}\left(\texttt{Im}\left(x_d\right); \delta^{\text{HL}}_{\text{progress}}\right)\right) = 0\right)\end{pmatrix}
\end{aligned}
$$

$\Phi^{\text{HL}}_{\text{sensor}}$ captures the requirement that the sensor produces values that are close to the real state x at the time when m ticks. The predicate $\texttt{Close}(v, v'; \delta)$ indicates that two symbols v, v' are evaluated to quantities that are close up to a parameter δ for some notion of distance (taken to be ℓ_2 for the case study). $\Phi^{\text{HL}}_{\text{bound\_var}}$ assumes that the state x has bounded variation. $\Phi^{\text{HL}}_{\text{timing}}$ requires the clock period of m to lie between $T^m_{\min}$ and $T^m_{\max}$; (we also assume that the nominal m period T^m_{avg} lies between these bounds); the assumption also requires the difference in physical time between a tick of m and the tick of ℓ from which m reads ℓ's data to be less than T^m_{fresh}. This means that the data from ℓ that is read from m cannot be too old. $\Phi^{\text{HL}}_{\text{A\_init}}$ requires x to be close to a value x_i at the beginning of the system execution.

With respect to guarantees, $\Phi^{\text{HL}}_{\text{G\_init}}$ means that the first trajectory output by the high-level block will start close to x_i. $\Phi^{\text{HL}}_{\text{rsp\_dyn}}$ is a predicate stating that the high-level controller will always generate trajectories x_d that should be within the competence of the low-level block to follow. In the context of this work, we require that the produced trajectories are dynamically feasible, i.e., there exists a feedback controller which is able to track the generated trajectories. $\Phi^{\text{HL}}_{\text{dynamics}}$ states that the beginning of every new trajectory provided by the high-level block has to be close to the value of the previous trajectory at time T^m_{avg}. $\Phi^{\text{HL}}_{\text{traj\_bound\_var}}$ is a promise that the trajectories provided by the high-level controller will have bounded variation.

Finally, in order to be able to promise that the system will make progress towards its objective, the high-level block will make use of the cost function $\mathcal{V}$. The high-level controller makes a guarantee $\Phi^{\text{HL}}_{\text{progress}}$ which says that either the cost at the beginning of a trajectory x_d is larger than the cost at the beginning of the next trajectory x_d, or that all points that are close to the trajectory have a cost of zero. The function $\texttt{Inflate}(A, \delta)$ takes a set A and returns the set of all points that are δ-close to any point of A. That is, each trajectory either improves the cost or stays fixed at cost equal to zero.

2) Low-level controller: The low-level controller takes as inputs trajectories x_d that the system's state x has to follow and promises that it can make the system follow these trajectories with a given accuracy. We have the following contract $\mathcal{C}^{\text{LL}} = (A^{\text{LL}}, G^{\text{LL}})$ for the low-level controller:

$$
\begin{aligned}
A^{\text{LL}}\colon\ & \phi^{\text{LL}}_{\text{timing}} \wedge \phi^{\text{LL}}_{\text{rsp\_dyn}} \wedge \phi^{\text{LL}}_{\text{dynamics}} \wedge \phi^{\text{LL}}_{\text{bound\_var}} \\
& \phi^{\text{LL}}_{\text{timing}}\colon \rho\ell.\ \Box\begin{pmatrix}\left(T^\ell_{\min} \le r(1) - r \le T^\ell_{\max}\right) \wedge \\ \left(r - m^r < T^\ell_{\text{fresh}}\right)\end{pmatrix} \\
& \phi^{\text{LL}}_{\text{rsp\_dyn}}\colon \rho\ell.\ \Box\texttt{RespectDynamics}(x_d) \\
& \phi^{\text{LL}}_{\text{dynamics}}\colon \rho\ell.\ \Box\begin{pmatrix}(m \ne m(-1)) \wedge (m \ge 0) \\ \rightarrow \texttt{Close}\left(x_d(0)(0), x; \delta^{\text{LL}}_{\text{dynamics}}\right)\end{pmatrix} \\
& \phi^{\text{LL}}_{\text{bound\_var}}\colon \rho\ell.\ \Box\texttt{BoundedVariation}(x; D_x) \\
G^{\text{LL}}\colon\ & \phi^{\text{LL}}_{\text{upd}} \wedge \phi^{\text{LL}}_{\text{tracking}} \\
& \phi^{\text{LL}}_{\text{upd}}\colon \rho\ell.\ \Box\begin{pmatrix}\textbf{if}\ \ (m \ne m(-1)) \wedge (m \ge 0)\ \ \textbf{then} \\ (\texttt{upd} = \ell)\ \ \textbf{else}\ \ (\texttt{upd} = \texttt{upd}(-1))\end{pmatrix} \\
& \phi^{\text{LL}}_{\text{tracking}}\colon \rho\ell.\ \Box\begin{pmatrix}(\ell - \texttt{upd} > 0) \rightarrow \\ \texttt{Close}\left(x, x_d(0)\left(T^\ell_{\text{avg}}(\ell - \texttt{upd})\right); \delta^{\text{LL}}_{\text{tracking}}\right)\end{pmatrix}
\end{aligned}
$$

The assumptions of this contract are as follows. $\phi^{\text{LL}}_{\text{timing}}$ assumes that ℓ's period is bounded below and above, and that the data read from clock m is not too old. $\phi^{\text{LL}}_{\text{rsp\_dyn}}$ requires trajectories received from m to respect certain physical limits, i.e., that they satisfy the state and input constraints used in the analysis of the low level controller. $\phi^{\text{LL}}_{\text{dynamics}}$ makes sure that when ℓ detects the generation of a new trajectory x_d from m, then the starting point of that trajectory should be close to the value of the state. $\phi^{\text{LL}}_{\text{bound\_var}}$ requires the state x to have bounded variation.

Regarding guarantees, $\phi^{\text{LL}}_{\text{upd}}$ is a helper statement that defines the variable $\texttt{upd}$. This variable contains the last value of ℓ when a new trajectory was received from m. $\phi^{\text{LL}}_{\text{tracking}}$ guarantees that the low-level controller will make the system follow the given trajectory x_d. Observe that $\phi^{\text{LL}}_{\text{tracking}}$ is enforced for all values of the trajectory x_d, except its first point. For the first point of the trajectory, the low level controller makes the assumption $\phi^{\text{LL}}_{\text{dynamics}}$ on the state. T^ℓ_{avg} is the nominal period of ℓ and respects the bounds of $\phi^{\text{HL}}_{\text{timing}}$.

3) Estimator: The estimator will guarantee that the state estimates are always accurate on the clock ticks of ℓ, yielding the contract $\mathcal{C}^{\text{Est}} = (A^{\text{Est}}, G^{\text{Est}})$, where $A^{\text{Est}}\colon \rho\ell.\ \texttt{True}$ and $G^{\text{Est}}\colon \phi^{\text{Est}}_{\text{sensor}}$, with $\phi^{\text{Est}}_{\text{sensor}}\colon \rho\ell.\ \square\texttt{Close}\left(\hat{x}, x; \delta^{\text{Est}}_{\text{sensor}}\right)$.

4) Timing design: We understand the timing component of the system as "network design" in the sense that enforcing timing constraints is implemented by applying networking and clock synchronization technologies. This component will have the following contract $\mathcal{C}^{\text{Tmg}} = (A^{\text{Tmg}}, G^{\text{Tmg}})$:

$$A^{\text{Tmg}}\colon \rho r.\ \texttt{True}$$
$$G^{\text{Tmg}}\colon \phi^{\text{Tmg}}_{\ell\text{-timing}} \wedge \phi^{\text{Tmg}}_{m\text{-timing}}$$
$$\phi^{\text{Tmg}}_{\ell\text{-timing}}\colon \rho\ell.\ \square\left(\begin{pmatrix} T^{\ell}_{\text{min}} \le r(1) - r \le T^{\ell}_{\text{max}} \end{pmatrix} \wedge \begin{pmatrix} r - m^{r} < T^{\ell}_{\text{fresh}} \end{pmatrix}\right)$$
$$\phi^{\text{Tmg}}_{m\text{-timing}}\colon \rho m.\ \square\left(\begin{pmatrix} (T^{m}_{\text{min}} \le r(1) - r \le T^{m}_{\text{max}}) \wedge \\ (r - \ell^{r} < T^{m}_{\text{fresh}}) \end{pmatrix}\right)$$

B. System-level analysis

We have component-level contracts for all layers in our system. The following result connects these specifications with the desired top-level stability objective.

Theorem 1. *Consider a two-layer control system formed by a high-level controller and a low-level controller. If each layer satisfies contracts $\mathcal{C}^{HL}$ and $\mathcal{C}^{LL}$, respectively, the estimator satisfies $\mathcal{C}^{Est}$, and their interconnection satisfies $\mathcal{C}^{Tmg}$, then the top-level system will satisfy the top level contract (A, G), where*

$$A\colon \phi^{HL}_{A\_init} \wedge \phi^{HL}_{bound\_var} \quad G\colon \rho\ell.\ \Diamond\square\left(\mathcal{V}(x) = 0\right), \tag{2}$$

provided that the following conditions hold:

$$\delta^{Est}_{sensor} + T^{m}_{fresh} D_x \le \delta^{HL}_{sensor} \tag{3}$$
$$\delta^{HL}_{G\_init} + \delta^{HL}_{A\_init} + D_x T^{\ell}_{fresh} \le \delta^{LL}_{dynamics} \tag{4}$$
$$\delta^{LL}_{tracking} + \delta^{HL}_{dynamics} + (T^{m}_{fresh} + T^{\ell}_{fresh}) D_x + D_d \Delta T^{m} \le \delta^{LL}_{dynamics} \tag{5}$$
$$\delta^{LL}_{dynamics} \le \delta^{HL}_{progress}, \tag{6}$$

where $\Delta T^{m} = T^{m}_{max} - T^{\ell}_{avg} \left\lfloor \frac{T^{m}_{min} - \left(T^{m}_{fresh} + T^{\ell}_{fresh}\right)}{T^{\ell}_{\max}} \right\rfloor$.

Proof. Contract composition yields the specification formed by interconnecting the components whose specifications we have available. Our system-level specification is $\mathcal{C}^{\text{Sys}} = \mathcal{C}^{\text{HL}} \parallel \mathcal{C}^{\text{LL}} \parallel \mathcal{C}^{\text{Est}} \parallel \mathcal{C}^{\text{Tmg}}$, which we compute by applying the `ContractComposition` routine of Algorithm 1 of [8]. We now verify how this composition yields a system satisfying the assumptions of all components.

a) High-level controller: We consider the assumptions: $\phi^{\text{HL}}_{\text{init}}$ is an assumption on the initial state that we need to conserve at the system-level. $\phi^{\text{HL}}_{\text{bound\_var}}$ is also a system-level assumption. $\phi^{\text{HL}}_{\text{timing}}$ is satisfied by $\phi^{\text{Tmg}}_{m\text{-timing}}$ of our timing design. $\phi^{\text{HL}}_{\text{sensor}}$ needs to be analyzed. We observe that[1]

$$\frac{\phi^{\text{Est}}_{\text{sensor}} \quad \phi^{\text{Tmg}}_{m\text{-timing}} \quad \phi^{\text{HL}}_{\text{bound\_var}}}{\rho m.\ \texttt{Close}\left(\hat{x}, x; \delta^{\text{Est}}_{\text{sensor}} + T^{m}_{\text{fresh}} D_x\right).}$$

We need the relation (3) to hold in order to satisfy $\phi^{\text{HL}}_{\text{sensor}}$.

[1]This notation means that the formula below the horizontal bar is a valid deduction from the conjunction of the formulas on top of the bar.

b) Low-level controller: $\phi^{\text{LL}}_{\text{timing}}$ is satisfied by $\phi^{\text{Tmg}}_{\ell\text{-timing}}$ of the timing design. $\phi^{\text{LL}}_{\text{bound\_var}}$ is an assumption on the dynamics of the physical system and should therefore be a system-level assumption. $\phi^{\text{LL}}_{\text{rsp\_dyn}}$ is satisfied by the guarantee $\phi^{\text{HL}}_{\text{rsp\_dyn}}$. The satisfaction of $\phi^{\text{LL}}_{\text{dynamics}}$ requires an inductive argument. For the proof that relation (4) is required for $\phi^{\text{LL}}_{\text{dynamics}}$ to hold in the initial case and that $\phi^{\text{LL}}_{\text{dynamics}}$ holds inductively when (5) is true, please see Section 6.2.2 of the extended version of this letter [7].

c) Progress: Our analysis indicates that the composition of all contracts so far defined yields a situation in which all contracts have their assumptions met—provided the system parameters meet conditions (3), (4), and (5). Now we verify whether the system makes progress towards its goal. We observe that the fact that $\mathcal{V}$ takes values in a well order means that we can carry out the following deductions:

$$\frac{\dfrac{\phi^{\text{HL}}_{\text{progress}}}{\rho m.\ \Diamond\square\left(\mathcal{V}\left(\texttt{Inflate}\left(\texttt{Im}(x_d); \delta^{\text{HL}}_{\text{progress}}\right)\right) = 0\right)} \quad \phi^{\text{LL}}_{\text{upd}} \quad \phi^{\text{LL}}_{\text{tracking}} \quad \phi^{\text{LL}}_{\text{dynamics}}}{\rho\ell.\ \Diamond\square\left(\mathcal{V}(x) = 0\right)}$$

provided that (6) holds. □

C. Component-level verifications

Theorem 1 shows that if we implement control layers adhering to the specifications introduced in Section IV-A and satisfying constraints (3)-(6), the system will satisfy the desired stability property (2). These constraints impose timing requirements in the system. For instance, (5) imposes a limit on the maximum compute time of the high-level controller. This means that constraints (3)-(6) can be used to increase the robustness of the design by ensuring that they are satisfied with margin. Now we verify that each implementation of our control layers satisfies its own contract.

1) Low-level controller: In order to use the methods presented in [4] which combine a low level controller with (1), we begin by showing that applying zero order held inputs, i.e., with $\rho\ell.\ \square\left(\bar{u} = k_{\text{fbl}}(x, \ell - \texttt{upd})\right)$, results in bounded exogenous disturbance to the error dynamics. For the following discussion, let $x'(t)$ denote the solution to the system dynamics with continuous time control applied, and $x(t)$ the solution with zero order held inputs. Plugging in the solution to the differential equation with $x'(0) = x(0)$ and adding and subtracting $g(x'(\tau))\bar{u}$ yields

$$\|x(t) - x'(t)\| = \int_0^t \frac{d}{dt}(x(\tau) - x'(\tau))d\tau \tag{7}$$
$$\le \int_0^t (L_f + L_g\|\bar{u}\|)\|x(\tau) - x'(\tau)\| + \|g(x'(\tau))\|\|u(\tau) - \bar{u}\|d\tau,$$

where L_f and L_g represent the local Lipschitz constants of the drift vector and actuation matrix, respectively. From the perspective of the low level controller, we assume via $\phi^{\text{LL}}_{\text{rsp\_dyn}}$ that $\|u(t)\| \le U$ and that $x(t) \in \mathcal{X}$, a compact set. These will be necessary for the analysis, and will be explicitly enforced in the proposed MPC formulation. From these constraints, we know that there exists a $G > 0$ such that $\|g(x'(\tau))\| < G$ and that all local Lipschitz constants are global over $\mathcal{X}$. Combining these facts and using the Bellman-Gronwall Lemma [11] leads to $\|x(t) - x'(t)\| \le UGt^2 e^{(L_f + L_g U)t} \triangleq t\rho(t)$,

where $\rho \in \mathcal{K}_\infty$, a class $\mathcal{K}$ infinity function. Next, let $e'(t) \triangleq x'(t) - x_d(t)$ denote the error dynamics with continuous time control applied, whereby the feedback linearizing controller k_{fbl} yields $\dot{e}'(t) = A_{\text{cl}}e'$ for A_{cl} a stable matrix. Taking $e(t) = x(t) - x_d(t)$ to be the error with zero order held inputs, we have $\dot{e} = w(t) + A_{\text{cl}}e$, where $w(t) = \frac{d}{dt}(x - x') + A_{\text{cl}}(x' - x)$.

Plugging in the terms developed in (7) results in $\|w(t)\| \leq \|w(T^{\ell}_{\text{max}})\| \leq T^{\ell}_{\text{max}}((L_f + 2L_gU + \|A_{\text{cl}}\|)\rho(T^{\ell}_{\text{max}}) + GU)$. As this bound is a composition of class-$\mathcal{K}$ functions in time, for all $\delta_{\text{w}} > 0$ there exists an $\epsilon > 0$ such that $T^{\ell}_{\text{max}} < \epsilon$ results in $\|w(t)\| \leq \delta_{\text{w}}$. Fixing an allowable δ_{w} and thereby upper bounding T^{ℓ}_{max}, by integrating the error dynamics and using the comparison lemma we have $\|e(t)\| \leq \|x(0) - x_d(0)\|Me^{-\lambda t} + T^{\ell}_{\text{max}}\delta_{\text{w}}$, for some $M, \lambda > 0$ as determined by the convergence rate of the low level controller. Therefore, in order to produce the guarantee of $\phi^{\text{LL}}_{\text{tracking}}$, we must enforce that $\delta^{\text{LL}}_{\text{dynamics}}Me^{-\lambda T^{\ell}_{\text{min}}} + T^{\ell}_{\text{max}}\delta_{\text{w}} \leq \delta^{\text{LL}}_{\text{tracking}}$.

2) MPC: Lemma 2 in [4] demonstrates that the trajectories produced by (1) are dynamically feasible, i.e., able to be exactly tracked via the feedback linearizing controller k_{fbl}, implying satisfaction of $\phi^{\text{HL}}_{\text{rsp\_dyn}}$. As the MPC program produces solutions to a linear system with bounded state and control inputs, we have that the resulting desired trajectory x_d will have bounded variation, satisfying $\phi^{\text{HL}}_{\text{traj\_bound\_var}}$.

Next, we ensure that the above MPC program is recursively feasible, which requires showing that the enforced set $\mathcal{E}$ is a robust invariant for the system. Choosing $\mathcal{E}$ such that $B\left(0, \delta^{\text{LL}}_{\text{tracking}}\right) \subset \mathcal{E}$ results in a set that can be rendered invariant by the low level controller, even during sampling. Therefore, we choose $\mathcal{E}$ to meet this condition as well as the design requirements imposed by the system analysis in Section IV-B. We then appeal to the results in [4] to prove recursive feasibility of the MPC algorithm used therein.

Finally, let $V \colon \mathcal{X} \to \mathbb{R}$ denote the sum of the running and terminal cost of MPC, often used as a Lyapunov function in stability proofs. As we have effectively transformed our nonlinear MPC program to a linear one, we can use standard MPC results [3] to state that $\rho m.\ (V(x_d(1)(0)) < V(x_d(0)(0)))$. Since the guarantees of the MPC contract involve $\mathcal{V}$, which takes values in a well-order, we can define $\mathcal{V}$ by quantizing the function V. Then we can choose a vicinity around x_g and define $\mathcal{V}(p) = 0$ for $p \in \mathcal{E}$. In that case, $\mathcal{V}$ takes values in a well-order, and the MPC block satisfies $\phi^{\text{HL}}_{\text{progress}}$.

3) Simulation results: We investigate the use of MCL contracts as a design tool towards achieving stability and state constraint satisfaction for an architecture consisting of a high-level and a low-level control block designed in isolation. In both cases, the same control architecture and time delay of $T^{\ell}_{\text{fresh}} = T^{m}_{\text{max}} = T^{m}_{\text{min}}$ was used, i.e., the time delay is equal to one m clock cycle, and the m clock does not have jitter. In the first case shown in Figure 2a, the MPC and FBL controllers were independently implemented, but their interconnection was not verified against formal specifications. Therefore, our analysis did not yield any guarantees of system behavior, and indeed we find that our system violates the desired state constraints $x(t) \in \mathcal{X}$. In Figure 2b, the margin $\mathcal{E}$ was modified such that (5) holds with equality. As a result of adhering to the MCL contracts, the composite control hierarchy maintains the desired system-level specification in (2) despite having independently designed components.

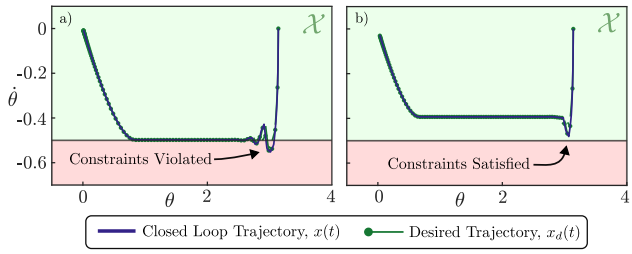

Fig. 2. a) Nominal controller whose design process did not use MCL contract verification fails to meet system objectives. b) Control blocks which adhere to MCL contracts satisfy the constraint, despite being designed in isolation.

V. Concluding Remarks

We considered an effective way of specifying control layers running on multiple clocks using MCL. Once the specifications were expressed, we used MCL contracts to prove system-level stability properties using the local properties of each layer. Our framework is extensible to systems with additional layers. We showed in the case study how to verify that each control block satisfies its own specification. Our simulation results showed that a violation of the system-level constraints led to failure. Future work involves applications to more complex systems, such as bipedal robots.

References

[1] F. Allgower, R. Findeisen, and Z. K. Nagy. Nonlinear model predictive control: From theory to application. *J.-Chinese Institute Of Chemical Engineers*, 35(3):299–316, 2004.

[2] Z. Artstein. Stabilization with relaxed controls. *Nonlinear Analysis: Theory, Methods & App.*, 7(11):1163–1173, 1983.

[3] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[4] N. Csomay-Shanklin, A. J. Taylor, U. Rosolia, and A. D. Ames. Multi-Rate Planning and Control of Uncertain Nonlinear Systems: Model Predictive Control and Control Lyapunov Functions, Mar. 2022. arXiv:2204.00152.

[5] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter. Perceptive locomotion through nonlinear model predictive control, Aug. 2022. arXiv:2208.08373.

[6] I. Incer. *The Algebra of Contracts*. PhD thesis, EECS Department, University of California, Berkeley, May 2022.

[7] I. Incer, N. Csomay-Shanklin, A. Ames, and R. M. Murray. Specifying and analyzing networked and layered control systems operating on multiple clocks. *arXiv:2402.11666*, 2024.

[8] I. Incer et al. Pacti: Scaling assume-guarantee reasoning for system analysis and design. *arXiv:2303.17751*, 2023.

[9] E. Lee and A. Sangiovanni-Vincentelli. A framework for comparing models of computation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(12):1217–1229, 1998.

[10] U. Rosolia and A. D. Ames. Multi-rate control design leveraging control barrier functions and model predictive control policies. *IEEE Control Systems Letters*, 5(3):1007–1012, 2021.

[11] S. Sastry. Mathematical Background. In S. Sastry, editor, *Nonlinear Systems: Analysis, Stability, and Control*, Interdisciplinary Applied Mathematics, pages 76–126. Springer, New York, NY, 1999.

[12] E. D. Sontag. A 'universal' construction of Artstein's theorem on nonlinear stabilization. *Sys. & Control Let.*, 13(2):117–123, 1989.