Automated verification of electrical systems: literature review and case studies of stand-alone solar photovoltaics[⋆]

Alessandro Trindade[1][0000−0001−8262−2919] and Lucas Cordeiro[2][0000−0002−6235−4272]

[1] Federal University of Amazonas, Av. Gen. Rodrigo Octavio, 6200, Coroado I,
69077-000 Manaus AM Brazil alessandrotrindade@ufam.edu.br

[2] The University of Manchester, Kilburn Building, Oxford Road, Manchester M13
UK lucas.cordeiro@manchester.ac.uk

Abstract. Around 1 billion of people do not have electrical energy nowadays on Earth. Energy poverty alleviation has become an important political issue. Several initiatives and policies have been proposed to deal with poor access to sources of energy in many developing countries. Particular attention is given to use renewables and stand-alone solar photovoltaic systems. Tools to evaluate electrification projects are available, but they are based on simulations that do not cover all aspects of the design space. The use of automated verification in electrical systems is a fresh subject with few issued literature. This paper do a state-of-art review about the use of automated verification on electrical systems and perform a comparative of tolls applied to five stand-alone solar photovoltaic systems. Information about the use of a commercial simulator and monitoring data from the deployed systems are used to validate the research. Data from experimental results shows that only some flaws can be detected by automated verification and that ESBMC verification engine obtain the best results, regardless using a high-end or low-end setup.

Keywords: Automated verification · Model Checking · Electrical systems · Solar photovoltaic systems.

1 Introduction

Lack of access to clean and affordable energy is considered a core dimension of poverty [19]. Progress has been made worldwide. The number of people without electricity access fell below 1 billion for the first time in 2017 [20]. To provide universal electricity for all, decentralized systems, led by solar PV in off-grid and mini-grid systems, will be the least-cost solution for three-quarters of the additional connections needed, but that grid extension will still have a role to play, especially in urban areas [20].

[⋆] Supported by Newton Fund (ref. 261881580) and FAPEAM (Amazonas State Foundation for Research Support, calls 009/2017 and PROTI Pesquisa 2018).

In order to simulate or evaluate a PV system there are a myriad of specialized tools available in the market, such as RETScreen, HOMER, PVWatts, SAM, and Hybrid2 [28, 33, 12, 6, 23]; and even general purpose simulation tools such as PSpice, Saber, or MATLAB/Simulink package [16, 3]. However, those tools are based on running experiments in simulation models. Simulation is employed before the system design is concluded but it has the drawback of an incomplete coverage since the verification of all possible combinations and potential failures of a system is unfeasible [10].

Formal methods based on model checking offer a great potential to obtain a more effective and faster verification in the design process [10]. Any kind of system can be specified as computer programs using mathematical logic, which constitutes the intended (correct) behavior; then, one can try to give a formal proof or otherwise establish that the program meets its specification.

In this study, an updated review about the use of verification tools in electrical systems is performed, and is presented five case studies, where different model checking tools are compared in terms of performance and reliability when validating stand-alone solar PV systems. In order to perform the comparation and case studies, a mathematical model of each component of a stand-alone PV system, as panel solar, charge controller, batteries, inverter, and electrical load is used. The project requirements, as battery autonomy and power demand, besides weather conditions, as solar irradiance and ambient temperature, are translated as part of the computer program and automatically verified during the formal process. The model checking tools report in which conditions a system does not meet the user requirements. A key benefit to this approach is that it helps in the detection of flaws in the design phase of system development, thereby considerably improving system reliability [2]. The case studies are not theoretical and were deployed in 2018, with feedback information collected from monitoring systems and through interview with the dwellers.

2 State-of-art in automated verification of Electrical systems

This section summarizes, in a chronological way, the research done by previous studies related to the subject of this research.

The conversion of traditional power grid into a smart grid, a fundamental example of a cyber-physical system, raises a number of issues that require novel methods and applications. In 2012, it was considered a specific Chinese Smart Grid implementation as a case study and address the verification problem for performance and energy consumption [34]. It was employed the stochastic model checking approach and present a modelling and analysis study using PRISM model checker. The focus was how Cyber-physical systems integrate information and communication technology functions to the physical elements of a system for monitoring and controlling purposes. In this context, it was applied automated verification of certain quantitative properties of the system.

In 2015, it was showed that related to power system protection schemes, the approach of using Monte-Carlo simulation and manual interpretation of results had the limitation of the incomplete coverage of all possible operating conditions due to the time complexity of simulation [32]. It was proposed an automated simulation-based verification technique to verify the correctness of protection settings efficiently using hybrid automata-temporal-logic framework. The initial focus were relay operations and test-case generation to ensure early detection of design errors.

Other work issued in 2015 was a framework named Modana to achieve an integrated process from modeling with SysML/MARTE to analysis with Statistical Model Checking (SMC) for Cyber-Physical Systems (CPSs) in terms of Non Functional Properties (NFP) such as time, energy, etc [8]. To demonstrate the capability of Modana framework, the authors modelled energy-aware buildings as a case study, and discussed the analysis on energy consumption in different scenarios. The focus is in smart buildings and HVAC (Heating, ventilation, and air conditioning) systems.

Back in 2017, it was suggested the use of formal methods to verify and certifiably control the behavior of computational devices interacting over a shared and smart infrastructure [1]. It was shown that formal techniques can provide compelling solutions not only when safety-critical goals are the target, but also to tackle verification and synthesis problems on populations of such devices. This work argued that alternative solutions based on classical analytical techniques or on approximate computations are prone to errors with global repercussions. Two different areas were targeted: thermal loads and electricity-generating devices interacting over a smart energy network or over a local power grid. The researcher discussed the aggregation of large populations of thermostatically-controlled loads and of photovoltaic panels, and the corresponding problems of energy management in smart buildings, of demand-response on smart grids, and respectively of frequency stabilization and grid robustness. The focus is in controlling the behavior of the components, verifying the smart grid, dealing with a 'system of systems' and with 'internet of things'. The author uses approximate model checking of stochastic and hybrid models.

In 2018, is was proposed a methodology for the verification of the Cyber Physical Energy Systems (CPES), with the methodology applied to PV panels and its distributed power point tracking [13]. The approach relied on representing the unpredictable behavior of the environment in order to cover all feasible possible scenarios. Processed by JModelica, the simulation results covered the system's complete dynamic behavior. It was evident at this work the time consuming issue, with almost three days of computer effort to verify the design space of one hour from PV panels behavior.

Another work issued in 2018 was the approach to modeling smart grid components using a formal specification. It was used a state-based formal specification language namely Z and it was demonstrated the application of Z on four smart grid components [2]. The presented formal specification can be considered as first steps towards modeling of smart grid using a Software Engineering formalism. The starting point of this work it was that a smart home can be considered as an integrated system consists of various objects and system, that communicates and interact with each other. The approach is base in Petri nets and from the premise that modeling of the smart home leads to clear understanding the overall behavior of the smart grid.

3 Automated Verification Using Model Checking

Validation is the process of determining whether a design meets the user requirements, whereas verification is the process of determining whether a design meets a set of requirements, specifications, and regulations [10]. If the requirements, specifications, and regulations are given in a formal language, then it may be possible to automate the verification process, thus resulting in a process known as *formal verification.* Verification may form part of a validation process. While simulation and testing explore some of the possible behaviors and scenarios of the system, leaving open the question of whether the unexplored trajectories may contain a flaw, formal verification conducts an exhaustive exploration of all possible behaviors. Thus, when a design is pronounced correct by a formal verification method, it implies that all behaviors have been explored, and the questions of adequate coverage or a missed behavior becomes irrelevant [9].

Formal verification is a systematic approach that applies mathematical reasoning to obtain guarantees about the correctness of a system; one successful method in this domain is model checking [9].

The process of model checking can be split into three main components: modeling, specification, and verification method. In modeling, a model (normally mathematical) of the system is created; in specification, normally a list of properties to be satisfied by the system is established, i.e., the requirements, normally expressed in a temporal logic form as Computational Tree Logic (CTL) or Linear Temporal Logic (LTL). However, there is a main disadvantage of model checking: the state explosion problem. In order to tackle this problem, many different techniques were developed in the last decades. One of the first major advances was symbolic model checking with binary decision diagrams (BDDs). In this approach, a set of states is represented by a BDD instead of by listing each state individually, which is often exponentially smaller in practice. Another promising approach to overcome state explosion problem is Bounded Model Checking (BMC) [5]. BMC is a method that checks a model up to a given path in the path length. BMC algorithms traverse a finite state machine for a fixed number of steps, k, and checks whether a property violation occurs with this bound. It uses Boolean Satisfiability (SAT) or Satisfiability Module Theories (SMT) solvers to check the generated formula from BMC.

SAT problem is a problem of determining whether there are certain conditions or interpretations that satisfy a given Boolean expression [10]. SMT decides the satisfiability of a fragment of first-order formulae using a combination of different background theories and thus generalizes SAT by supporting uninterpreted functions, linear and non-linear arithmetic, bit-vectors, tuples, arrays, and other decidable first-order theories [10]. The SAT or SMT solvers search the model for conditions (value of variables) that make the formula satisfiable. If a SAT or SMT solver finds a substitution for the formula/function then the substitute induces a counterexample and is said to be *satisfiable*, i.e., it is satisfiable *iff* the verified system contains errors.

3.1 CBMC

The C Bounded Model Checker (CBMC) is considering the best-known model verification tool to validate code in ANSI-C and C++ [22]. CBMC demonstrates the violation of assertions in C programs, or proves safety of the assertions under a given bound. CBMC implements a bit-precise translation of an input C program, annotated with assertions and with loops unrolled to a given depth, into a formula. If the formula is satisfiable, then an execution leading to a violated assertion exists [22].

CBMC's verification flow can be summarized in four stages: (i) Front-end that can read, perform parse and type checker of the C/C++ code; (ii) Intermediate Representation. CBMC uses intermediate representation, converting all non-linear control flow, such as if or switch statements, loops and jumps, into equivalent guarded goto statements; aiming to reduce the verification effort; (iii) Middle end. CBMC performs symbolic execution by eagerly unwinding loops up to a fixed bound, which can be specified by the user on a per-loop basis or globally, for all loops and finally; (iv) Back-end. With support to SAT and SMT solvers. Conversely, if the formula is unsatisfiable, no assertion can be violated within the given unwinding bounds.

3.2 ESBMC

ESBMC (or Efficient SMT-based Bounded Model Checker) is an open source, permissively licensed (Apache 2), cross platform bounded model checking for C and C++ programs [15], which supports the verification of LTL properties with bounded traces [25]. ESBMC comes as an alternative to overcome limitations of the system modeling, especially considering that the system complexity is increasing and SMT has richer theories than SAT to represent models. ESBMC's verification flow can be summarized in three stages: (i) a front-end that can read and compile C/C++ code, where the formal specification of the system to be verified is first handled; (ii) preprocessing steps to deal with the representation of the code, control flow and unwinding of loops, and the model simplification, thereby aiming to reduce the verification effort; and finally (iii) the SMT solving stage, where all the constraints and properties of the system to be verified are encoded into SMT and checked for satisfiability. If the SMT formula is shown to be satisfiable (SAT), a counterexample is presented; otherwise, the formula is unsatisfiable (UNSAT), i.e., there are no errors up to the given unwinding bound.

ESBMC exploits the standardized input language of SMT solvers (SMT-LIB[3] logic format) to make use of a resource called *assertion stack*. An assertion, in SMT solvers, is a constraint over the variables in a formula that must hold if the formula is satisfiable [24]. New assertions can be added to or old assertions removed from this stack, depending on the evaluated value of variables. This enables ESBMC, and the respective solver, to learn from previous checks, optimizing the search procedure and potentially eliminating a large amount of formula state space to be searched, because it solves and disregards data during the process, incrementally. This technique is called 'incremental SMT' [31] and allows us to reduce the memory overhead, mainly when the verified system is complex and the computing platform does not have large amount of memory to deal with all the design space state.

3.3 Ultimate Automizer

Ultimate Automizer is an automatic software verification tool for C programs that is able to check safety and liveness properties. The tool implements an automata-based instance of the CEGAR scheme. CEGAR or Counterexample-Guides Abstraction Refinement is a technique that iteratively refines an abstract model using counterexamples and use three concepts: precision, feasibility check, and refinement [4].

In each iteration, Automizer pick a trace (sequence of statements) that leads from the initial location to the error location and check whether the trace is feasible or infeasible. If the trace is feasible, an error is reported to the user; otherwise it is computed a sequence of predicates along the trace as a proof of the traces infeasibility. The predicates are a sequence of interpolants since each predicate interpolates between the set of reachable states and the set of states from which we cannot reach the error. In the refinement step of the CEGAR loop, Automizer tries to find all traces whose infeasibility can be shown with the given predicates and subtract these traces from the set of (potentially spurious) error traces that have not yet been analyzed. It is used automata to represent sets of traces. The difference to a classical CEGAR-based predicate abstraction is that Automizer have to do any logical reasoning (e.g., SMT solver calls) that involves predicates of different CEGAR iterations [18].

4 Automated verification of Stand-alone Solar PV Systems

4.1 Stand-alone Solar PV Systems

The mathematical modeling of the PV system is based on modular blocks, as illustrated in Fig.1. The PV generator, which can be a panel or an array, is a semiconductor device that can convert solar energy into DC electricity. In Fig.1, there are two variables that depend on the site where the system is deployed

[3] http://smtlib.cs.uiowa.edu/

and the weather (i.e., solar irradiance G and temperature T). For night hours or rainy days, we need to hold batteries, where power can be stored and used. The use of battery as a storage form implies the presence of a charge controller [17]. The PV arrays produce DC and therefore when the PV system contains an AC load, a DC/AC conversion is required. That converter is called of inverter; and the AC load dictates the behavior of AC electrical load from the house that will be fed by the system. A wide variety of models exists for estimation of power output of PV modules (and $I - V$ or $P - V$ curves). However, the present work will rely on the simplified model of 1-diode. It was demonstrated that it has a small error rate, between 0.03% and 4.68% from selected PV panels tested [30]. The model uses formulas from different references [17], [30], and [29].

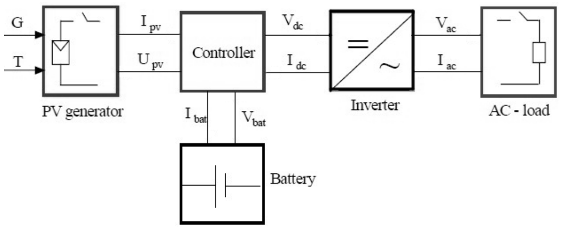

Fig. 1. Block diagram for a typical stand-alone PV system [17].

To model the batteries, it was used the most common based on lead-acid [11]. Related to the charge controller, it was used the 4-steps controller processes with MPPT (maximum power point tracking) to decide if the PV array must be disconnected from the system, or if the PV can be reconnected, or if the load is disconnected and when the load is reconnected [17]. The inverter model is based on efficiency, i.e., the relation between the AC output power and the DC input power of the component [17]. The AC load, from every case study, is defined by a 24 hour curve who reflects the estimated use of all electrical loads from the house. Load information was collected during a presential survey performed in each house.

4.2 Automated Verification Methodology

In order to evaluate the automated tools when verifying stand-alone solar photovoltaic systems, we created a methodology where each tool will process the ANSI-C code with constraints and properties from the PV system that are provided by the user, and the tool will automatically verify if the PV system requirements are met. If it returns a failure (i.e., SAT), then the tool provides a counterexample, i.e., a sequence of states that leads to the property violation. However, if the verification succeeds (i.e., UNSAT), there is no failure up to the bound k; therefore, the PV system will present its intended behavior up to the bound k.

The methodology can be described as a three stage process. In Step 1, the PV input data (e.g., load power demand and load energy consumption) and the formulae to check the sizing project, the mathematical model, the limits of the weather non-deterministic variables, are all written as an ANSI-C code [21]. In Step 2, the sizing check of the PV system takes place to make sure that the components were selected according to critical month method [27]. In Step 3, weather variables (e.g., solar irradiance and ambient temperature) will be systematically explored by our verification engine based on maximum and minimum values from the site, where the PV system will be deployed. In addition, depending on one of the desired properties of the system such as battery autonomy, energy availability, or even system power supply, our verification engine is able to indicate a failure if those properties are not met; in this particular case, it provides a diagnostic counterexample that shows in which conditions the property violation occurred.

Algorithm 1 describes the pseudo-code used to perform the automated verification. The batteries are assumed to have initial SOC of 80% (Line 1). Information related to average temperature (T) and solar irradiance (G), maximum and minimum annual, are given to the algorithm in Lines 7 to 10 using non-deterministic variables from ESBMC to explore all possible states and the *assume* macro to constrain the non-deterministic values using a given range. In order to reduce the computational effort of the algorithm, every 24h-day was considered as a time-step of 1 hour, and it was split into two parts: (a) one where it is possible to occur PV generation, during daylight, but dependent on the sun and weather conditions; and (b) one that includes all the remaining day (without PV generation). Therefore, our approach depends on specific data about the solar irradiation levels to define the average amount of hours of PV generation.

After that, the model from PV generator is used in the function call of Line 7, to produce the voltage and current considering the states of G and T. With respect to every hour considered, the conditional *if-else-endif* statements from Lines 8, 11, 13, 16 and 19, will perform the charge or discharge of batteries according to the value of different variables: if there is PV generation (which depends on G and T), the updated state of charge from batteries, the house's load and the set-up information of the PV system.

Next, representing the time of the day where PV generation is not possible anymore, starting in Line 22, the algorithm will only discharge the batteries (using the 1 hour step) until a new charging process (at the next day) starts. Specific *assert* in Line 26 will check the state of charging from batteries, and they will violate the property if their levels reach the minimum that represents a discharged battery; therefore, the PV system is unable to supply energy to the house. Nevertheless, if the verification engine does not fail, then we can conclude that the PV system does not need further corrections up to the given bounds.

shall we provide further details about the case studies?

Algorithm 1 Model checking algorithm for stand-alone PV

```
 1: $SOC \leftarrow 80\%$
    *LOOP Process*
 2: for $h = 1$ to $Hours\,of\,PV\,generation$ do
 3:    $G \leftarrow nondet\_uint(\,)$ {$G$ is non-deterministic variable}
 4:    $T \leftarrow nondet\_uint(\,)$ {$T$ is non-deterministic variable}
 5:    assume ($Gmin \leq G \leq Gmax$) {restricting $G$ values}
 6:    assume ($Tmin \leq T \leq Tmax$) {restricting $T$ values}
 7:    $Imax, Vmax \leftarrow PV generation MODEL(G, T)$
 8:    if ($battery\,is\,empty$) AND ($PV\,is\,generating$) then
 9:       charge battery in 1h
10:       PV feed the house
11:    else if ($battery\,is\,empty$) AND NOT($PV\,is\,generating$) then
12:       FAIL with assert macro
13:    else if NOT($battery\,is\,empty$) AND ($PV\,is\,generating$) then
14:       stop battery charge
15:       PV feed the house
16:    else if NOT($battery\,is\,empty$) AND NOT($PV\,is\,generating$) then
17:       discharge battery in 1h
18:       Battery feed the house
19:    end if
20:    $h \leftarrow (h + 1)$
21: end for
    *Start of battery autonomy verification:*
22: $AutonomyCount \leftarrow 1$
23: while $AutonomyCount \leq autonomy$ do
24:    $SOC \leftarrow SOC - (24 - Hours\,of\,PV\,generation)\,h\,discharge$
25:    $AutonomyCount \leftarrow AutonomyCount + 1)$
26:    assert NOT($Battery\,empty$)
       {Perform similar for-LOOP as defined in line 2-21}
27: end while
28: return ( )
```

5 Experimental Evaluation

5.1 Description of the Case Studies

We have performed five case studies to evaluate the tools, as described by Table 1.

Table 1. Case studies: stand-alone solar PV systems.

Item	House 1	House 2	House 3	House 4	House 5
PV Panels	3x 325W: in series				4x 325W: 2x series and 2x parallel
Batteries	4x220Ah: 2x series and 2x parallel autonomy: 48h				4x 120Ah batteries in series autonomy: 6h
Charge Controller	With MPPT of 150V/35A				
Inverter	700W with surge of 1,600W				1,200W with surge of 1,600W
Power demand	253W	263W	283W	501W	915W
GPS Coordinates	2^o44’50.0”S 60^o25’47.8”W				3^o4’20.208”S 60^o0’30.168”W
Details	Riverside indigenous community Rural Area of Manaus - Brazil				Urban house Manaus-Amazonas-Brazil

The annual average temperature (T) in Manaus is from 23^oC to 32^oC; and irradiance (G) varies from 274 W/m^2 to 852 W/m^2 when there is sunlight (Lines 5 and 6 of Algorithm 1). Another characteristic of Manaus, based on historical weather data [14], is related to the fact that only during 8 hours of the day is possible to have PV generation, from 8:00h to 16:00h.

5.2 Objectives and Setup

Our experimental evaluation aims to answer two research questions:

RQ1 **(soundness)** Does the automated verification provide correct results?
RQ2 **(performance)** How does the tools compare each other?

It was used two different setups of hardware and different verification engines, combined with different Solvers. The aim was to evaluate different environments and possible user limitation in terms of main processor and RAM memory. All the experiments were performed without a predefined timeout.

Mostly of experiments were conducted on an otherwise idle Intel octa core Xeon CPU E5-2640 with 2.4 GHz and 264 GB of RAM, running Ubuntu 18.04.1 LTS 64-bits. It was called high-end setup and the verification engines demanded until 90G bytes of RAM.

In addition, it was considered a low-end setup, using specifically the incremental option of ESBMC verification engine, where the demand of memory is low (2G Bytes of RAM). The experiments were conducted on an Intel Core i7-2600 (8-cores), with 3.4 GHz and 32 GB of RAM, running Ubuntu 18.04.1 LTS 64-bits.

Concerning the high-end configuration: (i) verification engine ESBMC, version v5.1.0 was used with the SMT solver Boolector version 2.4.1 [7][4]; (ii) verification engine CBMC 5.6 and MiniSat [22][5]; (iii) Ultimate Automizer version 0.1.24 [18][6].

Concerning the low-end configuration: ESBMC v5.1 was used with the SMT incremental mode[7] enabled with the goal of reducing memory usage; we have also used the SMT solver Z3 version 4.7.1 [26].

In order to evaluate the verification methods and its performance, we have considered five case studies and also compared its performance to the HOMER Pro tool. Experimental setup of HOMER Pro: Intel Core i5-4210 (4-cores), with 1.7 GHz and 4 GB of RAM, running Microsoft Windows 10; we have used HOMER Pro v3.12.0.

What are the command-lines used for all tools? Versions? Solvers? Solvers versions?

5.3 Results

Table 2 summarize the results concerning the use of the automated verification tools applied to the five case-studies of a solar PV system. The time expressed at the Table answer the RQ2. Worth to mention that using simulation tool (HOMER Pro) in all case studies, the 1,200W was the only one that was proved to not meet the requirement of battery autonomy; all the 700W systems had no indication of flaws during simulation. The simulation took less than five seconds to be performed in each case study.

The engines ESBMC and ESBMC Incremental accused that the 1,200W system (House 5) has sizing error related to the right number of solar panels. Related to the 700W systems (Houses 1-4) the verification engines ESBMC and ESBMC incremental indicate that there is a situation of low state of charge from batteries that will cause the cut of the load, just after the 16:00h of the day, when the weather condition was not enough to charge the batteries.

The CBMC tool, was not successful to produce results. It finished the verification of all cases, but the log file indicated that the Solver did not present SAT or UNSAT answer. We tried the MiniSAT and even the Boolector Solvers.

Related to UAutomizer, ...

[4] The command-line used to perform verification: $ esbmc filename.c --no-bounds-check --no-pointer-check --no-div-by-zero-check --unwind 300 --boolector

[5] The command-line used to perform verification: $ cbmc filename.c –unwind 300

[6] The command-line used to perform verification: $ ./Ultimate -tc config/AutomizerReach.xml -i filename.c

[7] The command-line used to perform the verification is: $ esbmc filename.c --no-bounds-check --no-pointer-check --no-div-by-zero-check --unwind 300 --smt-during-symex --smt-symex-guard --z3

Table 2. Summary of the case-studies comparative and the automated tools.

Model Checker (SAT/UNSAT: time and message)				
Case	ESBMC (Boolector)	CBMC (MiniSat)	UAutomizer	ESBMC Incremental (Z3)
House 1	7.44h (low SOC after 16:00h)	Inconclusive (during solver)	XX	409.3h (low SOC after 16:00h)
House 2	5.74h (low SOC after 16:00h)	Inconclusive (during solver)	XX	611.2h (low SOC after 16:00h)
House 3	Inconclusive (during solver)	Inconclusive (during solver)	XX	615.8h (low SOC after 16:00h)
House 4	5.52h (low SOC after 16:00h)	Inconclusive (during solver)	XX	620.8h (low SOC after 16:00h)
House 5	0.55h (Sizing: number of panels)	Inconclusive (during solver)	XX	63.3h (Sizing: number of panels)

The flaws from 700W systems were confirmed with the dwellers who own the systems by an interview process and from a local monitoring system with the Raspberry Pi controller: at least once a month is usual the system to turn off, normally in raining or clouds days, further affirming RQ1; related to the 1,200W system, we identified that, in fact, it does not meet the battery autonomy most of the time (mainly when all loads are turned on), and this was double checked with the monitoring system from the charge controller from the manufacturer who showed that the maximum power or surge power were not exceeded, thus affirming RQ1; this behavior is expected since the system was purchased as an off-the-shelf solution and not as a customized design for the electrical charges of the house

6 Conclusions and Future Work

We have described and evaluated different automated verification methods to check whether a given PV system meets its specification using software model checking techniques. We have considered five case studies from real photovoltaic systems deployed in five different sites, ranging from 700 W to $1,200$ W. ESBMC had the better results, in both setups (high-end and low-end configurations). Although the verification method takes longer than simulation methods, it is able to find specific conditions that lead to failures in a PV system previously validated by a commercial simulation tool. In particular, the proposed method was successful in finding sizing errors and indicating in which conditions a PV system can fail. Future work will focus in other verification engines and different solvers, besides the improvement of the algorithm in order to obtain better performance.

References

1. Abate, A.: Verification of networks of smart energy systems over the cloud. In: Bogomolov, S., Martel, M., Prabhakar, P. (eds.) Num. Soft. Verif. vol. LNCS 10152, pp. 1–14 (2017)
2. Akram, W., Niazi, M.A.: A formal specification framework for smart grid components. Complex Adaptive Systems Modeling **6**(1), 5 (Sep 2018)
3. Benatiallah, A., Benatiallah, D., Ghaitaoui, T., Harrouz, A., Mansouri, S.: Modelling and simulation of renewable energy systems in Algeria. Int. J. of Sc. and App. Inf. Tech **7**(1), 17–22 (2017)
4. Beyer, D., Löwe, S.: 16th international conference on Fund. Appr. to Soft. Eng., chap. Explicit-State software model checking based on CEGAR and interpolation. Springer-Verlag (2013)
5. Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: Symbolic Model Checking without BDDs. In: TACAS. LNCS, vol. 1579, pp. 193–207 (1999)
6. Blair, N., Dobos, A., Freeman, J., Neises, T., Wagner, M.: System Advisor Model, SAM 2014.1.14: General Description. Tech. rep., National Renewable Energy Laboratory, Colorado (2014)
7. Brummayer, R., Biere, A.: Boolector: An efficient SMT solver for bit-vectors and arrays. In: Tools and Alg. for the Const. and An. of Sys. (TACAS). vol. LNCS 5505, pp. 174–177 (2009)
8. Cheng, B., Wang, X., Liu, J., Du, D.: Modana: An integrated framework for modeling and analysis of energy-aware CPSs. In: IEEE 39th Annual Computer Software and Applications Conference (2015)
9. Clarke, E., Klieber, W., Miloš Nováček, Zuliani, P.: Model Checking and the State Explosion Problem, pp. 1–30. Springer, Berlin (2012)
10. Clarke, E.M., Henzinger, T.A., Veith, H.: Introduction to model checking. In: Handbook of Model Checking., pp. 1–26 (2018)
11. Copetti, J., Lorenzo, E., Chenlo, F.: A general battery model for PV system simulation. Prog. in Photovoltaics: Res. and App. **1**(4), 283–292 (1993). https://doi.org/10.1002/pip.4670010405
12. Dobos, A.: PVWatts Version 5 Manual. Tech. rep., National Renewable Energy Laboratory, Colorado (2014)
13. Driouich, Y., Parente, M., Tronci, E.: A methodology for a complete simulation of cyber-physical energy systems. In: IEEE Work. on Envir., Energ., and Struc. Monit. Syst. (EESMS) (2018)
14. EnergyPlus: Weather data by location, https://goo.gl/A82Jqh, acessed: 09.07.2018
15. Gadelha, M., Monteiro, F., Morse, J., Cordeiro, L., Fischer, B., Nicole, D.: ESBMC 5.0: An industrial-strength C model checker. In: 33^{rd} ACM/IEEE Int. Conf. on Aut. Soft. Engin. (ASE'18). pp. 888–891. ACM, New York, NY, USA (2018). https://doi.org/10.1145/3238147.3240481
16. Gow, J., Manning, C.: Development of a photovoltaic array model for use in power-electronics simulation studies. In: Proceedings of the 14th IEE Electric Power Applications Conference. vol. 146, pp. 193–200 (1999)
17. Hansen, A., Sørensen, P., Hansen, L., Bindner, H.: Models for a stand-alone PV system. No. 1219, Forskningscenter Risoe (2001)
18. Heizmann, M., Chen, Y.F., Dietsch, D., Greitschus, M., Hoenicke, J., Li, Y., Nutz, A., Musa, B., Schilling, C., Schindler, T., Podelski, A.: TACAS 2018, vol. LNCS 10806, chap. Ultimate Automizer and the Search for Perfect Interpolants, pp. 447–451. Springer (2018)

19. Hussein, M., Leal Filho, W.: Analysis of energy as a precondition for improvement of living conditions and poverty reduction in sub-Saharan Africa. In: Scientific Research and Essays. vol. 7, pp. 2656–2666 (2012)
20. IEA International Energy Agency: World Energy Outlook 2018. IEA, Paris (2018)
21. International Organization for Standardization: ISO/IEC 9899:2018 Information Technology – Programming Languages – C, https://www.iso.org/standard/74528.html., acessed: 14.11.2018
22. Kroening, D., Tautschnig, M.: CBMC C Bounded Model Checker (competition contribution). In: Tools and Alg. for the Const. and An. of Sys. (TACAS). vol. LNCS 8413, pp. 389–391 (2014)
23. Mills, A., Al-Hallaj, S.: Simulation of hydrogen-based hybrid systems using Hybrid2. Int. J. of Hydrog. Energy **29**(10), 991–999 (2004)
24. Morse, J.: Expressive and efficient bounded model checking of concurrent software. Ph.D. thesis, University of Southampton (2015)
25. Morse, J., Cordeiro, L.C., Nicole, D.A., Fischer, B.: Model checking LTL properties over ANSI-C programs with bounded traces. Software and System Modeling **14**(1), 65–81 (2015)
26. Moura, L.D., Bjørner, N.: Z3: An Efficient SMT Solver. In: Tools and Alg. for the Const. and An. of Sys. (TACAS). vol. LNCS 4963, pp. 337–340 (2008)
27. Pinho, J., Galdino, M.: Manual de Engenharia para Sistemas Fotovoltaicos. CEPEL CRESESB, Rio de Janeiro/RJ (2014)
28. Pradhan, S., Singh, S., Choudhury, M., Dwivedy, D.: Study of cost analysis and emission analysis for grid connected PV systems using retscreen 4 simulation software. Int. J. of Eng. Res. & Tech. **4**(4), 203–207 (2015)
29. Ross, R.: Flat-plate photovoltaic array design optimization. In: San Diego, C. (ed.) 14th IEEE Photovoltaic Specialists Conference. pp. 1126–1132 (1980)
30. Saloux, E., Teyssedou, A., Sorin, M.: Explicit model of photovoltaic panels to determine voltages and currents at the maximum power point. Solar Energy **85**(5), 713–722 (2011)
31. Schrammel, P., Kroening, D., Brain, M., Martins, R., Teige, T., Bienmüller, T.: Incremental bounded model checking for embedded software. Formal Asp. Comput. **29**(5), 911–931 (2017)
32. Sengupta, A., Mukhopadhyay, S., Sinha, A.: Automated verification of power system protection schemesPart I: Modeling and specifications. In: IEEE Tran. on Power Del. vol. 30, pp. 2077–2086 (2015)
33. Swarnkar, N., Gidwani, L., Sharma, R.: An application of HOMER Pro in optimization of hybrid energy system for electrification of technical institute. In: Int. Conf. on Energ. Eff. Tech. for Sust. (ICEETS). pp. 56–61 (2016)
34. Yksel, E., Zhu, H., Nielson, H.R., Huang, H., Nielson, F.: Modelling and analysis of smart grid: A stochastic model checking case study. In: Sixth International Symposium on Theoretical Aspects of Software Engineering (2012)