Parametric Analysis of Computer Systems

Farn Wang Pao-Ann Hsiung
Institute of Information Science, Academia Sinica, Taiwan, ROC
+886-2-7883799 x 1717 FAX:+886-2-7824814 farn@iis.sinica.edu.tw

March 18, 1997

Abstract
A general parametric analysis problem which allows the usage of parameter variables in both the real-time automata and the specifications is proposed and solved.

1 Introduction

A successful real-world project management relies on the satisfaction of various timing and nontiming restraints which may compete with each other for resources. Examples of such restraints include timely responses, budget, domestic or international regulations, system configurations, environments, compatibilities, ... In this work, we define and algorithmically solves the *parametric analysis problem* of computer systems which allows for the formal description of system behaviors and design requirements with various timing and nontiming parameter variables and asks for a general conditions on all solutions to those parameter variables.

The design of our problem was influenced by previous work of Alur et al. [AHV93] and Wang [Wang95] which will be discussed briefly later. Our parametric analysis problem is presented in two parts : an automaton with nontiming parameter variables and a specification with both timing and nontiming parameter variables. The following example is adapted from the railroad crossing example and shows how such a platform can be useful.

Example 1 *:* **Railroad Gate Controller**

The popular railroad crossing example consists of a train monitor and gate-controller. In figure 1, we give a parametric version of the automaton descriptions of the monitor and controller respectively. The ovals represent meta-states while arcs represent transitions. By each transition, we label the transition condition and the clocks to be reset to zero on the transition. The global state space can be calculated as the Cartesian-product of local state spaces.

The safety requirement is that whenever a train is at the crossing, the gate must be in the D mode (gate is down). The more money you spend on monitor, the more precise you can tell how far away a train is approaching. Suppose we now have two monitor types, one costs 1000 dollars and can tell if a train is coming to the crossing in 290 to 300 seconds; the other type costs 500 and can tell if a train is coming to the crossing in 200 to 350 seconds.

We also have two gate-controller types. One costs 900 dollars and can lower the gate in 20 to 50 seconds and skip the U mode (gate is up) when a train is coming to the crossing and the controller is in the R mode (gate-Raising mode). The other type costs 300 dollars and can lower the gate in 100 to 200 seconds and cannot skip the U mode once the controller is in the R mode.

Suppose now the design of a rail-road crossing gate-controller is subjected to the budget constraint : the cost of the monitor ($\$_M$) and that of controller ($\$_C$) together cannot exceed 1500 dollars. We want to make sure under this constraint, if the safety requirement can still be satisfied. This can be expressed in our logic

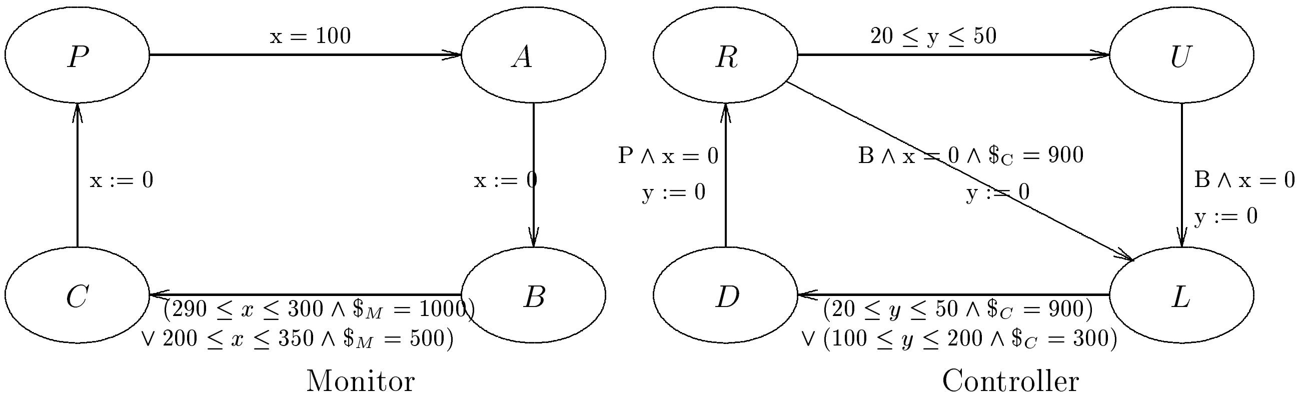

Figure 1: Railroad Gate Controller Example

PCTL as $\$_M + \$_C \le 1500 \wedge \forall\Box(C \rightarrow D)$. Here $\forall\Box$ is a modal operator from CTL [CE81, CES86] which means for all computations henceforth, the following statement must be true. $\|$

Our system behavior descriptions are given in *statically parametric automata (SPA)* and our specifications are given in *parametric computation tree logic (PCTL)*. The outcome of our algorithm are Boolean expressions, whose literals are linear inequalities on the parameter variables, and can be further processed with standard techniques like simplex, simulated annealing, ... to extract useful design feedback.

In the remainder of the introduction, we shall first briefly discuss related work on the subject, and then sketch an outline of the rest of the paper.

1.1 Related work

In the earliest development [CE81, CES86], people use finite-state automata to describe system behavior and check to see if they satisfy specification given in branching-time temporal logic CTL. Such a framework is usually called *model-checking.* A *CTL (Computation Tree Logic)* formula is composed of binary propositions $(p, q, \ldots)$, Boolean operators $(\neg, \vee, \wedge)$, and branching-time modal operators $(\exists\mathcal{U}, \exists\bigcirc, \forall\mathcal{U}, \forall\bigcirc)$. $\exists$ means *"there exists"* a computation. $\forall$ means *"for all"* computations. $\mathcal{U}$ means something is true "until" something else is true. $\bigcirc$ means *"next state."* For example, $\exists p\mathcal{U}q$ says there exists a computation along which p is true until q is true. Since there is no notion of real-time (clock time), only ordering among events are considered.

The following shorthands are generally accepted besides the usual ones in Boolean algebra. $\exists\Diamond\phi_1$ is for $\exists \mathit{true}\,\mathcal{U}\phi_1$; $\forall\Box\phi_1$ for $\neg\exists\Diamond\neg\phi_1$; $\forall\Diamond\phi_1$ for $\forall \mathit{true}\,\mathcal{U}\phi_1$; and $\exists\Box\phi_1$ for $\neg\forall\Diamond\neg\phi_1$. Intuitively $\Diamond$ means *"eventually"* while $\Box$ means *"henceforth."*

CTL model-checking has been used to prove the correctness of concurrent systems such as circuits and communication protocols. In 1990, the platform was extended by Alur et al. to *Timed CTL (TCTL) model-checking problem* to verify dense-time systems equipped with resettable clocks [ACD90]. Alur et al. also solve the problem in the same paper with an innovative state space partitioning scheme.

In [CY92], the problems of deciding the earliest and latest times a target state can appear in the computation of a timed automaton was discussed. However, they did not derive the general conditions on parameter variables.

In 1993, Alur et al. embark on the reachability problem of real-time automata with parameter variables [AHV93]. Particularly, they have established that in general, the problem has no algorithm when three clocks are compared with parameter variables in the automata [AHV93]. This observation greatly influences the design of our platform.

In 1995, Wang propose another platform which extends the TCTL model-checking problem to allow for timing parameter variables in TCTL formulae [Wang95]. His algorithm gives back Boolean conditions whose literals are linear equalities on the timing parameter variables. He also showed that his parametric timing analysis problem is PSPACE-hard while his analysis algorithm is of double-exponential time complexity.

Henzinger's HyTech system developed at Cornell also has parametric analysis power[AHV93, HHWT95]. However in their framework, they did not identify a decidable class for the parametric analysis problem and their procedure is not guaranteed to terminate. In comparison, our framework has an algorithm which can generate the semilinear description of the working solutions for the parameter variables.

1.2 Outline

Section 2 presents our system behavior description language : the *Statically Parametric Automaton* (SPA). Section 3 defines *Parametric Computation Tree Logic (PCTL)* and the *Parametric Analysis Problem.* Section 4 presents the algorithm, proves its correctness, and analyzes its complexity. Section 5 concludes the paper.

We also adopt $\mathcal{N}$ and $\mathcal{R}^+$ as the sets of nonnegative integers and nonnegative reals respectively.

2 Statically parametric automata (SPA)

In an SPA, people may combine propositions, timing inequalities on clock readings, and linear inequalities of parameter variables 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 atomic propositions, a set C of clocks, and a set H of parameter variables, the syntax of a *state predicate* η of P, C, and H, has the following syntax rules.

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

where $p \in P$, $x, y \in C$, $a_i, c \in \mathcal{N}$, $\alpha_i \in H$, $\sim \in \{\leq, <, =, \geq, >\}$, and η_1, η_2 are state predicates. Notationally, we let $B(P,C,H)$ be the set of all state predicates on P, C, and H. Note the parameter variables considered in H are static because their value do not change with time during each 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 Automata**

A *Statically Parametric Automaton* (SPA) is a tuple $(Q, q_0, P, C, H, \chi, E, \rho, \tau)$ with the following restrictions.

- Q is a finite set of meta-states.
- $q_0 \in Q$ is the initial meta-state.
- P is a set of atomic propositions.
- C is a set of clocks.
- H is a set of parameters variables.
- $\chi : Q \mapsto B(P,C,H)$ is a function that labels each meta-state with a condition true in that meta-state.
- $E \subseteq Q \times Q$ is the set of transitions.
- $\rho : E \mapsto 2^C$ defines the set of clocks to be reset during each transition.
- $\tau : E \mapsto B(P,C,H)$ defines the transition triggering conditions. ||

An SPA starts execution at its meta-state q_0. We shall assume that initially, all clocks read zero. In between meta-state transitions, all clocks increment their readings at a uniform rate. The transitions of the SPA may be fired when the triggering conditon is satisfied. With different interpretation to the parameter variables, it may exhibit different behaviors. During a transition from meta-state q_i to q_j, for each $x \in \rho(q_i, q_j)$, the reading of x will be reset to zero. There are state predicates with parameter variables on the states as well as transitions. These parameters may also appear in the specifications of the same analysis problem instance.

Definition 2 *:* **State**

A *state* s of SPA $A = (Q, q_0, P, C, H, \chi, E, \rho, \tau)$ is a mapping from $P \cup C$ to $\{true, false\} \cup \mathcal{R}^+$ such that for each $p \in P$, $s(p) \in \{true, false\}$ and for each $x \in C$, $s(x) \in \mathcal{R}^+$, where $\mathcal{R}^+$ is the set of nonnegative real numbers. $\|$

The same SPA may generate different computations under different interpretation of its parameter variables. An *interpretation*, $\mathcal{I}$, for H is a mapping from $\mathcal{N} \cup H$ to $\mathcal{N}$ such that for all $c \in \mathcal{N}$, $\mathcal{I}(c) = c$. An SPA $A = (Q, q_0, P, C, H, \chi, E, \rho, \tau)$ is said to be interpreted with respect to $\mathcal{I}$, when all state predicates in A have their parameter variables interpreted according to $\mathcal{I}$.

Definition 3 *:* **Satisfaction of interpreted state predicates by a state**

State predicate η is satisfied by state s under interpretation $\mathcal{I}$, written as $s \models_{\mathcal{I}} \eta$, iff

- $s \not\models_{\mathcal{I}} false$;
- $s \models_{\mathcal{I}} p$ iff $s(p) = true$;
- $s \models_{\mathcal{I}} x - y \sim c$ iff $s(x) - s(y) \sim c$;
- $s \models_{\mathcal{I}} x \sim c$ iff $s(x) \sim c$;
- $s \models_{\mathcal{I}} \sum a_i \alpha_i \sim c$ iff $\sum a_i \mathcal{I}(\alpha_i) \sim c$;
- $s \models_{\mathcal{I}} \eta_1 \vee \eta_2$ iff $s \models_{\mathcal{I}} \eta_1$ or $s \models_{\mathcal{I}} \eta_2$; and
- $s \models_{\mathcal{I}} \neg \eta_1$ iff $s \not\models_{\mathcal{I}} \eta_1$. $\|$

Now we are going to define the computation of SPA. For convenience, we adopt the following conventions.

An SPA $A = (Q, q_0, P, C, H, \chi, E, \rho, \tau)$ is *unambiguous* iff for all states s, there is at most one $q \in Q$ such that for some I, $s \models_{\mathcal{I}} \chi(q)$. Ambiguous SPA's can be made unambiguous by incorporating meta-state names as propositional conjuncts in the conjunctive normal forms of the $\chi()$-state predicate of each meta-state. For convenience, from now on, we shall only talk about unambiguous SPA's. When we say an SPA, we mean an unambiguous SPA.

Given an SPA $A = (Q, q_0, P, C, H, \chi, E, \rho, \tau)$, an interpretation $\mathcal{I}$ for H, and a state s, we let s^Q be the meta-state in Q such that $s \models_{\mathcal{I}} \chi(s^Q)$. If there is no meta-state $q \in Q$ such that $s \models_{\mathcal{I}} \chi(q)$, then s^Q is undefined.

Given two states s, s', there is a *meta-state transition* from s to s' in A under interpretation $\mathcal{I}$, in symbols $s \rightarrow_{\mathcal{I}} s'$, iff

- s^Q, s'^Q are both defined,
- $(s^Q, s'^Q) \in E$,
- $s \models_{\mathcal{I}} \tau(s^Q, s'^Q)$, and
- $\forall x \in C \left((x \in \rho(s^Q, s'^Q) \Rightarrow s'(x) = 0) \wedge (x \notin \rho(s^Q, s'^Q) \Rightarrow s'(x) = s(x)) \right)$.

Also, given a state s and a $\delta \in \mathcal{R}^+$, we let $s + \delta$ be the state that agrees with s in every aspect except for all $x \in C$, $s(x) + \delta = (s + \delta)(x)$.

Definition 4 *:* **s-run of interpreted SPA**

Given a state s of SPA $A = (Q, q_0, P, C, H, \chi, E, \rho, \tau)$ and an interpretation $\mathcal{I}$, a computation of A starting at s is called an *s-run* and is a sequence $((s_1, t_1), (s_2, t_2), \ldots\ldots)$ of pairs such that

- $s = s_1$; and
- for each $t \in \mathcal{R}^+$, there is an $i \in \mathcal{N}$ such that $t_i \geq t$; and
- for each integer $i \geq 1$, s_i^Q is defined and for each real $0 \leq \delta \leq t_{i+1} - t_i$, $s_i + \delta \models_{\mathcal{I}} \chi(s_i^Q)$; and
- for each $i \geq 1$, A goes from s_i to s_{i+1} because of
  - a meta-state transition, i.e. $t_i = t_{i+1} \wedge s_i \rightarrow_{\mathcal{I}} s_{i+1}$; or
  - time passage, i.e. $t_i < t_{i+1} \wedge s_i + t_{i+1} - t_i = s_{i+1}$. $\|$

3 PCTL and parametric analysis problem

Parametric Computation Tree Logic (PCTL) is used for specifying the design requirements and is defined with respect to a given SPA. Suppose we are given an SPA $A = (Q, q_0, P, C, H, \chi, E, \rho, \tau)$. 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 η is a state predicate in $B(P, C, H)$, ϕ_1 and ϕ_2 are PCTL formulae, and θ is an element in $\mathcal{N} \cup H$.

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

With different interpretations, a PCTL formula may impose different requirements. We write in notations $s \models_{\mathcal{I}} \phi$ to mean that ϕ is satisfied at state s in A under interpretation $\mathcal{I}$. The satisfaction relation is defined inductively as follows.

- If ϕ is a state predicate, then $s \models_{\mathcal{I}} \phi$ iff ϕ is satisfied by s as a state predicate under $\mathcal{I}$.
- $s \models_{\mathcal{I}} \phi_1 \vee \phi_2$ iff either $s \models_{\mathcal{I}} \phi_1$ or $s \models_{\mathcal{I}} \phi_2$
- $s \models_{\mathcal{I}} \neg\phi_1$ iff $s \not\models_{\mathcal{I}} \phi_1$
- $s \models_{\mathcal{I}} (\exists\phi_1 \mathcal{U}_{\sim\theta}\phi_2)$ iff there are an s-run $= ((s_1, t_1), (s_2, t_2), \ldots)$ in A, an $i \geq 1$, and a $\delta \in [0, t_{i+1} - t_i]$, s.t.
  - $t_i + \delta \sim t_1 + \mathcal{I}(\theta)$,
  - $s_i + \delta \models_{\mathcal{I}} \phi_2$,
  - for all $0 \leq j < i$ and $\delta' \in [0, t_{j+1} - t_j]$, $s_j + \delta' \models_{\mathcal{I}} \phi_1$, and
  - for all $\delta' \in [0, \delta)$, $s_i + \delta' \models_{\mathcal{I}} \phi_1$.
- $s \models_{\mathcal{I}} (\forall\phi_1 \mathcal{U}_{\sim\theta}\phi_2)$ iff for every s-run $= ((s_1, t_1), (s_2, t_2), \ldots)$ in A, for some $i \geq 1$ and $\delta \in [0, t_{i+1} - t_i]$,
  - $t_i + \delta \sim t_1 + \mathcal{I}(\theta)$,
  - $s_i + \delta \models_{\mathcal{I}} \phi_2$,
  - for all $0 \leq j < i$ and $\delta' \in [0, t_{j+1} - t_j]$, $s_j + \delta' \models_{\mathcal{I}} \phi_1$, and
  - for all $\delta' \in [0, \delta)$, $s_i + \delta' \models_{\mathcal{I}} \phi_1$.

Given an SPA A, a PCTL formula ϕ, and an interpretation $\mathcal{I}$ for H, we say A is a *model* of ϕ under $\mathcal{I}$, written as $A \models_{\mathcal{I}} \phi$, iff $s \models_{\mathcal{I}} \phi$ for all states s such that $s^Q = q_0$.

We now formally define our problem.

Definition 5 *:* **Statically Parametric Analysis Problem**
Given an SPA A and a specification (PCTL formula) ϕ, the *parametric analysis problem instance* for A and ϕ, denoted as *PAP*(A, ϕ), is formally defined as the problem of deriving the general condition of all interpretation $\mathcal{I}$ such that $A \models_{\mathcal{I}} \phi$. $\mathcal{I}$ is called a *solution* to $PAP(A, \phi)$ iff $A \models_{\mathcal{I}} \phi$. ||

We will show that such conditions are always expressible as Boolean combinations of linear inequalities of parameter variables.

4 Parametric analysis

In this section, we shall develop new data-structures, *parametric region graph* and *conditional path graph*, to solve the parametric analysis problem. Parametric region graph is similar to the region graph defined in [ACD90] but it contains parametric information. A region is a subset of the state space in which all states exhibit the same behavior with respect to the given SPA and PCTL formula.

Given a parametric analysis problem for A and ϕ, a modal subformula ϕ_1 of ϕ, and the parametric region

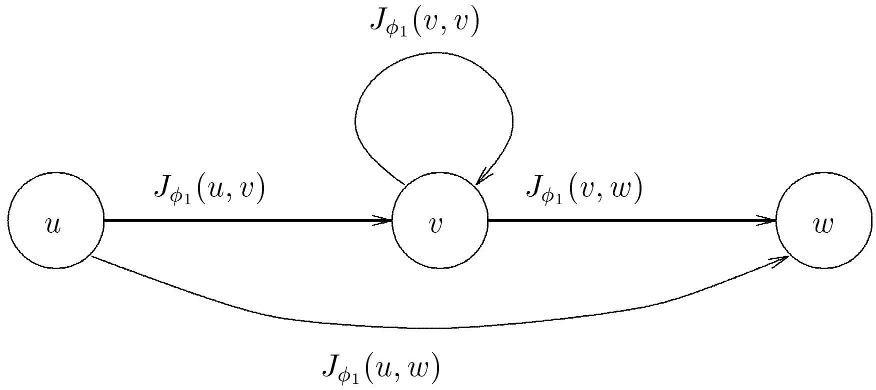

Figure 2: Railroad Gate Controller Example

graph with region sets V, the *conditional path graph* for ϕ_1 is a fully connected graph of V whose arcs are labeled with sets of pairs of the form : (π, T) where π is a static state predicate and T is an integer set. Conveniently, we call such pairs *conditional time expressions (CTE)*. Alternatively, we can say that the conditional path graph J_{ϕ_1} for ϕ_1 is a mapping from $V \times V$ to the power set of CTE's. For a $v, v' \in V$, if $(\pi, T) \in J_{\phi_1}(v, v')$, then for all interpretation $\mathcal{I}$, $t \in T$, and $s \in v$, if π is satisfied by $\mathcal{I}$, then there is a finite s-run of time t ending at an $s' \in v'$ such that ϕ_1 is satisfied all the way through the run except at s'. In subsection 4.2, we shall show that all our modal formula evaluations can be decomposed to the computation of conditional time expressions.

The kernel of this section is a Kleene's closure procedure which computes the conditional path graph. Its computation utilizes the following four types of integer set manipulations.

- $T_1 \cup T_2$ means $\{a_1 \mid a_1 \in T_1 \text{ or } a_2 \in T_2\}$.
- $T_1 + T_2$ 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.
- $\overline{T_1}$ is the complement of T_1, i.e., $\{a_1 \mid a_1 \in \mathcal{N}; a_1 \notin T_1\}$.

It can be shown that all integer sets resulting from such manipulations in our algorithm are semilinear.[1] Semilinear expressions are convenient notations for expressing infinite integer sets constructed regularly. They are also closed under the four 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 $a + c*$. 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 four manipulations can all be transformed back into PNF. Due to page-limit, we shall skip the details here.

The intuition behind our algorithm for computing the conditional path graph is a vertex bypassing scheme. Suppose, we have three regions u, v, w whose connections in the conditional path graph is shown in Figure 2. Then it is clear that by bypassing region v, we realized that $J_{\phi_1}(u, w)$ should be a superset of

$$\{(\pi_1 \wedge \pi_2 \wedge \bigwedge_{(\pi_3, T_3) \in D} \pi_3,\ T_1 + T_2 + \sum_{(\pi_3, T_3) \in D} T_3*) \mid (\pi_1, T_1) \in J_{\phi_1}(u, v); (\pi_2, T_2) \in J_{\phi_1}(v, w); D \subseteq J_{\phi_1}(v, v)\}$$

Our conditional path graph construction algorithm utilizes a Kleene's closure framework to calculate all the arc labels.

In subsection 4.1, we kind of extend the regions graph concepts in [ACD90] and define parametric region graph. In subsection 4.2, we define conditional path graph, present algorithm to compute it, and present our labelling algorithm for parametric analysis problem. In subsections 4.3 and 4.4, we briefly prove the

[1] A seminlinear 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 $a, b_1, \ldots, b_n \in \mathcal{N}$.

algorithm's correctness and analyze its complexity.

4.1 Parametric region graph

The brilliant concept of region graphs were originally discussed and used in [ACD90] for verifying dense-time systems. A region graph partitions its system state space into finitely many behavior-equivalent subspaces. Our parametric region graphs extend from Alur et al's region graph and contains information on parameter variable restrictions. Beside parameter variables, our parametric region graphs have an additional clock κ which gets reset to zero once its reading reaches one. κ is not used in the user-given SPA 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 formulae of PCTL. The reading of κ is always between 0 and 1, that is, for every state s, $0 \leq s(\kappa) \leq 1$.

The *timing constants* in an SPA 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 subformulae 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 $K_{A:\phi}$ be the largest timing constant used in both A and ϕ for the given parametric analysis problem instance.

For each $\delta \in \mathcal{R}^+$, we define *fract*(δ) as the fractional part of δ, i.e. *fract*$(\delta) = \delta - \lfloor\delta\rfloor$.

Definition 6 : Regions

Given an SPA $A = (Q, q_0, P, C, H, \chi, E, \rho, \tau)$ and a PCTL formula ϕ for A, two states s, s' of A, $s \cong_{A:\phi} s'$ (i.e. s and s' are equivalent with respect to A and ϕ) iff the following conditions are met.

- For each $p \in P$, $s(p) = s'(p)$.
- For each $x - y \sim c$ used in A or ϕ, $s(x) - s(y) \sim c$ iff $s'(x) - s'(y) \sim c$.
- For each $x \in C$, if either $s(x) \leq K_{A:\phi}$ or $s'(x) \leq K_{A:\phi}$, then $\lfloor s(x)\rfloor = \lfloor s'(x)\rfloor$.
- For every $x, y \in C \cup \{0, \kappa\}$, *fract*$(s(x)) \leq$ *fract*$(s(y))$ iff *fract*$(s'(x)) \leq$ *fract*$(s'(y))$. where $s(0) = s'(0) = 0$.

$[s]$ denotes the equivalent class of A's states, with respect to relation $\cong_{A:\phi}$, to which s belongs and it is called a *region*. $\|$

Note because of our assumption of unambiguous SPA's, we know that for all $s' \in [s]$, $s'^Q = s^Q$. Using the above definition, parametric region graph is defined as follows.

Definition 7 : Parametric Region Graph (PR-graph)

The *Parametric Region Graph* (PR-graph) for an SPA $A = (Q, q_0, P, C, H, \chi, E, \rho, \tau)$ and a PCTL formula ϕ is a directed graph $G_{A:\phi} = (V, F)$ such that the vertex set V is the set of all regions and the arc set F consists of the following two types of arcs.

- An arc (v, v') may represent *meta-state transitions* in A. That is, for every $s \in v$, there is an $s' \in v'$ such that $s \rightarrow s'$.
- An arc (v, v') may be a *time arc* and represent passage of time in the same meta-state. Formally, for every $s \in v$, there is an $s' \in v'$ such that
  - $s + \delta = s'$ for some $\delta \in \mathcal{R}^+$;
  - there is no $\dot{s}$ and $\dot{\delta} \in \mathcal{R}^+$, $0 < \dot{\delta} < \delta$, s.t. $[\dot{s}] \neq v$, $[\dot{s}] \neq v'$, $s + \dot{\delta} = \dot{s}$, and $\dot{s} + \delta - \dot{\delta} = s'$.

Just as in [Wang95], propositional value-changings within the same meta-states are taken care of automatically.

For each (v, v') in F, we let $\epsilon(v, v') =\uparrow$ if going from states in v to states in v', the reading of κ increments from a noninteger to an integer; $\epsilon(v, v') =\downarrow$ if going from states in v to states in v', the reading of κ increments from an integer to a noninteger; otherwise $\epsilon(v, v') = 0$. Also $v \models$ *fract*$(\kappa) = 0$ iff for all $s \in v$, $s(\kappa)$ is an

KClosure$_{\phi_1}(V,F)$
/* It is assumed that for all regions $v \in V$, we know the static state predicate condition $L^{\phi_1}(v)$ which makes ϕ_1 satisfied at v. */
{
(1) For each $(v,w) \in F$, if $\epsilon(v,w) =\uparrow$, {
(1) then let $J_{\phi_1}(v,w) := \{(L^{\phi_1}(v) \wedge v(\chi(v^Q)) \wedge v'(\chi(v'^Q)) \wedge v(\tau(v^Q, v'^Q)), 1)\}$;
(2) else let $J_{\phi_1}(v,w) := \{(L^{\phi_1}(v) \wedge v(\chi(v^Q)) \wedge v'(\chi(v'^Q)) \wedge v(\tau(v^Q, v'^Q)), 0)\}$.
}
(2) for each $v \in V$, do {
(1) for each $u, w \in V$, let
$$J_{\phi_1}(u,w) := J_{\phi_1}(u,w) \cup \left\{ (\pi_1 \wedge \pi_2 \wedge \textstyle\bigwedge_{(\pi_3,T_3)\in D} \pi_3,\ T_1 + T_2 + \sum_{(\pi_3,T_3)\in D} T_3 *) \;\middle\vert\; \begin{array}{l} (\pi_1,T_1) \in J_{\phi_1}(u,v); \\ (\pi_2,T_2) \in J_{\phi_1}(v,w); \\ D \subseteq J_{\phi_1}(v,v) \end{array} \right\}$$
}
}

Table 1: Construction of the conditional path graph

integer.

Also we conveniently write $v \models_{\mathcal{I}} \phi_1$ for some PCTL formula ϕ_1 when for all $s \in v(s \models_{\mathcal{I}} \phi_1)$. Similarly, we let v^Q be the meta-state such that for all $s \in v(v^Q = s^Q)$. ||

Since regions have enough informations to determine the truth values all propositions and clock inequalities used in a parametric analysis problem, we can define the mapping from state predicates to static state predicates through a region. Formally, given a region v and a state predicate η, we write $v(\eta)$ for the static predicate constructed according to the following rules.

- $v(\mathit{false})$ is *false*.
- $v(p)$ is *true* iff $\forall s \in v(s(p) = \mathit{true})$; $v(p)$ is *false* otherwise.
- $v(x - y \sim c)$ is *true* iff $\forall s \in v(s(x) - s(y) \sim c)$; $v(x - y \sim c)$ is *false* otherwise.
- $v(x \sim c)$ is *true* iff $\forall s \in v(s(x) \sim c)$; $v(x - y \sim c)$ is *false* otherwise.
- $v(\sum a_i\alpha_i \sim c)$ is $\sum a_i\alpha_i \sim c$.
- $v(\eta_1 \vee \eta_2) = v(\eta_1) \vee v(\eta_2)$.
- $v(\neg\eta_1) = \neg(\eta_1)$.

For convenience, we let $\langle\kappa\rangle v$ be the region in a PR-graph that agrees with v in every aspect except that for all $s' \in \langle\kappa\rangle v$, $s'(\kappa) = 0$. Given a PCTL formula ϕ and a path (cycle) $\Gamma = \langle v_1 v_2 \ldots v_m\rangle$, Γ is called a *ϕ-path* (*ϕ-cycle*) iff there is an interpretation $\mathcal{I}$ such that for each $1 \le i < m$ and $s \in v_i$, $s \models_{\mathcal{I}} \phi$.

4.2 Labeling Algorithm

To compute the parametric condition for a parametric modal formula like $\exists\phi_1\mathcal{U}_{\sim\theta}\phi_2$ at a region, we can instead decompose the formula into a Boolean combinations of path conditions and then compute those path conditions. For example, suppose under interpretation $\mathcal{I}$, we know there exists a ϕ_1-path $v_1v_2\ldots v_n$ of time 5. Then a sufficient condition for all states in v_1 satisfying $\exists\phi_1\mathcal{U}_{\le\theta}\phi_2$ is that $\mathcal{I}(\theta) \ge 5 \wedge v_n \models_{\mathcal{I}} \phi_2 \wedge \bigwedge_{1\le i<n} v_i \models \phi_1$. Now we define our second new data structure : *conditional path graph* to prepare for the presentation of the algorithm.

Definition 8 *:* **Conditional path graph**
Given a region graph $G_{A:\phi} = (V, F)$ and a subformula ϕ_1 of ϕ, the *conditional path graph* for ϕ_1, denoted as J_{ϕ_1} is a mapping from $V \times V$ to the power set of conditional time expressions such that for all $v, v' \in V$, if $(\pi, T) \in J_{\phi_1}(v, v')$, then for all interpretation $\mathcal{I}$ satisfying π, $t \in T$, and $s \in v$, there is a finite s-run of time t ending at an $s' \in v'$ such that ϕ_1 is satisfied all the way through the run except at s'. $\|$

The procedure for computing $J_{\phi_1}()$ is presented in table 1. Once the conditional path graph has been constructed for ϕ_1 using KClosure$_{\phi_1}()$, we can then turn to the labeling algorithm in table 2 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 connect the conditional time expressions in the arc labels to parametric conditions ?" Suppose, we want to examine if from v to v', there is a run satisfying the parametric requirement of $\geq \theta$. The condition can be derived as $\bigvee_{(\pi,T)\in J_{\phi_1}(v,v')} \pi \wedge T \geq \theta$ where $T \sim \theta$ with semilinear expressions T in PNF and (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 since we assume that the operands are in PNF, we do not have to pay attention to the case of $+, *, -$.

Table 2 presents the labeling alogrithm for $L^{\phi}(v)$. This algorithm maps pairs of vertices and temporal logic formulas to a Boolean combination of linear inequalities with parameter variables as free variables. Also note the labeling algorithm relies on the special case of $\exists\Box_{\geq 0}\phi_j$ which essentially says there is an infinite computation along which ϕ_j is always true.

Also the presentation in table 2 only covers some typical cases. For the remaining cases, please check the appendix.

4.3 Correctness

The following lemma establishes the correctness of our labeling algorithm.
LEMMA 1 *Given* PAP(A, ϕ)*, an interpretation $\mathcal{I}$ for H, and a vertex v in $G_{A:\phi}$, after executing $L^{\phi}(v)$ in our labeling algorithm, $\mathcal{I}$ satisfies $L^{\phi}(v)$ iff $v \models_{\mathcal{I}} \phi$.*
proof : The proof follows a standard structural induction on ϕ, which we often saw in related model-checking literature, and very much resembles the one in [Wang95]. Due to page-limit, we shall omit it here. $\|$

4.4 Complexity

According to our construction, the number of regions in $G_{A:\phi}$, denoted as $|G_{A:\phi}|$, is at most $3|Q| \cdot (K_{A:\phi} + 1)^C \cdot (|C| + 1)!$ where coefficient 3 and constant +1 reflect the introduction of ticking indicator κ. The inner loop of KClosure$_{\phi_1}$ will be executed for $|G_{A:\phi}|^3$ times. Each iteration takes time proportional to $|J_{\phi_1}(u, v)||J_{\phi_1}(v, w)|2^{|J_{\phi_1}(v,v)|}$. The conditional path graph arc labels, i.e. $J_{\phi_1}(u, v)$, roughly corresponds to the set of simple paths from u to v, although they utilize the succinct representation of semilinear expressions. Thus according to the complexity analysis in [Wang95], we find that procedure KClosure$_{\phi_1}()$ has complexity doubly exponential to the size of $G_{A:\phi}$, and thus triply exponential to the size of input, assuming constant time for the manipulation of semilinear expressions.

We now analyze the complexity of our labeling procedure. In table 2, procedure $L^{\phi_i}()$ invokes KClosure$_{\phi_j}()$ at most once. Label(A, ϕ) invokes $L^{\phi_i}()$ at most $|G_A||\phi|$ times. Thus the complexity of the algorithm is roughly triply exponential to the size of A and ϕ, since polynomials of exponentialities are still exponentialities.

Finally, PCTL satisfiability problem is undecidable since it is no easier than TCTL satisfiability problem[ACD90].

Label(A, ϕ) {
  (1) construct the PR-graph $G_{A:\phi} = (V, F)$;
  (2) for each $v \in V$, recursively compute $L^{\phi}(v)$;
}

$L^{\phi_i}(v)$ {
switch(ϕ_i){
case (*false*), $L^{false}(v) := false$;

case (p) where $p \in P$, $L^p(v) := true$ if $v \models p$, else $L^p(v) := false$;

case ($x - y \sim c$), if either x or y is zero in v, evaluate $x - y \sim c$ as in the next case; else $x - y \sim c$ is evaluated to the same value as it is in any region u such that $(u, v) \in F$.

case ($x \sim c$), $L^{\phi_i}(v) := true$ if $v \models (\phi_i)$, else $L^{\phi_i}(v) := false$;

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 ($\neg\phi_j$), $L^{\neg\phi_j}(v) := \neg L^{\phi_j}(v)$;

case ($\exists\Box_{\geq 0}\phi_j$), {
  (1) KClosure$_{\phi_j}(V, F)$;
  (2) let $L^{\exists\Box_{\geq 0}\phi_j}(v)$ be $\bigvee_{u \in V}\left(\left(\bigvee_{(\pi,T)\in J_{\phi_j}(\langle\kappa\rangle v, u)} \pi\right) \wedge \left(\bigvee_{(\pi,T)\in J_{\phi_j}(u,u)}(\pi \wedge T > 0)\right)\right)$.
}

case ($\exists\phi_j\mathcal{U}_{\geq\theta}\phi_k$), {
  (1) KClosure$_{\phi_j}(V, F)$;
  (2) let $L^{\exists\phi_j\mathcal{U}_{\geq\theta}\phi_k}(v)$ be $\bigvee_{u \in V}\left(\left(\bigvee_{(\pi,T)\in J_{\phi_j}(\langle\kappa\rangle v, u)}(\pi \wedge T \geq \theta)\right) \wedge L^{\phi_k}(u) \wedge L^{\exists\Box_{\geq 0}\mathrm{true}}(u)\right)$.
}

case ($\exists\phi_j\mathcal{U}_{\leq\theta}\phi_k$, $\exists\phi_j\mathcal{U}_{>\theta}\phi_k$, $\exists\phi_j\mathcal{U}_{<\theta}\phi_k$, or $\exists\phi_j\mathcal{U}_{=\theta}\phi_k$)
These cases are treated in ways similar to the above case and are left in table **??** which is appended at the end of the paper.

case ($\forall\phi_j\mathcal{U}_{\geq\theta}\phi_k$), {
  (1) KClosure$_{\phi_j}(V, F)$;
  (2) let $L^{\forall\phi_j\mathcal{U}_{\geq\theta}\phi_k}(v)$ be

$$\neg\left(\begin{array}{cl} & L^{\exists\Diamond_{<\theta}\neg\phi_j}(\langle\kappa\rangle v) \;\vee\; \left(\theta = 0 \wedge \left(L^{\exists\Box_{\geq 0}\neg\phi_k}(\langle\kappa\rangle v) \vee L^{\exists(\neg\phi_k)\mathcal{U}_{\geq 0}\neg(\phi_j\vee\phi_k)}(\langle\kappa\rangle v)\right)\right) \\ \vee & \left(\theta > 0 \wedge \bigvee_{u_1,u_2 \in V}\left(\begin{array}{cl} & \bigvee_{(\pi,T)\in J_{\phi_j}(\langle\kappa\rangle v, u_1)}(\pi \wedge T = \theta - 1) \wedge L^{\phi_j}(u_1) \wedge \epsilon(u_1, u_2) =\uparrow \\ \wedge & \left(L^{\exists\Box_{\geq 0}\neg\phi_k}(u_2) \vee L^{\exists(\neg\phi_k)\mathcal{U}_{\geq 0}\neg(\phi_j\vee\phi_k)}(u_2)\right)\end{array}\right)\right)\end{array}\right)$$
}

case ($\forall\phi_j\mathcal{U}_{\leq\theta}\phi_k$, $\forall\phi_j\mathcal{U}_{>\theta}\phi_k$, $\forall\phi_j\mathcal{U}_{<\theta}\phi_k$, or $\forall\phi_j\mathcal{U}_{=\theta}\phi_k$),
These cases are treated in ways similar to above case and are left in table **??** which is appended at the end of the paper.
}

Table 2: Labeling algorithm

5 Conclusion

With the success of CTL-based techniques in automatic verification for computer systems [Bryant86, BCMDH90, HNSY92], it would be nice if a formal theory appealing to the common practice of real-world projects can be developed. We feel hopeful that the insight and techniques used in this paper can be further applied to help verifying reactive systems in a more natural and productive way.

Acknowledgements

The authors would like to thank Prof. Tom Henzinger. His suggestion to use dynamic programming to solve PTCTL parametric timing analysis problem triggered the research.

References

[ACD90] Alur, R., Courcoubetis, C., and Dill, D.L. (1993), Model-Checking in Dense Real-Time, *Information and Computation* **104**, Nr. 1, pp. 2–34.

[AD90] Alur, R. and Dill, D., (1990), Automata for modeling real-time systems, in "Automata, Languages and Programming: Proceedings of the 17th ICALP," Lecture Notes in Computer Science, **443**, pp. 332–335, Springer-Verlag, Berlin/New York.

[AH90] Alur, R. and Henzinger, T.A. (1990), Real-Time Logics : Complexity and Expressiveness, in "Proceedings, 5th IEEE LICS."

[AHV93] Alur, R., Henzinger, T.A., and Vardi, M.Y. (1993), Parametric Real-Time Reasoning, in "Proceedings, 25th ACM STOC," pp. 592–601.

[BCMDH90] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L.Dill, L.J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond, IEEE LICS, 1990.

[Bryant86] R.E. Bryant. Graph-based Algorithms for Boolean Function Manipulation, IEEE Trans. Comput., C-35(8), 1986.

[CE81] 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.

[CES86] 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.

[CY92] 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.

[HHWT95] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: the next generation. In proceedings of the 16th Annual Real-Time System Symposium, pages 56-65. IEEE Computer Society Press, 1995.

[HNSY92] T.A. Henzinger, X. Nicollin, J. Sifakis, S. Yovine. Symbolic Model Checking for Real-Time Systems, IEEE LICS 1992.

[Wang95] Wang, F. (1995), Parametric Timing Analysis for Real-Time Systems, in "Proceedings, 10th IEEE Symposium on Logic in Computer Science."

APPENDIX : Elaboration on cases in Label(A, ϕ)

{

(1) KClosure$_{\phi_j}(V, F)$;

(2) if ϕ_i is of the form $\exists\phi_j\mathcal{U}_{>\theta}\phi_k$, let $L^{\exists\phi_j\mathcal{U}_{>\theta}\phi_k}(v)$ be

$$\bigvee_{u\in V}\left(\begin{array}{ll} & \left(\bigvee_{(\pi,T)\in J_{\phi_j}(\langle\kappa\rangle v,u)}(\pi\wedge T>\theta)\vee\left(\bigvee_{(\pi,T)\in J_{\phi_j}(\langle\kappa\rangle v,u)}(\pi\wedge T=\theta)\wedge u\models \mathit{fract}(\kappa)\neq 0\right)\right)\\ \wedge & L^{\phi_k}(u)\wedge L^{\exists\Box_{\geq 0}\mathrm{true}}(u)\end{array}\right)$$

(3) if ϕ_i is of the form $\exists\phi_j\mathcal{U}_{\leq\theta}\phi_k$, let $L^{\exists\phi_j\mathcal{U}_{\leq\theta}\phi_k}(v)$ be

$$\bigvee_{u\in V}\left(\begin{array}{ll} & \left(\bigvee_{(\pi,T)\in J_{\phi_j}(\langle\kappa\rangle v,u)}(\pi\wedge T<\theta)\vee\left(\bigvee_{(\pi,T)\in J_{\phi_j}(\langle\kappa\rangle v,u)}(\pi\wedge T=\theta)\wedge u\models \mathit{fract}(\kappa)= 0\right)\right)\\ \wedge & L^{\phi_k}(u)\wedge L^{\exists\Box_{\geq 0}\mathrm{true}}(u)\end{array}\right)$$

(4) if ϕ_i is of the form $\exists\phi_j\mathcal{U}_{<\theta}\phi_k$, let $L^{\exists\phi_j\mathcal{U}_{<\theta}\phi_k}(v)$ be

$$\bigvee_{u\in V}\left(\bigvee_{(\pi,T)\in J_{\phi_j}(\langle\kappa\rangle v,u)}(\pi\wedge T<\theta)\wedge L^{\phi_k}(u)\wedge L^{\exists\Box_{\geq 0}\mathrm{true}}(u)\right)$$

(5) if ϕ_i is of the form $\exists\phi_j\mathcal{U}_{=\theta}\phi_k$, let $L^{\exists\phi_j\mathcal{U}_{=\theta}\phi_k}(v)$ be

$$\bigvee_{u\in V}\left(\bigvee_{(\pi,T)\in J_{\phi_j}(\langle\kappa\rangle v,u)}(\pi\wedge T=\theta)\wedge u\models \mathit{fract}(\kappa)=0\wedge L^{\phi_k}(u)\wedge L^{\exists\Box_{\geq 0}\mathrm{true}}(u)\right)$$

(6) if ϕ_i is of the form $\forall\phi_j\mathcal{U}_{>\theta}\phi_k$, let $L^{\forall\phi_j\mathcal{U}_{>\theta}\phi_k}(v)$ be

$$\neg\left(\begin{array}{l} L^{\exists\Diamond_{\leq\theta}\neg\phi_j}(\langle\kappa\rangle v)\\ \vee\bigvee_{u_1,u_2\in V}\left(\begin{array}{l}\bigvee_{(\pi,T)\in J_{\phi_j}(\langle\kappa\rangle v,u_1)}(\pi\wedge T=\theta)\wedge L^{\phi_j}(u_1)\\ \wedge\,\epsilon(u_1,u_2)=\downarrow\wedge\left(L^{\exists\Box_{\geq 0}\neg\phi_k}(u_2)\vee L^{\exists(\neg\phi_k)\mathcal{U}_{\geq 0}\neg(\phi_j\vee\phi_k)}(u_2)\right)\end{array}\right)\end{array}\right)$$

(7) if ϕ_i is of the form $\forall\phi_j\mathcal{U}_{\leq\theta}\phi_k$, let $L^{\forall\phi_j\mathcal{U}_{\leq\theta}\phi_k}(v)$ be

$$\neg\left(\begin{array}{ll} & \left(\bigvee_{u_1,u_2\in V}\left(\bigvee_{(\pi,T)\in J_{\neg\phi_k}(\langle\kappa\rangle v,u_1)}(\pi\wedge T=\theta)\wedge L^{\neg\phi_k}(u_1)\wedge\epsilon(u_1,u_2)=\downarrow\wedge L^{\exists\Box_{\geq 0}\mathrm{true}}(u_2)\right)\right)\\ \vee & L^{\exists(\neg\phi_k)\mathcal{U}_{\leq\theta}\neg(\phi_j\vee\phi_k)}(\langle\kappa\rangle v)\end{array}\right)$$

(8) if ϕ_i is of the form $\forall\phi_j\mathcal{U}_{<\theta}\phi_k$, let $L^{\forall\phi_j\mathcal{U}_{<\theta}\phi_k}(v)$ be

$$\neg\left(\begin{array}{ll} & \left(\bigvee_{u_1,u_2\in V}\left(\bigvee_{(\pi,T)\in J_{\neg\phi_j}(\langle\kappa\rangle v,u_1)}(\pi\wedge T=\theta-1)\wedge L^{\neg\phi_k}(u_1)\wedge\epsilon(u_1,u_2)=\uparrow\wedge L^{\exists\Box_{\geq 0}\mathrm{true}}(u_2)\right)\right)\\ \vee & L^{\exists(\neg\phi_k)\mathcal{U}_{<\theta}\neg(\phi_j\vee\phi_k)}(\langle\kappa\rangle v)\end{array}\right)$$

(9) If ϕ_i is of the form $\forall\phi_j\mathcal{U}_{=\theta}\phi_k$, let $L^{\forall\phi_j\mathcal{U}_{=\theta}\phi_k}(v)$ be

$$\neg\left(\begin{array}{l} L^{\exists\Diamond_{<\theta}\neg\phi_j}(\langle\kappa\rangle v)\\ \vee\left(\theta=0\wedge\left(\begin{array}{l}\bigvee_{u_1,u_2\in V}\left(\begin{array}{l}\bigvee_{(\pi,T)\in J_{\neg\phi_k}(\langle\kappa\rangle v,u_1)}(\pi\wedge T=0)\wedge L^{\neg\phi_k}(u_1)\\ \wedge\,\epsilon(u_1,u_2)=\downarrow\wedge L^{\exists\Box_{\geq 0}\mathrm{true}}(u_2)\end{array}\right)\\ \vee\, L^{\exists(\neg\phi_k)\mathcal{U}_{=0}\neg(\phi_j\vee\phi_k)}(\langle\kappa\rangle v)\end{array}\right)\right)\\ \vee\bigvee_{u_1,u_2\in V}\left(\begin{array}{l}\theta>0\wedge\bigvee_{(\pi,T)\in J_{\phi_j}(\langle\kappa\rangle v,u_1)}(\pi\wedge T=\theta-1)\wedge L^{\phi_j}(u_1)\wedge\epsilon(u_1,u_2)=\uparrow\\ \wedge\left(\begin{array}{l}\bigvee_{u_3,u_4\in V}\left(\begin{array}{l}\bigvee_{(\pi,T)\in J_{\neg\phi_k}(\langle\kappa\rangle u-2,u_3)}(\pi\wedge T=0)\wedge L^{\neg\phi_k}(u_3)\\ \wedge\,\epsilon(u_3,u_4)=\downarrow\wedge L^{\exists\Box_{\geq 0}\mathrm{true}}(u_4)\end{array}\right)\\ \vee\, L^{\exists(\neg\phi_k)\mathcal{U}_{=0}\neg(\phi_j\vee\phi_k)}(u_2)\end{array}\right)\end{array}\right)\end{array}\right)$$

}

Notations:

1. Ω (capital omega) .... a conditional semilinear expression (CSE)
2. α (alpha), β (beta), γ (gamma) .... parameters
3. χ (chi) .... a mapping from meta-states to state predicates
4. η (eta) .... a state predicate
5. κ (kappa) .... ticking indicator
6. ϕ (phi) .... a PCTL formula or specification
7. π (pi) .... condition in CSE (π, T)
8. ρ (rho) .... assignment labeling function for E
9. τ (tau) .... triggering condition labeling function for E
10. θ (theta) .... a parameter or integer constant
11. A .... a statically parametric automaton (SPA)
12. $B(P, C, H)$ .... a set of state predicates for P, C, and H
13. C .... a set of clock variables
14. D .... a set of conditional path-times between two regions
15. E .... a set of meta-state transitions in an SPA
16. F .... a set of transitions in $G_A = (V, F)$
17. G_A .... the Statically Parametric Region Graph of SPA A
18. H .... a set of parameters
19. $\mathcal{N}$(calligraphical capital N) .... the set of non-negative integers
20. P .... a set of atomic propositions
21. $\mathcal{R}^+$ .... set nonnegative real numbers
22. Q .... a set of meta-states
23. V .... a vertex set of regions in $G_A = (V, F)$
24. $\mathcal{I}$(calligraphical capital I) .... an interpretation for H
25. a, b, c, d .... non-negative integer constants
26. p, q, r .... atomic propositions
27. T .... a set of times in CSE (π, T)
28. v .... a region in V of $G_A = (V, F)$
29. x, y, z .... clock variables