

Control Synthesis for Parametric Timed Automata under Unavoidability Specifications

Ebru Aydin Gol

Abstract—Timed automata (TA) is used for modeling systems with timing aspects. A TA extends a finite automaton with a set of real valued variables called clocks, that measure the time and constraints over the clocks guard the transitions. A parametric TA (PTA) is a TA extension that allows parameters in clock constraints. In this paper, we focus on synthesis of a control strategy and parameter valuation for a PTA such that each run of the resulting TA reaches a target location within the given amount of time while avoiding unsafe locations. We propose an algorithm based on depth first analysis combined with an iterative feasibility check. The algorithm iteratively constructs a symbolic representation of the possible solutions, and employs a feasibility check to terminate the exploration along infeasible directions. Once the construction is completed, a mixed integer linear program is solved for each candidate strategy to generate a parameter valuation and a control strategy pair. We present a robotic planning example to motivate the problem and to illustrate the results.

I. INTRODUCTION

Timed automata (TA) [1] is used for modeling systems with timing aspects. A TA extends a finite automaton with a set of real valued variables called clocks that measure the time. The clocks enrich the semantics, and the constraints over the clocks restrict the behavior of the automaton. The examples of real-time systems modeled as timed automata includes rail-road crossing systems [2], scheduling problems [3], and pace-makers [4], [5].

The correctness of a TA against high level specifications such as safety, reachability and unavoidability can be verified via model checking algorithms that are implemented in off-the-shelf tools such as UPPAAL [6] and HyTech [7]. A reachability specification requires existence of an execution that reach a target set, whereas, an unavoidability (inevitability) specification requires each execution to reach a target set. Using a model checker to verify such a property requires a complete TA model, and designing it for a complex system (or problem) is a very challenging task. Parametric timed automata (PTA) simplifies the design problem by allowing the use of parameters in place of the numeric constants. Then, the model generation is completed via parameter synthesis: find a parameter valuation such that the resulting model satisfies the specification [8]. However, parameter synthesis problems are, in general, undecidable [9].

The control of timed automata problem deals with the synthesis of a controller that monitors and affects the behavior of

the timed automata such that the resulting controlled system satisfies the specification. In literature [10], [11], [12], [13], the timed automaton is assumed to have controllable and uncontrollable inputs (transitions), and a control strategy that restricts the controllable transitions by both assigning input symbols and delay values is synthesized. In the pioneering work [10], the authors restricted the transitions of a timed automaton by solving a turn-based timed game such that the resulting automaton satisfies a safety property (avoids “bad” states). An on-the-fly algorithm for safety and reachability specifications is developed in [12], [13] to generate a feedback controller that assigns a control input or a delay value to partial runs. In [11], a controller in the form of a timed transition system is synthesized for partially observable timed automata. A template-based controller synthesis method for safety specifications is studied in [14].

In this paper, we study the problem of synthesizing a control strategy and a parameter valuation pair for a PTA such that the resulting TA satisfies an unavoidability specification. In particular, we require each run to reach a set of target locations (L_T) within a given amount of time (D) while avoiding unsafe locations (L_A). We consider control strategies that map a TA path (sequence of locations and transitions) to an input and a delay value pair. It is important to note that the controlled TA can be non-deterministic. Thus, it is necessary to ensure that each possible run satisfies the constrained unavoidability specification. To solve this problem, we represent candidate strategies symbolically as a tree with respect to the specification. Then, we employ a Mixed Integer Linear Programming (MILP) to generate a control strategy and a parameter valuation pair from a symbolic tree. Furthermore, we present an efficient algorithm to construct the candidate solutions (trees). The algorithm constructs the candidate trees in a depth first manner and employs an MILP based feasibility check to terminate the exploration along the infeasible directions. Finally, we show that the algorithm is complete under a mild non-zero assumption [10].

As summarized, in general, the parameter and controller synthesis problems are studied separately. Here, we tune parameters and restrict transitions via controller synthesis such that the resulting automaton satisfies a specification, thus we combine both problems for constrained unavoidability specifications. Parameter and controller synthesis is previously studied under safety [15] and reachability [16]. In [15], a symbolic parameter synthesis method is extended to incorporate symbolic constraints over the TA inputs, whereas in [16], a path is searched in a depth first manner with an MILP encoding to find parameters.

This work has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 798482.

Ebru Aydin Gol is with the Department of Computer Engineering, Middle East Technical University, Ankara/TURKEY {ebrugol}@metu.edu.tr

II. PRELIMINARIES

A. Timed Automata

A timed automaton (TA) [1] is a finite-state machine extended with a finite set of real-valued clocks denoted by C . A clock $x \in C$ measures the time spent after its last reset. Clock constraints define timed conditions for transitions (guard). A clock constraint is defined with the following grammar $\phi := x \sim c \mid \phi \wedge \phi$ where $x \in C$ is a clock, $c \in \mathbb{N}$ is a constant and $\sim \in \{<, \leq, >, \geq\}$ (\mathbb{N} is the set of natural numbers). A constraint is called parametric if some of the numeric constants are represented with parameters. The set of clock constraints over C is defined as $\Phi(C)$. For a parametric clock constraint ϕ with P as its set of parameters and a parameter valuation $\gamma : P \rightarrow \mathbb{N}$, $\phi(\gamma)$ is the constraint obtained by replacing parameters in ϕ with the corresponding constants from γ , e.g, for $\phi = x > p_1 \wedge y \leq p_2$, and valuation $\gamma(p_1) = 3, \gamma(p_2) = 4$, $\phi(\gamma) = x > 3 \wedge y \leq 4$.

A clock valuation $v : C \rightarrow \mathbb{R}_{\geq 0}$ assigns non-negative real values to each clock. The notation $v \models \phi$ denotes that the clock constraint ϕ evaluates to true when each clock x is replaced with the corresponding valuation $v(x)$. Two operations are defined for clock valuations: delay and reset. For a clock valuation v and $d \in \mathbb{R}_{\geq 0}$, $v + d$ is the clock valuation obtained by adding d to each clock, i.e., $(v + d)(x) = v(x) + d$ for each $x \in C$. For $\lambda \subseteq C$, $v[\lambda]$ is the clock valuation obtained after resetting each clock from λ , i.e., $v[\lambda](x) = 0$ for each $x \in \lambda$ and $v[\lambda](x) = v(x)$ for each $x \in C \setminus \lambda$.

Definition 2.1 (Timed Automata): A *timed automaton* $\mathcal{A} = (L, l_0, \Sigma, C, \Delta)$ is a tuple, where L is a finite set of locations, $l_0 \in L$ is the initial location, Σ is a finite input alphabet, C is a finite set of clocks and $\Delta \subseteq L \times \Sigma \times 2^C \times \Phi(C) \times L$ is a finite transition relation.

For a transition $e = (l_s, a, \lambda, \phi, l_t) \in \Delta$, l_s is the source location, l_t is the target location, $a \in \Sigma$ is the input symbol, λ is the set of clocks reset on e and ϕ is the guard tested for enabling e . The set of enabled input symbols in a location l is denoted by $\Sigma(l) = \{a \mid (l, a, \lambda, \phi, l') \in \Delta\}$. The set of locations that can be reached from l under input a is defined as $Post(l, a) = \{l' \mid (l, a, \lambda, \phi, l') \in \Delta\}$. A clock $x_0 \in C$ is used to measure the time passed since the start of the execution, thus it is not reset on any transition of \mathcal{A} .

A TA is called parametric (PTA) if it contains a parametric clock constraint. Given a PTA \mathcal{A} with a set of parameters P and a valuation $\gamma : P \rightarrow \mathbb{N}$ for its parameters, $\mathcal{A}(\nu)$ is the TA obtained by replacing each parameter with the corresponding constant from the valuation ν .

The semantics of a TA is given by a timed transition system (TTS). An TTS is a tuple $\mathcal{T} = (S, s_0, \Gamma, \rightarrow)$, where S is a set of states, $s_0 \in S$ is an initial state, Γ is a set of symbols, and $\rightarrow \subseteq S \times \Gamma \times S$ is a transition relation. A transition $(s, a, s') \in \rightarrow$ is also shown as $s \xrightarrow{a} s'$.

Definition 2.2 (TTS semantics for TA): Given a timed automaton $\mathcal{A} = (L, l_0, \Sigma, C, \Delta)$, the timed transition system $\mathcal{T}(\mathcal{A}) = (S, s_0, \Gamma, \rightarrow)$ is defined as follows:

- $S = \{(l, v) \mid l \in L, v \in \mathbb{R}_{\geq 0}^{|C|}\}$,

- $s_0 = (l_0, \mathbf{0})$, where $\mathbf{0}(x) = 0$ for each $x \in C$,
- $\Gamma = \Sigma \cup \mathbb{R}_{\geq 0}$, and the transition relation defined by the following rules:

- delay transition: $(l, v) \xrightarrow{d} (l, v + d)$ if $v + d \models Inv(l)$
- discrete transition: $(l, v) \xrightarrow{a} (l', v')$ if there exists $(l, a, \lambda, \phi, l') \in \Delta$ such that $v \models \phi$, and $v' = v[\lambda]$.

A run ρ of \mathcal{A} is an alternating sequence of delay and discrete transitions:

$$\rho : (l_0, v_0) \xrightarrow{d_0} (l_0, v_0 + d_0) \xrightarrow{a_0} (l_1, v_1) \xrightarrow{d_1} \dots, \quad (1)$$

where v_0 is $\mathbf{0}$, $a_i \in \Sigma$ and $d_i \in \mathbb{R}_{\geq 0}$ for each $i \geq 0$. A run is called *maximal* if it is either infinite or can not be extended by a discrete transition. The set of all runs of \mathcal{A} is denoted by $\llbracket \mathcal{A} \rrbracket$. A path π of \mathcal{A} is an interleaving sequence of locations and transitions, $\pi : l_0, e_1, l_1, e_2, \dots$. A path π is *realizable* if there exists a delay sequence d_0, d_1, \dots such that $(l_0, v_0) \xrightarrow{d_0} (l_0, v_0 + d_0) \xrightarrow{a_0} (l_1, v_1) \xrightarrow{d_1} \dots$ is a run of \mathcal{A} , and for every $i \geq 1$, the i th discrete transition is taken according to e_i , i.e., $e_i = (l_{i-1}, a_{i-1}, \lambda_{i-1}, \phi_{i-1}, l_i)$, $v_{i-1} + d_{i-1} \models \phi_{i-1}$, and $v_i = (v_{i-1} + d_{i-1})[\lambda_{i-1}]$.

In this work, we study control strategies that assign a delay value and an input symbol to a finite path:

Definition 2.3 (Control Strategy): A control strategy $\mathcal{C} : (L \times \Delta)^n \times L \rightarrow \mathbb{R}_{\geq 0} \times \Sigma$, $n \geq 0$, for a TTS $\mathcal{T}(\mathcal{A}) = (S, s_0, \Gamma, \rightarrow)$ of a TA \mathcal{A} (Defn. 2.2) maps a path π of \mathcal{A} to a delay and input symbol pair. A run ρ as in (1) is generated in closed loop with a strategy \mathcal{C} if for each $n \geq 0$:

- (a) $\mathcal{C}(l_0, e_1, \dots, l_n) = (d_n, a_n)$,
- (b) there exists $e_{n+1} = (l_n, a_n, \lambda_n, \phi_n, l_{n+1}) \in \Delta$ such that $v_n + d_n \models \phi_n$, and $v_{n+1} = (v_n + d_n)[\lambda_n]$.

For a timed automaton \mathcal{A} and a valid strategy \mathcal{C} for \mathcal{A} , the set of all runs of \mathcal{A} that is generated in closed loop with \mathcal{C} is denoted by $\llbracket \mathcal{A}_{\mathcal{C}} \rrbracket$. A strategy only limits the transitions of \mathcal{A} , thus $\llbracket \mathcal{A}_{\mathcal{C}} \rrbracket \subseteq \llbracket \mathcal{A} \rrbracket$. Note that the resulting controlled TA can be non-deterministic since there can be multiple transitions satisfying condition-(b) from Def. 2.3

Definition 2.4 ((L_T, L_A, D) -satisfaction): Let $\mathcal{A} = (L, l_0, \Sigma, C, \Delta)$ be a timed automaton, $L_T \subset L$ and $L_A \subset L$ be subsets of its locations, and $D \in \mathbb{N}$ be a time bound. A run ρ as in (1) of \mathcal{A} satisfies the reach-avoid specification with deadline (L_T, L_A, D) if there exists $i \in \mathbb{N}$ such that $l_i \in L_T$, $l_j \notin L_A$ for each $j < i$, and $v_i(x_0) \leq D$.

Remark 2.1: This specification can be expressed as a temporal logic formula with bounded until operator $(\neg L_A U_{[0, D]} L_T)$. As we focus on this particular specification, further details on the syntax and semantics of temporal logics are not included. Furthermore, an alternative way to enforce the deadline is to add $x_0 \leq D$ to each transition that ends in a location $l \in L_T$. As our goal is to enforce the overall specification via controller and parameter synthesis, we integrate this to the specification instead of the TA.

III. PROBLEM FORMULATION

Problem 3.1: Given a PTA $\mathcal{A} = (L, l_0, \Sigma, C, \Delta)$ with parameter set P , an interval $I_p \subset \mathbb{N}$ for each parameter $p \in P$, and a reach-avoid specification (L_T, L_A, D) , find a

parameter valuation $\gamma : P \rightarrow \prod_{p \in P} I_p$ and a feedback control strategy \mathcal{C} as in Defn. 2.3 such that each run $\rho \in \llbracket \mathcal{A}_C(\gamma) \rrbracket$ satisfies (L_T, L_A, D) .

Intuitively, our goal is to find a parameter valuation γ , and restrict the behaviors of $\mathcal{A}(\gamma)$ via controller synthesis, such that each remaining run reaches L_T within D time units while avoiding L_A . Our solution for this problem constructs a symbolic exploration tree for the given PTA. Central to the proposed method is the iterative construction of the symbolic model equipped with a MILP based feasibility analysis guided by the specification. This approach avoids computation of symbolic states that can not be part of the solution, i.e., not reachable by a TA $\mathcal{A}_C(\gamma)$ solving Prop. 3.1.

The developed method is presented for PTA satisfying the following assumption. The extension of the method to TA violating the assumption is explained in Remark 4.1.

Assumption 3.1: For a TA $\mathcal{A} = (L, l_0, \Sigma, C, \Delta)$

if $(l_s, a, \lambda, \phi, l_t), (l_s, a, \lambda', \phi', l_t) \in \Delta$, then $\phi = \phi'$.

The assumption states that the guards of all transitions that leave the same state (l_s) under the same input (a) are the same ($\phi = \phi'$). The following example illustrates Prob. 3.1 over a time-constrained task planning problem for a robot.

Example 3.1: An example timed automaton is shown in Fig. 1. The automaton represents a task planning problem for a robot. The robot has three tasks a, b and c . It needs to complete either a or b and then c . Each task is represented with a location in the timed automata (l_a, l_b, l_c) . In addition, it is assumed that the machines (tools/room) that the robot needs for a task can be busy. In this case, the robot waits for at least p_1 time units (locations l'_a, l'_b, l'_c). Thus, when the robot decides to perform a task, say a , it either (1) reaches location l'_a , and then it can move to l_a , or (2) it reaches l_a without waiting. The other tasks are represented similarly. The task durations have relative constraints. For example, the bound for the duration of task c should be “more than two times and less than three times” of the bounds defined for the duration of task a . These relative constraints are captured with the parametric constraints. The parameter intervals are $I_{p_1} = I_{p_2} = \{2, 3, 4\}$. Further details are given in Fig. 1. The input alphabet of the TA is $\Sigma = \{a, b, c, d\}$. The goal is to generate a strategy \mathcal{C} and a parameter valuation γ for p_1 and p_2 such that each run $\rho \in \llbracket \mathcal{A}_C(\gamma) \rrbracket$ reaches l_t in 15 time units without visiting l_d , i.e, the specification is (L_T, L_A, D) with $L_T = \{l_t\}$, $L_A = \{l_d\}$, and $D = 15$.

IV. CONTROL AND PARAMETER SYNTHESIS

In this section, we present the proposed method to solve Prob. 3.1, and prove the correctness of the result. The method first constructs an exploration tree that symbolically represents the TA runs, and then solves an optimization problem for each candidate solution (a sub-tree) represented in the tree. We first formally define the exploration tree, and the associated candidate solutions with respect to the specification (L_T, L_A, D) . Then, we present an algorithm to synthesize a control strategy-parameter valuation pair without constructing the whole tree, which can be infinite.

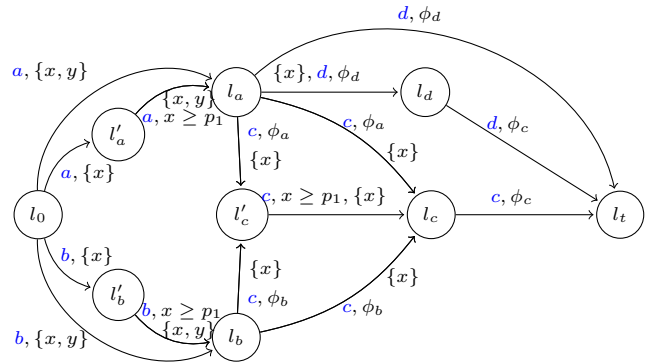


Fig. 1. The timed automaton from Ex. 3.1. l_0 is the initial location. The control inputs, reset sets and the constraints are shown next to the transitions. For example, the transition from l'_c to l_c is $(l'_c, c, \{x\}, x \geq 4, l_c)$. The parametric constraints are $\phi_a := p_2 \leq x \wedge x \leq p_1$, $\phi_b := x \geq 5p_1$, $\phi_c := 2p_1 \leq x \wedge x \leq 3p_2 \wedge y \geq 12$, and $\phi_d := x \geq p_2 \wedge y \geq 12$.

Definition 4.1 (Exploration Tree): The exploration tree of a PTA $\mathcal{A} = (L, l_0, \Sigma, C, \Delta)$ is denoted by $\mathcal{E}(\mathcal{A})$ and it is a rooted tree defined in the following way:

- The root r is labelled by the initial location l_0 .
- If $m \in \mathcal{E}(\mathcal{A})$ is a tree node labelled by $l \in L$, then for each $a \in \Sigma(l)$, and for each $(l, a, \lambda, \phi, l')$ $\in \Delta$ there exists a node $m' \in \mathcal{E}(\mathcal{A})$ that is labelled by l' and is an a -successor of m .

The label and the set of a -successors of a node m are denoted by $m(l)$ and $\mathcal{E}(\mathcal{A}, m, a)$, respectively.

An exploration tree characterizes all possible paths of \mathcal{A} . If \mathcal{A} includes a cycle, i.e., if it has a path $\pi = l_0, e_1, l_1, e_2, \dots$, with $l_i = l_j$ for some $i \neq j$, then $\mathcal{E}(\mathcal{A})$ is infinite. By the tree definition, there is a one-to-one mapping between a tree path from root to a node and an automaton path. Given a node $m \in \mathcal{E}(\mathcal{A})$, the path from root r to m is uniquely defined as $\pi_{r \rightarrow m} = m_0, \dots, m_n$ where m_0 is r , m_n is m , and for each $i = 0, \dots, n-1$ there exists $a_i \in \Sigma$ such that $m_{i+1} \in \mathcal{E}(\mathcal{A}, m_i, a_i)$. The corresponding automaton path is $\pi_{r \rightarrow m}^{\mathcal{A}} = l_0, e_1, l_1, e_2, \dots, l_n$ where for each $i = 0, \dots, n$, $l_i = m_i(l)$, and for each $i = 1, \dots, n$, $e_i = (l_{i-1}, a_i, \lambda_i, \phi_i, l_i) \in \Delta$ for some λ_i and ϕ_i (a_i is as in $\pi_{r \rightarrow m}$). Before introducing subtrees characterizing control strategies, we present an MILP based method to decide whether a path is realizable within the given time limit D . This method is extended to sub-trees for controller synthesis.

Proposition 4.1: Let $\mathcal{A} = (L, l_0, \Sigma, C, \Delta)$ be a parametric timed automaton with parameter set P , $\{I_p\}_{p \in P}$ be the set of parameter ranges, and $\pi = l_0, e_1, l_1, e_2, \dots, l_n$ be a path of \mathcal{A} . There exists a parameter valuation γ such that π is realizable on $\mathcal{A}(\gamma)$ within D time units if and only if MILP (2) with the decision variables γ_p , $p \in P$ and d_0, \dots, d_{n-1} is feasible.

$$\gamma_p \in I_p \quad \text{for each } p \in P \text{ and} \quad (2a)$$

$$d_i \in \mathbb{R}_{\geq 0} \quad \text{for each } i = 0, \dots, n-1 \quad (2b)$$

$$M(x, \pi, i) \sim g(c) \quad \text{for each } i = 1, \dots, n, \quad \text{and for each } x \sim c \text{ from } \phi_i \quad (2c)$$

$$\sum_{i=0}^{n-1} d_i \leq D, \quad (2d)$$

where $g(c)$ is γ_p if c is parameter p , otherwise, i.e., if $c \in \mathbb{N}$, $g(c) = c$, and

$$M(x, \pi, i) = d_k + d_{k+1} + \dots + d_{i-1} \text{ and} \quad (3)$$

$$k = \max(\{m \mid x \in \lambda_m, m < i\} \cup \{0\}).$$

The value of a clock x on a particular transition of π is represented as the sum of the delay variables since the last reset of x via $M(\cdot)$ (3). In particular, clock x equals to $M(x, \pi, i)$ on the i -th transition e_i along π .

Example 4.1: Consider the TA introduced in Ex. 3.1 and its path $\pi_1 = l_0, l'_b, l_b, l'_c, l_c$ (edges are omitted for brevity). Delay values d_0, d_1, d_2, d_3 , are the positive real valued variables and parameters γ_{p_1} and γ_{p_2} are the integer valued variables (with domain $\{2, 3, 4\}$) of the corresponding MILP (2). The MILP constraints are $C_1 : d_1 - \gamma_{p_1} \geq 0$, $C_2 : d_2 - 5\gamma_{p_1} \geq 0$, $C_3 : d_3 - \gamma_{p_1} \geq 0$ (from (2c)), and $C_4 : d_0 + d_1 + d_2 + d_3 \leq 15$ (2d). This MILP is feasible. Now, consider the extended path $\pi_2 = l_0, l'_b, l_b, l'_c, l_c, l_t$. It has an additional delay variable d_4 . Its constraints are C_1, C_2, C_3 as in π_1 and $C_5 : d_4 - 2\gamma_{p_1} \geq 0$, $C_6 : -d_4 + 3\gamma_{p_2} \geq 0$, $C_7 : d_2 + d_3 + d_4 \geq 0$ and $C_8 : d_0 + d_1 + d_2 + d_3 + d_4 \leq 15$. In this case, the MILP is infeasible.

Definition 4.2 (Proper Sub-tree): A proper sub-tree $\bar{\mathcal{E}}(\mathcal{A})$ of an exploration tree $\mathcal{E}(\mathcal{A})$ with respect to (L_T, L_A, D) has the following properties

- 1) The root r of $\bar{\mathcal{E}}(\mathcal{A})$ is labelled by the initial location l_0 .
- 2) For each node $m \in \bar{\mathcal{E}}(\mathcal{A})$, m is also node of $\mathcal{E}(\mathcal{A})$, and
 - a) $m(l) \in L \setminus L_A$,
 - b) if $m(l) \in L_T$, then m does not have a successor,
 - c) if $m(l) \notin L_T$, then there exists $a^m \in \Sigma(m(l))$, such that for each $(l, a^m, \lambda, \phi, l')$ $\in \Delta$ there exists $m' \in \bar{\mathcal{E}}(\mathcal{A}, m, a^m)$ with $m'(l) = l'$, and for each $b \neq a^m$, $\bar{\mathcal{E}}(\mathcal{A}, m, b) = \emptyset$.
 - d) if m is not root, there is $m' \in \bar{\mathcal{E}}(\mathcal{A})$ such that $m \in \mathcal{E}(\mathcal{A}, m', a)$.

The proper sub-tree definition ensures that locations from the avoid set L_A are not included in the tree (a), the leaf nodes are labelled by the target locations (L_T) (b), a unique input $a \in \Sigma$ is assigned to each internal node (non-leaf) and each location that is reachable under the assigned input is represented by the corresponding nodes (c), and the tree is connected (d). A proper sub-tree symbolically characterizes a candidate solution in terms of an input assignment, and integrates specifications L_T and L_A .

Next, we define a control strategy \mathcal{C} and a parameter valuation γ from a proper sub-tree $\bar{\mathcal{E}}(\mathcal{A})$ by solving a MILP over $\{\gamma_p \mid p \in P\}$ and $\{d_m \mid m \in \text{Int}(\bar{\mathcal{E}}(\mathcal{A}))\}$, where $\text{Int}(\bar{\mathcal{E}}(\mathcal{A}))$ is the set of internal (non-leaf) nodes of $\bar{\mathcal{E}}(\mathcal{A})$.

$$\gamma_p \in I_p \quad \text{for each } p \in P \text{ and} \quad (4a)$$

$$d_m \in \mathbb{R}_{\geq 0} \quad \text{for each } m \in \text{Int}(\bar{\mathcal{E}}(\mathcal{A})) \quad (4b)$$

$$M^{\mathcal{E}}(x, m') \sim g(c) \quad \text{for each } m \in \text{Int}(\bar{\mathcal{E}}(\mathcal{A}))$$

$$\text{and for each } x \sim c \text{ from } m^\phi \quad (4c)$$

$$\sum_{m \in \pi_r \rightarrow m_t} d_m \leq D \quad \text{for each } m_t \in \text{Leaf}(\bar{\mathcal{E}}(\mathcal{A})) \quad (4d)$$

where $\text{Leaf}(\bar{\mathcal{E}}(\mathcal{A}))$ is the set of leaf nodes of $\bar{\mathcal{E}}(\mathcal{A})$, ϕ^m is the guard of a transition leaving $m(l)$ under input a^m , m' is an a^m successor of m (as in Defn. 4.2-2-c), i.e., $(m(l), a^m, \lambda, \phi^m, m') \in \Delta$, $g(c)$ is as defined in (2), and

$$M^{\mathcal{E}}(x, m') = M(x, \pi_{r \rightarrow m'(l)}^A, \text{length}(\pi_{r \rightarrow m'(l)}^A)). \quad (5)$$

As in (2) and (3), each clock x is mapped to sum of the delay values since its last reset based on the path from the initial location to the position of the constraint via $M^{\mathcal{E}}(x, m')$ (5). With a slight abuse of notation, $\text{length}(\pi_{r \rightarrow m'(l)}^A)$ is used to denote the index of the last transition along the path $\pi_{r \rightarrow m'}$ (automaton path obtained from the tree path from root r to m'). Furthermore, the indices in (3) are considered as relative indices in $\pi_{r \rightarrow m'}$ and assumed to map to $\{d_m \mid m \in \text{Int}(\bar{\mathcal{E}}(\mathcal{A}))\}$ in order not to complicate the notation. Note that ϕ^m is uniquely defined by Assumption 3.1. Essentially, the tree represents several paths. The delay variables are associated with the tree nodes and they are shared among the paths. If this MILP is feasible, then each of these paths is realizable via the corresponding delay sequence. On the other hand, if the MILP (2) defined for a path is not feasible, then the tree MILP (4) can not be feasible. This property is exploited in Sec. V. Finally, even if the MILPs (2) defined for the tree paths are all feasible, the tree MILP might not be feasible. Next, we define a control strategy $\mathcal{C}(\cdot)$ from a feasible solution of this MILP, and prove that $\mathcal{C}(\cdot)$ and γ obtained from MILP (4) solves Prob. 3.1.

Proposition 4.2: Let $\mathcal{A} = (L, l_0, \Sigma, C, \Delta)$ be a parametric timed automaton with parameter set P , $\{I_p\}_{p \in P}$ be the set of parameter ranges, and $\bar{\mathcal{E}}(\mathcal{A})$ be a proper sub-tree of \mathcal{A} with respect to specification (L_T, L_A, D) . Let MILP (4) be feasible for $\bar{\mathcal{E}}(\mathcal{A})$, and d_m^* for each $m \in \text{Int}(\bar{\mathcal{E}}(\mathcal{A}))$, γ_p^* for each $p \in P$ be a solution, and let control strategy \mathcal{C} w.r.t. $\bar{\mathcal{E}}(\mathcal{A})$ and $d_m^* \in \bar{\mathcal{E}}(\mathcal{A})$ be defined as:

$$\mathcal{C}(\pi = l_0, e_1, \dots, e_n, l_n) = \quad (6)$$

$$\begin{cases} (d_m^*, a^m) & \text{if } \exists m \in \text{Int}(\bar{\mathcal{E}}(\mathcal{A})) \text{ s.t. } \pi_{r \rightarrow m}^A = \pi \\ (\perp, \infty) & \text{otherwise} \end{cases}$$

Then, each $\rho \in \llbracket \mathcal{A}_{\mathcal{C}}(\gamma^*) \rrbracket$ satisfies (L_T, L_A, D) .

For the given automaton path π , the control strategy generates the delay and control action pair (d_m^*, a^m) associated with the last node m of the corresponding tree path $\pi_{r \rightarrow m}$ ($\pi_{r \rightarrow m}^A = \pi$). Note that the strategy $\mathcal{C}(\cdot)$ (6) is defined until the target set is reached due to the particular reachability specification. As this proposition highlights, a proper sub-tree of the exploration tree characterizes a family of solutions by assigning an input to finite paths identified in the tree. Then, the solution of the MILP defines a strategy (as in (6)) by simultaneously finding parameter valuations for \mathcal{A} and delay values.

Example 4.2: The exploration tree $\mathcal{E}(\mathcal{A})$ of the TA given in Fig. 1 is finite and it has two proper sub-trees $\bar{\mathcal{E}}_1(\mathcal{A})$ and $\bar{\mathcal{E}}_2(\mathcal{A})$ such that $\bar{\mathcal{E}}_1(\mathcal{A}, r, a) \neq \emptyset$ (assigns a to l_0) and $\bar{\mathcal{E}}_2(\mathcal{A}, r, b) \neq \emptyset$ (assigns b to l_0). Note that no proper sub-tree assigns input d to a node m with $m(l) = l_a$ since $\text{Post}(l_a, d) \cap L_A \neq \emptyset$. The MILP constructed for

$\bar{\mathcal{E}}_2(\mathcal{A})$ is infeasible. In particular, $\bar{\mathcal{E}}_2(\mathcal{A})$ includes π_2 from Ex. 4.1 and the MILP (2) defined for π_2 is infeasible, which is sufficient for infeasibility of the tree MILP. On the other hand, the MILP (4) defined for $\bar{\mathcal{E}}_1(\mathcal{A})$ is feasible ($\gamma_{p_1} = 3, \gamma_{p_2} = 3$). $\bar{\mathcal{E}}_1(\mathcal{A})$ includes 4 paths that end in $\{l_t\}$: $\pi_3 : l_0, l'_a, l_a, l'_c, l_c, l_t$, $\pi_4 : l_0, l'_a, l_a, l_c, l_t$, $\pi_5 : l_0, l_a, l'_c, l_c, l_t$, $\pi_6 : l_0, l_a, l_c, l_t$. The resulting strategy as defined in Prop. 4.2 is (edges are omitted from the paths in $C(\cdot)$):

$$\begin{aligned} C(l_0) &= (0, a), C(l_0, l'_a) = (3, a), C(l_0, l_a) = (3, c), \\ C(l_0, l'_a, l_a) &= (3, c), C(l_0, l_a, l'_c) = (3, c), C(l_0, l_a, l_c) = (9, c), \\ C(l_0, l'_a, l_a, l'_c) &= (3, c), C(l_0, l'_a, l_a, l_c) = (9, c), \\ C(l_0, l_a, l'_c, l_c) &= (6, c), C(l_0, l'_a, l_a, l'_c, l_c) = (6, t), \end{aligned}$$

Remark 4.1: For a TA violating Assumption 3.1, a strategy can be computed by considering all guards associated with the location and control input in (4c). In particular, consider location l_s and input a such that $(l_s, a, \lambda, \phi, l_t), (l_s, a, \lambda', \phi', l'_t) \in \Delta$, with $\phi \neq \phi'$. Adding a constraint as in (4c) to the MILP for each inequality from $\phi \wedge \phi'$ guarantees that each symbolic path encoded in the tree (Defn. 4.2-c) will be realizable when the MILP is feasible.

V. SYNTHESIS ALGORITHMS

In this section, we present an iterative method to construct the exploration tree as in Defn. 4.1, and a control strategy via a proper sub-tree (Defn. 4.2) as shown in Prop. 4.2. The method is summarized in Alg. 1. The algorithm starts with the initialization of the root node (line 1) and expands the tree recursively by analyzing the input symbols and the corresponding transitions in a depth-first manner (described in Alg. 2). For each considered input symbol, the feasibility of the corresponding automaton path is checked via MILP (2) (line 5 of Alg. 2). Thus, the exploration only continues through promising directions. Once the exploration tree construction terminates, MILP (4) is solved for each proper subtree until a feasible solution is found (lines 3-7 of Alg. 1).

Algorithm 1 Synthesis($\mathcal{A}, \mathcal{P}, (L_T, L_A, D)$)

Require: A PTA $\mathcal{A} = (L, l_0, \Sigma, C, \Delta)$ with parameter set P , $\mathcal{P} = \{I_p \mid p \in P\}$ parameter intervals for each $p \in P$, specification (L_T, L_A, D) .

Ensure: Control strategy \mathcal{C} and parameter valuation γ such that each run from $\llbracket \mathcal{C}(\mathcal{A}(\gamma)) \rrbracket$ satisfies (L_T, L_A, D) .

- 1: Initialize root r of $\mathcal{E}(\mathcal{A})$ with $r(l) = l_0$.
 - 2: $ps = \text{ForwardAnalysis}(r, r, \mathcal{A}, P, \mathcal{P}, (L_T, L_A, D))$
 - 3: **for each** $i \in \{1, \dots, ps\}$ **do** \triangleright Enumerate each possible solution.
 - 4: $\bar{\mathcal{E}}_i(\mathcal{A}) = \text{GetSolutionTree}(i)$
 - 5: $\mathbf{d}, \gamma, \text{soln} = \text{Synthesis}(\bar{\mathcal{E}}_i(\mathcal{A}))$
 - 6: **if soln then return** $\mathcal{C}, \gamma = \text{Controller}(\bar{\mathcal{E}}_i(\mathcal{A}), \mathbf{d}, \gamma)$
 - 7: **end for**
 - 8: **return** No Solution
-

The forward analysis method (Alg. 2) takes an exploration tree node m as input, constructs the sub-tree rooted at m recursively, and returns the number of different sub-trees

Algorithm 2 ForwardAnalysis($r, m, \mathcal{A}, \mathcal{P}, (L_T, L_A, D)$)

Require: r is the root node, m is a node reachable from r , $\mathcal{A}, P, \mathcal{P}$, and (L_T, L_A, D) are as in Alg. 1.

Ensure: Construct tree, and return the number of possible proper trees that include m .

- 1: **if** $m(l) \in L_T$ **then return** 1
 - 2: **if** $m(l) \in L_A$ **then return** 0
 - 3: $ps^m = 0$ \triangleright The number of candidate solutions for m .
 - 4: **for each** $a \in \Sigma(m(l))$ **do** \triangleright For each admissible action.
 - 5: **if** $\text{IsFeasible}(\text{root} - m - a)$ **then**
 - 6: $ps = 1$
 - 7: **for each** $l' \in \text{Post}(m(l), a)$ **do**
 - 8: Create m' with $m'(l) = l'$
 - 9: Set $\mathcal{E}(\mathcal{A}, m, a) = \mathcal{E}(\mathcal{A}, m, a) \cup \{m'\}$
 - 10: $ps = ps \times \text{ForwardAnalysis}(\text{root}, m', \mathcal{A}, \mathcal{P}, S)$
 - 11: **end for**
 - 12: **if** $ps == 0$ **then** \triangleright No soln. from input a
 - 13: Delete $\mathcal{E}(\mathcal{A}, m, a)$ \triangleright Remove each sub-tree.
 - 14: **else**
 - 15: $ps^m = ps^m + ps$
 - 16: **end if**
 - 17: **end if**
 - 18: **end for**
 - 19: **return** ps^m
-

that can be part of a proper sub-tree (a candidate solution Defn. 4.2) through m . It can be regarded as the number of different candidate solutions that contain m . Reaching a location from the target set (line 1) or from the avoid set (line 2) terminates the recursive construction. Otherwise, each admissible input is considered for the node (line 4). First, the feasibility of the timed automaton path induced by the exploration tree path from root to m and input a (line 5) is checked via MILP (2) from Prop. 4.1 (e.g. considering a location $l' \in \text{Post}(m(l), a)$ as the final location of the path). If this MILP is not feasible, i.e., the path is not realizable by any parameter valuation, the corresponding sub-trees of the exploration tree ($m' \in \mathcal{E}(\mathcal{A}, m, a)$) are not constructed. On the other hand, if it is feasible, the exploration continues for each $l' \in \text{Post}(m(l), a)$ recursively (lines 7-11).

The number of candidate solutions (proper sub-trees) associated with node m and input a , denoted by ps , is the product of the number of solutions associated with the a -successors of m , i.e. $ps = \prod_{m' \in \mathcal{E}(\mathcal{A}, m, a)} ps^{m'}$. Note that each combination of these alternative choices can yield a different proper sub-tree of $\mathcal{E}(\mathcal{A})$. Furthermore, if $ps^{m'}$ is 0 for a node $m' \in \mathcal{E}(\mathcal{A}, m, a)$, then the specification is not satisfiable through m' . As \mathcal{A} can reach $m'(l)$ non-deterministically when a is applied at m , ps is also set to 0, and each sub-tree associated with $m' \in \mathcal{E}(\mathcal{A}, m, a)$ is removed (line 12). Otherwise, the number of possible solutions through m is incremented by ps reflecting the sub-trees assigning a to m .

A sub-tree constructed by Alg. 2 (extracted in line 4 of Alg. 1) satisfies conditions of Defn. 4.2. The first condition (1) follows from the initialization in line 1 of Alg. 1.

The condition that a node of the sub-tree belongs to the exploration tree (e.g. cond. (2)) trivially holds since nodes are added via $Post(m(l), a)$ relation (line 7). The first base condition (line 1) ensures that a child node is not constructed for a node m when $m(l) \in L_T$ (2-b). The second base condition (line 2) ensures that $m(l) \notin L_A$ for any $m \in \tilde{\mathcal{E}}_i(\mathcal{A})$ since nodes with 0 number of possible solutions are removed (see line 10 and 12) (2-a). The connectivity (2-d) and the control assignment (2-c) conditions are satisfied by the enumeration performed with respect to the number of possible proper sub-trees (ps).

Note that since \mathcal{A} is non-deterministic, the feasibility analysis performed for paths (line 5 of Alg. 2) is not sufficient to generate a control strategy. However, as the specification requires each run to satisfy the property, it is sufficient to prune violating runs. In particular, the feasibility of MILP from (2) is a necessary condition for the feasibility of the MILP (4) of the proper sub-trees that contain the path. Alg. 2 returns the number ps of the proper subtrees of $\mathcal{E}(\mathcal{A})$ that pass the path based feasibility check. In Alg. 1, each proper sub-tree $\tilde{\mathcal{E}}_i(\mathcal{A})$ is extracted (line 4), MILP (4) for the tree $\tilde{\mathcal{E}}_i(\mathcal{A})$ is solved (line 5), and if this MILP is feasible, a control strategy $\mathcal{C}(\cdot)$ as in (6) w.r.t. the MILP solution is returned. By Prop. 4.2, we conclude that a strategy generated by Alg. 1 solves Prop. 3.1.

Alg. 1 exhaustively searches all possible strategies via Alg. 2. Thus, if Alg. 1 reaches line 8, then a solution to Prop. 3.1 does not exist. Consequently, when the algorithm terminates, either a strategy and parameter valuation pair solving Prop. 3.1 is generated or a solution does not exist. A final possibility is that the algorithm might not terminate. In particular, if \mathcal{A} has a loop, then $\mathcal{E}(\mathcal{A})$ is infinite, and in this case Alg. 2 might fail to terminate. Next, we state an assumption that avoids zero behavior by guaranteeing that the time progresses at each cycle ($l_i = l_j$ on a path):

Assumption 5.1: For a TA $\mathcal{A} = (L, l_0, \Sigma, C, \Delta)$, if an infinite run $\pi : l_0, e_1, l_1, e_2, \dots$ is realizable by a delay sequence d_0, d_1, \dots , then for a positive constant ϵ :

$$\text{if } l_i = l_j, j > i \text{ then } d_i + d_{i+1} \dots d_{j-1} > \epsilon$$

Finally, we can guarantee that Alg. 1 finds a solution when one exists if timed automata \mathcal{A} satisfies Assumption 5.1. By the well-known pigeon hole principle, a path of length $|L| \cdot k$ includes a location at least k times. By Assumption 5.1, if such a path is realizable, then the total duration of the corresponding delay variables are lower bounded by $k \cdot \epsilon$. Thus, the length of a path induced by the exploration tree path is upper bounded by $\frac{D}{\epsilon}$, as otherwise the resulting MILP (2) is infeasible due to the time bound D . Consequently, if Assumption 5.1 holds, the depth of the tree generated by Alg. 2 is bounded and the synthesis algorithm always terminates.

Example 5.1: We run Alg. 1 on the TA \mathcal{A} introduced in Ex. 3.1. As shown in Ex. 4.2, path π_2 is infeasible. Thus, ps is set to 0 for root r and input b . In addition, $ps = 0$ is assigned to trees with $\mathcal{E}(\mathcal{A}, m, d) \neq \emptyset$ in line 2. As MILPs (2) defined for paths π_3, π_4, π_5 and π_6 are feasible,

$ps = 1$ in Alg. 2 (line 2). As illustrated in Ex. 4.2, the corresponding MILP is feasible and results in a control strategy solving Prop. 3.1.

VI. CONCLUSION

In this paper, we studied the controller and parameter synthesis problem for a PTA under unavailability specifications with a deadline. We presented the candidate solutions symbolically with sub-trees of the exploration tree, and developed an algorithm to generate such trees. The algorithm is based on depth-first analysis and it uses an iterative feasibility check to terminate the exploration along infeasible directions. Finally, we presented an MILP based method to generate a feedback control strategy and a parameter valuation pair from a sub-tree such that the resulting TA satisfies the given specification.

REFERENCES

- [1] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.
- [2] F. Wang, "Formal verification of timed systems: a survey and perspective," *Proceedings of the IEEE*, vol. 92, pp. 1283–1305, Aug 2004.
- [3] A. David, J. Illum, K. G. Larsen, and A. Skou, "Model-based framework for schedulability analysis using UPPAAL 4.1," in *Model-based design for embedded systems*, pp. 117–144, 2009.
- [4] M. Kwiatkowska, A. Mereacre, N. Paoletti, and A. Patanè, "Synthesising robust and optimal parameters for cardiac pacemakers using symbolic and evolutionary computation techniques," in *Hybrid Systems Biology* (A. Abate and D. Šafránek, eds.), (Cham), pp. 119–140, Springer International Publishing, 2015.
- [5] Z. Jiang, M. Pajic, R. Alur, and R. Mangharam, "Closed-loop verification of medical devices with model abstraction and refinement," *Int. J. Softw. Tools Technol. Transf.*, vol. 16, p. 191?213, Apr. 2014.
- [6] G. Behrmann, A. David, K. G. Larsen, J. Hakansson, P. Petterson, W. Yi, and M. Hendriks, "Uppaal 4.0," in *International Conference on the Quantitative Evaluation of Systems*, QEST '06, (Washington, DC, USA), pp. 125–126, IEEE Computer Society, 2006.
- [7] T. A. Henzinger, J. Preussig, and H. Wong-Toi, "Some lessons from the hytech experience," in *IEEE Conference on Decision and Control (Cat. No.01CH37228)*, vol. 3, pp. 2887–2892, 2001.
- [8] A. Jovanovic, D. Lime, and O. H. Roux, "Integer parameter synthesis for real-time systems," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 445–461, 2015.
- [9] E. André, "What's decidable about parametric timed automata," *Int. J. Softw. Tools Technol. Transf.*, vol. 21, pp. 203–219, Apr. 2019.
- [10] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis, "Controller synthesis for timed automata," *IFAC Proceedings Volumes*, vol. 31, no. 18, pp. 447 – 452, 1998. 5th IFAC Conference on System Structure and Control 1998 (SSC'98), Nantes, France, 8-10 July.
- [11] P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit, "Timed control with partial observability," in *Computer Aided Verification* (W. A. Hunt and F. Somenzi, eds.), pp. 180–192, Springer Berlin Heidelberg, 2003.
- [12] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime, "Efficient on-the-fly algorithms for the analysis of timed games," in *CONCUR 2005 – Concurrency Theory* (M. Abadi and L. de Alfaro, eds.), pp. 66–80, Springer Berlin Heidelberg, 2005.
- [13] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime, "Uppaal-tiga: Time for playing games!," in *Computer Aided Verification* (W. Damm and H. Hermanns, eds.), pp. 121–125, Springer Berlin Heidelberg, 2007.
- [14] B. Finkbeiner and H.-J. Peter, "Template-based controller synthesis for timed systems," in *Tools and Algorithms for the Construction and Analysis of Systems* (C. Flanagan and B. König, eds.), pp. 392–406, Springer Berlin Heidelberg, 2012.
- [15] A. Étienne, M. Knapik, W. Penczek, and L. Petrucci, "Controlling actions and time in parametric timed automata," in *2016 16th International Conference on Application of Concurrency to System Design (ACSD)*, pp. 45–54, 2016.
- [16] E. A. Gol, "Control synthesis for parametric timed automata under reachability," *Turk J Elec Eng & Comp Sci*, pp. 1–14, 2021 (to appear).