A Metric Temporal Logic Specification Interface for Real-Time Discrete-Event Control

Amrith Dhananjayan and Kiam Tian Seow

***Abstract*—In supervisory control of timed discrete-event systems (TDES's), a conceptually well-founded finitary control synthesis framework is developed, and it requires specifications to be prescribed as finite (-trace) automata in the form of timed transition graphs (TTG's). However, prescribing real-time specifications as TTG's is a non-trivial task that must be resolved before the formal framework could expect to become widely used. In addressing this specification problem, metric temporal logic (MTL) is proposed in this paper as a control specification language for use with the TTG-based control synthesis framework. MTL is a designer-friendly formalism due to its human or natural language expressiveness and readability, and is well-suited for specifying real-time control specifications. In automating TTG prescription, this paper proposes an MTL interface to the control synthesis framework. The interface is proved to be a correct and complete translation algorithm that converts finitary control specifications written in state-based MTL formulae for a given TDES model into deterministic finite TTG's. Integrated, the MTL interface and the control synthesis framework combine the human expressiveness and readability of MTL with the algorithmic computability of TTG's, and together provide a new and convenient MTL specification-based design tool for automated prescription of TTG's and real-time control synthesis. Illustrative examples demonstrate the utility of the MTL interface.**

***Index Terms*—Timed discrete-event systems, timed transition graph, metric temporal logic, formal specification, supervisory control**

I. Introduction

Supervisory control of discrete-event systems proposed by Ramadge and Wonham [1], [2], along with its many extensions [3], [4], [5], [6], provides an effective formal framework to model and control complex systems. The logical framework has been applied to a wide variety of applications such as manufacturing systems [7], [8], [9] and intelligent transportation [10], to name a few. For handling real-time systems, Brandin and Wonham [11] have augmented the framework with timing features for supervisory control of timed discrete-event systems (TDES's), providing an effective framework for real-time control synthesis that has found application in domains such as task scheduling [12], [13], multiprocessor resource allocation [14] and transformer voltage control [15].

In the Brandin-Wonham framework [11], [16], [17], a TDES to be controlled is modeled as a finite (-trace) automaton in the form of a timed transition graph (TTG) that encompasses all the possible timed execution sequences of the system. The TTG control system model formulation is useful as it is able to capture the time urgency of control with the control concept of forcible events [11], and a rich set of control-theoretic results [16], [17] has been developed for the model with control synthesis tool support (e.g., TTCT [18]). As a design paradigm, the automated control-theoretic tool is attractive as the supervisor (or controller) for a TDES that it synthesizes is guaranteed to conform to a given control specification in a nonblocking and maximally permissive fashion. By nonblockingness, every initiated task is allowed to run to completion; and by maximal permissiveness, the largest set of feasible TDES execution sequences within the specification are allowed. Besides, research is being actively pursued to evolve TTG-based control synthesis into a practical foundation of greater realism (e.g., [19], [20], [21], [22], [23], [24]). Essentially, the synthesis also requires the specification to be given as a TTG.

In practice, control requirements are initially described in natural language statements; human designers understand these statements and prescribe TTG specifications accordingly. However the task of prescribing in finite automata that TTG's are is non-trivial [25], and must be resolved before the supervisory control framework [11] could expect to become widely used, motivating the use of alternative specification languages that are more human or natural language expressive and readable [4]. Language expressiveness refers to the ease of description and readability refers to the fluency of expression for clarity that aids in understanding a given specification described in the language. Moreover, most designers would find it easier to write and understand a specification in a descriptive language such as *temporal logic*, rather than in a prescriptive one that *automata* is [26]. Adding a temporal logic translation interface to the TTG control synthesis framework can mitigate the difficulty of specifying directly in TTG's, as it would support the alternate writing of a class of control requirements as temporal logic formulae that the interface can convert to TTG specifications for algorithmic control synthesis. That writing in temporal logic is generally easier and clearer is due to the natural language readability and expressiveness of temporal logic furnished by its temporal operators that define a fragment of the English language [27]. This practical utility of temporal logic as a designer specification language is intuitively evident, and is supported by the experience of the temporal logic research community [28], [29]. However, to the best of our knowledge, there is relatively little or no prior research work that, in the context of a finite-state TDES model, translates temporal logic formulae to TTG's to harness the specification benefit of temporal logic for TTG control synthesis.

Metric temporal logic (MTL) [30] is well recognized as a human designer-friendly formalism for writing real-time specifications [31]. In this paper, this version of temporal logic

This work was supported by the Singapore Ministry of Education, under NTU-AcRF Tier 1 Grant No: RG65/07.

The authors are with the School of Computer Engineering, Nanyang Technological University, Republic of Singapore 639798. {amri0005,asktseow}@ntu.edu.sg

is proposed as the real-time specification language for use with the TTG-based control synthesis framework [11], [3]. MTL is chosen because its underlying timing semantics can be readily adapted to match the TTG model in terms of qualitatively defining the passage of time as ticks of the global clock. Apart from offering simple syntax and semantics for descriptively writing qualitative specifications that are paraphrastic in natural language, MTL also allows timing constraints to be concurrently specified using its time-bounded temporal operators. In automating TTG prescription for TTG-based control synthesis, an MTL interface is proposed. This interface is a correct and complete translation algorithm which converts finitary control specifications, easily expressed in state-based MTL formulae, to deterministic (finite) TTG specifications. State-based MTL formulae express control requirements in terms of the state information of TDES's. Integrated, the MTL interface and the real-time control synthesis framework combine the expressiveness and readability of MTL with the computability of TTG's, and provide a convenient MTL specification-based design tool for automated prescription of TTG's for real-time control synthesis. Importantly, while the TTG-based control synthesis tool guarantees producing supervisors that synthesize "the specifications right", the interface fills the gap of assisting designers to prescribe "the right specifications". It helps that recent research efforts [32], [33] allow MTL specifications of common real-time requirements to be obtained directly from descriptions given in structured natural language (English). These techniques, which have been successfully applied in several real-world applications [34], [35], [36], have made it possible even for non-specialist designers to write MTL specifications for realistic systems [37].

In a related work [31], an algorithm is proposed to directly translate control requirements given as MTL formulae to automata that realize controllers satisfying these requirements. However, the control setting considered therein is fundamentally different from that of the Brandin-Wonham framework that our MTL interface is developed for. In that setting [31], control concepts such as controllability and observability do not arise naturally, uncontrollable events are characterized by nondeterminism, and the output controller may not be maximally permissive.

In some early [38], [39] efforts, timed automata [40] is used to model TDES's for supervisory control. Timed automata is a popular dense time model for real-time systems [40] and unlike TTG's [9], the notion of time used is continuous and not discrete-event. It should be noted that with the notion of time as tick events, a TTG (system) model is directly amenable to supervisory control since supervisory control actions are event-based. In real-time system modeling and analysis, some researchers prefer to use timed automata over their purely discrete-event counterparts such as TTG's. However, being time continuous, the more general timed automaton model is not amenable to supervisory control without time eventization [38], which entails the problem of converting timed automata to finite transition systems that is PSPACE-hard [40], [41].

The theoretical and algorithmic development of the proposed MTL interface for automated prescription of TTG's is decidedly a contribution to the TTG-based control systems literature. We believe that the MTL interface for TTG-based control synthesis would add value to the design framework as it adapts to the common background of TDES control theorists and practitioners by not making too many changes in their traditional research and practice of designing controllers; for example, the TTG modeling formalism that they are familiar with is not changed. Motivated as it is by the specification needs of control designers, the translation ability of the MTL interface could in turn influence the control designers' thinking.

The rest of the paper is organized as follows. Section II reviews the relevant background in TDES's and MTL. Section III presents the new MTL translation algorithm, and establishes its correctness and completeness. Section IV demonstrates the utility of the translation algorithm as a specification interface with three examples. Section V concludes the paper.

II. Background

A. Timed Discrete-Event Systems (TDES's)

We review the TDES control framework proposed by Brandin and Wonham [11]. In this framework, a TDES is modeled as a timed transition graph (TTG), which is a finite automaton displayed as an edge-labeled directed graph such that its nodes denote states of the TDES and edges denote timed transitions.

The base model of a TTG is a finite automaton $G_{act} = (A_{act}, \Sigma_{act}, \delta_{act}, a_0, A_m)$ called an *activity transition graph* (ATG). The ATG describes the untimed behavior of a TTG. In ATG G_{act}, A_{act} is the finite set of activities, Σ_{act} is the finite alphabet of events, $\delta_{act} : \Sigma_{act} \times A_{act} \to A_{act}$ is the (partial) activity transition function, $a_0 \in A_{act}$ is the initial activity and $A_m \subseteq A_{act}$ is the set of marked activities.

Let $\mathbf{N}$ represent the set of nonnegative integers. Each event $\sigma \in \Sigma_{act}$ is associated with a *lower time bound* $l_\sigma \in \mathbf{N}$ and an *upper time bound* $u_\sigma \in \mathbf{N} \cup \{\infty\}$ such that $l_\sigma \leq u_\sigma$. Σ_{act} is partitioned into two subsets, the set $\Sigma_{spe} = \{\sigma \in \Sigma_{act} \mid u_\sigma \in \mathbf{N}\}$ of prospective events and the set $\Sigma_{rem} = \{\sigma \in \Sigma_{act} \mid u_\sigma = \infty\}$ of remote events. The upper time bounds of remote events are infinite while that of prospective events are finite. For modeling purposes l_σ would typically denote a delay and u_σ, a hard deadline.

For $\sigma \in \Sigma_{act}$ let

$$T_\sigma = \begin{cases} [0, u_\sigma], & \text{if } \sigma \in \Sigma_{spe} \\ [0, l_\sigma], & \text{if } \sigma \in \Sigma_{rem} \end{cases} .$$

T_σ is said to be the *timer interval* for σ.

Formally, a transition system

$$\mathcal{G} \overset{\text{def}}{=} [\Psi, G_{act}]$$

is an ATG with state information incorporated. Here Ψ denotes the finite set of propositional state symbols of $\mathcal{G}$ such that the domain $Range(u)$ over which each $u \in \Psi$ ranges is $\{true, false\}$. The activity set A_{act} is the Cartesian product of the ranges of the state symbols in Ψ (i.e., $A_{act} \overset{\text{def}}{=} \prod_{u\in\Psi}\text{Range}(u)$), so that for any activity $a \in A_{act}$, we denote the value of $u \in \Psi$ assigned by a to be $a[u]$ over its domain. The state information in $a \in A_{act}$ is uniquely characterized by the formula $\prod_{u\in\Psi}(u = a[u])$. Henceforth in this paper, for notational convenience, Ψ is implicitly assumed and we simply use symbol G_{act} to refer to the ATG of any TTG with state information.

A TTG $G = (Q, \Sigma, \delta, q_0, Q_m)$ models a TDES. The model has the lower and upper time bounds of events incorporated into its transition structure. The state set Q is defined as $Q = A_{act} \times \prod\{T_\sigma \mid \sigma \in \Sigma_{act}\}$, such that a state $q \in Q$ is an element of the form $q = (a, \{t_\sigma \mid \sigma \in \Sigma_{act}\})$, where $a \in A_{act}$ and $t_\sigma \in T_\sigma$. For each $q \in Q$, the component t_σ is called the *timer* of σ in q. Intuitively, each system state is characterized by an activity and a timer value for each event in Σ_{act}. Let $Act(q)$ denote the activity of state q, i.e., for $q = (a, \{t_\sigma \mid \sigma \in \Sigma_{act}\})$, $Act(q) = a$. The set of events Σ is defined as $\Sigma = \Sigma_{act} \cup \{tick\}$, where the additional event $tick$ is used to represent the advancement of one unit of time. The initial state $q_0 \in Q$ is defined as $q_0 := (a_0, \{t_{\sigma,0} \mid \sigma \in \Sigma_{act}\})$, such that $t_{\sigma,0} = \begin{cases} u_\sigma, & \text{if } \sigma \in \Sigma_{spe} \\ l_\sigma, & \text{if } \sigma \in \Sigma_{rem} \end{cases}$. The set $Q_m \subseteq Q$ of marked states is given by a subset of $A_m \times \prod\{T_\sigma \mid \sigma \in \Sigma_{act}\}$, comprising a marked activity with suitable assignment of the timers. The state transition function $\delta : \Sigma \times Q \to Q$ is defined as follows. $\delta(\sigma, q)$ is defined for any $q = (a, \{t_\sigma \mid \sigma \in \Sigma_{act}\}) \in Q$ and $\sigma \in \Sigma$, written $\delta(\sigma, q)!$, if and only if any of the following three conditions holds.

1) $\sigma = tick$ and $(\forall\tau \in \Sigma_{spe}) t_\tau > 0$;
2) $\sigma \in \Sigma_{spe}$, $\delta_{act}(\sigma, a)!$, and $0 \le t_\sigma \le u_\sigma - l_\sigma$;
3) $\sigma \in \Sigma_{rem}$, $\delta_{act}(\sigma, a)!$, and $t_\sigma = 0$.

We write $\neg\delta(\sigma, q)!$ to denote that $\delta(\sigma, q)!$ is not defined.

Whenever $\delta(\sigma, q)!$, an entrance state $q' = (a', \{t'_\tau \mid \tau \in \Sigma_{act}\})$ such that $\delta(\sigma, q) = q'$ is defined as follows:

- when $\sigma = tick$, $a' = a$ and $\forall\tau \in \Sigma_{act}$,
$$t'_\tau = \begin{cases} t_\tau - 1, & \text{if } \delta_{act}(\tau, a)! \text{ and } t_\tau > 0 \\ t_\tau, & \text{otherwise} \end{cases};$$
- when $\sigma \in \Sigma_{act}$, $a' = \delta_{act}(\sigma, a)$, $t'_\sigma = t_{\sigma,0}$ and $\forall\tau \in \Sigma_{act}$ such that $\tau \ne \sigma$,
$$t'_\tau = \begin{cases} t_\tau, & \text{if } \delta_{act}(\tau, a')! \\ t_{\tau,0}, & \text{otherwise} \end{cases}.$$

Note that the function δ is deterministic in the sense that for $\sigma_1, \sigma_2 \in \Sigma$ defined at any given state $q \in Q$ (i.e., $\delta(\sigma_1, q)!$ and $\delta(\sigma_2, q)!$), $\sigma_1 = \sigma_2$ implies $\delta(\sigma_1, q) = \delta(\sigma_2, q)$. Consequently TTG G is said to be deterministic.

In TTG's, the event $tick$ is used to represent timing behavior. Denoting the duration of an event $\sigma \in \Sigma$ as $\varphi(\sigma)$, we have
$$\varphi(\sigma) = \begin{cases} 0, & \text{if } \sigma \in \Sigma_{act} \\ 1, & \text{if } \sigma = tick \end{cases}.$$

Let Σ^* be the set containing all finite strings of events in Σ, including the empty string ε. A string t' is considered a *prefix* of t'', if there exists some string s such that $t's = t''$. A *language* L over Σ is a subset of Σ^*. A language L_1 is said to be a *sublanguage* of language L_2 if $L_1 \subseteq L_2$. The *prefix closure* of a language L, denoted by $\bar{L}$, is the language consisting of all prefixes of its strings. As each string s in Σ^* is a prefix of itself, we have $L \subseteq \bar{L}$. A language L is *prefix-closed* if $L = \bar{L}$.

We describe the behavior of G by its prefix-closed language $L(G)$ and marked language $L_m(G)$. Formally,
$$\begin{aligned} L(G) &= \{s \in (\Sigma)^* \mid \delta(s, q_0)!\}, \\ L_m(G) &= \{s \in L(G) \mid \delta(s, q_0) \in Q_m\}. \end{aligned}$$
By definition, $L_m(G) \subseteq L(G)$ is the subset of strings in $L(G)$ which end in any of the states in Q_m, and is a distinguished subset or sublanguage. By designer choice, the set Q_m represents completed tasks (or sequences of tasks) carried out by the system that the model G is intended to represent.

A state $q \in Q$ is said to be *reachable* if $(\exists s \in (\Sigma)^*)$ $\delta(s, q_0) = q$, and *coreachable* if $(\exists s \in (\Sigma)^*)$ $\delta(s, q) \in Q_m$. G is considered *reachable* if all its states are reachable, and *coreachable* if all its states are coreachable, i.e., $\overline{L_m(G)} = L(G)$. G is said to be *trim* if it is both reachable and coreachable. If G is not trim, then a trim automaton, denoted by $Trim(G)$, can be computed by deleting from G every state that is either not reachable or not coreachable.

B. Metric Temporal Logic (MTL)

The control specification language adopted in this paper is MTL [30], [31]. In MTL, temporal operators have timing constraints associated with them, supporting the specification of quantitative temporal requirements that impose time deadlines on the behavior of TDES's.

1) Syntax: MTL formulae are constructed from a finite set of propositional symbols $\mathcal{P}$; the Boolean connectives $\neg$ (not) and $\wedge$ (and); and the temporal connectives $\bigcirc_{\sim t}$ ($next$), $\Box_{\sim t}$ ($always$) and $U_{\sim t}$ ($until$), where $\sim$ denotes $<, \le, >$ or $\ge$ and t is a non-negative integer. The following formula formation rules apply.

1) every propositional symbol $p \in \mathcal{P}$ is an MTL formula;
2) if ω, ω_1 and ω_2 are MTL formulae, so are $\neg\omega$, $\bigcirc_{\sim t}\omega$, $\Box_{\sim t}\omega$, $\omega_1 U_{\sim t} \omega_2$ and $\omega_1 \wedge \omega_2$.

The following equivalences ($\equiv$) are also used in addition to these basic rules to define related connectives $\vee$ (or), $\to$ ($implies$), and operator $\Diamond_{\sim t}$ ($eventually$):
$$\omega_1 \vee \omega_2 \equiv \neg(\neg\omega_1 \wedge \neg\omega_2), \tag{1}$$
$$\omega_1 \to \omega_2 \equiv \neg\omega_1 \vee \omega_2, \tag{2}$$
$$\Diamond_{\sim t}\omega \equiv true\ U_{\sim t}\ \omega. \tag{3}$$

Propositional constants $true$ and $false$ are defined, respectively, by the equivalences
$$true \equiv \neg\omega \vee \omega,$$
$$false \equiv \neg\omega \wedge \omega.$$

If a timing constraint $\sim t$ is associated with a temporal connective, then the modal formula should hold within a time period that is satisfied by the relation $\sim t$. For example, an MTL formula $\Box_{\le t}\ \omega$ is read as "always ω in the closed time interval $[0, t]$".

2) Semantics: A finite string of events over an event set Σ can be considered as a mapping
$$e : \{0, 1, \cdots, j, \cdots, \cdots\} \to \Sigma$$

such that

$$e \overset{\text{def}}{=} e(0)e(1)\cdots e(j)\cdots, \quad e(j) \in \Sigma.$$

Then, we can say that e is an event string generated by G if there is a "labeling" ρ of the string by states of G

$$\rho : \{0, 1, \cdots, j, \cdots, \cdots\} \to Q$$

such that

$$\rho \overset{\text{def}}{=} \rho(0)\rho(1)\cdots\rho(j)\cdots, \quad \rho(j) \in Q$$

for which

1) $\rho(0) = q_0$;
2) $\rho(j+1) = \delta(e(j), \rho(j))$.

Such a finite labeling ρ is called a state trajectory of G. The j-suffix of ρ, denoted by $\rho^{(j)}$, is a finite trajectory

$$\rho(j)\rho(j+1)\cdots\rho(i)\cdots, \; j \geq 0.$$

We have $\rho^{(0)} = \rho$. The j-prefix of ρ, denoted by ρ_j, is the finite state trajectory

$$\rho(0)\rho(1)\cdots\rho(j), \; j \geq 0.$$

The state trajectory ρ_j is marked if $\rho(j) \in Q_m$.

MTL formulae expressed over a given TDES G are interpreted over models of the form $(\rho, \pi, \mathcal{T})$, where

$$\pi : \{0, 1, \cdots, j, \cdots, \cdots\} \times \mathcal{P} \to \{false, \; true\}$$

is a binary function evaluating propositional symbol $p \in \mathcal{P}$ in j-th state $\rho(j)$, i.e.,

$$\pi(j, p) = \begin{cases} true, & \text{if } p \text{ holds in } Act(\rho(j)) \\ false, & \text{otherwise} \end{cases},$$

and $\mathcal{T} : \{0, 1, \cdots, j, \cdots, \cdots\} \to \{0, 1, \cdots, j, \cdots, \cdots\}$ is a monotonic function which assigns the time stamp $\mathcal{T}(j)$ to position j.

If a propositional symbol p holds (i.e., is evaluated to be $true$) at $\rho(j)$, we simply write $\models^{\rho(j)} p$, with the model $(\rho, \pi, \mathcal{T})$ considered to be understood. If $\rho^{(j)}$ satisfies an MTL formula ω, we write $\models^{\rho^{(j)}} \omega$. Note that the evaluations of a propositional symbol p in the j-th state and over a j-suffix ρ^j are logically equivalent, i.e., $\models^{\rho(j)} p \; \equiv \models^{\rho^{(j)}} p$.

Along with the standard rules for Boolean connectives, MTL uses the following rules for temporal operators to establish the satisfaction of a suffix state trajectory over an MTL formula. Given MTL formulae ω, ω_1 and ω_2, a state index j, and a propositional symbol p, we have

- $\models^{\rho^{(j)}} p$ iff $\pi(j, p) = \; true$;
- $\models^{\rho^{(j)}} \bigcirc_{\sim t}\omega$ iff $\mathcal{T}(j+1) \sim \mathcal{T}(j) + t$ and $\models^{\rho^{(j+1)}} \omega$;
- $\models^{\rho^{(j)}} \Box_{\sim t}\omega$ iff for all $k, k \geq j$, $\models^{\rho^{(k)}} \omega$ whenever $\mathcal{T}(k) \sim \mathcal{T}(j) + t$;
- $\models^{\rho^{(j)}} \omega_1 U_{\sim t} \omega_2$ iff there exists $k, k \geq j$, such that $\mathcal{T}(k) \sim \mathcal{T}(j) + t$ and $\models^{\rho^{(k)}} \omega_2$ and for all $l, j \leq l \leq k$, $\models^{\rho^{(l)}} \omega_1$ whenever $\mathcal{T}(l) \sim \mathcal{T}(j) + t$.

3) Expansion Rules: An MTL formula can be assessed over a state trajectory as a $present$ condition that holds in the current state of the trajectory and a $future$ condition that must hold in the next state of the trajectory. This is formalized using the $\bigcirc_d$-formula, where d represents the time elapsed when a state transition occurs in a state trajectory and has a strictly non-negative real value. More formally, a subformula in the expansion having $\bigcirc_d$ as the main operator is a $future$ condition that should hold in the next state occurring d units of time later. The semantics of this formula is

$$\models^{\rho^{(j)}} \bigcirc_d \omega \text{ iff } \mathcal{T}(j+1) - \mathcal{T}(j) = d \text{ and } \models^{\rho^{(j+1)}} \omega.$$

The expansion of MTL formulae is based on the equivalences (4) through (8).

4) Disjunctive Expanded Form: An MTL formula is said to be in disjunctive expanded form (DEF) if it is expressed by a disjunction of a finite number of subformulae, i.e., of the form $\bigvee_j (present_j \wedge \bigcirc_d future_j)$, where $present_j$ is a conjunction of literals[1] and $future_j$ a conjunction of literals and formulae having $\bigcirc$, $\Box$ or U as the main connective. An MTL formula is transformed to its equivalent DEF by applying equivalences (4)-(8), the usual distributive laws between Boolean connectives and the following equivalence:

$$\bigcirc_d(\omega_1 \wedge \omega_2) \equiv (\bigcirc_d\omega_1 \wedge \bigcirc_d\omega_2). \tag{9}$$

Subsequently in this paper, we will note the expansion of a formula ω with respect to (w.r.t) a state transition of time duration d as

$$\bigvee_j f^j \wedge \; \bigcirc_d \overline{\omega}^j.$$

Let $Pre(\omega, d)$ denote the set containing the f^j terms in the expansion of an MTL formula ω w.r.t a state transition of time duration d and $Fut(f^j)$ denote the $future$ condition corresponding to f^j, i.e., for a DEF $\bigvee_j f^j \wedge \; \bigcirc_d \overline{\omega}^j$, we have $Fut(f^j) = \overline{\omega}^j$.

5) Finitary Control Requirements: Clearly, an MTL specification ω for a TDES G (for finitary nonblocking control) is to restrict the behavior of G to a marked sublanguage [4]. In other words, ω "selects" marked finite state trajectories ρ_m of G such that $\models^{\rho_m} \omega$ (read as ρ_m satisfies ω). Essentially, this would require removing those (finite) state trajectories of G that violate ω. Such a formula belongs to the bounded response class [42], [43] of MTL. In other words, a control requirement that we specify in MTL for finitary control of a TDES is necessarily a bounded response formula.

Remark 1. The apparent restriction to finitary control requirements, which represents a large part of what designers are interested in [42], is not as constraining as one would imagine [44], especially in the case of real-time systems [43]. Liveness or infinitary control requirements, specified for non-terminating behaviors, would typically require that something should "eventually" occur without stating an upper time bound for the occurrence. Consider, for example, a requirement that some process should terminate eventually. This would also include situations where process termination occurs after a limitless (and impractical) period of time. While this is a liveness requirement, specifying that the termination should occur within (say) 100 units of time is a more relevant (finitary) control requirement.

[1]A literal is a propositional symbol or its negation.

$$\bigcirc_{\sim t}\omega \equiv \begin{cases} \bigcirc_d \ \omega, & \text{if } d \sim t \\ false, & \text{otherwise} \end{cases} \tag{4}$$

$$\Box_{\leq t}\omega \equiv \begin{cases} \omega \wedge \bigcirc_d \Box_{\leq t-d} \ \omega, & \text{if } d \leq t \\ \omega, & \text{otherwise} \end{cases} \tag{5}$$

$$\Box_{\geq t}\omega \equiv \begin{cases} \bigcirc_d \Box_{\geq t-d} \ \omega, & \text{if } d \leq t \text{ and } t \neq 0 \\ \bigcirc_d \Box_{\geq 0} \ \omega, & \text{if } d \geq t \text{ and } t \neq 0 \\ \omega \wedge \bigcirc_d \Box_{\geq 0} \ \omega, & \text{if } t = 0 \end{cases} \tag{6}$$

$$\omega_1 U_{\leq t} \omega_2 \equiv \begin{cases} \omega_2 \vee (\omega_1 \wedge \neg\omega_2 \wedge \bigcirc_d \ \omega_1 \ U_{\leq t-d} \ \omega_2), & \text{if } d \leq t \\ \omega_2, & \text{otherwise} \end{cases} \tag{7}$$

$$\omega_1 U_{\geq t} \omega_2 \equiv \begin{cases} \bigcirc_d \ \omega_1 \ U_{\geq t-d} \ \omega_2, & \text{if } d \leq t \text{ and } t \neq 0 \\ \bigcirc_d \ \omega_1 \ U_{\geq 0} \ \omega_2, & \text{if } d > t \text{ and } t \neq 0 \\ \omega_2 \vee (\omega_1 \wedge \neg\omega_2 \wedge \bigcirc_d \ \omega_1 \ U_{\geq 0} \ \omega_2), & \text{if } t = 0 \end{cases} \tag{8}$$

6) Conjunctive Normal Form: An MTL formula is said to be in normal form, if every constituent subformula (within the scope of their outermost temporal operator) in its DEF does not contain liveness modalities [45]. It is said to be positive if only its propositional symbols are negated. Any MTL formula can be transformed to its equivalent positive form by using De Morgan's laws and the following equivalences:

$$\neg(\omega_1 U_{\sim t} \omega_2) \equiv (\Box_{\sim t} \neg\omega_2) \vee (\neg\omega_2 U_{\sim t}(\neg\omega_1 \wedge \omega_2)), \tag{10}$$

$$\neg(\Box_{\sim t}\omega) \equiv \Diamond_{\sim t}\neg\omega, \tag{11}$$

$$\neg \bigcirc_{\sim t} \omega \equiv (\bigcirc_{\sim t}\neg\omega) \vee \bigcirc_{\not\sim t} true, \tag{12}$$

where $\not\sim$ is used to denote the converse[2] of the ordering relation $\sim$ [45].

The positive normal form is a conveniently recognizable structure of a bounded response MTL formula and is expressed using temporal modalities $\bigcirc_{\sim t}$, $\Box_{\sim t}$, $U_{<t}$ or $U_{\leq t}$. Finally, an MTL formula in positive normal form with $\Box_{\sim t}$ as its outermost operator is said to be in invariance normal form.

In what follows, we consider a class of MTL formulae representing (finitary) control requirements in conjunctive normal form (CNF). A CNF is defined as a conjunction of j subformulae, $j \geq 1$, where each subformula is in positive normal form. Because the bounded response class is closed under conjunction [46], a CNF is a bounded response formula. Also, a CNF is said to be in invariance normal form if all its subformulae are in invariance normal form.

III. Translation and Analysis

In this section, we present the development and analysis of a translation algorithm as a specification interface to the real-time (TTG) control synthesis framework. The algorithm automatically prescribes a TTG specification H for a TDES G from an MTL specification ω in CNF specifying a finitary control requirement. The basic operation in the algorithm is to compute a trim TTG that generates a marked sublanguage of TDES G, obtained by retaining all strings in $L_m(G)$ whose corresponding state trajectories satisfy ω.

[2]The converse of a relation is obtained by switching the order of the elements in the relation. The converse of ordering relations $\leq$, $<$, $>$, and $\geq$ are $\geq$, $>$, $<$, and $\leq$, respectively

A. Procedure $Expand$

For an input MTL specification ω in CNF, Procedure $Expand$ returns the expansion of ω in DEF w.r.t a state transition of time duration d.

```
Procedure Expand(ω, d)
Input: An MTL formula $\omega$ (in CNF) and time duration $d$;
Output: An expansion of $\omega$ w.r.t $d$ (in DEF);
1 begin
2     Expand $\omega$ w.r.t $d$ using equivalences given in Section
      II-B3. Do logic arithmetic on the expanded formula
      to reduce it to a finite number of terms of the form
      $f^j \wedge \bigcirc_d\overline{\omega}^j$, where $f^j$ is a state formula such that
      the conjunction of any two such formulae is $false$
      and $\overline{\omega}^j$ is an MTL formula;
3     return $\bigvee_j f^j \wedge \bigcirc_d\overline{\omega}^j$;
```

A CNF has the following property.

Property 1. *Given an MTL formula ω in CNF and time duration d. For $f^i, f^j \in Pre(\omega, d)$ such that $f^i \neq f^j$, we have $f^i \wedge f^j = false$.*

The satisfaction of this property is facilitated by logic arithmetic (including the use of appropriate validity assertions) and expansion rules given in Section II-B3.

B. Translation Algorithm

We now present our translation algorithm that translates an MTL specification ω in CNF for a TDES $G = (Q, \Sigma, \delta, q_0, Q_m)$ to a TTG specification H.

The MTL specification ω expanded w.r.t a state transition of time duration d into DEF $\bigvee_j(f^j \wedge \bigcirc_d\overline{\omega}^j)$ can be intuitively interpreted as a $present$ condition f^j that is to be satisfied immediately and a corresponding $future$ condition $\overline{\omega}^j$ that should be satisfied in future, d unit of time later, where d is 0 or 1.

The algorithm begins by obtaining the DEF $\bigvee_j f^j \wedge \bigcirc_d \overline{\omega}^j$ of ω w.r.t $d = 0$ using Procedure $Expand$[3]. If any f^j holds at q_0, then its corresponding $future$ condition $\overline{\omega}^j$ should be satisfied by all state trajectories starting from q_0, prompting the algorithm to associate $\overline{\omega}^j$ with q_0, by assigning $(q_0, \overline{\omega}^j)$ as the initial state of H.

Formally, satisfaction of an MTL formula ω_i at q_i over a state trajectory $q_i \xrightarrow{\sigma} q_{i+1} \cdots$ is established by evaluating a $present$ condition at q_i and postponing a $future$ condition to be evaluated at q_{i+1}, $\varphi(\sigma)$ unit of time later. H is computed by:

1) recursively applying MTL expansion rules (given in Section II-B3) on ω_i w.r.t $d = \varphi(\sigma)$ and transforming it into DEF $\bigvee_j (f^j \wedge \bigcirc_d \overline{\omega}^j)$;
2) selecting a disjunct $f^j \wedge \bigcirc_d \overline{\omega}^j$ such that its $present$ condition f^j is satisfied by the state transition $q_i \xrightarrow{\sigma} q_{i+1}$ (i.e., f^j holds at q_{i+1}) and associating the corresponding $future$ condition $\overline{\omega}^j$ with q_{i+1} by assigning $(q_{i+1}, \overline{\omega}^j)$ as a state of H; and
3) defining state transition $\zeta(\sigma, (q_i, \omega_i)) = (q_{i+1}, \overline{\omega}^j)$.

We now present our algorithm.

Theorem 1. *Given an input MTL specification ω in CNF for a TDES $G = (Q, \Sigma, \delta, q_0, Q_m)$, the output $Trim(H)$ of Algorithm 1 is deterministic.*

Proof. Let $H = (Y, \Sigma, \zeta, y_0, Y_m)$. Consider an arbitrary state $(q, \overline{\omega}) \in Y$ and event $\sigma \in \Sigma$ of H. By Property 1, for $\overline{\omega}$ expanded w.r.t the time duration $\varphi(\sigma)$ of σ into equivalent DEF $\bigvee_j f^j \wedge \bigcirc_d \overline{\omega}^j$, we have for two nonidentical $f^{j_1}, f^{j_2} \in Pre(\overline{\omega}, \varphi(\sigma))$ that $f^{j_1} \wedge f^{j_2} = false$. As a result (and as G is also deterministic), for each $\sigma \in \Sigma$ at most one $f^j \in Pre(\overline{\omega}, \varphi(\sigma))$ holds at $\delta(\sigma, q)$. Hence, there exists at most one state of H, defined as $(\delta(\sigma, q), Fut(f^j))$ such that $\zeta(\sigma, (q, \overline{\omega})) = (\delta(\sigma, q), Fut(f^j))$. It follows that $Trim(H)$ is deterministic. ■

Definition 1. *Let $\rho_m = (q_0, \overline{\omega}^0)(q_1, \overline{\omega}^1) \cdots (q_i, \overline{\omega}^i) \cdots (q_m, \overline{\omega}^m)$ be a marked finite state trajectory of H, for which there exists a corresponding string $e_{m-1} = e(0)e(1) \cdots e(i) \cdots e(m-1) \in L_m(H)$ such that for all $i,\ 0 < i \le m$, $(q_{i+1}, \overline{\omega}^{i+1}) = \zeta(e(i), (q_i, \overline{\omega}^i))$ and $q_{i+1} = \delta(e(i), q_i)$. Then, $Trim(H)$ is said to be correct w.r.t CNF ω if*

1) $\exists f^0 \in Pre(\omega, 0), \models^{q_0} f^0$ and $(q_0, \overline{\omega}^0) \in Y$ is the initial state y_0 of H;
2) for every $e_{m-1} \in L_m(H)$, there exists a state trajectory $(q_0, \overline{\omega}^0)(q_1, \overline{\omega}^1) \cdots (q_m, \overline{\omega}^m)$ such that $\models^{q_i} f^i$ for some $f^i \in Pre(\overline{\omega}^{i-1}, \varphi(e(i-1)))$, for all $i,\ 0 < i \le m$;

where $\overline{\omega}^i = Fut(f^i)$.

Definition 2. *Let $\rho_m = q_0 q_1 \cdots q_i \cdots q_m$ be a marked finite state trajectory of TDES G, for which there exists a corresponding string $e_{m-1} = e(0)e(1) \cdots e(m-1) \in L_m(G)$ such that for all $i,\ 0 < i \le m$, $q_{i+1} = \delta(e(i), q_i)$. Then, $Trim(H)$ is said to be complete w.r.t TDES G if*

[3]$d = 0$ because the initial time stamp $\mathcal{T}(0) = 0$.

Algorithm 1: A CNF-to-TTG Translation

```
Input: An MTL specification $\omega$ (in CNF) for TDES
       $G = (Q, \Sigma, \delta, q_0, Q_m)$;
Output: A trim TTG specification of
        $H = (Y, \Sigma, \zeta, y_0, Y_m)$ on $G$;
begin
    $Y = \emptyset$;
    $Y_m = \emptyset$;
    Let $\bigvee_j f^j \wedge \bigcirc_d \overline{\omega}^j = Expand(\omega, 0)$;
    if any $f^j$ holds at $q_0$ then
        $y_0 := \{(q_0, \overline{\omega}^j)\}$;
        $Y := Y \cup \{y_0\}$;
        if $q_0 \in Q_m \wedge \overline{\omega}^j$ is in invariance normal form then
            $Y_m := Y_m \cup \{y_0\}$;
    $Y' = \emptyset$;
    while $Y - Y' \neq \emptyset$ do
        Select any $(q_1, \omega_1) \in Y$;
        foreach $\sigma \in \Sigma$ such that $\delta(\sigma, q_1)!$ do
            $\bigvee_j f^j \wedge \bigcirc_d \overline{\omega}^j = Expand(\omega_1, \varphi(\sigma))$;
            if any $f^j$ holds at $\delta(\sigma, q_1)$ then
                Let $(q_2, \omega_2) = (\delta(\sigma, q_1), \overline{\omega}^j)$;
                $Y := Y \cup \{(q_2, \omega_2)\}$;
                Define $\zeta(\sigma, (q_1, \omega_1)) = (q_2, \omega_2)$;
                if $\delta(\sigma, q_1) \in Q_m \wedge \overline{\omega}^j$ is in invariance normal form then
                    $Y_m := Y_m \cup \{(q_2, \omega_2)\}$;
        $Y' := Y' \cup \{(q_1, \omega_1)\}$;
    return $Trim(H)$ where $H = (Y, \Sigma, \zeta, y_0, Y_m)$;
```

1) $\exists f^0 \in Pre(\omega, 0), \models^{q_0} f^0$ and $(q_0, \overline{\omega}^0) \in Y$ is the initial state y_0 of H;
2) for every ρ_m of G with $q_m \in Q_m$, if there exists a state trajectory $(q_0, \overline{\omega}^0)\ (q_1, \overline{\omega}^1) \cdots (q_m, \overline{\omega}^m)$ of H such that for all $i,\ 0 < i \le m$, $\models^{q_i} f^i$ for some $f^i \in Pre(\overline{\omega}^{i-1}, \varphi(e(i-1)))$, then $e_{m-1} \in L_m(H)$;

where $\overline{\omega}^i = Fut(f^i)$.

Theorem 2. *The output $Trim(H)$ of Algorithm 1 is correct w.r.t an input CNF specification ω and complete w.r.t the given TDES G.*

Proof. Given an input CNF specification ω for a TDES G, it follows from Definitions 1 and 2 that the output $Trim(H)$ of Algorithm 1 is correct w.r.t ω and complete w.r.t G. ■

C. Computational Complexity

The number of different subformulae that can be produced using equivalences (4)-(8) for an MTL formula ω is $2^{|closure(\omega)|}$, where $closure(\omega)$ denotes the set containing all its subformulae. It can be easily shown that $|closure(\omega)| \le 2N$ [31], where N denotes the number of Boolean and temporal connectives in ω. Let $\mathscr{T}$ be the maximum value that occurs as a timing constraint associated with the temporal connectives of ω. Then there can be at most $\mathscr{T}+1$ different time arguments. Hence, within H, there can be at most $|Q|2^{2^{N(\mathscr{T}+1)}}$ states, where $|Q|$ denotes the cardinality of set Q. It follows that the worst case complexity of Algorithm 1 is exponential in N and $\mathscr{T}$.

Although our algorithm is of exponential complexity in the worst case, its translation complexity is expected to be much lower in practice. It has been empirically shown that in the very large majority of cases encountered, the inherent exponential nature of translation algorithms of this kind may be of little practical significance [47], [48], [31]. For a temporal logic formula that a system designer could think of for a system specification, it is almost always very short and so the translated automaton obtained (or TTG in the TDES context considered) incurs a reasonable time complexity [26].

IV. Specification Interface and Application Examples

A. A Specification Interface for TTG-based Control Synthesis

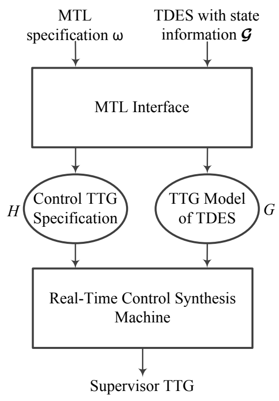

MTL specification ω
TDES with state information 𝒢
MTL Interface
H
Control TTG Specification
TTG Model of TDES
G
Real-Time Control Synthesis Machine
Supervisor TTG

Fig. 1. MTL Interface to a TTG-based Control Synthesis Machine

Real-time supervisor synthesis engines or machines such as TTCT [18] have been developed to automatically synthesize supervisors from TTG specifications. Our translation algorithm can be implemented as an MTL interface for use with a TTG-based control synthesis machine, integrated as depicted in Fig. 1.

B. Application Examples

We now present three examples to demonstrate the utility of our MTL interface. Using the first example, we also explain the workings of the translation algorithm. In all the examples, every activity of each ATG model formulated is characterized by a unique propositional symbol, i.e., the symbol is $true$ only at the activity it defines and is $false$ elsewhere. In the TTG models of these examples shown in Figs. 2, 3 and 4, the event timer value defined for each state is implicitly understood and not shown. The ATG and TTG models are graphically represented by edge-labeled directed graphs, with a node denoting an activity or state and an edge denoting an event-labeled transition. The initial activity or state is represented by a node with an entering arrow, and a marked activity or state is represented by a double-concentric circle.

TABLE I
Propositional Symbol Definitions for Task Scheduling

Propositional symbol	Activity
$Dormant$	Task is dormant
$Arrived$	Task has arrived
$Pending$	Task is pending execution
$Ending$	Task is ending execution

1) Example 1: Task Scheduling: Recent work [12] has proposed a preemptive scheduling scheme that uses the Brandin-Wonham framework for supervisory control of TDES's to automatically synthesize schedulers for systems with sporadic tasks. Herewith, a scheduler accepts a newly arrived instance of a sporadic task only if the task instance can be completed without missing the deadline for the task and other previously accepted tasks. The synthesis procedure requires a deadline specification TTG for each task. Such a TTG specification, which has to be prescribed by a designer, should include all possible (timed) execution sequences of the task that meet its deadline. Later in this example, we will show how our MTL interface can automate the prescription of deadline specification TTG's, making the interface-integrated control framework more human designer-friendly.

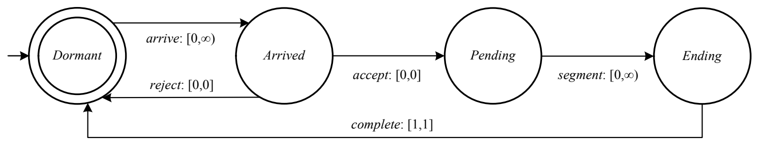
We consider a sporadic task, modeled as TDES G, that can $arrive$ at any arbitrary time. Once an instance of the task has arrived, a scheduler can $accept$ or $reject$ the task instance. While modeling, the task is divided into segments such that each segment takes one unit of time to execute. In the Brandin-Wonham framework, this can be represented in a TTG by a transition of event $tick$ following each instantaneous $segment$. Once a task instance is accepted, each segment of the task is executed one by one, until finally the task instance is $complete$. For simplicity and without loss of generality, in this example we consider the case where the task has only one execution segment. The ATG model of G along with time bounds of events is given in Fig. 2(a). Table I defines a unique propositional symbol for every activity of this ATG. The TTG G having timing constraints explicitly represented using transitions of $tick$ is given in Fig. 2(b).

A deadline specification for a task asserts that once an instance of the task is accepted, it should complete execution within a certain period of time. Considering 2 time units as the deadline for the task under consideration, this specification can be easily written in MTL as a CNF

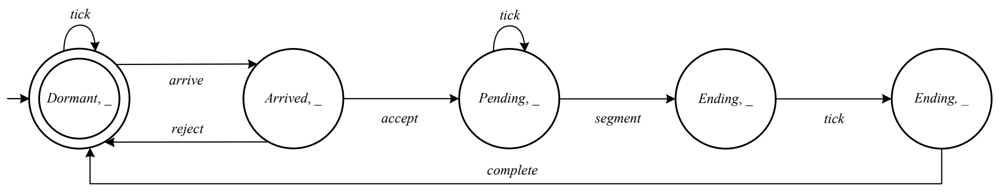
$$\omega \;=\; \Box_{\geq 0}[Pending \rightarrow \Diamond_{\leq 2}\; Dormant].$$

This MTL formula can be paraphrased as "whenever an instance of the task is $Pending$ execution, it must return to $Dormant$ (after completing execution) within 2 time units". The deadline specification TTG H, obtained from ω and G by applying Algorithm 1, is given in Fig. 2(c). Note that we represent $true\ U_{\leq t}\ Dormant$ as ϕ_t for brevity.

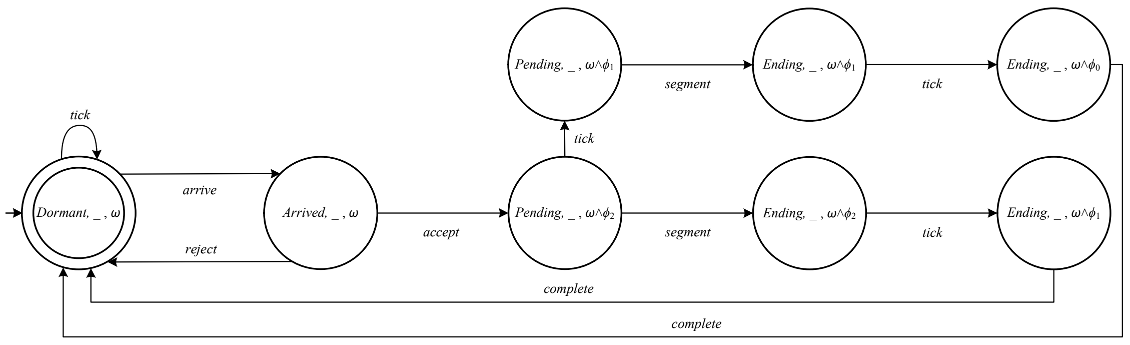
The computational workings to produce H by Algorithm 1 are explained as follows. Initially, the algorithm uses Procedure $Expand$ to expand the MTL specification $\omega = \Box_{\geq 0}[Pending \rightarrow \Diamond_{\leq 2}\; Dormant]$ for G into its DEF $(\neg Pending \vee Dormant) \;\wedge\; \bigcirc_d(\omega) \vee (Pending \;\wedge$



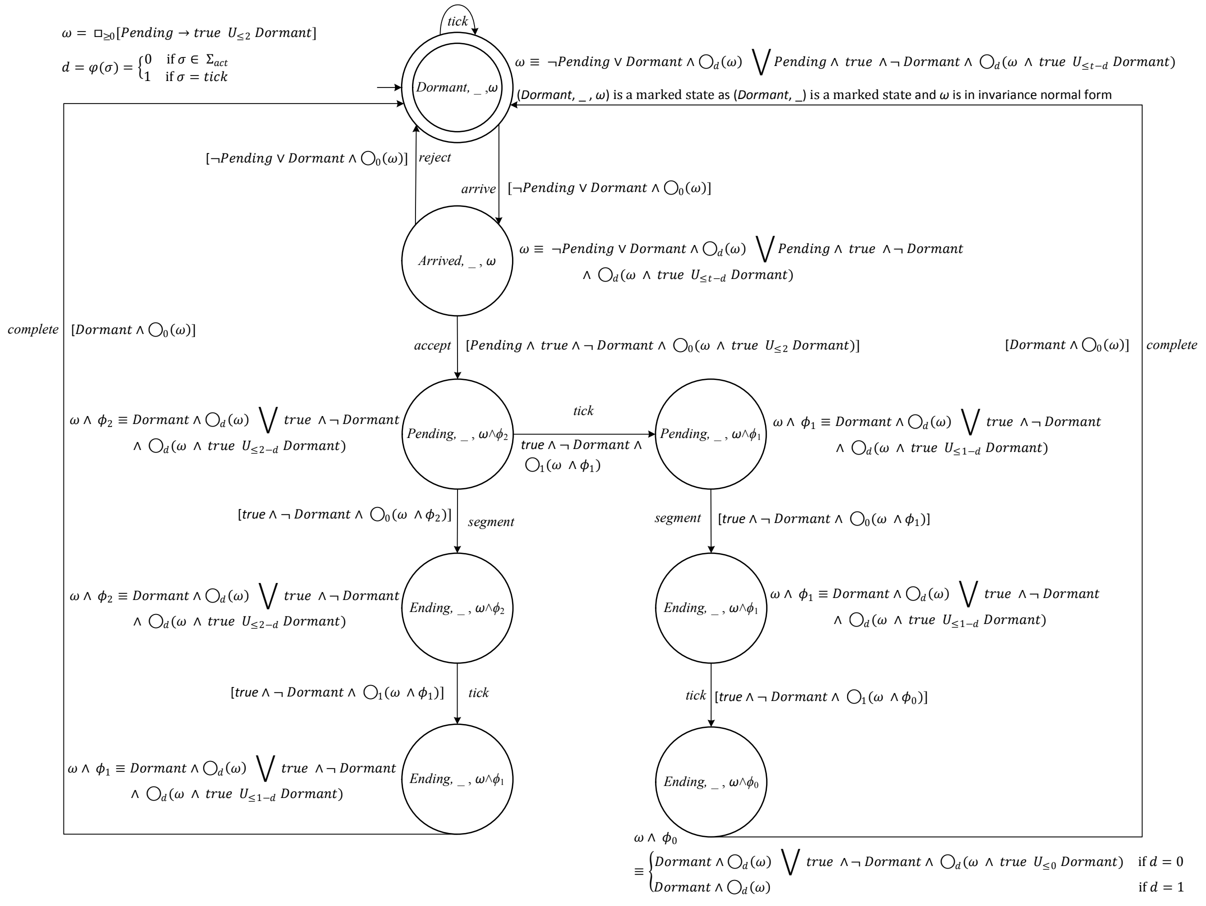
(a) Task ATG



(b) Task TTG



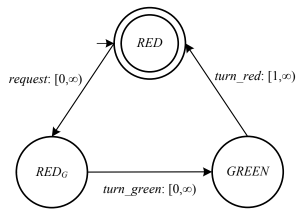
(c) Deadline Specification TTG



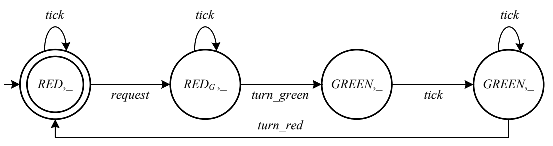
ω = □≥0[Pending → true U≤2 Dormant]
d = φ(σ) = {0 if σ ∈ Σact; 1 if σ = tick
tick
Dormant, _ ,ω
ω ≡ ¬Pending ∨ Dormant ∧ ◯d(ω) ⋁ Pending ∧ true ∧¬ Dormant ∧ ◯d(ω ∧ true U≤t−d Dormant)
(Dormant, _ , ω) is a marked state as (Dormant, _) is a marked state and ω is in invariance normal form
[¬Pending ∨ Dormant ∧ ◯0(ω)] reject
arrive [¬Pending ∨ Dormant ∧ ◯0(ω)]
Arrived, _ , ω
ω ≡ ¬Pending ∨ Dormant ∧ ◯d(ω) ⋁ Pending ∧ true ∧¬ Dormant ∧ ◯d(ω ∧ true U≤t−d Dormant)
complete [Dormant ∧ ◯0(ω)]
accept [Pending ∧ true ∧¬ Dormant ∧ ◯0(ω ∧ true U≤2 Dormant)]
[Dormant ∧ ◯0(ω)] complete
ω ∧ ϕ2 ≡ Dormant ∧ ◯d(ω) ⋁ true ∧¬ Dormant ∧ ◯d(ω ∧ true U≤2−d Dormant)
Pending, _ , ω∧ϕ2
tick
true ∧¬ Dormant ∧ ◯1(ω ∧ϕ1)
Pending, _ , ω∧ϕ1
ω ∧ ϕ1 ≡ Dormant ∧ ◯d(ω) ⋁ true ∧¬ Dormant ∧ ◯d(ω ∧ true U≤1−d Dormant)
[true ∧¬ Dormant ∧ ◯0(ω ∧ϕ2)] segment
segment [true ∧¬ Dormant ∧ ◯0(ω ∧ϕ1)]
ω ∧ ϕ2 ≡ Dormant ∧ ◯d(ω) ⋁ true ∧¬ Dormant ∧ ◯d(ω ∧ true U≤2−d Dormant)
Ending, _ , ω∧ϕ2
Ending, _ , ω∧ϕ1
ω ∧ ϕ1 ≡ Dormant ∧ ◯d(ω) ⋁ true ∧¬ Dormant ∧ ◯d(ω ∧ true U≤1−d Dormant)
[true ∧¬ Dormant ∧ ◯1(ω ∧ϕ1)] tick
tick [true ∧¬ Dormant ∧ ◯1(ω ∧ϕ0)]
ω ∧ ϕ1 ≡ Dormant ∧ ◯d(ω) ⋁ true ∧¬ Dormant ∧ ◯d(ω ∧ true U≤1−d Dormant)
Ending, _ , ω∧ϕ1
Ending, _ , ω∧ϕ0
ω ∧ ϕ0 ≡ {Dormant ∧ ◯d(ω) ⋁ true ∧¬ Dormant ∧ ◯d(ω ∧ true U≤0 Dormant) if d = 0; Dormant ∧ ◯d(ω) if d = 1

(d) Computation of Deadline Specification TTG

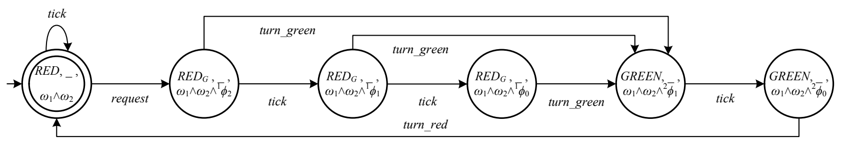
Fig. 2. Task Scheduling



(a) Traffic Light ATG



(b) Traffic Light TTG



(c) TTG Specification

Fig. 3. Pedestrian Traffic Light

$true \wedge \neg Dormant) \wedge \bigcirc_d(\omega \wedge true\, U_{\leq t-d}\, Dormant)$, for $d = 0$. The procedure accomplishes the expansion using equivalences (4)-(9) and the usual distributive laws between Boolean connectives as described in Section II-B4. As $\neg Pending \vee Dormant \in Pre(\omega, 0)$ holds at the initial state $q_0 = (Dormant, \_)$ of G (line 5), the algorithm associates $Fut(\neg Pending \vee Dormant) = \omega$ with q_0 to compute the initial state $y_0 = (Dormant, \_, \omega)$ of H (line 6). As $q_0 \in Q_m$ and ω is in invariance normal form (line 8), $y_0 = (Dormant, \_, \omega)$ is identified as a marked state (line 9). For each computed state (q_1, ω_1) of H (lines 11-12) and each event such that a TDES transition is defined from the associated TDES state q_1 (line 13), the algorithm uses Procedure $Expand$ to expand the associated MTL formula ω_1 (line 14) w.r.t $d = 1$ if the event under consideration is $tick$, and otherwise sets $d = 0$. If any $present$ condition of the expansion is satisfied by the destination TDES state of the transition (line 15), then the corresponding $future$ condition is associated with the TDES state to compute a new state of H (line 17). A transition of the event under consideration is defined from (q_1, ω_1) to the newly computed state (line 18), which is assigned as marked (line 20) if its associated TDES state is a marked state of the TDES and its associated MTL formula is in invariance normal form (line 19). These steps are repeated until no new state is computed. These computational steps produce the TTG H given in Fig. 2(c), and are depicted in Fig. 2(d) where, beside each state, the expansion of its associated MTL formula is given and beside each defined transition, the $present$ condition of the expansion that the state transition satisfies is given.

2) Example 2: Pedestrian Traffic Light: The second example is that of a traffic light for pedestrians. The traffic light, modeled as a TDES G, can turn red ($turn\_red$) or green ($turn\_green$), and can accept request to turn green ($request$). The ATG model of G along with time bounds of events is given in Fig. 3(a). Table II defines a unique propositional symbol for every activity of this ATG. The TTG G having timing constraints explicitly represented using transitions of $tick$ is given in Fig. 3(b).

A real-time control requirement for G asserts that the traffic light must turn green within 2 time units following a request to turn green. To prevent the system from indefinitely displaying green, it is also required that a traffic light that has turned green should not remain so for more than 1 time unit. These statements can easily be translated into MTL as a CNF $\omega = \Box_{\geq 0}[RED_G \rightarrow RED_G\ U_{\leq 2}\ GREEN] \wedge \Box_{\geq 0}[GREEN \rightarrow \Diamond_{\leq 1}\neg GREEN]$.

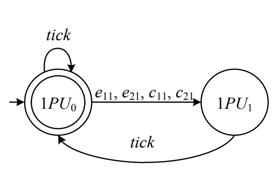
This MTL formula can be paraphrased as "whenever there is a request to turn the traffic light green when it is red (i.e., $RED_G = true$), the light should remain so until it turns green (i.e., $GREEN = true$) within 2 time units and whenever the traffic light is green (i.e., $GREEN = true$), it should stop being green (i.e., $GREEN = false$) within 1 time unit". By equivalence (3), the MTL formula is equivalent to $\Box_{\geq 0}\,[RED_G \rightarrow RED_G\ U_{\leq 2}\ GREEN] \wedge \Box_{\geq 0}[GREEN \rightarrow true\ U_{\leq 1}\neg GREEN]$. The TTG specification obtained by applying Algorithm 1 is given in Fig. 3(c). Note that we represent $RED_G\ U_{\leq t}\ GREEN$ as ${}^1\phi_t$ and $true\ U_{\leq t}\ \neg GREEN$ as ${}^2\phi_t$ for brevity.

TABLE II
PROPOSITIONAL SYMBOL DEFINITIONS FOR TRAFFIC LIGHT

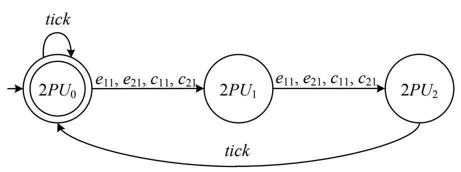
Propositional symbol	Activity
RED	Traffic light is red
RED_G	Traffic light is red after a request to turn green
$GREEN$	Traffic light is green

TABLE III
PROPOSITIONAL SYMBOL DEFINITIONS FOR MULTIPROCESSOR RESOURCE ALLOCATION

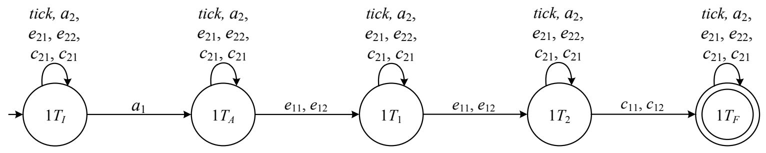
Propositional symbol	Activity
iPU_{j}	Processing unit $PU_i, i \in \{1, 2\}$ has executed j task segments
iT_I	Task i is dormant
iT_A	Task i has arrived
iT_{j}	j segments of Task i have been executed
iT_F	Task i has finished execution



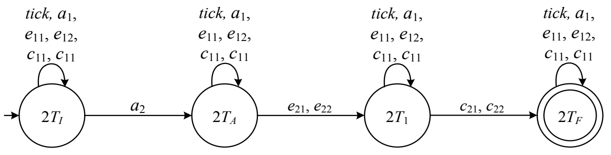
(a) Processing Unit TTG, PU_1


tick
$2PU_0$
$e_{11}, e_{21}, c_{11}, c_{21}$
$2PU_1$
$e_{11}, e_{21}, c_{11}, c_{21}$
$2PU_2$
tick

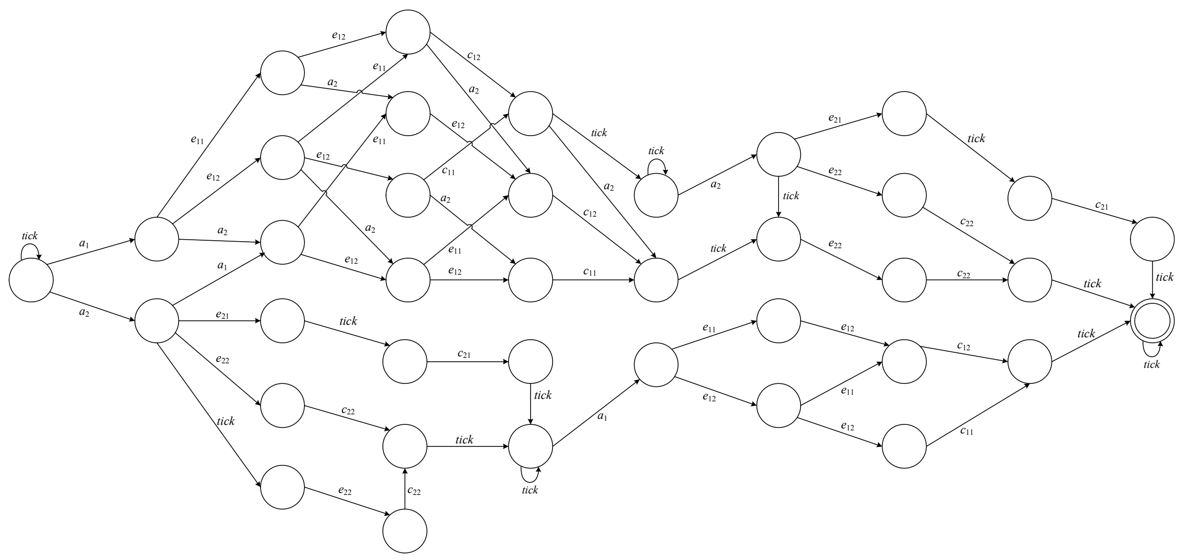
(b) Processing Unit TTG, PU_2



(c) Task TTG, T_1



(d) Task TTG, T_2



(e) TTG Specification

Fig. 4. Multiprocessor Resource Allocation

3) Example 3: Multiprocessor Resource Allocation: We now consider the example of resource allocation in a multiprocessor having two or more processing units (adapted from [14]). A controller that allocates tasks to the processing units of the multiprocessor such that all the given real-time constraints on task executions are satisfied can be automatically generated once a TDES of the processing units in the multiprocessor interacting with a set of tasks and a control specification of the real-time constraints are modeled as TTG's.

In this example, we consider the case where two tasks are to be allocated on a multiprocessor with two processing units. Each processing unit has its own computing capacity and a task executing on a processing unit having computing capacity b for τ time units completes $b\tau$ units of execution.

In this example, we consider a TDES G consisting of two processing units modeled by TTG's PU_1 and PU_2, and two tasks modeled by TTG's T_1 and T_2. Table III defines a unique propositional symbol for every activity of the respective ATG's. These ATG's are structurally similar to their TTG's and not shown.

PU_1 and PU_2 have computing capacity of 1 and 2, respectively. This means that PU_1 can execute 1 segment of a task in one time unit, while PU_2 can execute 2 segments. The TTG models of PU_1 and PU_2 are given in Figs. 4(a) and 4(b), respectively.

Each task TTG $T_i, i \in \{1, 2\}$, is modeled as follows. Arrival of an instance of task T_i is denoted by an event a_i, and each T_i is divided into v_i segments, where each segment has an execution time of one time unit. The execution of segment $k \in [1, v_k)$ of task T_i in processing unit PU_j is denoted by event symbol e_{ij}. We use an event symbol c_{ij} to denote the execution of the last segment of a task T_i in PU_j. In this example, we consider a case where $v_1 = 3$ and $v_2 = 2$. The TTG models of T_1 and T_2 are given in Figs. 4(c) and 4(d), respectively.

The TDES G is formed by the composition [11] of TTG's T_1, T_2, PU_1 and PU_2.

The real-time controller has to allocate the two tasks to the two processing units such that the deadline for each task T_i, denoted by D_i is not violated. In this example, we consider the case where $D_1 = 1$ and $D_2 = 2$. This deadline requirement can be easily given as an MTL formula $\omega \equiv \Box_{\geq 0}[1T_A \rightarrow \Diamond_{\leq 1} 1T_F] \wedge \Box_{\geq 0}[2T_A \rightarrow \Diamond_{\leq 2} 2T_F]$. The TTG specification H obtained by translating ω using Algorithm 1 is given in Fig. 4(e).

In summary, by the three examples, it should be clear that without the MTL interface, attempting to prescribe by hand the translated specification TTG's (or their equivalents) could be tedious and error-prone. Unlike reading an MTL formula, it is often not easy to interpret the control meaning of a given TTG against the specification statement in English that it supposedly formalizes without a structural understanding of the given TDES model as well, and so there is less confidence of the correctness and completeness of the TTG prescribed by hand. The examples demonstrate that MTL formulas for control requirements are often easy to write and read, and this supports clear interpretation and give more confidence to control designers in determining if the written formulas do reflect the specification statements. Once an MTL formula is determined as specifying the right specification, the proposed MTL interface converts it to a TTG that is guaranteed to be correct and complete, underlining the significant utility of the interface.

V. CONCLUSION

In this paper, we have proposed an MTL specification translation interface for finitary control of TDES's. The interface can translate MTL specifications from the bounded response class into TTG's. The translation is made within the context of the TDES model defined in the conceptually well-founded control framework initiated by Brandin and Wonham [11], and is proved to be correct and complete. Importantly, it enables automated TTG prescription based on writing specifications in MTL that is practically expressive and readable, and this should mitigate, if not solve, the difficult problem of specifying real-time control requirements directly as TTG models in the Brandin-Wonham control framework. Three examples presented illustrate the utility of the interface.

The output TTG specification of the proposed interface represents the full nonblocking behavior of TDES G, and a human designer who wants to further ascertain that the translated TTG models the intended requirement may find it difficult to comprehend and intuitively validate it against the MTL counterpart. Recent work has proposed the concept of specification transparency to facilitate human comprehension of graphical specifications [49], [50], [51]. In particular, a framework to automatically restructure TTG specifications into transparent TTG specifications that are easier to comprehend has been proposed in [52]. Our interface, in conjunction with the TTG specification transparency framework [52], should lead to a more effective specification-synthesis paradigm, where the ease of specification in MTL combined with the comprehensibility of transparent TTG's should inspire even more confidence that a control specification in TTG - a mandatory real-time computational model for control synthesis in the Brandin-Wonham framework - does indeed capture the intended requirement.

To experimentally ascertain the usefulness of the proposed MTL interface, future work includes conducting an observational study on TDES control design by human designers, with and without the interface. Developing complexity mitigation techniques for the proposed translation algorithm is also a significant subject for future research.

REFERENCES

[1] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal of Control and Optimization*, vol. 25, no. 1, pp. 206–230, January 1987.

[2] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81 – 98, January 1989.

[3] W. M. Wonham, "Supervisory control of discrete-event systems," Systems Control Group, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada, Tech. Rep., 2013. [Online]. Available: http://www.control.toronto.edu/cgi-bin/dldes.cgi

[4] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer, 2008.

[5] S. Takai and R. Kumar, "Decentralized diagnosis for nonfailures of discrete event systems using inference-based ambiguity management," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 40, no. 2, pp. 406 –412, March 2010.

[6] K. T. Seow, "Organizational control of discrete-event systems: A hierarchical multiworld supervisor design," *IEEE Transactions on Control Systems Technology*, 2013, Accepted.

[7] W. M. Wonham, "Supervisory control theory: Models and methods," in *Workshop on Discrete Event Systems Control, 24th International Conference on Application and Theory of Petri Nets (ATPN 2003)*, Eindhoven, The Netherlands, June 2003, pp. 1–14.

[8] M. P. Fanti and M. Zhou, "Deadlock control methods in automated manufacturing systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 34, no. 1, pp. 5 – 22, January 2004.

[9] H. R. Golmakani, J. K. Mills, and B. Benhabib, "Deadlock-free scheduling and control of flexible manufacturing cells using automata theory," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 36, no. 2, pp. 327 – 337, March 2006.

[10] K. T. Seow and M. Pasquier, "Supervising passenger land-transport systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 3, pp. 165–176, September 2004.

[11] B. A. Brandin and W. M. Wonham, "Supervisory control of timed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 39, no. 2, pp. 329–342, 1994.

[12] Seong-Jin Park and Kwang-Hyun Cho, "Real-time preemptive scheduling of sporadic tasks based on supervisory control of discrete event systems," *Information Sciences*, vol. 178, pp. 3393–3401, 2008.

[13] P. C. Y. Chen and W. M. Wonham, "Real-time supervisory control of a processor for non-preemptive execution of periodic tasks," *Real-Time Systems*, vol. 23, no. 3, pp. 183–208, 2002.

[14] V. Janarthanan and P. Gohari, "Multiprocessor scheduling in supervisory control of discrete-event systems framework," *Control and Intelligent Systems*, vol. 35, no. 4, pp. 360–366, September 2007.

[15] A. A. Afzalian, S. M. Noorbakhsh, and W. M. Wonham, "Discrete-event supervisory control for under-load tap-changing transformers (ultc): from synthesis to plc implementation," *Discrete Event Simulations*, pp. 285–310, 2010.

[16] F. Lin and W. M. Wonham, "Supervisory control of timed discrete-event systems under partial observation," *IEEE Transactions on Automatic Control*, vol. 40, no. 3, pp. 558 – 562, 1995.

[17] K. C. Wong and W. M. Wonham, "Hierarchical control of timed discrete-event systems," *Discrete Event Dynamic Systems*, vol. 6, pp. 275–306, 1996.

[18] "TTCT," http://www.control.utoronto.ca/people/profs/wonham/.

[19] R. Zhang, K. Cai, Y. Gan, Z. Wang, and W. M. Wonham, "Supervision localization of timed discrete-event systems," *Automatica*, 2013, Accepted.

[20] M. Nomura and S. Takai, "A synthesis method for decentralized supervisors for timed discrete event systems," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 96, no. 4, pp. 835–839, 2013.

[21] S.-J. Park and K.-H. Cho, "Nonblocking supervisory control of timed discrete event systems under communication delays: The existence conditions," *Automatica*, vol. 44, no. 4, pp. 1011–1019, 2008.

[22] T.-J. Ho, "Controller synthesis for some control problems in timed discrete-event systems," in *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 5, 1997, pp. 4613–4618.

[23] P. Gohari and W. M. Wonham, "Reduced supervisors for timed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 48, no. 7, pp. 1187–1198, 2003.

[24] A. Saadatpoor and W. M. Wonham, "State based control of timed discrete event systems using binary decision diagrams," *Systems & control letters*, vol. 56, no. 1, pp. 62–74, 2007.

[25] J.-M. Roussel and A. Giua, "Designing dependable logic controllers using the supervisory control theory," in *Proceedings of the 16th IFAC World Congress*, 2005.

[26] K. T. Seow, "Integrating temporal logic as a state-based specification language for discrete-event control design in finite automata," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 3, pp. 451–464, July 2007.

[27] J. Thistle and W.M.Wonham, "Control problems in a temporal logic framework," *International Journal of Control*, vol. 44, no. 4, pp. 943–976, 1986.

[28] R. Gotzhein, "Temporal logic and applications : A tutorial," *Computer Networks and ISDN systems*, vol. 24, no. 3, pp. 203–218, 1992.

[29] V. Goranko, "Temporal logics for specification and verification," *Proceedings of the European Summer School in Logic, Language and Information (ESSLI'09)*, 2009.

[30] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Systems*, vol. 2, pp. 255–299, 1990.

[31] M. Barbeau, F. Kabanza, and R. St.-Denis, "A method for the synthesis of controllers to handle safety, liveness, and real-time constraints," *IEEE Transactions on Automatic Control*, vol. 43, pp. 1543–1559, 1998.

[32] S. Konrad and B. H. C. Cheng, "Real-time specification patterns," in *Proceedings of the 27th International Conference on Software Engineering*, St. Louis, MO, USA, 2005, pp. 372–381.

[33] P. Bellini, P. Nesi, and D. Rogai, "Expressing and organizing real-time specification patterns via temporal logics," *Journal of Systems and Software*, vol. 82, no. 2, pp. 183 – 196, 2009.

[34] A. Post and J. Hoenicke, "Formalization and analysis of real-time requirements: A feasibility study at BOSCH," in *Verified Software: Theories, Tools, Experiments*. Springer Berlin / Heidelberg, 2012, vol. 7152, pp. 225–240.

[35] C. Zhou, R. Kumar, D. Bhatt, K. Schloegel, and D. Cofer, "A framework of hierarchical requirements patterns for specifying systems of interconnected simulink/stateflow modules," in *Software Engineering and Knowledge Engineering*, Boston, Massachusetts, USA, July 2007, pp. 179–184.

[36] L. Grunske, K. Winter, and R. Colvin, "Timed behavior trees and their application to verifying real-time systems," in *Proceedings of the Australian Software Engineering Conference*, Melbourne, Victoria, Australia, 2007, pp. 211–222.

[37] N. Abid, S. D. Zilio, and D. L. Botlan, "A Real-Time Specification Patterns Language," Laboratoire d'analyse et d'architecture des systèmes - LAAS, Tech. Rep., 2011. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00593965

[38] H. Wong-Toi and G. Hoffmann, "The control of dense real-time discrete event systems," in *Proceedings of the 30th IEEE Conference on Decision and Control*, 1991, pp. 1527–1528.

[39] A. Khoumsi and M. Nourelfath, "An efficient method for the supervisory control of dense real-time discrete event systems," in *Proceedings of the 8th International Conference on Real-Time Computing Systems (RTCSA)*, 2002.

[40] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[41] A. Dubey, "A discussion on supervisory control theory in real-time discrete event systems," Institute for Software Integrated Systems, Vanderbilt University," Technical Report, 2009.

[42] O. Maler, D. Nickovic, and A. Pnueli, "On synthesizing controllers from bounded-response properties," in *Computer Aided Verification*, 2007, vol. 4590, pp. 95–107.

[43] W. Kuijper and J. van de Pol, "Compositional control synthesis for partially observable systems," in *CONCUR 2009 - Concurrency Theory*, 2009, vol. 5710, pp. 431–447.

[44] Z. Manna and A. Pnueli, "Clocked transition systems," Stanford University, Stanford, CA, USA, Tech. Rep., 1996.

[45] F. Kabanza, M. Barbeau, and R. St.-Denis, "Planning control rules for reactive agents," *Artificial Intelligence*, vol. 95, no. 1, pp. 67 – 113, 1997.

[46] T. A. Henzinger, "It's about time: Real-time logics reviewed," in *CONCUR'98 Concurrency Theory*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, vol. 1466, pp. 439–454.

[47] P. Wolper, "Constructing automata from temporal logic formulas: A tutorial," in *Lectures on Formal Methods and Performance Analysis*, 2001, vol. 2090, pp. 261–277.

[48] E. A. Emerson, T. Sadler, and J. Srinivasan, "Efficient temporal reasoning (extended abstract)," in *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, Austin, Texas, United States, 1989, pp. 166–178.

[49] M. T. Pham, A. Dhananjayan, and K. T. Seow, "On the transparency of automata as discrete-event control specifications," in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, USA, May 2010, pp. 1474–1479.

[50] M. T. Pham, A. Dhananjayan, and K. T. Seow, "On specification transparency: Towards a formal framework for designer comprehensibility of discrete-event control specifications in finite automata," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 1, pp. 139 – 148, 2013.

[51] A. Dhananjayan and K. T. Seow, "On specification informatics in discrete-event systems: State-transparency for clarity of finite automata as control specifications," in *Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics*, Rome, Italy, July 2012, pp. 357 – 367.

[52] A. Dhananjayan and K. T. Seow, "Automating timed specification transparency for human designer validation of real-time discrete-event control requirements," in *Proceedings of the IEEE International Conference on Automation Science and Engineering*, Seoul, South Korea, August 2012, pp. 908 – 913.