Parametric Optimization of Open Real-Time Systems

Farn Wang[*]
Institute of Information Science, Academia Sinica, Taipei, Taiwan, ROC
+886-2-27883799 x 1717 FAX:+886-2-27824814 farn@iis.sinica.edu.tw

Hsu-Chun Yen
Dept. of Electrical Engineering, National Taiwan University, Taipei, Taiwan, ROC
yen@cc.ee.ntu.edu.tw

Abstract

For controllable timed automata, a general parametric optimization framework based on automata-theory is proposed. The framework is general enough to incorporate both the parametric analysis problem and the controller synthesis problem of computer systems. We propose an algorithm for the construction of the characterization of the parameter constraints and controller synthesis, which in turn yields a linear programming solution to parametric optimization.

1 Introduction

As increasing efforts have been devoted to applying CAV techniques to real-world systems, it becomes urgent to design appropriate models and analytical techniques to deal with *parametric optimization* of real-time systems. Many (if not most) of the conventional CAV techniques are only capable of classifying a given system as 'good' or 'bad.' To make things even more complicated, the behavior of many of the real-world systems is often influenced by various engineering constraints, e.g. assumptions on environments in which the systems reside. Suppose the performance of a system is evaluated once with respect to a given constraint setting. Once the constraint changes, traditionally either the evaluation process is restarted, or the performance under the new constraint is calculated using the so-called *extrapolation* technique utilizing results from known system constraints. The former is somewhat time-consuming, whereas the latter suffers from imprecision. In addition, neither technique is appropriate for answering a question like: *find the environment assumptions under which the system performs best.* It is therefore highly desirable to employ an evaluation strategy that is *parametric* in nature. That is, the variations of the engineering constraints are treated as parameters, and the evaluation ends up including such parameters as part of the performance measure.

We demonstrate in this paper a general framework within which parametric optimization is carried out in a *parametric* fashion for real-time systems modeled by *controllable timed automata*. As we shall see later, our framework incorporates both *parametric analysis* and *controller synthesis*, which are two issues that have received increasing attention in the CAV community lately [3, 4, 5, 6, 7, 11, 13, 16, 17, 18]. Aside from being more general than the problem of parametric analysis or controller synthesis alone, a unique feature of our solution to parametric optimization lies in that we are able to construct a characterization of the parameter constraints and controller synthesis, which in turn yields a linear programming solution

[*]To whom all correspondence should be sent.

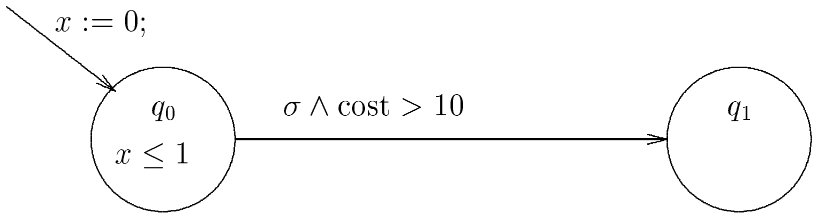


Figure 1: A simple controllable timed automaton.

to parametric optimization. Such a characterization is valuable both in the construction of a solution controller and in the derivation of the optimal performance. By doing so, re-evaluation of the performance of a system under a new constraint setting is as easy as solving the parameterized inequalities with respect to a new set of parameters. More interestingly, by encapsulating the environment's parameters into our framework of performance evaluation, it becomes feasible to find out the best system performance by solving an optimization version of the parametric inequalities.

Example 1 **A simple automaton.** To give the reader a better feel for the issue of parametric optimization of controllable timed automata, consider Figure 1 in which a simple controllable timed automaton is shown. Two operation modes are represented by the ovals in which we have mode names q_0 and q_1, and invariance conditions $x \leq 1$ and *true*, respectively. Between the modes, there is an arc for a transition labeled with a triggering condition (above). The triggering condition contains a special symbol σ which represents the enabling signal from the controller. In this paper, we shall adopt the approach in [6, 7, 13] which assumes that σ is an uninterpreted Boolean function of *regions* [1]. Notice that the automaton is *parametric* in the sense that "cost" is a static parameter for some optional functions of the automaton.

An example specification which we may want to analyze is written in *PCTL* (*Parametric CTL*, defined in Section 2) as follows.

$$\phi \equiv (\text{cost} < 100) \wedge (\theta \geq 1) \wedge (\forall\Box_{<\theta} q_0)$$

The formula says that the cost must be less than 100 dollars, parameter θ is greater than or equal to 1, and we want to find out the characterization of θ such that q_0 is true in all computations within θ time units from the initial state. As the transition from q_0 to q_1 is 'controlled' by the environment through the control symbol σ, whether ϕ holds depends not only on the values of the static parameters but also on the control policy imposed on the automaton by the controller (i.e., environment).

Suppose we are given an optimization metric like 'cost $-\ \theta$.' For the aforementioned 'parametric' timed automaton, specification, and optimization metric, *parametric optimization* in our setting boils down to finding a control policy under which the optimization metric is maximized for a parameterized automaton meeting the specification.

Our framework is capable of analyzing trace-oriented optimization problems of real-time systems. For example, with the same automaton used in figure 1, we may have another specification $\phi' = \text{cost} < 12 \wedge \forall\Box(q_0 \rightarrow \forall\Diamond_{\leq\theta} q_1)$, which says that q_1 will eventually be true in time $\leq \theta$ whenver q_0 is true, and an optimization metric $-\theta$. Then the parametric optimization problem asks for the deadline value in all computations, from a q_0 state to a q_1 state, subject to the restriction of ϕ. Thus it is easy to see that our framework is more general than previous ones like [10].

In example 4 of subsection 3.5 after we have presented our alogirhtms, we shall present the answers to the parametric optimization problems with these two just-mentioned sets of specifications and optimization metrics. ||

Our parametric optimization problem can be thought of as a generalization of both *parametric analysis* ("what parameter settings make a system correct?") [3, 4, 11, 16, 17, 18] and *controller synthesis* ("what controller if any induces a correct behavior?") [5, 6, 7, 13] of timed automata. For parametric analysis, parameter variables are associated with either temporal operators or timed automata, and problems such as *'Does there* exist *a valuation of the parameters so that the system meets a given property?'* or *'Is it the case that for* all *valuations of the parameters the given property always holds for the system?'* are being thought. It has been shown that for parametric timed automata in which parameter variables are allowed to be compared with clocks, the verification problem is in general undecidable [4]. If, however, parameter variables are in existence in the specification (i.e., temporal formulas) only, then the problem becomes decidable [4]. In the context of parametric analysis, Alur *et al.* [3] considered the so-called 'model measuring' problem for *parametric linear time temporal logic* (PLTL). Model measuring is an extension of the standard model checking problem in that the latter only returns 'yes/no' answers, whereas the former provides answers to a number of questions regarding the set of parameter valuations for which the given specification is fulfilled. Emerson and Trefler [11], on the other hand, investigated the model checking problem for *parameterized real-time computation tree logic* (PRTCTL). In [3], PLTL is defined over conventional Kripke structures, whereas in [11], both untimed and (discrete) timed structures are considered for PRTCTL. With respect to dense-time automata, Wang [16] gave a complete characterization of the set of parameter valuations satisfying a specification expressed in *parametric computation tree logic* (PCTL) in terms of a set of linear inequalities. The work of [16] has subsequently been generalized in [17], which shows that parametric analysis remains decidable for the model in which timed automata are augmented by *static parameters* (i.e., nontiming parameters) and temporal formulas are parameterized by both timing and nontiming parameters. In comparison with its predecessors in parametric analysis of timed automata, Wang's algorithm in [17], based upon the technique of dynamic programming, is easier to understand, implement, and analyze.

What makes controller synthesis an important issue is that many interesting real-world systems tend to be *open* in nature, meaning that their behaviors are influenced by the environment. Since the seminal work of Ramadge and Wonham [14], the use of *automata* and *formal languages* to reason about *controllability* of *discrete event dynamic systems* has received much attention in the control community in the past decade. Being recognized as one of the most popular models for representing real-time systems, *timed automata* [2] have naturally become the underlying model for which various *controller synthesis* issues are investigated, aside from a very successful role such a model has played in the *verification* aspects of real-time systems. Consider a dynamic system (modeled by a timed automaton) whose behaviors are to be *controlled* in a certain way so as to meet certain predefined requirements. The *controller synthesis* problem, simply speaking, is to find out whether, for a given system, there is a controller through which the interaction between the system and the controller results in only computations of 'good' behavior. (If such a controller exists, it is also desirable to construct it effectively.) The interested reader is referred to [6] for a *symbolic* approach for controller synthesis. As opposed to providing only yes/no answers in the conventional framework of controller synthesis, a recent article [5] dealt with *quantitative* properties of behaviors for controllable timed automata.

In this paper, we move a step further from previous work [3, 4, 5, 6, 7, 11, 13, 16, 17, 18] by considering the controller synthesis issue for *parametric timed automata* with respect to system requirements specified by *parametric computation-tree logic (PCTL)* (see [16, 17]). By explicitly allowing static parameters in our model, a richer parametric optimization framework, in comparison with that of [5], is provided. To the best of our knowledge, our work is the first that addresses parametric approach for the optimization of synthesized controller. Unlike the (fixed-point based) backward reachability approach employed in [13] for the controller synthesis problem of timed automata, we generalize the parametric analysis technique devised in [17] to derive, for a given parametric timed automaton A and a PCTL formula ϕ, a complete characterization of the parameter constraint and controller synthesis which is satisfiable if and only if there exists a controller forcing A to meet property ϕ. The characterization contains the information for both controller synthesis and parameter constraint, which enables us to formulate a unified framework in integer linear programming for *parametric optimization* of real-time systems. We feel that our approach is interesting in its own right and may have applications to the analysis of related problems for real-time systems.

The remainder of this paper is organized as follows. Section 2 introduces the model of *statically parametric plants*, *parametric computation tree logic*, as well as the *parametric optimization problem.* An algorithm, together with an illustrating example, is demonstrated in Section 3 for solving the parametric optimization problem. Section 4 concludes our work.

2 Parametric optimization problem

A parametric optimization problem instance in our framework is given as a tuple $\langle A, \phi, \lambda \rangle$ such that A is a controllable timed automaton (statically parametric plant, defined in subsection 2.1) for the description of the system behavior, ϕ is a temporal logic formula for the requirements on the system behaviors, and λ is a linear expression of parameters for the performance measurement. The aim of the problem is to find a valuation (interpretation) of the parameters that maximizes λ and makes A satisfy ϕ under the interpretation for some control strategy. A framework for minimization of λ can be similarly defined by changing the signs of coefficients in λ.

2.1 Statically parametric plant (SPP)

An SPP is a timed automaton extended with linear constraints of static parameters and transition controls. In an SPP, people may combine control signals, timing inequalities on clock readings, and linear inequalities of static parameters to write the invariance and transition conditions. Such a combination is called a *state predicate* and is defined formally in the following. Given a set P of basic propositions, a set X of clocks, and a set H of parameter variables, a *state predicate* η of P, X, and H has the following syntax rules.

$$\eta ::= \mathit{false} \;\mid\; p \;\mid\; x - y \sim c \;\mid\; x \sim c \;\mid\; \sum a_i \alpha_i \sim c \;\mid\; \eta_1 \vee \eta_2 \;\mid\; \neg \eta_1$$

where $p \in P$, $x, y \in X$, $a_i, c \in \mathcal{N}$, $\alpha_i \in H$, $\sim \in \{\leq, <, =, \geq, >\}$, and η_1, η_2 are state predicates. Notationally, we let $B(P, X, H)$ be the set of all state predicates on P, X, and H. Note the parameter variables considered in H are static because their values do not change with time in computation of an automaton. A state predicate with only $\sum a_i \alpha_i \sim c$ type literals is called *static*.

Definition 1 (Statically parametric plant): A *statically parametric plant* (SPP) is a tuple $(Q, q_0, X, H, \mu, E, \tau, \pi)$ with the following restrictions.

- Q is a finite set of *modes* (operation modes, or control locations).
- $q_0 \in Q$ is the *initial mode.*
- X is a set of clocks with readings in $\mathcal{R}^+$, i.e., the set of nonnegative reals.
- H is a set of parameter variables with values in $\mathcal{N}$, i.e., the set of nonnegative integers.
- μ is a mapping from Q such that for each $q \in Q$, $\mu(q) \in B(\emptyset, X, H)$ is the invariance condition true in q.
- $E \subseteq Q \times Q$ is the set of transitions.

- $\tau : E \mapsto B(\{\sigma\}, X, H)$ is a mapping which defines the transition-triggering conditions. Here σ is a control signal symbol representing the enabling/disabling signal from the controller. Conceptually, σ is an uninterpreted Boolean function of states whose presence on edge e suggests such an edge to be '*controllable*'.
- $\pi : E \mapsto 2^X$ defines the set of clocks to be reset during each transition. ||

Figure 1 displays a simple example of an SPP in which $Q = \{q_0, q_1\}$, $X = \{x\}$, $H = \{cost\}$, and σ is the control signal symbol associated with the only transition of the plant. Notice that in this example, the invariance conditions (defined by μ) associated with q_0 and q_1 are $x \leq 1$ and *true*, respectively, although in general, parameter variables are allowed to take part in the invariance conditions. An SPP starts its execution at its initial mode q_0. We shall assume that initially, all clocks read zero. In between mode transitions, all clocks increment their readings at a uniform rate. A transition of an SPP may be fired when the triggering condition of the transition is satisfied. During a transition from modes q_0 to q_1, for each $x \in \pi(q_0, q_1)$, the reading of x is reset. It is worthy of pointing out that in our setting, it is legal to let time elapse even in the presence of an enabled transition. (The reader is encouraged to contrast our model with that of [5] in which transition firings are assumed to be 'urgent.') For instance, in state q_0 with $x = 0.5$ for the SPP depicted in Figure 1, the computation may either stay in q_0 while letting the clock run, or exercise the transition from q_0 to q_1, provided that $cost > 10$ and the controller assigns *true* to σ. The behavior of an SPP depends not only on the interpretation of the parameter variables, but also on the control policy enforced by the environment during the course of its computation. The interested reader should notice that although the triggering condition hinders on the control signal symbol σ as well as on the static parameter *cost*, they play entirely different roles as far as how they enable or disable the transition is concerned. The values of static parameters are given prior to the execution of the SPP, whereas the control signals are disabled/enabled on a step-by-step basis by the controller as the computation proceeds.

Note that we allow control signals σ to participate in the construction of triggering conditions. This is different from the controller definition in [13] in which at any moment at most one controllable transition can be enabled.

Definition 2 (State): A *state* of SPP $A = (Q, q_0, X, H, \mu, E, \tau, \pi)$ is a pair (q, ν) such that $q \in Q$ and ν is a mapping from X to $\mathcal{R}^+$ (i.e., ν represents the current clock readings). Let U_A be the state set of A. ||

Definition 3 (Controller): Given an SPP A, a *controller* χ is a Boolean function $U_A \mapsto \{true, false\}$ which intuitively denotes the action on the control signal (i.e., σ) to enable or disable transitions according to the current states. Since the controller does not depend on the history, it is also a *simple controller.* ||
We write $\sigma_\chi(s)$ to denote the truth value of σ at state s under controller χ. (If χ is clear from the context, $\sigma(s)$ is used as a shorthand.) Apparently, it is useless to enable a transition in a mode other than the source mode of the transition. Thus, at a given state (q, ν), it is reasonable to only consider controller χ such that χ disables all the transitions whose source nodes are not in mode q. It should be noted that the same SPP may generate different computations under different controllers and interpretations of its parameter variables.

Definition 4 (Interpretation): An *interpretation* $\mathcal{I}$ for H is a mapping from $\mathcal{N} \cup H$ to $\mathcal{N}$ such that for all $c \in \mathcal{N}$, $\mathcal{I}(c) = c$. ||

Definition 5 (Satisfaction of state predicate): A state (q,ν) satisfies state predicate η under controller χ and interpretation $\mathcal{I}$, written as $(q,\nu) \models^{\chi}_{\mathcal{I}} \eta$, iff

- $(q,\nu) \not\models^{\chi}_{\mathcal{I}} \mathit{false}$;
- $(q,\nu) \models^{\chi}_{\mathcal{I}} \sigma$ iff $\chi((q,\nu))$ (in words, the control signal σ is enabled by χ at (q,ν));
- $(q,\nu) \models^{\chi}_{\mathcal{I}} x - y \sim c$ iff $\nu(x) - \nu(y) \sim c$;
- $(q,\nu) \models^{\chi}_{\mathcal{I}} x \sim c$ iff $\nu(x) \sim c$;
- $(q,\nu) \models^{\chi}_{\mathcal{I}} \sum a_i\alpha_i \sim c$ iff $\sum a_i\mathcal{I}(\alpha_i) \sim c$;
- $(q,\nu) \models^{\chi}_{\mathcal{I}} \eta_1 \vee \eta_2$ iff $(q,\nu) \models^{\chi}_{\mathcal{I}} \eta_1$ or $(q,\nu) \models^{\chi}_{\mathcal{I}} \eta_2$; and
- $(q,\nu) \models^{\chi}_{\mathcal{I}} \neg\eta_1$ iff $(q,\nu) \not\models^{\chi}_{\mathcal{I}} \eta_1$.

If for all χ, we have $(q,\nu) \models^{\chi}_{\mathcal{I}} \eta$, then we may write $(q,\nu) \models_{\mathcal{I}} \eta$. If for all $\mathcal{I}$, we have $(q,\nu) \models_{\mathcal{I}} \eta$, then we may write $(q,\nu) \models \eta$. ||

Definition 6 (Transitions): Given two states $(q,\nu),(q',\nu')$, there is a *mode transition* from ν to ν' in A under controller χ and interpretation $\mathcal{I}$, in symbols $(q,\nu) \rightarrow^{\chi}_{\mathcal{I}} (q',\nu')$, iff $(q,q') \in E$, $(q,\nu) \models^{\chi}_{\mathcal{I}} \mu(q) \wedge \tau(q,q')$, $(q',\nu') \models^{\chi}_{\mathcal{I}} \mu(q')$, $\forall x \in \pi(q,q')(\nu'(x) = 0)$, and $\forall x \notin \pi(q,q')(\nu'(x) = \nu(x))$. In words, for the transition $(q,\nu) \rightarrow (q',\nu')$ along the edge (q,q') to take place under controller χ and interpretation $\mathcal{I}$, it must be the case that the starting and ending invariance conditions (i.e., $\mu(q)$ and $\mu(q')$, respectively) hold in modes q and q', respectively, and the associated triggering condition $\tau(q,q')$ is met as well. Meanwhile, all the clocks specified in $\pi(q,q')$ are reset to zero, while the remaining clock readings remain unchanged. (That is, transition firing is assumed to take place instantaneously.) ||

For ease of expression, given a state ν and a $\delta \in \mathcal{R}^+$, we let $(q,\nu)+\delta = (q,\nu+\delta)$ be the state that agrees with (q,ν) in every aspect except for all $x \in X$, $\nu(x)+\delta = (\nu+\delta)(x)$.

Definition 7 ((q,ν)-run of controlled and interpreted SPP): An infinite computation of $A = (Q,q_0,X,H,\mu,E,\tau,\pi)$ starting at state (q,ν) under controller χ and interpretation $\mathcal{I}$ is called a (q,ν)-*run* and is a sequence $((q_1,\nu_1,t_1),(q_2,\nu_2,t_2),\ldots)$ such that

- $q = q_1$ and $\nu = \nu_1$;
- for each $t \in \mathcal{R}^+$, there is an $i \in \mathcal{N}$ such that $t_i \geq t$ (meaning that the run is diverging);
- for each integer $i \geq 1$ and for each real $0 \leq \delta \leq t_{i+1} - t_i$, $(q_i,\nu_i)+\delta \models^{\chi}_{\mathcal{I}} \mu(q_i)$ (meaning that the invariance condition $\mu(q_i)$ continuously holds throughout the time interval $[t_i,t_{i+1}]$); and
- for each $i \geq 1$, A goes from (q_i,ν_i) to (q_{i+1},ν_{i+1}) because of
  - a mode transition, i.e., $t_i = t_{i+1} \wedge (q_i,\nu_i) \rightarrow^{\chi}_{\mathcal{I}} (q_{i+1},\nu_{i+1})$; or
  - time passage, i.e., $t_i < t_{i+1} \wedge (q_i,\nu_i) + t_{i+1} - t_i = (q_{i+1},\nu_{i+1})$. ||

2.2 Parametric computation-tree logic

Parametric Computation Tree Logic (PCTL) is used for specifying the design requirements and is defined with respect to a given SPP. Suppose we are given an SPP $A = (Q,q_0,X,H,\mu,E,\tau,\pi)$. A PCTL formula ϕ for A has the following syntax rules.

$$\phi ::= \eta \quad | \quad \phi_1 \vee \phi_2 \quad | \quad \neg\phi_1 \quad | \quad \exists\phi_1\mathcal{U}_{\sim\theta}\phi_2 \quad | \quad \forall\phi_1\mathcal{U}_{\sim\theta}\phi_2$$

Here $\eta \in B(Q,X,H)$, ϕ_1 and ϕ_2 are PCTL formulas, and $\theta \in \mathcal{N} \cup H$. Note that mode names are used as basic propositions for the specification of timely mode changes. $\exists$ means "there exists a computation." $\forall$ means "for all computations." $\phi_1\mathcal{U}_{\sim\theta}\phi_2$ means that along a computation, ϕ_1 is true until ϕ_2 becomes true and ϕ_2 happens with time $\sim \theta$. For example, in a requirement like cost = deadline+5 $\wedge \forall q_0 \mathcal{U}_{\leq \text{deadline}} q_1$, parameters "cost" and "deadline" are related and we require that for all computations, q_0 is true until q_1 becomes true in "deadline" time units.

The parameter variable subscripts of modal formulas can also be used as parameter variables in SPP. Also we adopt the following standard shorthands : *true* for $\neg$*false*, $\phi_1 \wedge \phi_2$ for $\neg((\neg\phi_1) \vee (\neg\phi_2))$, $\phi_1 \rightarrow \phi_2$ for $(\neg\phi_1) \vee \phi_2$, $\exists\Diamond_{\sim\theta}\phi_1$ for $\exists$*true* $\mathcal{U}_{\sim\theta}\phi_1$, $\forall\Box_{\sim\theta}\phi_1$ for $\neg\exists\Diamond_{\sim\theta}\neg\phi_1$, $\forall\Diamond_{\sim\theta}\phi_1$ for $\forall$*true* $\mathcal{U}_{\sim\theta}\phi_1$, $\exists\Box_{\sim\theta}\phi_1$ for $\neg\forall\Diamond_{\sim\theta}\neg\phi_1$.

With different controllers and interpretations, a PCTL formula may impose different requirements.

Definition 8 (Satisfaction of PCTL formulas): We write in notations $(q,\nu) \models^{\chi}_{\mathcal{I}} \phi$ to mean that ϕ is satisfied at state (q,ν) in A under controller χ and interpretation $\mathcal{I}$. The satisfaction relation is defined inductively as follows.

- The base case of $\phi \in B(Q,X,H)$ was previously defined except that $(q,\nu) \models^{\chi}_{\mathcal{I}} q'$ iff $q = q'$;
- $(q,\nu) \models^{\chi}_{\mathcal{I}} \phi_1 \vee \phi_2$ iff either $(q,\nu) \models^{\chi}_{\mathcal{I}} \phi_1$ or $(q,\nu) \models^{\chi}_{\mathcal{I}} \phi_2$
- $(q,\nu) \models^{\chi}_{\mathcal{I}} \neg\phi_1$ iff $(q,\nu) \not\models^{\chi}_{\mathcal{I}} \phi_1$
- $(q,\nu) \models^{\chi}_{\mathcal{I}} (\exists\phi_1 \mathcal{U}_{\sim\theta}\phi_2)$ iff there exist a (q,ν)-run = $((q_1,\nu_1,t_1),(q_2,\nu_2,t_2),\ldots)$ in A, an $i \geq 1$, and a $\delta \in [0, t_{i+1} - t_i]$, s.t.
    - $t_i + \delta \sim t_1 + \mathcal{I}(\theta)$,
    - $(q_i,\nu_i) + \delta \models^{\chi}_{\mathcal{I}} \phi_2$,
    - for all j, δ' s.t. either $(0 \leq j < i) \wedge (\delta' \in [0, t_{j+1} - t_j])$ or $(j = i) \wedge (\delta' \in [0,\delta))$, $(q_j,\nu_j) + \delta' \models^{\chi}_{\mathcal{I}} \phi_1$.

  (In words, there exists a (q,ν)-run along which ϕ_2 eventually holds at some point in time ($\sim t_1 + \mathcal{I}(\theta)$) in the time interval $[t_i, t_{i+1}]$, for some i, and before reaching that point ϕ_1 always holds.)
- $(q,\nu) \models^{\chi}_{\mathcal{I}} (\forall\phi_1 \mathcal{U}_{\sim\theta}\phi_2)$ iff for every (q,ν)-run = $((q_1,\nu_1,t_1),(q_2,\nu_2,t_2),\ldots)$ in A, for some $i \geq 1$ and $\delta \in [0, t_{i+1} - t_i]$,
    - $t_i + \delta \sim t_1 + \mathcal{I}(\theta)$,
    - $(q_i,\nu_i) + \delta \models^{\chi}_{\mathcal{I}} \phi_2$,
    - for all j, δ' s.t. either $(0 \leq j < i) \wedge (\delta' \in [0, t_{j+1} - t_j])$ or $(j = i) \wedge (\delta' \in [0,\delta))$, $(q_j,\nu_j) + \delta' \models^{\chi}_{\mathcal{I}} \phi_1$.

Given an SPP A, a PCTL formula ϕ, a controller χ, and an interpretation $\mathcal{I}$ for H, we say A is a *model* of ϕ under χ and $\mathcal{I}$, written as $A \models^{\chi}_{\mathcal{I}} \phi$, iff $(q_0, \mathbf{0}) \models^{\chi}_{\mathcal{I}} \phi$ where $\mathbf{0}$ is the mapping that maps all clocks to zeros. $\|$

2.3 Formal definition of problem

A *performance measure* is just a linear expression like $\sum a_i\alpha_i$ where the a_i's are integers (negative or nonnegative) and the α_i's are parameters in H. It represents a metric that the users want to maximize in their system design.

Definition 9 (Parametric optimization problem): Given an SPP A, a PCTL formula ϕ, and a performance measure λ $(=\sum a_i\alpha_i)$, the *parametric optimization problem instance* for A, ϕ, and λ, denoted as $PO(A,\phi,\lambda)$, is formally defined as the problem of deriving the value $\max\{\sum a_i \mathcal{I}(\alpha_i) \mid \exists\chi(A \models^{\chi}_{\mathcal{I}} \phi)\}$ if it exists. $\|$

3 Algorithm

Our algorithm consists of two steps. First, we extend the parametric analysis algorithm for computer systems [17, 18] with controller-choice information. The modified algorithm can then generate a constraint describing the sufficient and necessary condition of a controller (with parameters) for a given PO(A, ϕ, λ). The second step then uses various techniques in linear algebra to derive the maximum of λ under the constraint.

For the first step, we shall define *controlled region graphs (CR-graphs)* and *controlled path characterization (CP-characterization)* for parametric analysis of controllable timed automata. CR-graphs are like the parametric region graphs introduced in [17, 18]. They are also similar to the region graphs defined in [1] but contain parametric information. A region is a subset of the state space in which all states exhibit the same behavior with respect to the given SPP and PCTL formula.

CP-characterization is derived for each pair of regions in a CR-graph. For each $t \in \mathcal{N}$, it gives a sufficient and necessary condition for the existence of a finite run of t time units from the source region to the destination region. CP-characterization will be useful for the construction of constraints associated with existential path quantifiers ($\exists$).

We need the following three types of integer set manipulations. Given $T_1, T_2 \subseteq \mathcal{N}$,

- $T_1 \cup T_2$ means $\{a \mid a \in T_1 \text{ or } a \in T_2\}$.
- $T_1 + T_2$ means $\{a_1 + a_2 \mid a_1 \in T_1; a_2 \in T_2\}$.
- T_1* means $\{0\} \cup \bigcup_{i \in \mathcal{N}} \sum_{1 \leq j \leq i} T_1$, where $\sum_{1 \leq j \leq i} T_1$ means the addition of i consecutive T_1.

As we shall see later, the notion of *semilinear sets*[1] is crucial in our algorithm of constructing the CP-characterizations. In fact, it will be shown that all integer sets resulting from the above manipulations in our algorithm are semilinear. Semilinear expressions are convenient notations for expressing infinite integer sets constructed regularly. They are also closed under the three manipulations. There are also algorithms to compute the manipulation results. Specifically, we know that all semilinear expressions can be represented as the union of a finite number of sets like $a + c*$ (a shorthand for $\{a + c * h \mid h \geq 0\}$). Such a special form is called *periodical normal form (PNF)*. It is not difficult to prove that given operands in PNF, the results of the three manipulations can all be transformed back into PNF[15]. Due to page-limit, we shall skip the details here.

3.1 CR-graphs

The classic concept of region graphs was originally discussed and used in [1] for verifying dense-time systems. Our CR-graphs are extended from region graphs with constraints on parameter variables. Beside parameter variables, our CR-graphs have an auxiliary clock κ which gets reset to zero once its reading reaches one. The reading of κ is always between 0 and 1, that is, for every state (q, ν), $0 \leq \nu(\kappa) \leq 1$. κ is not used in the user-given SPP and is added when we construct the regions for the convenience of parametric timing analysis. It functions as a ticking indicator for evaluating timed modal formulas of PCTL. ¿From now on, we shall assume that $\kappa \in X$.

The *timing constants* in an SPP A are the integer constants c that appear in conditions such as $x - y \sim c$ and $x \sim c$ in A. The timing constants in a PCTL formula ϕ are the integer constants c that appear in subformulas like $x - y \sim c, x \sim c, \exists\phi_1 \mathcal{U}_{\sim c}\phi_2$, and $\forall\phi_1 \mathcal{U}_{\sim c}\phi_2$. Let $C_{A:\phi}$ be the largest timing constant used in both A and ϕ for the given PO(A, ϕ, λ).

Given a state (q, ν), $(q, \nu) \models (\mathit{fract}(\kappa) = 0)$ iff $\nu(\kappa)$ is an integer.

Definition 10 (Regions): A *region* of a state (q, ν) for $PO(A, \phi, \lambda)$ is a pair $(q, [\nu])$ such that $[\nu]$, called *clock region* of ν, is the notation for the set of timing inequalities characterizing ν, that is,

$$[\nu] = \left\{ x \sim c \,\middle|\, \begin{array}{l} x \in X; \nu(x) \sim c \\ 0 \leq c \leq C_{A:\phi}; \end{array} \right\} \cup \left\{ x - y \sim c \,\middle|\, \begin{array}{l} x, y \in X; 0 \leq c \leq C_{A:\phi}; \\ \nu(x) - \nu(y) \sim c \end{array} \right\} \cup \{\mathit{fract}(\kappa) = 0\}$$

[1]A semilinear integer set is expressible as the union of a finite number of integer sets like $\{a + b_1 j_1 + \ldots + b_n j_n \mid j_1, \ldots, j_n \in \mathcal{N}\}$ for some $a, b_1, \ldots, b_n \in \mathcal{N}$.

where c is a non-negative integer. Given a (q,ν), we shall say $(q,\nu)\in(q,[\nu])$. Specifically, $[\mathbf{0}]$ is the clock region of mapping $\mathbf{0}$. $\|$

Our region definition resembles the one in [13]. Interested readers are referred to [1] for an alternative definition of regions based on the notion of *region equivalence*.

Definition 11 (Controlled region graph, CR-graph): The CR-graph for a $PO(A,\phi,\lambda)$ with $A=(Q,q_0,X,H,\mu,E,\tau,\pi)$ is a directed labelled graph $G_{A:\phi}=(V,F)$ such that the vertex set V is the set of all regions and the arc set $F\subseteq V\times V$ consists of the following two types of arcs (v,v') with $v=(q,\Lambda)$ and $v'=(q',\Lambda')$.

- [**mode transitions**] $(v,v')\in F$ iff for all $(q,\nu)\in v$, there are some χ, $\mathcal{I}$, and $(q',\nu')\in v'$ such that $(q,\nu)\rightarrow^{\chi}_{\mathcal{I}}(q',\nu')$.
- [**time passage**] $q=q'$ and for every state $(q,\nu)\in v$, there is a state $(q,\nu')\in v'$ such that
  - $(q,\nu)+\delta=(q,\nu')$ for some $\delta\in\mathcal{R}^+$; and
  - there is no $\dot{\delta}\in\mathcal{R}^+$, $0<\dot{\delta}<\delta$, s.t. $(q,\nu)+\dot{\delta}$ is not in v or v' $\|$

According to [6, 7, 13], in the controller synthesis problem of timed automata, there is a controller which is a function of states iff there is a controller which is a function of regions. So we shall follow their approach and treat σ simply as a function of regions. We want to derive arc constraints on static parameters and controller choices in CR-graphs so as to construct CP-characterizations. For convenience, given a region $v=(q,\Lambda)$ and a state predicate η, we write $v(\eta)$ for the static state predicate and synthesis decision extracted from η according to the following rules.

- $v(\mathit{false})$ is *false*
- $v(\sigma)=\sigma(v)$
- $v(x-y\sim c)$ is *true* if $x-y\sim c\in\Lambda$; or *false* otherwise.
- $v(x\sim c)$ is *true* if $x\sim c\in\Lambda$; or *false* otherwise.
- $v(\sum a_i\alpha_i\sim c)$ is $\sum a_i\alpha_i\sim c$.
- $v(\eta_1\vee\eta_2)=v(\eta_1)\vee v(\eta_2)$.
- $v(\neg\eta_1)=\neg(\eta_1)$.

A mode transition arc $(v,v')\in F$ with $v=(q,\Lambda),v'=(q',\Lambda')$ iff (1) $\mu(q)\wedge\tau(q,q')$ is satisfied at all states in v; (2) after the transition, $\mu(q')\wedge\bigwedge_{p\in\pi(q,q')(\Lambda)}p$ is satisfied at all states in v', where $\pi(q,q')\,(\Lambda)$ is the new clock region identical to Λ except all clocks in $\pi(q,q')$ are reset to zeros. The two conditions can be formulated as follows.

$$\mathit{xtion}(v,v')\equiv\left(v\left(\bigwedge_{p\in\Lambda}p\wedge\mu(q)\wedge\tau(q,q')\right)\wedge v'\left(\bigwedge_{p'\in\Lambda'}p'\wedge\mu(q')\wedge\bigwedge_{p\in\pi(q,q')(\Lambda)}p\right)\right)$$

Note that formulas like $\bigwedge_{p\in\Lambda}p\wedge\mu(q)\wedge\tau(q,q')$ extract the truly active part in $\mu(q)\wedge\tau(q,q')$ in region (q,Λ). Then the applications of $v()$ and $v'()$ extract the constraints on static parameters as well as on the control symbol σ, should they exist.

The constraints on time passage arcs hinge on the basic constraints on relations between Λ and Λ' which are free of static parameters. Such constraints can be determined with standard techniques like symbolic weakest preconditon calculation [12]. Thus we shall assume the availability of such a procedure $\mathit{timed}(\Lambda,\Lambda')$ which is true iff Λ,Λ' are related in the region time passage relation. Thus a time passage arc $(v,v')\in F$ with $v=(q,\Lambda),v'=(q',\Lambda')$ iff (1) $q=q'$; (2) $\mathit{timed}(\Lambda,\Lambda')$; (3) $\mu(q)$ is satisfied at all states in v,v'. For convenience, we let

$$\mathit{timed}(v,v')\ \equiv\ (q=q')\wedge\mathit{timed}(\Lambda,\Lambda')\wedge v\left(\mu(q)\wedge\bigwedge_{p\in\Lambda}p\right)\wedge v'\left(\mu(q')\wedge\bigwedge_{p'\in\Lambda'}p'\right)$$

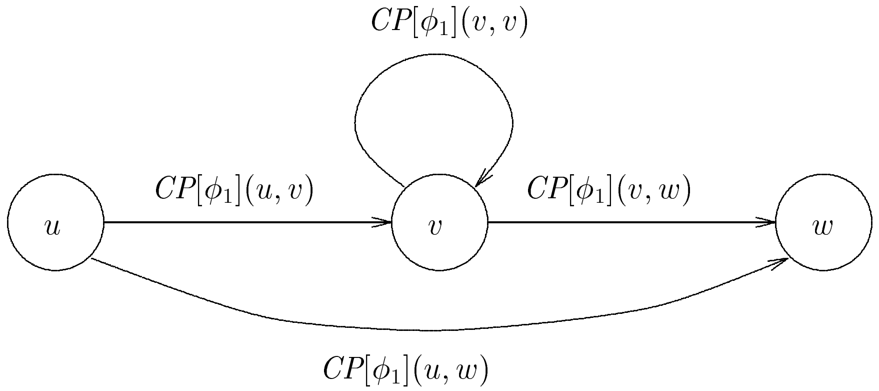


Figure 2: Central operation in our Kleene's closure algorithm.

3.2 CP-characterization

Given a PCTL formula ϕ_1 and a path $\Gamma = \langle v_1 v_2 \dots v_m \rangle$ with $v_i = (q_i, \Lambda_i)$, Γ is called a ϕ_1-*path* iff for some controller χ and some interpretation $\mathcal{I}$, we can embed a valid computation in the path such that along the path, all regions except the last one satisfies ϕ_1. Likewise, a cycle $\Gamma = \langle v_1 v_2 \dots v_1 \rangle$ is a ϕ_1-*cycle* if all regions along the cycle satisfy ϕ_1. The ϕ_1-path (cycle) is of t time units long iff along the path (cycle), exactly t arcs have the reading of κ increments from a noninteger to an integer.

To represent CP-characterizations, we shall use pairs like (η, T), where η is a state predicate and T is an integer set. Suppose, for each $v \in V$, the constraint for subformula ϕ_1 satisfied by states in v is notationally $L[\phi_1](v)$. For $v, v' \in V$ and a subformula ϕ_1, the notation for CP-characterization for v, v' is $CP[\phi_1](v, v')$. Conceptually, there is a finite ϕ_1-path $v_1, \dots, v_m$, with $v_i = (q_i, \Lambda_i)$, of t time units long iff there is a $(\eta, T) \in CP[\phi_1](v_1, v_m)$ such that $t \in T$, the path is a valid computation with respect to η which specifies the satisfiable constraints on parameters and controller-choice, and the controller choice is consistent at the replicated regions in the path.

Now we shall give a procedure for the derivation of the CP-characterization for each pair of regions. The kernel of the procedure is a Kleene's closure computation with an intuitive scheme of vertex-bypassing. Suppose we have three regions u, v and w whose connections in the CR-graph are shown in Figure 2. By bypassing region v, we realize that $CP[\phi_1](u,w)$ should be a minimal superset of

$$H[\phi_1](u,v,w) = \left\{ (\eta_1 \wedge \eta_2 \wedge \textstyle\bigwedge_{(\eta_3,T_3)\in D} \eta_3,\ T_1 + T_2 + \sum_{(\eta_3,T_3)\in D} T_3 *) \ \middle|\ \begin{array}{l} (\eta_1, T_1) \in CP[\phi_1](u,v); \\ (\eta_2, T_2) \in CP[\phi_1](v,w); \\ D \subseteq CP[\phi_1](v,v) \end{array} \right\}$$

considering all intermediate nodes v. Note that in the calculation of $CP[\phi_1](u,w)$, the time set component T always remains semilinear.

The procedure for computing $CP[\phi_1]()$ is presented in the following.

```
KClosure[$\phi_1$]($V, \bar{F}$) /* $\bar{F} \subseteq F$. It is also assumed that for all regions $v \in V$, we know constraint $L[\phi_1](v)$
which makes $\phi_1$ satisfied at every state in $v$. */
{

    For each $(u,w) \notin \bar{F}$, $CP[\phi_1](u,w) := \emptyset$;
```

```
  For each $(u,w) \in \bar{F}$ with $u = (q_u, \Lambda_u)$ and $w = (q_w, \Lambda_w)$, do {
    let $\eta := L[\phi_1](u) \wedge (\mathit{xtion}(u,w) \vee \mathit{timed}(u,w))$.
    if $\mathit{fract}(\kappa) \neq 0 \in \Lambda_u$ and $\mathit{fract}(\kappa) = 0 \in \Lambda_w$, $CP[\phi_1](u,w) := \{(\eta, 1)\}$;
      else $CP[\phi_1](u,w) := \{(\eta, 0)\}$.
  }
  For $i := 0$ to $|V|$, do
    Iteratively for each $v \in V$, do
      for each $u, w \in V$, let $CP[\phi_1](u,w) := CP[\phi_1](u,w) \cup H[\phi_1](u,v,w)$;
}
```

The first two for-loops are for the purpose of setting up the initial values for paths of length one (i.e., directly connected edges). Notice that one unit of time is charged to an edge should κ change from nonzero to zero. One important thing in the design of KClosure[]() is to ensure that the controller always makes consistent decision in each region. This is enforced by the individual controller-choice constraints for each arc along the path. More precisely, if, for example, σ is set to 'true' at some point in time for a region u, and set to 'false' later in the computation for the same region, then both $\sigma(u)$ and $\neg\sigma(u)$ appear (conjunctively) in the predicate characterizing the path, guaranteeing that all satisfiable paths be controller-choice consistent. Lemma 1 establishes the correctness of KClosure$[\phi_1]()$.

LEMMA 1 *: Suppose we are given the labeling function $L[\phi_1]()$ for a PCTL formula ϕ_1 on (V,F) and a natural number $t \in \mathcal{N}$. After running algorithm KClosure$[\phi_1](V,F)$, the following two statements are equivalent.*

1. *there is a $(\eta, T) \in \mathrm{CP}[\phi_1](v, v')$ such that η is satisfiable and $t \in T$;*
2. *there is a computation from a state $(q,\nu) \in v$ to a state in $(q',\nu') \in v'$ under some controller χ and some interpretation $\mathcal{I}$ of t time units such that ϕ_1 is satisfied in all but the last state during the computation.*

Proof sketch: The forward direction from item 1 to item 2 is easy to prove. The backward direction relies on a proof to show that the choice-consistency constraint (logically i.e., $(\mu(q) \wedge \bigwedge_{p\in\Lambda} p) \wedge \tau(q,q'))$ is sufficient. In the jargon of [13], we need to show that the existence of $C_{A:\phi}$-polyhedral solution controller is a necessary condition for the existence of any solution controller. The proof idea is to transform a non-$C_{A:\phi}$-polyhedral solution controller to a $C_{A:\phi}$-polyhedral one. Then we can prove the new controller also satisfies the same set of modal formulas, as the old one, by structural induction on formulas. ||

3.3 Nonzenoness

Zenoness is an undesirable anomaly in real-time computations such that clock readings converge to finite values. Certainly, we do not want such an anomaly sneaks in the constraints derived for the existence of interpretations and controllers. To avoid zenoness, we shall adopt the same approach used in [16]. A state is nonzeno iff from that state on, there is always a computation along which κ gets reset infinitely often. In PCTL, that is $\exists\Box_{\geq 0}\phi_j$ for some $\phi_j \neq \mathit{false}$. This can be expressed as the following constraints on regions.

$$L[\exists\Box_{\geq 0}\phi_j](v) \equiv \bigvee_{u\in V}\left(\left(\bigvee_{(\eta,T)\in CP[\phi_j](\langle\kappa\rangle v,u)} \eta\right) \wedge \left(\bigvee_{(\eta,T)\in CP[\phi_j](u,u)} (\eta \wedge T > 0)\right)\right)$$

where $\langle\kappa\rangle v$ is the region in a CR-graph that agrees with v in every aspect except that if $\langle\kappa\rangle v = (q,\Lambda)$, then $\mathit{fract}(\kappa) = 0 \in \Lambda$. The constraint essentially says that from $\langle\kappa\rangle v$, we can reach a cycle of nonzero time.

3.4 Labeling Algorithm

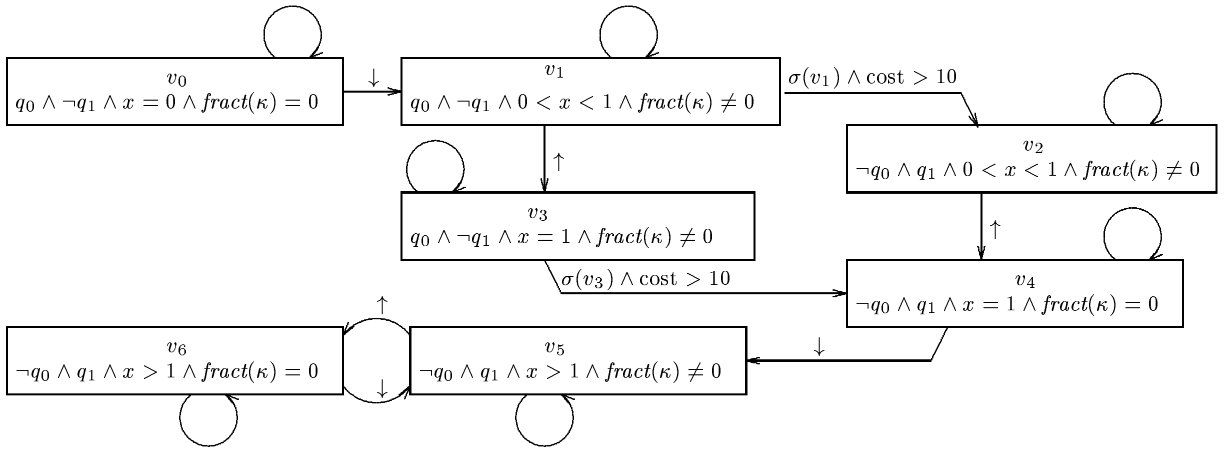
Once the CP-characterizations for ϕ_1 have been constructed successfully, we can then turn to the labeling algorithm to calculate the parametric conditions for the modal formulas properly containing ϕ_1. However, there is still one thing which we should define clearly before presenting our labeling algorithm, that is : "How should we derive parameter constraints from things like (η, T) in CP-characterizations?" Suppose, we want to examine if from v to v', there is a run with time $\geq \theta$. To do this, we define semilinear conditions in the form of $T \sim \theta$ with semilinear expressions T in PNF, and the (numerical or variable) parameter θ is calculated according to the following rewriting rules.

- $a + c* \sim \theta \implies a + cj \sim \theta$ where j is a new integer variable never used before.
- $T_1 \cup T_2 \sim \theta \implies (T_1 \sim \theta) \vee (T_2 \sim \theta)$.

Note that since we assume that the operands are in PNF, we do not have to pay attention to the case of $+$ and $*$. Then, the condition that there is a run with time $\geq \theta$ from v to v' can be calculated as $\bigvee_{(\eta,T)\in CP[\phi_1](v,v')} \eta \wedge T \geq \theta$.

In the following, we present the labeling algorithm for $L[\phi](v)$ in the traditional inductive case analysis of formula ϕ.

```
Label($A, \phi$) {
  (1) construct the CR-graph $G_{A:\phi} = (V, F)$;
  (2) return $L[\phi]((q_0, [\mathbf{0}]))$;
}
$L[\phi_i](v)$ /* $v = (q, \Lambda)$ */ {
switch($\phi_i$){
case ($\mathit{false}$), $L[\mathit{false}](v) := \mathit{false}$;
case ($\phi_i = q'$) where $q' \in Q$, $L[q'](v) := \mathit{true}$ if $q = q'$, else $L[q'](v) := \mathit{false}$;
case ($x - y \sim c$ or $x \sim c$), $L[\phi_i](v) := \mathit{true}$ iff $\phi_i \in \Lambda$;
case ($\sum a_i\alpha_i \sim d$), $L[\sum a_i\alpha_i \sim d](v) := \sum a_i\alpha_i \sim d$;
case ($\phi_j \vee \phi_k$), $L[\phi_j \vee \phi_k](v) := L[\phi_j](v) \vee L[\phi_k](v)$;
case ($\neg\eta$), $L[\neg\eta](v) := \neg L[\eta](v)$;
case ($\exists\Box_{\geq 0}\phi_j$), {
   (1) KClosure$[\phi_j](V, F)$;
   (2) $L[\exists\Box_{\geq 0}\phi_j](v) := \bigvee_{u\in V} \left(\left(\bigvee_{(\eta,T)\in CP[\phi_j](\langle\kappa\rangle v,u)} \eta\right) \wedge \left(\bigvee_{(\eta,T)\in CP[\phi_j](u,u)} (\eta \wedge T > 0)\right)\right)$
 }
case ($\exists\phi_j\mathcal{U}_{<\theta}\phi_k$) {
   (1) KClosure$[\phi_j](V, F)$;
   (2) $L[\exists\phi_j\mathcal{U}_{\leq\theta}\phi_k](v) := \bigvee_{u\in V} \left(L[\phi_k](u) \wedge L[\exists\Box_{\geq 0}\mathit{true}](u) \wedge \bigvee_{(\eta,T)\in CP[\phi_j](\langle\kappa\rangle v,u)} (\eta \wedge T < \theta)\right)$
 }
case ($\exists\phi_j\mathcal{U}_{\geq\theta}\phi_k$, $\exists\phi_j\mathcal{U}_{>\theta}\phi_k$, $\exists\phi_j\mathcal{U}_{<\theta}\phi_k$, $\exists\phi_j\mathcal{U}_{=\theta}\phi_k$) can be treated similarly as the last case.
case ($\forall\phi_j\mathcal{U}_{\leq\theta}\phi_k$), {
   (1) KClosure$[\phi_j](V, F)$;
```



v_0
$q_0 \wedge \neg q_1 \wedge x = 0 \wedge fract(\kappa) = 0$
$\downarrow$
v_1
$q_0 \wedge \neg q_1 \wedge 0 < x < 1 \wedge fract(\kappa) \neq 0$
$\sigma(v_1) \wedge \text{cost} > 10$
v_2
$\neg q_0 \wedge q_1 \wedge 0 < x < 1 \wedge fract(\kappa) \neq 0$
$\uparrow$
v_3
$q_0 \wedge \neg q_1 \wedge x = 1 \wedge fract(\kappa) \neq 0$
$\uparrow$
$\sigma(v_3) \wedge \text{cost} > 10$
v_4
$\neg q_0 \wedge q_1 \wedge x = 1 \wedge fract(\kappa) = 0$
$\uparrow$
$\downarrow$
v_6
$\neg q_0 \wedge q_1 \wedge x > 1 \wedge fract(\kappa) = 0$
$\downarrow$
v_5
$\neg q_0 \wedge q_1 \wedge x > 1 \wedge fract(\kappa) \neq 0$

Figure 3: A simple region graph to illustrate the algorithm.

$$(2)\ L[\forall \phi_j \mathcal{U}_{\leq\theta} \phi_k](v) := \neg \left(\begin{array}{l} L[\exists(\neg\phi_k)\mathcal{U}_{\leq\theta}\neg(\phi_j \vee \phi_k)](\langle\kappa\rangle v) \\ \vee\bigvee_{u_1=(q_1,\Lambda_1),u_2=(q_2,\Lambda_2)\in V} \left(\begin{array}{l} fract(\kappa) = 0 \in \Lambda_1 \wedge fract(\kappa) \neq 0 \in \Lambda_2 \\ \wedge L[\neg\phi_k](u_1) \wedge L[\exists\Box_{\geq 0} true](u_2) \\ \wedge\bigvee_{(\pi,T)\in CP[\neg\phi_k](\langle\kappa\rangle v, u_1)} (\pi \wedge T = \theta) \end{array} \right) \end{array} \right)$$

}

case $(\forall\phi_j\mathcal{U}_{>\theta}\phi_k, \forall\phi_j\mathcal{U}_{\geq\theta}\phi_k, \forall\phi_j\mathcal{U}_{<\theta}\phi_k, \forall\phi_j\mathcal{U}_{=\theta}\phi_k)$ can be treated similarly as the last case.

}

There are two things worthy of mention in the algorithm. First, nonzenoness is properly handled because we require in the algorithm that all computations have a suffix computation satisfying $\exists\Box_{\geq 0}(\ldots)$. Second, the case $\forall\phi_j\mathcal{U}_{\leq\theta}\phi_k$ is handled as the negation of the existence of two types of counter examples. The first counter example type $\exists(\neg\phi_k)\mathcal{U}_{\leq\theta}\neg(\phi_j \vee \phi_k)$ says that ϕ_k is not fulfilled before ϕ_j becomes false in time $\leq \theta$. The rest is for the second counter example type which says that along some computation, ϕ_k is never true in time $\leq \theta$. Note that to characterize the interval which stop right at integer θ, we need constraint $fract(\kappa) = 0 \in \Lambda_1 \wedge fract(\kappa) \neq 0 \in \Lambda_2$.

Example 2 **A simple CR-graph.** For the automaton in Figure 1, the associated region graph is shown in Figure 3. To succinctly represent a region, we shall only put down the (true and false) mode names and those indispensable inequalities as a conjunction. All those inequalities which can be deduced from others are omitted. Also for ease of explanation, some of the edges of the region graph are annotated with $\uparrow$, $\downarrow$, or constraints (over σ and parameters) under which the associated transitions become enabled. An arc (v, v') is annotated with "$\uparrow$" if $v \not\models fract(\kappa) = 0$ and $v' \models fract(\kappa) = 0$; "$\downarrow$" if $v \models fract(\kappa) = 0$ and $v' \not\models fract(\kappa) = 0$.

Note that to make a valid characterization, $\theta \geq 1 \wedge \forall\Box_{<\theta} q_0$ must be satisfied. This means that the transition from region v_1 to v_2 must be disabled while the one from v_3 to v_4 must be enabled. In turn, this means that σ must be false at $q \wedge 0 < x < 1$ and true at $q \wedge x = 1$. With the region graph, we can derive relations $xtion()$ and $timed()$.

$(\mathit{timed}(v_0, v_1), 0) \in \mathit{CP}[\mathit{true}](v_0, v_1)$	$(\mathit{xtion}(v_1, v_2), 0) \in \mathit{CP}[\mathit{true}](v_1, v_2)$	
$(\mathit{timed}(v_1, v_3), 1) \in \mathit{CP}[\mathit{true}](v_1, v_3)$	$(\mathit{timed}(v_2, v_4), 1) \in \mathit{CP}[\mathit{true}](v_2, v_4)$	**(1)**
$(\mathit{xtion}(v_3, v_4), 0) \in \mathit{CP}[\mathit{true}](v_3, v_4)$	$(\mathit{timed}(v_4, v_5), 0) \in \mathit{CP}[\mathit{true}](v_4, v_5)$	
$(\mathit{timed}(v_5, v_6), 1) \in \mathit{CP}[\mathit{true}](v_5, v_6)$	$(\mathit{timed}(v_6, v_5), 0) \in \mathit{CP}[\mathit{true}](v_6, v_5)$	
$(\mathit{true}, 0) \in \mathit{CP}[\mathit{true}](v_0, v_1)$	$(\sigma(v_1) \wedge \text{cost} > 10, 0) \in \mathit{CP}[\mathit{true}](v_1, v_2)$	
$(\mathit{true}, 1) \in \mathit{CP}[\mathit{true}](v_1, v_3)$	$(\mathit{true}, 1) \in \mathit{CP}[\mathit{true}](v_2, v_4)$	**(2)**
$(\sigma(v_3) \wedge \text{cost} > 10, 0) \in \mathit{CP}[\mathit{true}](v_3, v_4)$	$(\mathit{true}, 0) \in \mathit{CP}[\mathit{true}](v_4, v_5)$	
$(\mathit{true}, 1) \in \mathit{CP}[\mathit{true}](v_5, v_6)$	$(\mathit{true}, 0) \in \mathit{CP}[\mathit{true}](v_6, v_5)$	
$(\sigma(v_1) \wedge \text{cost} > 10, 0) \in \mathit{CP}[\mathit{true}](v_0, v_2)$	$(\mathit{true}, 1) \in \mathit{CP}[\mathit{true}](v_0, v_3)$	
$(\sigma(v_1) \wedge \text{cost} > 10, 1) \in \mathit{CP}[\mathit{true}](v_1, v_4)$	$(\sigma(v_3) \wedge \text{cost} > 10, 1) \in \mathit{CP}[\mathit{true}](v_1, v_4)$	
$(\mathit{true}, 1) \in \mathit{CP}[\mathit{true}](v_2, v_5)$	$(\sigma(v_3) \wedge \text{cost} > 10, 0) \in \mathit{CP}[\mathit{true}](v_3, v_5)$	**(3)**
$(\mathit{true}, 1) \in \mathit{CP}[\mathit{true}](v_4, v_6)$	$(\mathit{true}, 1*) \in \mathit{CP}[\mathit{true}](v_5, v_5)$	
$(\mathit{true}, 1*) \in \mathit{CP}[\mathit{true}](v_6, v_6)$		
$(\sigma(v_1) \wedge \text{cost} > 10, 1) \in \mathit{CP}[\mathit{true}](v_0, v_4)$	$(\sigma(v_3) \wedge \text{cost} > 10, 1) \in \mathit{CP}[\mathit{true}](v_0, v_4)$	
$(\sigma(v_1) \wedge \text{cost} > 10, 1 + 1*) \in \mathit{CP}[\mathit{true}](v_1, v_5)$	$(\sigma(v_3) \wedge \text{cost} > 10, 1 + 1*) \in \mathit{CP}[\mathit{true}](v_1, v_5)$	**(4)**
$(\sigma(v_3) \wedge \text{cost} > 10, 1 + 1 * +1*) \in \mathit{CP}[\mathit{true}](v_3, v_6)$	$(\mathit{true}, 2 + 1 * +1*) \in \mathit{CP}[\mathit{true}](v_2, v_6)$	
...	...	...

Table 1: Computation of KClosure[]()

$$\mathit{timed}(v_0, v_1) \equiv \mathit{true}; \mathit{xtion}(v_0, v_1) \equiv \mathit{false}; \mathit{timed}(v_1, v_2) \equiv \mathit{false};$$

$$\mathit{xtion}(v_1, v_2) \equiv \left(\begin{array}{l} \quad v_1(q_0 \wedge \neg q_1 \wedge 0 < x < 1 \wedge \mathit{fract}(\kappa) \neq 0 \wedge \sigma \wedge \text{cost} > 10) \\ \wedge v_2(\neg q_0 \wedge q_1 \wedge 0 < x < 1 \wedge \mathit{fract}(\kappa) \neq 0) \end{array} \right) \equiv \sigma(v_1) \wedge \text{cost} > 10$$

$$\mathit{timed}(v_1, v_3) \equiv \mathit{true}; \mathit{xtion}(v_1, v_3) \equiv \mathit{false}; \mathit{timed}(v_2, v_4) \equiv \mathit{true}; \mathit{xtion}(v_2, v_4) \equiv \mathit{false};$$

$$\mathit{timed}(v_3, v_4) \equiv \mathit{false};$$

$$\mathit{xtion}(v_3, v_4) \equiv \left(\begin{array}{l} \quad v_3(\neg q_0 \wedge q_1 \wedge x = 1 \wedge \mathit{fract}(\kappa) = 0 \wedge \sigma \wedge \text{cost} > 10) \\ \wedge v_4(\neg q_0 \wedge q_1 \wedge x = 1 \wedge \mathit{fract}(\kappa) = 0) \end{array} \right) \equiv \sigma(v_3) \wedge \text{cost} > 10$$

$$\mathit{timed}(v_4, v_5) \equiv \mathit{true}; \mathit{xtion}(v_4, v_5) \equiv \mathit{false}; \mathit{timed}(v_5, v_6) \equiv \mathit{true}; \mathit{xtion}(v_5, v_6) \equiv \mathit{false};$$

$$\mathit{timed}(v_6, v_5) \equiv \mathit{true}; \mathit{xtion}(v_6, v_5) \equiv \mathit{false}$$

After running algorithm KClosure$[q]$() on the region graph, we find that the computation of membership relations is that shown in Table 1. In the table, we group the formulas into rows with horizontal lines to make it more readable. The first two rows are set up for length one paths, while the remaining rows are obtained with the transitivity (by-passing) law. In the third row, because of the time 1 self-loops on regions v_5, v_6, we can deduce that $(\mathit{true}, 1*) \in \mathit{CP}[\mathit{true}](v_5, v_5)$, which means that we can cycle through region v_5 for an arbitrary number of times. $\|$

In the following discussion, our labeling algorithm is run on a small example to present the idea.

Example 3 **A test run of the labelling algorithm.** We illustrate our algorithm on the automaton shown in Figure 1 and PCTL specification cost $< 100 \wedge \theta \geq 1 \wedge \forall\Box_{<\theta} q_0$. The region graph is shown in Figure 3. We first have the following derivation.

$$\begin{array}{rl} & \text{cost} < 100 \wedge \theta \geq 1 \wedge \forall\Box_{<\theta} q_0 \\ \equiv & \text{cost} < 100 \wedge \theta \geq 1 \wedge \neg\exists\Diamond_{<\theta}((\neg q_0) \wedge \exists\Box_{\geq 0} \mathit{true}) \end{array}$$

According to our labelling algorithm, the characterization formula is

$$\begin{array}{l} L[\text{cost} < 100](\langle\kappa\rangle v_0) \wedge L[\theta \geq 1](\langle\kappa\rangle v_0) \\ \wedge \neg \bigvee_{u \in V} \Big(L[\neg q_0](u) \wedge L[\exists\Box_{\geq 0} \mathit{true}](u) \wedge \bigvee_{(\eta, T) \in \mathit{CP}[\mathit{true}](\langle\kappa\rangle v_0, u)} (\eta \wedge T < \theta) \Big) \end{array}$$

$\neg q_0$ is true only at v_2, v_4, v_5, v_6 and $\exists\Box_{\geq 0} \mathit{true}$ is true at all these four regions. Thus we have

$$\begin{array}{rl}
& \mathit{cost} < 100 \wedge \theta \geq 1 \wedge \neg \left(\begin{array}{ll} & \bigvee_{(\eta,T)\in \mathit{CP}[\mathit{true}](v_0,v_2)} (\eta \wedge \theta > T) \\ \vee & \bigvee_{(\eta,T)\in \mathit{CP}[\mathit{true}](v_0,v_4)} (\eta \wedge \theta > T) \\ \vee & \bigvee_{(\eta,T)\in \mathit{CP}[\mathit{true}](v_0,v_5)} (\eta \wedge \theta > T) \\ \vee & \bigvee_{(\eta,T)\in \mathit{CP}[\mathit{true}](v_0,v_6)} (\eta \wedge \theta > T) \end{array} \right) \\
\equiv & \text{cost} < 100 \wedge \theta \geq 1 \wedge \neg \left(\begin{array}{ll} & \sigma(v_1) \wedge \text{cost} > 10 \wedge \theta > 0 \\ \vee & \sigma(v_1) \wedge \text{cost} > 10 \wedge \theta > 1 \\ \vee & \sigma(v_3) \wedge \text{cost} > 10 \wedge \theta > 1 \\ \vee & \sigma(v_1) \wedge \text{cost} > 10 \wedge \theta > 1 + 1* \\ \vee & \sigma(v_3) \wedge \text{cost} > 10 \wedge \theta > 1 + 1* \\ \vee & \sigma(v_1) \wedge \text{cost} > 10 \wedge \theta > 2 + 1 * + 1* \\ \vee & \sigma(v_3) \wedge \text{cost} > 10 \wedge \theta > 2 + 1 * + 1* \end{array} \right) \\
\equiv & \text{cost} < 100 \wedge \theta \geq 1 \wedge \neg \left((\sigma(v_1) \wedge \text{cost} > 10 \wedge \theta > 0) \vee (\sigma(v_3) \wedge \text{cost} > 10 \wedge \theta > 1) \right) \\
\equiv & \text{cost} < 100 \wedge \theta \geq 1 \wedge (\neg\sigma(v_1) \vee \text{cost} \leq 10 \vee \theta \leq 0) \wedge (\neg\sigma(v_3) \vee \text{cost} \leq 10 \vee \theta \leq 1) \\
\equiv & \text{cost} < 100 \wedge \theta \geq 1 \wedge (\neg\sigma(v_1) \vee \text{cost} \leq 10) \wedge (\neg\sigma(v_3) \vee \text{cost} \leq 10 \vee \theta \leq 1)
\end{array}$$

This formula says that to make the solution existent, the discrete transition must be disabled with either $\sigma(v_1) = \mathit{false}$ or cost ≤ 10 at mode q_0 when $0 < x < 1$ is true. Furthermore, according to the last disjunction, if there is going to be a nonzeno computation, then it is necessary that $\sigma(v_3) = \mathit{true}$, cost > 10, and $\theta = 1$. $\|$

The following theorem establishes the correctness of our labeling algorithm. The proof parallels that of a similar algorithm presented in [17].

THEOREM 2 : *Given a* PO(A, ϕ, λ), *a subformula* ϕ_1, *and* $v = (q, \Lambda)$, *after executing* $L[\phi_1](v)$ *in our labeling algorithm,* $L[\phi_1](v)$ *is satisfiable iff for some* $\mathcal{I}$ *and for some* χ, *for any* $(q, \nu) \in v$, $(q, \nu) \models^{\chi}_{\mathcal{I}} \phi_1$. $\|$

3.5 Parametric optimization step

Given an SPP $A = (Q, q_0, X, H, \mu, E, \tau, \pi)$ and a PCTL formula ϕ, our labeling algorithm $\mathit{Label}(A, \phi)$ returns predicate $L[\phi]((q_0, [\mathbf{0}]))$ (a constraint on parameters and controller choices) in such a way that $L[\phi]((q_0, [\mathbf{0}]))$ is satisfiable iff for some interpretation $\mathcal{I}$ and some controller χ, $(q, \mathbf{0}) \models^{\chi}_{\mathcal{I}} \phi$. Then the constraint on parameters and controller choices is fully characterized by $L[\phi]((q_0, [\mathbf{0}]))$. In the following we shall demonstrate how to process $L[\phi]((q_0, [\mathbf{0}]))$ to solve our parametric optimization problem.

Step 1: $L[\phi]((q_0, [\mathbf{0}]))$ can be rearranged into the disjunctive normal form $\Delta_1 \vee \Delta_2 \vee \ldots \vee \Delta_m$, for some m. Each of the conjunctions can be further rearranged to two types of atoms: (type 1) $\sum a_i \alpha_i \sim c$ and (type 2) $\sigma(v)$. Type 2 is for consistency of controller choices. A conjunction is satisfiable iff the subconjunction of its type 1 atoms is satisfiable and the subconjunction of its type 2 atoms is also satisfiable. The satisfiability of conjunctions of type 2 atoms can be solved in the standard BDD or DBM technologies. $L[\phi]((q_0, [\mathbf{0}]))$ remains the same if those conjunctions with unsatisfiable subconjunctions of type 2 atoms are eliminated.

Step 2: Assume that $L[\phi]((q_0, [\mathbf{0}]))$ has no conjunctions with unsatisfiable subconjunctions of type 2 atoms. The constraint on parameters for the existence of any controllers is thus

$$\check{\Delta} = \bigvee_{1 \leq i \leq m; \check{\Delta}_i \text{ is the subconjunction of type 1 atoms of } \Delta_i} \check{\Delta}_i$$

Then the parametric optimization problem can be broken down to m subproblems which ask for the maximum of the objective function on linear inequality systems $\check{\Delta}_i$. Thus, the parametric optimization problem of controllable timed automata is reduced to that of *integer linear programming*, which is reasonably well-studied in the literature, although the size of the linear programming instance is likely to be exponential in the worst case. The answer to our optimization problem is the maximum of the answers to the m subproblems.

Example 4 *:* For the automaton, specification, and optimization metric $(\text{cost} - \theta)$ to maximize in example 1, the optimization metric is $\max(\text{cost} - \theta) = 10 - 0 = 10$ when only zeno computations exist; or $\max(\text{cost} - \theta) = 99 - 1 = 98$ when at least one nonzeno computation exists. Thus the optimization metric is 98 with respect specification *phi* and metric $\text{cost} - \theta$.

After running our algorithm for the second set of specification ϕ' and metric $-\theta$, we find that the optimization metric value $-\theta$ is -1, indicating that 1 is the minimum deadline from a q_0 state to a q_1 state subject to the restriction of ϕ'. Due to page-limit, we leave the details to the readers. ||

4 Conclusion

We have investigated the parametric optimization issue of real-time systems modeled by controllable timed automata augmented with static parameters. An algorithm has been proposed for deriving constraints over the static parameters as well as the synthesized controller that would provide an environment in which the system functions correctly. To the best of our knowledge, our work is the first in an attempt to investigate *parametric analysis*, *controller synthesis* and *parametric optimization* in a unified setting. By giving a complete characterization of the controller as well as the parameter valuations (satisfying a given specification) in terms of a set of linear inequalities, parametric optimization is then carried out in the framework of *integer linear programming*, which is relatively well-studied. The efficiency issue is one thing that has not been addressed much in this work. As region graphs of timed automata, in general, are exponential in size, we expect our algorithm to take exponential time in the worst case. One way to circumvent this inefficiency is to look into the possibility of incorporating the so-called *symbolic techniques*, which have been proven to be useful for controller synthesis (see, e.g., [6]). Analyzing the computational complexity of our algorithm (as well as the problem) and subsequently improving the algorithm (perhaps, based on symbolic approaches) are among our future research of parametric optimization.

References

[1] Alur, R., Courcoubetis, C., and Dill, D.L. (1993), Model-Checking in Dense Real-Time, *Information and Computation* **104**, Nr. 1, pp. 2–34.

[2] Alur, R. and Dill, D. (1990), Automata for Modeling Real-Time Systems, in "*Automata, Languages and Programming: Proceedings of the 17th ICALP*," LNCS 443, pp. 332–335, Springer-Verlag, Berlin/New York.

[3] Alur, R., Etessami, K., La Torre, S. and Peled, D. (1999), Parametric Temporal Logic for "Model Measuring," in "*Automata, Languages and Programming: Proceedings of the 26th ICALP*," Springer Verlag.

[4] Alur, R., Henzinger, T.A., and Vardi, M.Y. (1993), Parametric Real-Time Reasoning, in "*Proceedings, 25th ACM STOC*," pp. 592–601.

[5] Asarin, E. and Maler, O. (1999), As Soon as Possible: Time Optimal Control for Timed Automata, in F. Vaandrage and J. van Schuppen (Eds.), *Hybrid Systems: Computation and Control*, LNCS 1569, pp. 19-30.

[6] Asarin, E., Maler, O., and Pnueli, A. (1995), Symbolic Controller Synthesis for Discrete and Timed Systems, in Antsaklis, P., Kohn, W., Nerode, A., Sastry, S.(eds): *Hybrid Systems II.* LNCS 999,Springer 1995, pp. 1-20

[7] Asarin, E., Maler, O., Pnueli, A. and Sifakis, J. (1998), Controller Synthesis for Timed Automata, *Proc. IFAC Symposium on System Structure and Control*, Elsevier, pp. 469-474.

[8] Clarke, E. and Emerson, E.A. (1981), Design and Synthesis of Synchronization Skeletons using Branching-Time Temporal Logic, in "*Proceedings, Workshop on Logic of Programs*," LNCS 131, Springer-Verlag.

[9] Clarke, E., Emerson, E.A., and Sistla, A.P. (1986), Automatic Verification of Finite-State Concurrent Systems using Temporal-Logic Specifications, *ACM Trans. Programming, Languages, and Systems*, **8**, Nr. 2, pp. 244–263.

[10] Courcoubetis, C. and Yannakakis, M. (1992), Minimum and Maximum Delay Problems in Real-Time Systems. *Formal Methods in System Design* **1**: 385-415, Kluwer Academic Publishers; also in "Proceedings, 3rd CAV," 1991, Springer-Verlag, LNCS 575.

[11] Emerson, E. A., and Trefler, R. (1999), Parametric Quantitative Temporal Reasoning, *Proceedings of IEEE-CS Conference on Logic in Computer Science* (LICS), pp. 336–343.

[12] T.A. Henzinger, X. Nicollin, J. Sifakis, S. Yovine. Symbolic Model Checking for Real-Time Systems, IEEE LICS 1992.

[13] Maler, O. Pnueli, A. and Sifakis, J. (1995), On the Synthesis of Discrete Controllers for Timed Systems. *STACS'95*, LNCS 900, Springer-Verlag.

[14] Ramadge, P. and Wonham, W. (1987), Supervisory Control of a Class of Discrete Event Processes, *SIAM J. of Control and Optimization* **25**: 206-230.

[15] Wang, F. (1996), Scalable Compositional Reachability Analysis of Real-Time Concurrent Systems. In Proceedings of the 2nd IEEE RTAS (Real-Time Technology and Applications Symposium), Boston, June, 1996.

[16] Wang, F. (1996), Parametric Timing Analysis for Real-Time Systems, Information and Computation, Vol. 130, Nr 2, Nov. 1996, Academic Press, ISSN 0890-5401; pp 131-150. Also in "*Proceedings, 10th IEEE Symposium on Logic in Computer Science*," 1995.

[17] Wang, F., Parametric Analysis of Computer Systems, *Formal Methods in System Design*, pp.39-60, 17, 39-60, 2000.

[18] Wang, F. and Hsiung, P.-A. (1997), Parametric Analysis of Computer Systems, *AMAST'97*, LNCS 1349.