On the Feasibility of Probabilistic Model Checking to Analyze Battery Sustained Power Supply Systems

Marina Dioto[1(✉)], Eduardo Rohde Eras[2], and Valdivino Alexandre de Santiago Júnior[2]

[1] Instituto Edson Mororó de Moura (ITEMM), Parque Tecnológico São José dos Campos, São José dos Campos, SP, Brazil
marina.dioto@itemm.org.br

[2] Instituto Nacional de Pesquisas Espaciais (INPE), Av. dos Astronautas, 1758, São José dos Campos, São Paulo, SP, Brazil
eduardorohdeeras@gmail.com, valdivino.santiago@inpe.br

Abstract. Probabilistic Model Checking is a Formal Verification method which is able to guarantee, according to a specified probability, the correctness of a system that presents stochastic behavior. It is an approach which has been applied to several different application domains such as biology, communication and network protocols, security, dependability, just to name a few. In this paper, we realize about the feasibility of Probabilistic Model Checking to analyze power supply systems. We modelled and evaluated two types of systems: a solar power system and batteries of artificial satellites. Our findings show that Probabilistic Model Checking provides accurate results and can be used as a complementary approach to traditional simulation tools for Model-Driven Development of complex industrial applications.

1 Introduction

Power supply systems are a set of elements combined together in order to feed an electrical scheme. They are essential because they guarantee the full operation of the hardware during the time that the system was projected to perform. They must be developed with quality due to losses that can happen on their lack or malfunctioning. On this second scenario, they can also represent a risk to the people and the environment around, since they are based on power electronics. Although power supply systems comprise a wide range of source types, on this paper we will focus on battery sustained organizations. These are important to structures that do not have access to an energy generation source all the time, like photovoltaic panels. On these, the battery represents a power accumulator, bringing flexibility to the energy usage distributed along time. However, batteries are intrinsically unstable and reactive components. So, they must operate within a specific window of safety and reliability. This window is determined by

S. Misra et al. (Eds.): ICCSA 2019, LNCS 11620, pp. 743–757, 2019.
https://doi.org/10.1007/978-3-030-24296-1_59

a combination of parameters that, if overreached, could lead the battery to enter on thermal runaway. This reinforces the need for a robustness system.

To achieve this robustness, a strategy is to use Model Driven Development (MDD), which is a methodology that focuses on graphical representations and pre-built application components to enable visually construction of complex applications. This approach is meant (i) to increase productivity and design speed by maximizing compatibility between systems through reuse of standardized models; (ii) to simplify the design process due to models of recurring design patterns in the application domain; (iii) to reduce cost as a result of the replacement of real objects to computational models; (iv) to improve communication between individuals and teams due to patterned and visual language; and (v) to increase reliability as it enables several methods of validation and verification.

The main techniques used in MDD are Unified Model Language (UML), simulation, model transformation, code automatic generation and formal methods. Formal verification is a formal method approach defined as a mathematical analysis to prove or not the correctness of a system in relation to a certain specification or property. The methods for formal verification are basically divided in theorem proving and model checking. The latter is an automated technique that, given a finite state model of a system and a formal property, systematically verifies whether this property is satisfied by a certain state in the model [1].

A branch of this technique is the probabilistic model checking, which considers probabilistic temporal logic and models. It is important to complex systems, since these are subject to various phenomena of a stochastic nature, making it difficult to guarantee its absolute correctness.

In this context, the objective of this work is to study the feasibility of probabilistic model checking to analyse battery sustained power supply systems. We believe that it is relevant to perceive the benefits of a mathematical MDD-based approach to improve the quality of such systems which may be components of applications such as satellites, aircrafts, and mobile phones.

This paper is structured as follows. Section 2 presents an overview of Probabilistic Model Checking. Section 3 shows the case studies we considered in this research. In Sect. 4, we provide details on how we modelled both systems under the stochastic/probabilistic perspective. The evaluation of results is provided in Sects. 5, and 6 contains the information of related work and discussion. In Sect. 7, conclusions and future directions are described.

2 Probabilistic Model Checking

A given stochastic process $\{X_0, X_1, ..., X_n + 1, ...\}$ at the consecutive points of observation $0, 1, ..., n + 1$ constitutes a Discrete-Time Markov Chain (DTMC) if the following relation on the conditional probability mass function (pmf), that is the Markov property, holds for all $n \in \mathbb{N}_0$ and all $s_i \in S$: [2]

$$P(X_{n+1} = s_{n+1} | X_n = s_n, X_{n-1} = s_{n-1}, ..., X_0 = s_0) = P(X_{n+1} = s_{n+1} | X_n = s_n) \tag{1}$$

On DTMC it is used Probabilistic Computation Tree Logic (PCTL), a branching-time temporal logic, based on the Computational Tree Logic (CTL).

Continuous-Time Markov Chains (CTMCs), on the other hand, are distinct from DTMCs as state transitions may occur at arbitrary instants of time and not merely at fixed, discrete time points. A given stochastic process $\{X_t : t \in T\}$ constitutes a CTMC if for arbitrary $t_i \in \mathbb{R}_0^+$, with $0 = t_0 < t_1 < ... < t_n < t_n + 1, \forall n \in \mathbb{N}$, and $\forall s_i \in S = \mathbb{N}_0$ for the conditional probability density function (pdf), the following relation holds: [2]

$$P(X_{t_{n+1}} = s_{n+1} | X_{t_n} = s_n, X_{t_{n-1}} = s_{n-1}, .., X_{t_0 = s_0}) = P(X_{t_{n+1}} = s_{n+1} | X_{t_n} = s_n) \tag{2}$$

On CTMC it is used Continuous Stochastic Logic (CSL), which also extends CTL.

The probabilistic model checker used in this paper is PRISM [2], which comports both PCTL and CSL.

3 Power Supply Systems

3.1 Solar Power System

The probabilistic model of a solar photovoltaic power system described in [3] is based on Markov Chain Theory, modeling a stochastic insolation and load demands. The main goal of the original paper is to present a new approach to simulate a standalone photovoltaic system considering the probabilistic characteristics not explored by other computer simulations at the time.

Figure 1 is based on an original illustration from the paper, where we can see the basic structure of the photovoltaic system. The energy captured by the solar array is sent to the electrical demand or to the batteries, if the load demand is less then the array output. When the array output is not enough to supply the load demand, the batteries are used to do it. This flow is guaranteed by the controllers.

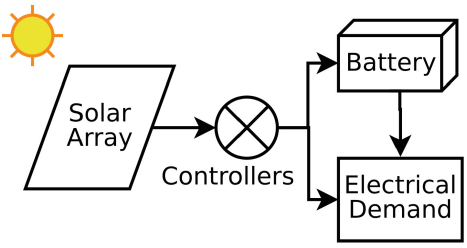

Fig. 1. Standalone photovoltaic power system

According to [3], "the insolation and the load demand are both random and accordingly the battery will behave in some stochastic fashion". The paper also

suggest to use discrete time to evaluate the system, which leads us to use a Discrete Time Markov Chain (DTMC) to create the model. So, considering a stochastic process $[X_t : t \geq 0]$, at each moment t in time, we have a state space W composed by the battery charging state S, the insulation level I and the load demand L:

$$W = \{S, I, L\} \tag{3}$$

where:

$$s = S_1, ..., S_m$$
$$i = I_1, ..., I_n$$
$$l = L_1, ..., L_y$$

so:

$$W = [w = (S_i, I_j, L_k) : i = 1, ..., m; j = 1, ..., n; k = 1, ..., y]$$

The insolation I value is generated in a stochastic way. The next insolation level depends only upon the current state, which creates a Markov behavior. The load demand is also unpredictable, but the paper suggests an average daytime L_d and nighttime L_n load demands given by the formulas:

$$L_d = \frac{L * H_d}{24}$$
$$L_n = \frac{L * H_n}{24}$$

where L is the total load demand, H_d is the average daytime hours and H_n is the average nighttime hours. The next battery State of Charge is given by a serie of equations called the energy flow equations. They predict two cases:
Case 1: $S_{min} < S_1 < S_{max}$. If the current battery charging state is between the minimum and the maximum levels of charge, the next state S_2 is given by the systems of equations:

$$A_O \geq L_d \begin{cases} S_2 = S_1 + A_O - L_d - \frac{L_n}{b} \\ b = 1 + f * (r - 1) \\ f = \frac{L_n}{L} \end{cases} \tag{4}$$

$$A_O < L_d \begin{cases} S_2 = S_1 + \frac{A_O - L}{b} \\ b = 1 + f * (r - 1) \\ f = \frac{L - A_O}{L} \end{cases} \tag{5}$$

Where A_O is the array output, b is the battery efficiency, r is the battery round trip efficiency and f is the fraction of the load met by the battery.
Case 2: $S_1 = S_{min}$. If the current state is the minimum charging state of the battery, the next state when the array output is greater than or equal to the load demand is given by the exactly same Equation System 4 shown in case 1,

but if we have a greater load demand then the array output, we don't leave the minimum charging state of the battery:

$$S_2 = S_1 = S_{min}$$

In both cases 1 and 2, if the next charging state predicted by the equation overflows the limits established by the constants s_{min} and s_{max}, there is a set of cutoff equations to follow:

$$S_2 < s_{min} \Rightarrow S_2 = S_{min}$$
$$S_2 > s_{max} \Rightarrow S_2 = S_{max}$$

3.2 Batteries of Artificial Satellites

A block diagram model for a battery charging and discharging system in artificial satellites is developed in [4], based on macroscopic principles which can be generalized to a wide variety of topologies and technologies of power supply and batteries. The main goal of the original paper is to discuss the causes and effects of thermal avalanches in these applications.

Figure 2 is based on the paper, in which it is possible to see the basic structure of a satellite power supply system. Its operation can be divided in two periods. The first one, during sunlight exposure, is refereed as "solar". In this case the satellite is fed by a Solar Array Generator (SAG), which is also responsible to recharge the battery (BAT) through a Battery Charge and Heating Controller (BCHC), also responsible to control a heater to maintain the optimum temperature of the batteries. The other phase is during the absence of solar exposition, referenced as "eclipse". In this, the BAT is responsible to supply the satellite while connected through a Battery Discharging Regulator (BDR), which regulates the voltages.

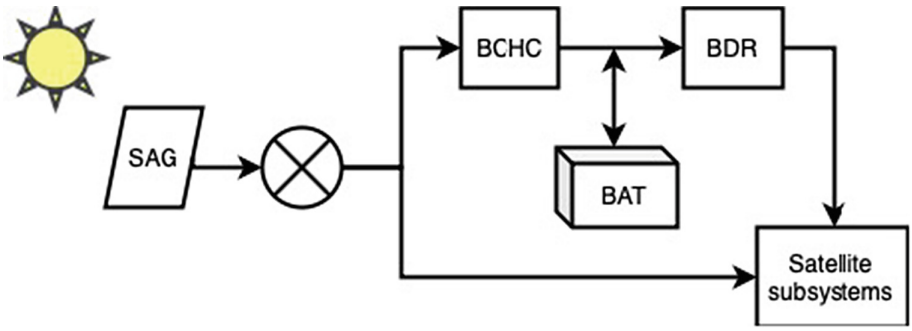

Fig. 2. Satellite power supply system block diagram

The orbital condition of the satellite, which alternates between the solar and the eclipse stages, creates a periodic excitation to the BAT. During the solar

period, the input current seen by the battery is positive, which comes from the BCHC (i_{BCHC}) performing its charge. During the eclipse, it becomes negative, causing its discharge to the BDR input (i_{BDR}).

To consider this situation, the author defines the following Boolean variable:

$$s = \begin{cases} 1, & \text{if } 0 < t < T_s \\ 0, & \text{if } T_s < t < T \end{cases} \tag{6}$$

in which t is the time, the interval $(0, T_s)$ represents the solar period and the interval (T_s, T), the eclipse period. This Boolean variable is therefore periodic with period equal to T.

Due to the endothermic and exothermic characteristics of the battery charging and discharging processes, the aforementioned excitation, in turn, causes a heat exchange defined by $q(t)$. This varies in time accordingly to the state that the model is: during charge $q(t) = q_c$ and during discharge $q(t) = q_d$, forcing it to warm up and cool down alternatively.

Regarding the thermal representation of the battery, the author defines a model in which the input is the heat $q(t)$, from the periodic excitation of the logic model, and the output is the battery temperature. The dynamics of the thermal model is therefore described as Eq. (7).

$$dx = \frac{1}{C_{bat}} f(x) + \frac{1}{C_{bat}} q(t) \tag{7}$$

in which x is the battery temperature, C_{bat} is the thermal capacitance, and $f(t)$ is the heat and cooling source function, defined as

$$f(x) = \begin{cases} P_M - k_2(x + 273)^4, & \text{if } x < 0 \\ k_1(x - 10)^2 - k_2(x + 273)^4, & \text{if } 0 \leq x \leq 10 \\ -k_2(x + 273)^4, & \text{if } x > 0 \end{cases} \tag{8}$$

in which P_M is power, k_1 is the coefficient of the heating source and k_2 is the coefficient of the radiator.

4 Modeling and Properties Formalization

4.1 Solar Power System Modeling and Properties Formalization

In order to model the suggested scenario from the original paper [3], the first approach is to recreate the state structure based on a 3-tuple $W = (S, I, L)$ seen at 3. Using the Prism syntax to model a Discrete Time Markov Chain (DTMC), one `module` was created for each element of the tuple:

```
DTMC

module battery_state
    \\Module contents
endmodule

module insolation
    \\Module contents
endmodule

module load
    \\Module contents
endmodule
```

Using this approach, there is no W variable: its existence is implied by the state of each module. But, as suggest by the paper, a single 24 h day will be simulated, so a time module was created to control the battery state through the hours:

```
module time
    t : [0..24] init 0;
    [c]t < 24 -> (t' = t + 1);
endmodule
```

The t time variable runs in the $0, ..., 24$ integer range, beginning at 0. The next state is achieved by the sum of 1 at the time variable until this reaches 24 h. Note a c synchronization variable at the front of the step guard condition: this variable also appears at the `battery_state` module to guarantee a synchronous transition of the charging state within the hours of the day.

The `insolation` module runs a binary variable to simulate the stochastic behavior of the sun condition: if the state is equal to 0 (no sun at all) it has 50% chance to turn into 1 (sunny) or stay at the same state. The same unpredictable approach is used if the current state equals 0.

```
module insolation
    i : [0..1] init 1;
    []i = 0 -> 0.5:(i' = 1) + 0.5:(i' = i);
    []i = 1 -> 0.5:(i' = 0) + 0.5:(i' = i);
endmodule
```

The paper originally considers a constant insolation in their simulations, but here we decided to test this feature to verify the versatility of the Prism model checker.

The load demand was also suggested to be constant in the paper. Here, however, we created a stochastic behavior of the load in the `load` module:

```
const int L = 3; //Maximum load
module load
    l : [0..L] init 1;
    []l > 0 & l < L -> 0.5:(l' = l + 1) + 0.5:(l' = l - 1);
    []l = 0 -> 0.5:(l' = l + 1) + 0.5:(l' = l);
    []l = L -> 0.5:(l' = l - 1) + 0.5:(l' = l);
endmodule
```

The daily load provided was 240 *amp-hrs* in the paper, but the use of a big integer as a constant would create a state explosion. So, this value was simplified by a factor of 100 and rounded. The maximum load L was empirically rounded up from $2,4$ to 3 to provide stability to the model, considering the inclusion of a stochastic insolation and load demand, both absent in the original simulation. The load l variable runs in the $0, ..., 3$ range in a probabilistic motion, in which the next state is given by the three conditions seen in the module.

The battery charging state is simulated in the `battery_state` module. Here, all the guard conditions have the synchronous variable c found in the `time` module. This is meant to create a state time-slice each hour.

```
module battery_state
    s : [s_min..s_max] init ceil((3 * s_max) / 4);

    //case a: AO >= Ld
    [c](s > s_min & s < s_max) & s < s2 & (i * AO) >= Ld-> (s' = s + 1); //a1
    [c](s > s_min & s < s_max) & s > s2 & (i * AO) >= Ld -> (s' = s - 1); //a1
    [c](s > s_min & s < s_max) & s = s2 & (i * AO) >= Ld -> (s' = s); //a1
    [c](s = s_min & (i * AO) >= Ld) -> (s' = s + 1); //a2

    //case b: AO < Ld
    [c](s > s_min & s < s_max) & (i * AO) < Ld -> (s' = s - 1); //b1
    [c](s = s_min  & Ld > (i * AO)) -> (s' = s); //b2

    [c]s = s_max -> (s' = s);
endmodule
```

In this module, the state s variable runs in a range limited by two constants: the maximum and the minimum battery load found in the paper:

```
const int  s_max = 42; //Ampere hour
const int  s_min = ceil(s_max * 0.5); //Maximum DoD = 50%
```

The maximum DoD (Depth of Discharge) was limited to 50%, as suggested by the paper. The 4200 Ah (Ampere hour) was shrunken to 42, using the 100

reduction factor to avoid state explosion. The initial state of the s variable was a value in the middle of the maximum and minimum charging state.

The energy flow equations provided in the Eqs. 4 and 5 were used to find the next battery charging state. Cases "a" and "b" were predicted considering an array output greater or smaller than the load demand. The increment or decrement in the battery level is given by comparing the current value of s with the next state s_2 value, given by the equations:

```
//Load
formula Ld = l * Hd/24; //Daytime Load
formula Ln = l * Hn/24; //Nighttime Load

//Case a
formula f = Ln/l;
formula b = 1 + f * (r - 1);
formula s2 = s + (i * AO) - Ld - Ln/b;
```

Given the fact that when the array output A_O is less than the load demand the battery charge always decreases, the "Case b" equations were not used. The value of the s variable was decreased by 1 instead. When the battery is in its minimum level, it can only rises in "Case a" or remain in "Case b". Once reached the maximum level of charge, the battery will stay there and the simulation is finished.

In order to verify the model, four properties were proposed to check the probability that the battery will get full or empty at the end of the day:

$$P_{>0.1}[t = 24 \wedge s = 42]$$

This property verifies if the state s is maximum (42) at the end of 24 h, with a probability grater than 10%. In the Prism notation, we have `P>0.1[F(t=24&s=42)]`.

$$P_{<0.1}[t = 24 \wedge s = 21]$$

Similar to the anterior, this property verifies if the state s is minimum (21) at the end of 24 h, with a probability less than 10%. In the Prism notation, we have `P<0.1[F(t=24&s=21)]`.

The next two properties state the actual probability of the given scenarios to happen:

$$P_{\bowtie}[t = 24 \wedge s = 42] \tag{9}$$

$$P_{\bowtie}[t = 24 \wedge s = 21] \tag{10}$$

The property shown in the Eq. 9 verifies the actual probability to end the day at full charge while the property in the Eq. 10 verifies the probability of running out of battery in one day. In the Prism notation, the properties are, respectively, `P=?[F(t=24&s=42)]` and `P=?[F(t=24&s=21)]`.

4.2 Batteries of Artificial Satellites Modeling and Properties Formalization

To model the proposed scheme from the original paper [4], it was used CTMC, as the central variable of interest, the temperature, is defined by the a rate dx. All the elements that have influence on this variation were identified and one `module` was created for each, using Prism syntax:

```
module time
    \\Module contents
endmodule

module orbital_logic
    \\Module contents
endmodule

module temperature
    \\Module contents
endmodule

module heat_cooling_source
    \\Module contents
endmodule
```

The t variable on the `time` module runs in the $0, ..., T$ integer range, beginning at 0. It represents the satellite orbit and behaves as a clock to the other modules. Its next state is given by the sum of 1 at the variable until it reaches T, when it returns to 0, characterizing its cyclic nature.

```
module time
    t : [0..T] init 0;
    [] t < T -> (t' = t + 1);
    [orbit_end] t >= T -> (t' = 0);
endmodule
```

The `orbital_logic` module represents the system's output response from the clock excitation. It runs the state variable s, which is a binary, and the heat variable $q(t)$. When $t < T_s$ (solar period), $s = 1$, and when $t \geq T_s$ (eclipse period), $s = 0$, as stated on Eq. (6). The heat variable $q(t)$ is also a two state element, with a constant value q_c during charge (solar period) and a constant value q_d during discharge (eclipse period).

```
module orbital_logic
    s : [0..1] init 1;
    q: [Qmin..Qmax];
    [solar] t < Ts -> 1:(s' = 1) + 1:(q' = qc);
    [eclipse] t >= Ts -> 1:(s' = 0) + 1:(q' = qd);
endmodule
```

The temperature flow distribution provided in Eq. (7) is expressed as the function for dx.

```
formula dx = ((invCbat)*(f + q));
```

This is used to estimate the battery temperature on module `temperature`, which runs the variable x. The rate of which the temperature is modified is determined by dx. As Prism do not understand negative rates, it was necessary to break the command into three different situations: $dx < 0$ (dx is negative), $dx > 0$ (dx is positive) and $dx = 0$ (dx is zero). The last one does not impact in change on x. Also, it is detected if the temperature is bellow the absolute zero ($Xmin$), evidencing an error in the code; and if the temperature is higher then the maximum accepted before the battery enters thermal runaway ($Xmax$).

```
module temperature
                x: [Xmin..Xmax];
                [dx_negative] dx < 0 & x>Xmin -> (-dx):(x'=x-1);
                [dx_positive] dx > 0 & x<Xmax -> (dx):(x'=x+1);
                [dx_zero] dx = 0 -> 1:(x'=x);
                [alert] x>=Xmax -> 1:(x'=x);
                [error] x<=Xmin -> 1:(x'=x);
        endmodule
```

The `heat_cooling_source` module basically transcripts Eq. (8). It runs a variable f that substitutes $f(t)$, which assumes different functions at three temperature sections.

```
module heat_cooling_source
    f : [fmin..fmax];
    [temperature_range1] x < 0 ->
        1:(f' = ceil(Pm - (k2*pow((x+273.0), 4))));
    [temperature_range2] x >= 0 & x <= 10 ->
        1:(f' = ceil(k1*pow((x-10.0), 2) - k2*pow((x+273), 4)));
    [temperature_range3] x > 10 ->
        1:(f' = ceil(-k2*pow((x+273.0), 4)));
endmodule
```

It was used cost and rewards to reason about quantitative measures relating to model behaviour: to count the number of cycles, checking how many times it passed through `orbit_end`; and to compute the `alerts` and `errors`.

```
rewards "cycles"
    [orbit_end] true : 1;
endrewards

rewards "alert"
    [alert] true : 1;
endrewards
```

```
rewards "error"
    [error] true : 1;
endrewards
```

With the aim of verifying the model, four properties were developed using the patterns presented on [5].

$$P_{<0.1}[\Diamond[T,T]x = X_{max}]$$

Being a Transient State Probability pattern, this property checks if the probability of having a thermal runway before $t < T$ is less than 10%. In the Prism notation, we have `P>0.1[F(T,T) x=Xmax]`.

$$P_{\geq 0.9}[\Box[0,100]x > X_{min}]$$

As a Probabilistic Invariant, this property certifies if no error will occur in the next 100 units of time, with a probability of at least 90%. In Prism, it becomes `P>=0.9[[0, 100] x > Xmin]`.

$$P_{\geq 0.95}[\Diamond[0,6015]s = 0]$$

This property was created using the Probabilistic Existence pattern. It certifies if a transition to the eclipse state should occur within 6015 seconds in 95% of the cases. In Prism, it is `P>=0.95[F[0, 6015] s = 0]`.

$$R_{\text{“}cycles\text{”}} =?[C \leq 100000]$$

This is a Cumulative Rewards pattern and it sums the number of cycles performed. In Prism, the same property is written as `R{"cycles"}=? [C<=100000]`.

5 Results Evaluation

5.1 Solar Power System Results

In this section, we present the results of our approach applied to the solar power system. The original values were simplified by the factor of 100 to avoid state explosion and were rounded to fit the discrete nature of the state model.

By verifying the four properties, we achieved the following results (Table 1):

Table 1. Results of the properties for the Solar Power System

Property	Result
$P_{>0.1}[t = 24 \wedge s = 42]$	True
$P_{<0.1}[t = 24 \wedge s = 21]$	True
$P_{\bowtie}[t = 24 \wedge s = 42]$	0.17649027
$P_{\bowtie}[t = 24 \wedge s = 21]$	0.08091588

The simulation in the original paper predicted a scenario in which the system is supplied for 98, 3% of the time. With ∼8,09% of probability to run out of battery at the end of the day, a 92% of system reliability is a very close result to the original one. The author of the paper achieved 13.32% of maximum battery in his simulations, while we got ∼17,64% of full charge at the end of the day, another close result.

5.2 Batteries of Artificial Satellites Results

Verifying the four properties stated on Sect. 4 for the satellite model, the following results were achieved (Table 2):

Table 2. Result of the properties for the Satellite Power System

Property	Result
$P_{<0.1}[\Diamond[T,T]x = X_{max}]$	True
$P_{\geq 0.9}[\Box[0,100]x > X_{min}]$	True
$P_{\geq 0.95}[\Diamond[0,6015]s = 0]$	True
$R_{\text{“}cycles\text{”}} =?[C \leq 100000]$	1.0

Making a simulation on Prism, it was obtained the two curves bellow. On the first one, it is possible to see the interaction between all variables, mainly the periodic ones. On the second one, the temperature element was isolated in order to have a closer look in its behavior (Figs. 3, 4).

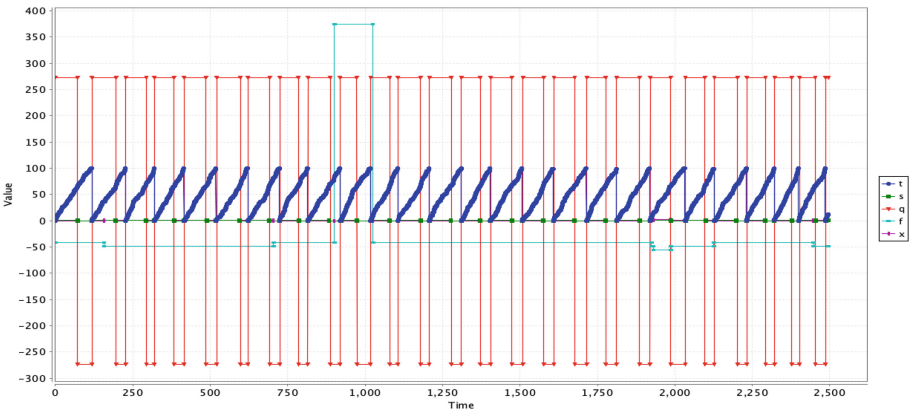

Fig. 3. Satellite Power System Simulation: variables interaction.

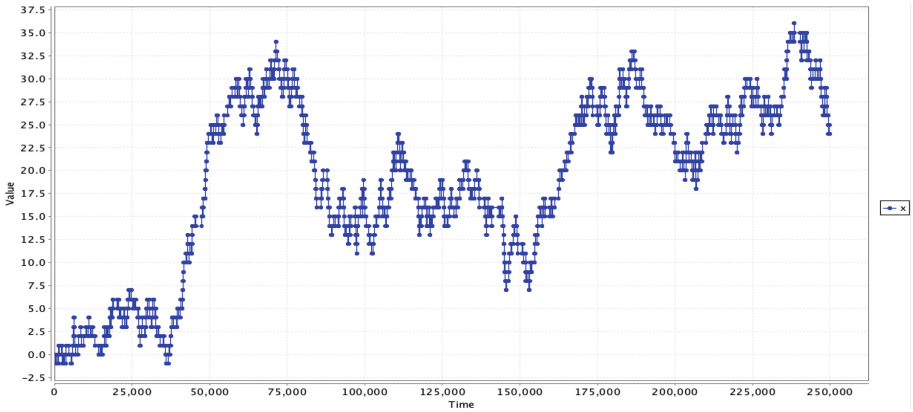

Fig. 4. Satellite Power System Simulation: temperature variation.

6 Discussion of Related Work

In this section, we focus on some relevant studies addressing the analysis of power supply systems.

In [6] the author introduces a battery power supply model based on discrete-time VDHL. It is done an event-driven simulation at a very high level of abstraction. The objective is to estimate battery life-time during design optimization. Different than the strategy used in this paper, the battery is represented only as an electrical circuit.

A mathematical simulation is done in [7] using experimental data to calculate parameters used on the model. The aim is to predict the battery state of charge of a hybrid solar-wind power supply system.

Another simulation method explored for the analysis of satellite power supply subsystem in [8] is the subspace identification. In this paper it is used n4sid together with control techniques to improve fidelity of CBERS-4 operational simulator.

A satellite power supply subsystem is also modeled in [9], but using Simulink. In this case, it was analysed the different performances for each configuration of the solar arrays.

Also in the space application, [10] uses Virtual Test Bed (VTB), an interdisciplinary computational environment, to model, simulate and virtual-prototype the battery power supply system. The author uses the results obtained to compare different chemicals.

It is possible to perceive the plurality of techniques in the literature to model and simulate battery sustained power supply systems. However, all of them have in common the deterministic approach. In other words, it is used past data or mathematical equations to predict the system's exact behavior. Probabilistic Model Checking, on the other hand, considers the stochastic characteristic of the battery. In this way, the objective is not to reproduce a past scenario as it happened once, but to infer patterns and probabilities about that system.

7 Conclusions

In this paper, it is discussed the use of Probabilistic Model Checking on battery sustained power supply systems. We considered two systems in our analysis: a solar power supply system based on DTMC and a satellite power supply system based on CTMC. The results achieved show a behavior similar to the expected, proving the feasibility of the method. Besides, it is obtained a series of probabilistic assumptions to help on the development and maintenance of projects using the studied subsystems. Since batteries are passive and unstable elements, it is difficult to predict its behavior, even through past data comparison. Thus, Probabilistic Model Checking turned out to be a more realistic estimation in this case than other methods based on reproduction of scenarios. Future directions include the investigation of other probabilistic models such as Markov Decision Processes (MDPs) and Probabilistic Timed Automata (PTA) as well as the application of probabilistic model checking to analyze other power supply systems.

Acknowledgments. The authors would like to thank ITEMM for the help in carrying out this work and CNPq for the financial support on the process number 130878/2018-9.

References

1. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge (2008)
2. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic model checking. In: Bernardo, M., Hillston, J. (eds.) SFM 2007. LNCS, vol. 4486, pp. 220–270. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72522-0_6
3. Safie, F.M.: Probabilistic modeling of solar power systems. In: Annual Reliability and Maintainability Symposium, 1989. Proceedings, pp. 425–430. IEEE (1989)
4. de Magalhães, R.O.: Esudo de avalanche térmica em um sistema de carga e descarga de baterias em satélites artificiais. Ph.D. thesis, INPE (2012)
5. Grunske, L.: Specification patterns for probabilistic quality properties. In: ACM/IEEE 30th International Conference on Software Engineering, ICSE 2008, pp. 31–40. IEEE (2008)
6. Benini, L., Castelli, G., Macii, A., Macii, E., Poncino, M., Scarsi, R.: A discrete-time battery model for high-level power estimation. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 35–41. ACM (2000)
7. Zhou, W., Yang, H., Fang, Z.: Battery behavior prediction and battery working states analysis of a hybrid solar–wind power generation system. Renew. Energy **33**(6), 1413–1423 (2008)
8. Rodrigues, I.P., Rego, L.F.M., Coimbra, T.d.S., Gruppelli, G.P., Ambrósio, A.M.: Identification and control techniques applied to an operational satellite simulator. Congresso Brasileiro de Automática, 22. (CBA) (2018)
9. Farid, H., El-Koosy, M., El-Shater, T., El-Koshairy, A., Mahmoud, A.: Simulation of a LEO satellite electrical power supply subsystem in-orbit operation. In: Proceedings of the 23rd European Photovoltaic Solar Energy Conference and Exhibition, Valencia, Spain (2008)
10. Jiang, Z., Dougal, R.A., Liu, S.: Application of VTB in design and testing of satellite electrical power systems. J. Power Sources **122**(1), 95–108 (2003)