Model Checking of Energy Consumption Behavior

Shin Nakajima

Abstract Energy consumption is one of the primary non-functional properties to be addressed early in software system development. Model-based analysis methods are introduced in order to supplement the current practice of runtime profiler techniques. In the present paper, the energy consumption analysis is classified as a duration-bounded cost constraint problem. Specifically, behavioral contracts based on Power Consumption Automata and properties written in terms of weighted linear temporal logic with freeze quantifiers are proposed. In addition, the problem is solved by model-checking of such logic formulas with respect to the automaton.

1 Introduction

Energy consumption is one of the primary non-functional concerns in software-intensive systems, from networked embedded systems to user-centric smartphones, which constitute any modern complex system infrastructure. A software-intensive system, even though functionally correct, may suffer from unexpected energy consumption. Although root causes include design flaws, post-analysis techniques are adapted for checking programs. In Android-based smartphone [1], for example, energy profilers (cf.[20][22]) are used to detect such energy bugs (*e-bugs*) [19]. Since a profiler is a runtime monitor used for checking the energy consumption of running programs, it has the same disadvantages that program testing methods have. Specifically, the profiler can be used only after all of the programs are completed, and the coverage is limited by the supplied test cases.

A model-based energy consumption analysis method is necessary in order to counter the disadvantage of the runtime profiler method. With appropriate design notations, the energy consumption behavior of software-intensive systems can be represented and considered early in development. This method is particularly useful

Shin Nakajima
National Institute of Informatics, Tokyo, Japan, e-mail: nkjm@nii.ac.jp

for finding design faults caused by, for example, improper use of power management application interfaces of the Android framework.

In the present paper, we propose a model-based formal analysis method for the energy consumption problem, which is formulated as a duration-bounded cost constraint problem. In addition to Power Consumption Automaton (PCA) [13] accounting for the energy consumption behavior, a new variant of linear temporal logic with freeze quantifiers (fWLTL) is introduced so that the duration-bounded cost constraint problem is solved by logic model-checking. For this purpose, we propose a new definition of a PCA in terms of a Timed Automaton with observers [5][6][7]. Previously in [13], the PCA was defined as an n-rate Timed System [4].

The reminder of the present paper is organized as follows. Section 2 describes the concept of the model-based analysis method. Sections 3 and 4 introduce the PCA and fWLTL respectively, and the model-checking method is introduced in Section 5. Related research is discussed in Section 6, and Section 7 concludes the paper.

2 Model-based Analysis

2.1 Energy Consumption Behavior

The energy consumption of a system is attributed to two types of sources. The first is the hardware infrastructure that constitutes the main part of the system including the CPU and memory. These are not controllable by application programs. The second type is peripheral devices that particular application programs use, which include network interfaces, and sensors or actuators. The LCD display, which consumes a lot of energy, also falls into this second category. An application program, running as a backend worker, may not use the display, because it does not provide a GUI for human users.

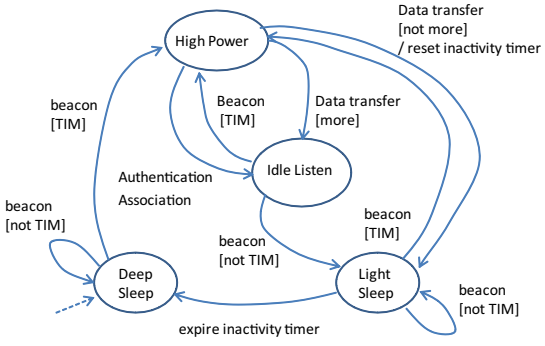

Fig. 1 Example of a PCA Diagram

Although hardware components are direct consumers of energy, application programs that are required in order to make use of these components are responsible for the consumption. Android-based systems [1], for example, use an aggressive power-saving strategy to make the system sleep when periods of inactivity are detected. Some application programs, however, will make the system awake even when a human user does not touch the screen for some time. The application program calls the power management API of the Android framework such as wake-lock methods, so that the display is kept awake as long as some active locks remain. In the default mode, the wake-lock is maintained by a reference-counting method. An acquired wake-lock must have a matching release call. Otherwise, the wake-lock is kept alive even after the caller program is destroyed. These methods control the energy consumption behavior of the hardware components, and thus the energy consumption is attributed to the caller program. Specifically, improper use of wake-locks result in energy bugs [13][19]. The energy consumption behavior of application programs results from an integrated view of their own functional behavior and the amount of energy consumed by the hardware components.

As an example of energy consumption behavior, Figure 1 shows a state-transition diagram of WiFi client hardware operating in a power-saving mode. The diagram consists of four states, called power states, and several state-transition edges. In Figure 1, `DeepSleep` is the initial state. A transition is enabled to `HighPower` state when a beacon signal, which starts data transfer, is received. The state then alternates between the `IdleListen` and `HighPower` states. In the `IdleListen`, the WiFi client awaits the arrival of a new data frame. When the data frame transfer is over, the `LightSleep` state is entered in order to prepare for a quick restart for data transfer in the near future. An inactivity timer, which causes the transition to the `DeepSleep` state, is also set. The diagram shows that a state-transition sequence consists of many instances of the four power states.

The energy consumption is different for each power state. The `HighPower` state consumes a lot of energy because it needs a large amount of computing resources in order to decode the transferred frames. In the `DeepSleep` state, the electric current is only necessary to activate a small number of hardware circuits and thus the energy consumption is small. If we let $F^j(t)$ be a function of time to represent the rate of consumed energy at a state indexed by j, the total power consumption from time a to b is $P^j = \int_a^b F^j(t)dt$. Since the power states are visited several times as the state transitions progress, the total energy consumption is calculated to be a sum of P^j; $P = \sum_{j=0}^{n} P^j = \sum_{j=0}^{n} \int_{a^j}^{b^j} F^j(t)dt$. We assume here that the state transition sequence is bounded, and that each visited state is indexed by j.

A PCA [13] is a formal model that accounts for the energy consumption behavior. It is informally represented by the diagram in Figure 1. The PCA introduces a linear approximation, so that $dP^j/dt = F^j(t) = M^j$ with a constant M^j for each power state. The constant M^j refers to an average rate of energy consumption at a particular power state j. Then, the total energy consumption P is $\sum_{j=0}^{n} M^j \times (b^j - a^j)$. If we know the constant M^j for each power state, for example, `HighPower` in Figure 1, then the PCA model concisely depicts the energy consumption behavior.

The PCA is regarded as a member of a subclass of Linear Hybrid Automata (LHA) [4] because the dynamics of power consumption variable P^j takes the form of $dP^j/dt = M^j$, and the inactivity timer X is a clock variable of $dX/dt = 1$. However, the PCA is simpler than the LHA in that P^j is an observer and does not have any effect on the state-transition behavior. Contrarily, the clock variable X controls state-transition timings. In Figure 1, for example, the transition from `LightSleep` to `DeepSleep` is enabled by time-out of the inactivity timer. Therefore, a PCA can be regarded as a Timed Automaton (TA) with observers [7], a Weighted Timed Automaton (WTA) [5], or a Priced Timed Automaton (PTA) [6]. These are members of a proper subclass of the LHA and an extension of the TA.

The dynamics in a variant of the PCA are extended to be probabilistic [15]. If we let R be a random variable that is independent and identically distributed (i.i.d.) with respect to probabilistic distribution $f(r)$, the dynamics are written as $dP^j/dt = \alpha(R) \times M^j$ using probabilistic values $\alpha(R)$ with $\alpha \geq 1$. The weight can be used to encode the probabilistic component. Probabilistic or statistical method [21] can be applied to such an extension.

2.2 Energy Consumption Properties

From a naive viewpoint, the properties that must be checked appear simple enough to state that the total amount of consumed energy must be less than a specified maximum value. Since the amount of consumed energy is proportional to the amount of time during which the hardware components are used, the above simple property is eventually violated if the hardware components are used indefinitely. Checking the properties must be limited with respect to time. Thus, the problem involves checking the *duration-bounded cost constraints*, in which the cost refers to the amount of energy consumption. This problem is more general than the duration-bounded reachability of the TA [2]. An expressive and flexible property language is needed to specify precisely and compactly both the durations and constraints.

The example in Figure 1 may be checked to see whether a simple property is satisfied, such that the total amount of energy consumed from `IdleListen` to `LightSleep` is less than a given maximum. A standard linear temporal logic (LTL) formula of the form $\Box(\texttt{IdleListen} \Rightarrow \Diamond\texttt{LightSleep})$ describes the temporal behavior of the desired property. The formula states that *it is always the case that the* `LightSleep` *state is eventually reached if it is in the* `IdleListen` *state*. This LTL formula is extended in order to refer to both the real-time timing and the real-valued energy. To represent the timing, Metric Temporal Logic (MTL) [18] and TPTL [3], for example, have been proposed. However, MTL and TPTL are not applicable to the real-valued energy consumption variables.

We introduce the freeze quantifier that extends the original proposal in TPTL [3] so that the quantifiers can refer to either the energy consumption values or the real-time clocks. If we use freeze quantifiers of the form $\exists x.\phi^x$, the above-mentioned property to be checked (φ_1) will be written as

$$\Box \downarrow^{\tau} x.\ \downarrow^{m} u.\ (\texttt{IdleListen} \Rightarrow \Diamond \downarrow^{\tau} y.\ \downarrow^{m} v.\ (\texttt{LightSleep} \wedge (y \leq x+10) \wedge (v \leq u+50)))\ ,$$

where x and y refer to the time points, while u and v record the consumed energy at specified states. Here, x and u are *frozen* at the state when the system is in the `IdleListen` state, and y and v are taken from the `LightSleep` state. If the start and target states of properties to be checked are pre-determined, then the algorithm in [2] or in [5][6] might be used. However, the formula of the type φ_1 may refer to any state proposition to specify the duration. Furthermore, imagine that we have another property φ_2.

$$\Box \downarrow^{\tau} x.\ (\texttt{DeepSleep} \Rightarrow\ \Diamond \downarrow^{\tau} y.\ (\texttt{HighPower} \wedge (y \leq x+100)))$$

Although φ_2 is less interesting, we can combine two properties to obtain $\varphi_1 \wedge \varphi_2$ or $\varphi_2 \Rightarrow \varphi_1$ in a flexible manner and describe the expressive property specifications.

If we have such an extension of the LTL with freeze quantifiers, model-based energy consumption analysis is formalized as a model-checking problem. In succeeding sections, we will formally define the PCA and the new extension of the LTL, and then investigate the model checking problem in detail.

3 Behavioral Model

3.1 Power Consumption Automaton

A Power Consumption Automaton (PCA) $\mathscr{A}$ over a set of atomic propositions *Prop* is defined as a Weighted Timed Automaton (WTA).

$$\langle\ Loc, C, D, \Sigma \cup \{\varepsilon\}, Edg, Flow, Inv, Lab\ \rangle$$

1. Loc is a finite set of locations. Each location corresponds to a power state.
2. C is a finite set of clock variables and D is a set of weight variables. C and D are disjoint ($C \cap D = \emptyset$). For a clock variable x ($\in C$), a constant $n (\in \mathscr{N})$, and an operator $\bowtie \in \{<, \leq, =, \geq, >\}$, constraints of the form $x \bowtie n$ and $x_1 - x_2 \bowtie n$ constitute a set of clock constraints $Z(C)$.
3. Σ is an alphabet that is a finite set of input symbols, and ε is an empty symbol.
4. Edg represents a set of transitions. It is a finite set $Loc \times Z(C) \times \Sigma \times 2^C \times Loc$. The element of Edg, (l_1, g, a, r, l_2), is written as $l_1 \xrightarrow{g,a,r} l_2$, where g is a guard condition in $Z(C)$, a is an input symbol ($\in \Sigma \cup \{\varepsilon\}$), and r refers to a set of clock variables ($\in 2^C$) to reset.

   Furthermore, the PCA is deterministic if the destination location l_2 is determined uniquely at a location l_1 with given a and g.
5. $Flow$ represents the dynamics to account for the change rate of weight variables. For non-negative real $\mathscr{R}_+$ and $\mathscr{R}_+^D$ being $D \to \mathscr{R}_+$, $Flow : Loc \to (\mathscr{R}_+^D \to \mathscr{R}_+^D)$.

For a particular weight variable p to keep track of energy consumption and a valuation $w \in \mathscr{R}_+^D$, $Flow(\ell)(w)(p) = dp/dt = M^\ell$.

6. *Inv* is a mapping to clock constraints. $Inv : Loc \rightarrow Z(C)$
7. *Lab* is a mapping to a set of atomic propositions. $Lab : Loc \rightarrow 2^{Prop}$

The above equation uses valuations for clock variables or weight variables $\nu \in \mathscr{R}_+^{C \cup D}$. Reset, delay, and multiplication are defined respectively below.

$$\nu[r](x) \;=\; \begin{cases} 0 & \text{if } x \in r \\ \nu(x) & otherwise, \end{cases}$$

$$\begin{aligned} (\nu + d)(x) &= \nu(x) + d, \\ (\nu \times e)(x) &= \nu(x) \times e \end{aligned}$$

For a probabilistic extension of the PCA, $Flow(\ell)(w)(p) = \alpha(R) \times M^\ell$, in which α is a probabilistic variable mentioned in Section 2.1.

Finally, two PCAs, where $(C_1 \cup D_1) \cap (C_2 \cup D_2) = \emptyset$, synchronize on the common set of input symbols $\Sigma_1 \cap \Sigma_2$. Whenever PCA_1 makes a discrete transition with a synchronization symbol $a \in (\Sigma_1 \cap \Sigma_2)$, PCA_2 also makes a transition.

3.2 Operational Semantics

Semantics of PCA $\mathscr{A}$ is given by a Labeled Transition System (LTS), $\langle\, S, T \,\rangle$. The state space S is a set of tuples consisting of a location l, a clock valuation v, and a weight valuation w. The clock invariants at location l are satisfied.

$$S = \{ (l, v, w) \in Loc \times \mathscr{R}_+^C \times \mathscr{R}_+^D \mid v \models Inv(l) \}$$

The transition T consists of regular and stutter transitions; $T = \{ \xrightarrow{d,e} \} \cup \{ \xrightarrow{d,\varepsilon} \}$. The details are explained below.

- Event-trigger discrete transitions $(l_1, v_1, w) \xrightarrow{e} (l_2, v_2, w)$

$$\exists (l_1 \xrightarrow{g,a,r} l_2) \in Edg \wedge v_1 \models g \wedge v_2 = v_1[r]$$

- Delayed transitions $(l, v, w_1) \xrightarrow{d} (l, v + d, w_2)$

$$\begin{aligned} & d \in \mathscr{R}_+ \wedge w_1 = f(0) \wedge w_2 = f(d) \wedge \\ & \quad \forall t \in]0, d[\mid\; v + t \models Inv(l) \wedge df/dt = Flow(l) \end{aligned}$$

The time advances by an amount of the delay (d) to become $v + d$, and the weight is updated. In the above, $f(t)$ refers to a continuous function differentiable in the open time interval $]0, d[$. For an energy consumption variable p, $Flow(\ell) = dp/dt = M^\ell$ at location ℓ, and thus $w_2(p) = M^\ell \times d \;+\; w_1(p)$.

- Null transitions $(l_1, v_1, w) \xrightarrow{\varepsilon} (l_2, v_2, w)$

$$\exists (l_1 \xrightarrow{g,\varepsilon,r} l_2) \in Edg \wedge v_1 \models g \wedge Flow(l) = \emptyset \wedge v_2 = v_1[r]$$

  The input symbol is empty (ε) and the weights are not changed.
- Regular transitions $(l_1, v_1, w_1) \xrightarrow{d,e} (l_2, v_2, w_2)$

  A delayed transition $(l_1, v_1, w_1) \xrightarrow{d} (l_1, v, w_2)$ followed by an event-trigger discrete transition $(l_1, v, w_2) \xrightarrow{e} (l_2, v_2, w_2)$.
- Stutter transitions $(l, v_1, w_1) \xrightarrow{d,\varepsilon} (l, v_2, w_2)$

  A delayed transition $(l, v_1, w_1) \xrightarrow{d} (l, v, w_2)$ followed by a self-loop null transition at location l, $(l, v, w_2) \xrightarrow{\varepsilon} (l, v_2, w_2)$

A transition sequence is a finite number of $\xrightarrow{d,e}$, or such sequences followed by an infinite number of $\xrightarrow{d,\varepsilon}$. The location l at which the stutter transition is defined is a final state.

Finally, a set of time points $\tau^j \in \mathscr{R}_+$ represents a time progression sequence, where $\tau^0 = 0$ and $\tau^{j+1} = \tau^j + d$ for a delayed transition $\xrightarrow{d}$. For a state $\sigma^j \in S$, a timed point is introduced as $\rho^j = (\sigma^j, \tau^j)$. Then, for a sequence of timed point $\rho = \rho^0 \rho^1 \cdots$, we have a set of timed sequences $L(\mathscr{A})$ generated by a PCA $\mathscr{A}$, which is written as $\rho \in L(\mathscr{A})$.

4 Property Specification

This section introduces Weighted Linear Temporal Logic with freeze quantifiers (fWLTL) as the property specification language.

4.1 Syntax

The syntax of fWLTL is shown below.

π	$:= c$	Constant $\in \mathscr{N}$ (Natural numbers)
	$\mid x + c$	Addition
ϕ	$:= p$	Atomic Proposition $\in Prop$
	$\mid \pi_1 \leq \pi_2$	Comparison
	$\mid \neg\, \phi$	Logical Negation
	$\mid \phi_1 \wedge \phi_2$	Conjunction
	$\mid \phi_1 \,\mathrm{U}\, \phi_2$	Until Operator
	$\mid \downarrow^m x \,.\, \phi^x$	Freeze Quantifier ranging over $C \cup D$
	$\mid \downarrow^\tau x \,.\, \phi^x$	Freeze Quantifier ranging over time points $\{\tau^j\}$

A variable $x \in Var$, where Var is a countable set of variables, appears free in an fWLTL formula ϕ^x. A closed formula ϕ does not have any free variable. Cost constraints constitute a set $Z(C \cup D)$ because we have $\pi_1 < \pi_2 \equiv \neg(\pi_2 \leq \pi_1)$, and $\pi_1 = \pi_2 \equiv (\pi_1 \leq \pi_2) \wedge (\pi_2 \leq \pi_1)$. Furthermore, the following standard abbreviations are used. $false \equiv p \wedge \neg p$, $true \equiv \neg false$, $\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$, $\phi_1 \Rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$, $\Diamond\phi \equiv true\ \mathrm{U}\ \phi$ (Eventually Operator), $\Box\phi \equiv \neg(\Diamond\neg\phi)$ (Globally Operator), $\phi_1\ \mathrm{R}\ \phi_2 \equiv \neg(\neg\phi_1\ \mathrm{U}\ \neg\phi_2)$ (Release Operator).

4.2 Semantics

We adapt the pointwise semantics for the fWLTL. A number of symbols are introduced.

- A timed state $\rho^j = (\sigma^j, \tau^j) = ((l_j, v_j, w_j), \tau^j)$
- A timed state sequence $\rho = \rho^0 \rho^1 \cdots$
- A countable set of variables Var
- An environment $\Gamma : Var \rightarrow \mathscr{R}_+$.
- $\Gamma[x := e]$ assigns x to a value e (e$\in\mathscr{R}_+$) in the environment Γ.

The following satisfiability relations $\models$ defined inductively show the relationship that $\langle \rho, \Gamma \rangle$ satisfies the fWLTL formula ϕ, $\langle \rho, \Gamma \rangle \models \phi$.

$\langle \rho^j, \Gamma \rangle \models c$	iff $c \in \mathscr{N}$
$\langle \rho^j, \Gamma \rangle \models x + c$	iff $\Gamma(x) + c \in \mathscr{R}_+$
$\langle \rho^j, \Gamma \rangle \models p$	iff $p \in Lab(l_j)$
$\langle \rho^j, \Gamma \rangle \models \pi_1 \leq \pi_2$	iff $\Gamma(\pi_1) \leq \Gamma(\pi_2)$
$\langle \rho^j, \Gamma \rangle \models \neg\,\phi$	iff $\langle \rho^j, \Gamma \rangle \not\models \phi$
$\langle \rho^j, \Gamma \rangle \models \phi_1 \wedge \phi_2$	iff $\langle \rho^j, \Gamma \rangle \models \phi_1$ and $\langle \rho^j, \Gamma \rangle \models \phi_2$
$\langle \rho^j, \Gamma \rangle \models \phi_1\ \mathrm{U}\ \phi_2$	iff $\langle \rho^k, \Gamma \rangle \models \phi_2$ for some $k \geq j$
	and $\langle \rho^i, \Gamma \rangle \models \phi_1$ for all i $(j \leq i < k)$
$\langle \rho^j, \Gamma \rangle \models \mathcal{I}^m x\,.\,\phi^x$	iff $\langle \rho^j, \Gamma[x := (v^j \cup w^j)(m)] \rangle \models \phi^x$
$\langle \rho^j, \Gamma \rangle \models \mathcal{I}^\tau x\,.\,\phi^x$	iff $\langle \rho^j, \Gamma[x := \tau^j] \rangle \models \phi^x$

5 Model Checking

Let $L(\mathscr{A})$ be a set of timed sequences generated by a PCA $\mathscr{A}$. Given a closed fWLTL formula ϕ, the model-checking problem $\mathscr{A}, \Gamma \models \phi$ is defined to ensure $\langle \rho^0, \Gamma_0 \rangle \models \phi$ with an initial empty environment Γ_0 and for all of the timed sequences ρ generated by $\mathscr{A}$ ($\rho \in L(\mathscr{A})$).

We now study the decidability of the above model-checking problem by investigating the existing results on formalisms relating to the PCA and the fWLTL. First,

the PCA is a kind of Weighted Timed Automaton (WTA) [5] or Priced Timed Automaton (PTA) [6]. The WTA/PTA are extensions of a Timed Automaton (TA). In the case of the TA, model-checking of the Metric Temporal Logic (MTL) is undecidable, and only model-checking of a fragment of MTL with respect to the TA is decidable [18]. Furthermore, although restricted to considering a subset of the MTL, model-checking of the WTA/PTA is undecidable, but over- and under-approximation techniques are applied to this model-checking [8].

The TPTL [3] is propositional temporal logic with freeze quantifiers, the satisfiability relation of which is defined by timed words generated by the TA, and the freezer quantifier refers to a time point of the binding state (*now*). The TPTL subsumes the MTL [18] as a proper subset, and thus model-checking of the TPTL with respect to the TA is undecidable. However, this model-checkig is decidable when we limit ourselves to the discrete time represented by Natural numbers $\mathscr{N}$.

Based on these existing studies, model-checking of the fWLTL formula with respect to the PCA is undecidable in general. The proposed logic fWLTL is more expressive than the TPTL because the fWLTL can *freeze* weight variables as well as time points. We must introduce some notion of approximations in order to enable analysis. In particular, we will adapt the time-bounded search and maximum time sampling strategy proposed for Real-time Maude [16][17]. Real-time Maude uses *explicit* time model rather than *implicit* or symbolic representation of time. The latter is used in the analysis methods of the TA or the WTA/PTA.

First, the time-bounded search is an under-approximation method in that the search is limited to a finite scope. However, the properties that must be checked are the duration-bounded cost constraints, and thus the time-bounded search seems reasonably effective for finding violations within an appropriate scope. Second, for the case of dense time systems using real-valued time, the state space becomes infinite even if the scope is bounded. Using the maximum time sampling strategy, the sampled points in the timed sequence can be limited to a finite number of points. The strategy is, in a sense, an extension of discrete time sampling where sampling points are selected from $\mathscr{N}$. It chooses sampling points using the maximum possible time advance where sampling points can be $\mathscr{R}_+$. In contrast, the discrete sampling method considers only time points in a constant interval, and thus may miss significant changes between the sampling points. In summary, PCA descriptions are to be translated into Real-time Maude. The translation may follow the method briefly outlined in [14]. This method, however, assumes the PCA to be n-RTS system.

When the PCA has probabilistic weights, model-checking is conducted statistically using the technique commonly known as statistical model-checking (SMC) [21]. Let R be a random variable that is independent and identically distributed (i.i.d.) with respect to a probabilistic distribution $f(r)$. Then, the timed sequences are dependent on the value R_i and thus are represented as $\rho(R_i)$, which is an element of $L(\mathscr{A}(R_i))$. For the fWLTL property ϕ, $B(R_i)$ is defined such that $B(R_i) \equiv (\langle\, \rho(R_i), \Gamma \,\rangle \models \phi)$. Here, $B(R_i)$ follows a Bernoulli distribution $Bin(1,p)$ in which p is the probability that ϕ is satisfied with respect to $\rho(R_i)$. The probability p can be estimated statistically by increasing the number of trials, based on the law of large numbers and the central limit theorem. Therefore, the SMC problem is to

generate such a timed sequence $\phi(R_i)$ for R_i and to check $\langle\ \rho(R_i), \Gamma\ \rangle \models \phi$, which can be done using the method of Real-time Maude because the explicit time model explained above is used.

6 Related Work

The present paper discussed a model-based formal analysis method for the energy consumption problem. Specifically, it was formulated as a duration-bounded cost constraint problem. Two formal notations, the Power Consumption Automaton (PCA) and the Linear Temporal Logic with freeze quantifiers (fWLTL) were proposed so that the duration-bounded cost constraint problem was solved by logic model-checking. In the following, we compare the notations of the present study with those of existing research.

The PCA is a kind of Linear Hybrid Automata (LHA) [4]. A PCA can be emulated by an n-rate Timed System (n-RTS) with stopwatch clock variables [13]. Since the energy consumption is observable and does not affect the behavioral specifications, we encode the energy variables as weights in the Weighted Timed Automaton (WTA) [5] or the Priced Timed Automaton (PTA) [6]. The weights in PCA are, however, defined only on states, and not on transition edges. In view of reachability analysis, which is the basis for automatic verification methods, the n-RTA is undecidable while the WTA/PTA, as well as Timed Automaton (TA), are decidable. For the TA, duration-bounded reachability is decidable [2]. Furthermore, optimal or minimum-cost reachability of the WTA/PTA is also shown to be decidable [5][6]. The duration-bounded cost constraints require a property specification language to be flexible enough to express various behavioral aspects. We introduced the fWLTL for expressing properties to check. The problem is solved, in principle, by using the model-checking method. The fWLTL can encode the duration-bounded reachability problem, but does not have any notion of evaluation functions for optimization problems.

Freeze quantifiers were first proposed in the TPTL [3], the semantics of which were defined in terms of timed words generated by the TA. The freeze quantifier of the TPTL refers to a time point of the binding state (*now*). The fWLTL extends this quantifier to the range over weight variables as well as clock variables. The quantified formula $x.\phi^x$ in the TPTL is expressed as $\exists^\tau x.\phi^x$ in the fWLTL. The freeze quantifiers in the Constraint LTL (cLTL) [11] can refer to variables other than clocks, and $\downarrow_{x=m}\phi^x$ in the cLTL is expressed as $\exists^m x.\phi^x$. Finally, Duration Calculus [9] is a formal framework for considering a general class of dense time intervals. Duration Calculus (DC) is very expressive and so the model-checking method is only known for a subclass of DC [12]. We have not yet compared the fWLTL with the DC in detail.

Statistical model-checking of the Linear Priced Timed Automaton (LPTA) is discussed in [10]. The non-deterministic delay in timed transition is replaced by a probabilistic distribution, and the semantics of the LPTA is interpreted to be a stochastic

process. Furthermore, the method uses monitor-based model-checking [8], which is based on the explicit time model, rather than an implicit or symbolic representation of time, as in the cases of the TA or the WTA/PTA [4][5][6]. The probabilistic extension of the PCA is weighted probabilistically and thus the role of probability is different from that in the LPTA.

7 Conclusion

In the present paper, we investigated a model-based analysis of the energy consumption problem from the view point of logic model-checking. The problem was formally defined, and it turned out that some approximation was necessary in order to conduct automated analyses. The view presented here is a first step towards the formal model-based analysis of energy consumption behavior. In the future, we intend to investigate in detail a method for using Real-time Maude as the basic verification engine, and to demonstrate the usefulness of the proposed method in detecting energy bugs of Android application programs.

Acknowledgment

The present research was supported in part by JSPS KAKENHI Grant Number 26330095.

References

1. Android. `http://developer.android.com`.
2. R. Alur, C. Courcoubetis, and T.A. Henzinger. Computing Accumulated Delays in Real-Time System, In *Proc. CAV 1993*, pp.181-193, 1993.
3. R. Alur and T.A. Henzinger. A Really Temporal Logic, *J. Accoc. Comp. Machin.*, Vol.41, No. 1, pp.181-204, 1994.
4. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The Algorithmic Analysis of Hybrid Systems, *Theor, Comp. Sci.*, No.138, pp.3-24, 1995.
5. R. Alur, S. La Torre, and G.J. Pappas. Optimal Paths in Weighted Timed Automata, In *Proc. HSCC 2001*, pp.49-62, 2001.
6. G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romjin, and F. Vaandrager. Minimum-Cost Reachability for Priced Timed Automata, In *Proc. HSCC 2001*, pp.147-161, 2001.
7. P. Bouyer, U. Fahrenberg, K.G. Larsen, and N. Markey. Timed Automata with Observers under Energy Constraints, *Proc. HSCC 2010*, 2010.
8. P. Bulychev, A. David, K.G. Larsen, A. Legay, G. Li, D.B. Poulsen, and A. Stainer. Monitor-Based Statistical Model Checking for Weighted Metric Temporal Logic, In *LPAR-18*, pp.168-182, 2012.

9. Z. Chaochen, C.A.R. Hoare, and A.P. Ravn. A Calculus of Durations, *Information Processing Letters*, vol.40, pp.269-276, 1991.
10. A. David, K.G. Larsen, A. Legay, M. Mikucionis, D.B. Poulsen, J. van Vliet, and Z. Wang. Statistical Model Checking for Networks of Priced Timed Automata, In *Proc. 9th FORMATS*, pp.80-96, 2011.
11. S. Demri, R. Lazić, and D. Nowak. On the Freeze Quantifier in Constraint LTL : Decidability and Complexity, *Information and Computation*, 205(1), pp.2-24, 2007.
12. R. Meyer, J. Faber, J. Hoenicke, and A. Rybalchenko. Model Checking Duration Calculus: a practical approach, *Formal Aspects of Computing*, vol.20, pp.481-505, 2008.
13. S. Nakajima. Model-based Power Consumption Analysis of Smartphone Applications, In *Proc. ACES-MB 2013*, 2013.
14. S. Nakajima. Everlasting Challenges with the OBJ Language Family, In *Proc. SAS 2014*, pp.478-493, 2014.
15. S. Nakajima. Platform-Dependence in Model-based Energy Consumption Analysis of Smartphone Applications, *NII Technical Report*, 2014.
16. P.C. Ölveczky and J. Meseguer. Semantics and Pragmatics of Real-Time Maude, *Higher-Order and Symbolic Computation*, vol.20, no.1-2, pp.161-196, 2007.
17. P.C. Ölveczky and J. Meseguer. Abstraction and Completeness for Real-Time Maude, *ENTCS*, vol.176, no.4, pp.5-27, 2007.
18. J. Ouaknine and J. Worrell. Some Recent Results in Metric Temporal Logic, In *Proc. FORMATS 2008*, pp.1-13, 2008.
19. A. Pathak, Y.C. Hu, and M. Zhang. Bootstrapping Energy Debugging on Smartphones: A First Look at Energy Bugs in Mobile Devices, In *Proc. Hotnets'11*, 2011.
20. A. Pathak, Y.C. Hu, and M. Zhang. Where is the energy spent inside my app?: Fine Grained Energy Accoutning on Smartphones with Eprof, In *Proc. EuroSys'12*, 2012.
21. H.L.S. Younes, M. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. Statistical Probablistic Model Checking, *J. STTT*, 8(3), pp.216-228, 2006.
22. L. Zhang, M.S. Gordon, R.P. Dick, Z.M. Mao, P. Dinda, and L. Yang. ADEL : An Automatic Detector of Engery Leaks for Smartphone Applications, In *Proc. CODES+ISSS'12*, 2012.