Validation a Finite State Model for Transit Electrification Simulation Software

Zakerifar40054463
Daniel(Aliasghar)
Concordia
Montreal,Canada
Daniel.zakerifar@gmail.com

Abstract- ***Electrification of bus is one the most expensive project in Montreal and Government is trying to find the lease expensive solution for that. This project is going to explain the software which is used for that and then check specification of the main algorithm for that. Here we used NUSMV and UPPAl.***

Keywords— Model checking, software, NUSMV.

I. INTRODUCTION

A. Model Checking

In computer science, model checking, or property checking is a method for checking whether a finite-state model of a system meets a given specification (a.k.a. correctness). This is typically associated with hardware or software systems, where the specification contains liveness requirements (such as avoidance of livelock) as well as safety requirements (such as avoidance of states representing a system crash).

In order to solve such a problem algorithmically, both the model of the system and its specification are formulated in some precise mathematical language. To this end, the problem is formulated as a task in logic, namely to check whether a structure satisfies a given logical formula. This general concept applies to many kinds of logic as well as suitable structures. A simple model-checking problem consists of verifying whether a formula in the propositional logic is satisfied by a given structure [1-6].

In this paper we want to make a model by NUSMV and check the specifications.

NuSMV is a reimplementation and extension of SMV symbolic model checker, the first model checking tool based on Binary Decision Diagrams (BDDs). The tool has been designed as an open architecture for model checking. It is aimed at reliable verification of industrially sized designs, for use as a backend for other verification tools and as a research tool for formal verification techniques.

NuSMV has been developed as a joint project between ITC-IRST (Istituto Trentino di Cultura in Trento, Italy), Carnegie Mellon University, the University of Genoa and the University of Trento.

NuSMV 2, version 2 of NuSMV, inherits all the functionalities of NuSMV. Furthermore, it combines BDD-based model checking with SAT-based model checking. It is maintained by Fondazione Bruno Kessler, the successor organization of ITC-IRST[7-9].

B. Eletrification

Société de transport de Montréal (STM) would like to plan the electrification of the bus route 211 in their transit network[1]. To optimize the capital expenditure on building the charging facility and purchasing electric buses, they need to make informed decisions on the configurations and numbers of both chargers and buses. For chargers, they need to decide the types of chargers to be used, how many of each types, and where to install them. For buses, they need to decide the battery size of the buses and the fleet size to run the current operating schedule of Route 211.

STM has decided to select chargers offered by HELIOX (https://www.heliox.nl/products) or ABB (https://new.abb.com/ev-charging). The following table shows the models of the chargers from two manufactures that STM has short listed. Note that STM will purchase chargers from a single manufacture and they will purchase a mix of both models from that manufacture. STM will only install chargers at Terminus Lionel-Groulx station and Terminus MacDonald.

The functions required from the simulator

In order to optimize their capital investments and, at the same time, make sure the electrification plan can accommodate the existing operating schedule, STM needed a simulation software to help them estimate the feasibility and robustness of various charging facility and bus configuration plans.

Two graphic user interfaces are required 1) Control Panel and 2) Display Panel. They can be separate panels or integrated into one bigger panel. The panels can be displayed as a web page or simple console graphic inputs and outputs.

The functions of the software are described in the following steps, which represent a typical scenario of using the software:

Step#1: Control Panel allows a user to

a. Select one charger manufacture and show the two charger models of the selected manufacture

b. User enters the cost of each of the two charger models belong to the selected manufacture.

c. Select one of the two battery options for NOVA LFSe+.

d. User enters the unit cost of LFSe+ with the selected battery option.

Step#2: Based on the selections and inputs in Step #1, the simulator computes a suggested configuration plan using a scheduling and assignment algorithm to be designed. The plan includes:

a. For each of the terminus, two charger models will be installed. For each charger model at each terminus, the number of chargers to be installed.
b. Number of buses to be purchased.
c. The overall capital expenditure for purchasing the chargers and buses.
Step#3: Given a configuration plan prescribed in Step#2, Display Panel should show the charging and trip schedule for each of the buses for a typical Tuesday Route 211 schedule.
Step#4: After the schedule is displayed, User is allowed to modify the configuration plan generated in
Step#2. Specifically, User can modify 1) the number of chargers for each of the charger models
at each of the terminus; 2) the number of buses to be purchased. After modification, the overall capital expenditure and the charging and trip schedule should be recalculated and updated accordingly.

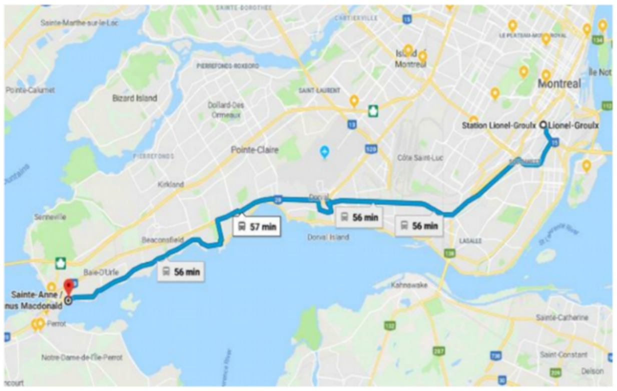


Figure 1: Electrification of One Bus Route STM 211 on Montreal Island

II. Modeling the algorithm and formulating specifications

For the one we can make a model for our algorithms.

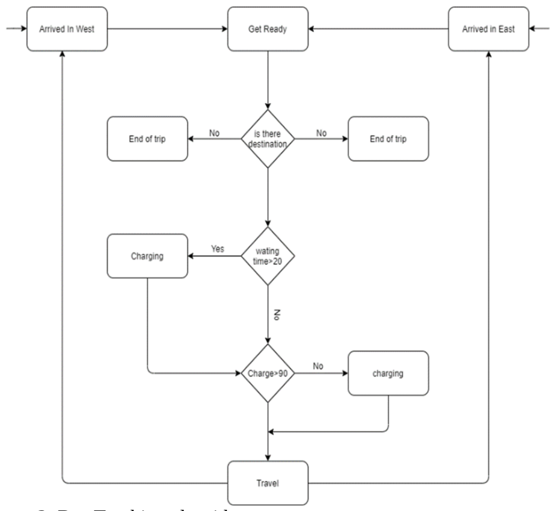


Figure 2: Bus Tracking algorithm

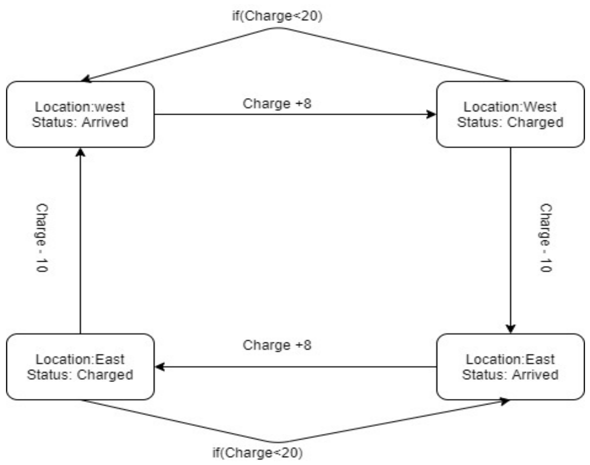


Figure 3: Simplification of the algorithm

Now can make our model in NUSMV or UPPAAL. In UPPAL we can make it graphically, and in NUSMV we can make it by writing some codes. The following are Model for algorithm in NUSMV and UPPAL.

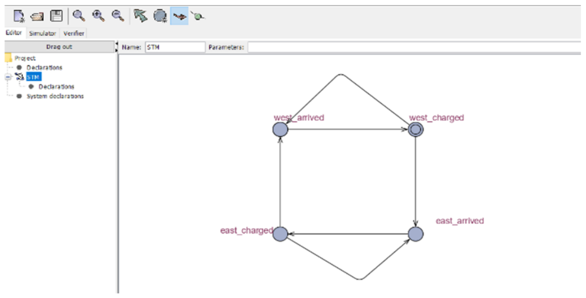


Figure 4: Algorithm model in UPPAAL

III. Requirements:

The following are our requirements:

1- The charge of the battery should be always more than zero. This is because when ever the charge of battery become zero, it can not move anymore. the formal language for that is (G(stateOfCharging>0))
2- More than that, to make sure that the bus has enough energy to reach charger station, we have to define the limit for that. For doing that we consider that limit as the amount of energy needed for reaching from one station to another station. We consider that limit equals 10. So The charge of battery should always be more than 10. The formal language for that is $\neg(F(x < 10))$
3- The energy consuming for one trip is 10 so for the condition that energy more than 10, we can have a trip with bus when it has at least 20. When is 20, after a one trip the energy become 10. So, it must go charging when it is 20. So the charge of battery often goes less than 20 but it always more than 10.
The formal language for that is (GF(x<20))
4- The charge of the battery can not go more than the capacity of battery which it is 100. The formal language for that is F(stateOfCharging>100) is false
5- We can not move anywhere until we get charged

check_ltlspec -p"!(status!=charged U status=arrived)“ Is true

6- The bus can not be in two stations at the same time. !(location==west and location==east)

7- when ever the charge of the battery become less than 20, it continue charging until it increase energy to 20.

G(stateOfCharging<20andstatus=charging $\longrightarrow$ XXstatus=charging)

8- when it become equals 100, the charging stops. (G(stateOFCharging==100→b.charging.stops)

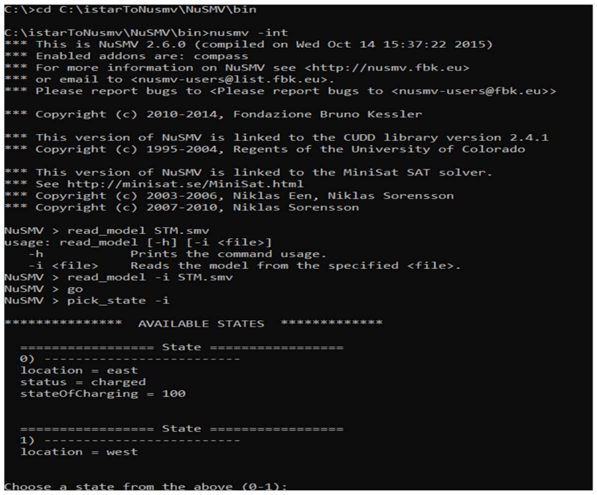
9- The bus starts from one of stations full charge and it goes to the next station. When it reaches there it become charged equals 8.

10- The bus can not arrived any where if does not move.

12 when the bus arrived there is no possibility that stops for ever

IV. Checking the specifications

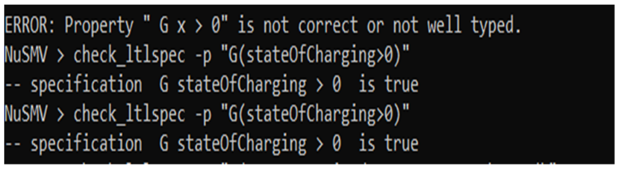
At first, we make a model for our system:



```
C:\>cd C:\istarToNusmv\NuSMV\bin

C:\istarToNusmv\NuSMV\bin>nusmv -int
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:37:22 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

NuSMV > read_model STM.smv
usage: read_model [-h] [-i <file>]
   -h           Prints the command usage.
   -i <file>    Reads the model from the specified <file>.
NuSMV > read_model -i STM.smv
NuSMV > go
NuSMV > pick_state -i

***************  AVAILABLE STATES  *************

  ================= State =================
  0) -------------------------
  location = east
  status = charged
  stateOfCharging = 100


  ================= State =================
  1) -------------------------
  location = west


Choose a state from the above (0-1):
```

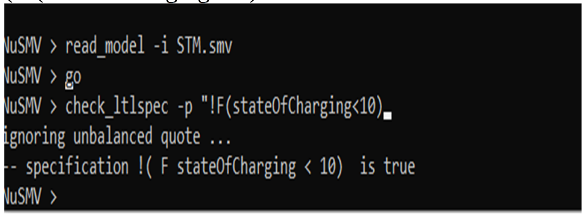
Then we try to check our specifications:

1- The charge of battery should always greater than zero G(stateOfCharge>0)



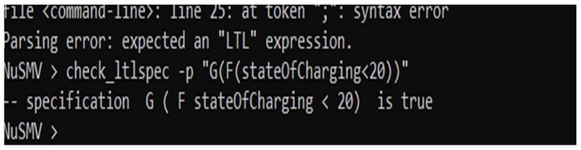
```
ERROR: Property " G x > 0" is not correct or not well typed.
NuSMV > check_ltlspec -p "G(stateOfCharging>0)"
-- specification  G stateOfCharging > 0  is true
NuSMV > check_ltlspec -p "G(stateOfCharging>0)"
-- specification  G stateOfCharging > 0  is true
```

2- The charge of battery should always be more than 10. The formal language for that is (!F(stateOfCharging<10)



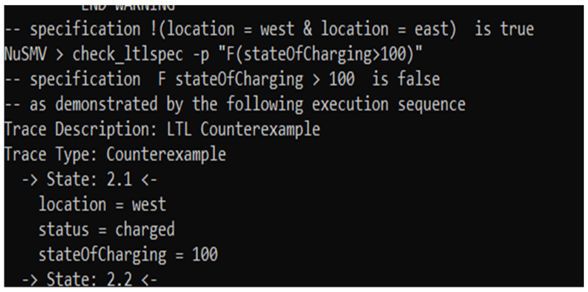
```
NuSMV > read_model -i STM.smv
NuSMV > go
NuSMV > check_ltlspec -p "!F(stateOfCharging<10)
ignoring unbalanced quote ...
-- specification !( F stateOfCharging < 10)  is true
NuSMV >
```

3- charge of battery often goes less than 20 but it always more than 10 and The formal language for that is (GF(x<20)



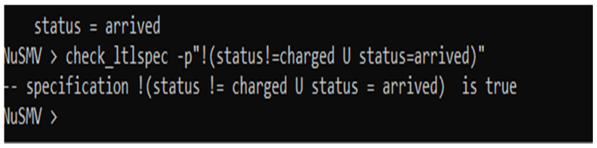
```
file <command-line>: line 25: at token ";": syntax error
Parsing error: expected an "LTL" expression.
NuSMV > check_ltlspec -p "G(F(stateOfCharging<20))"
-- specification  G ( F stateOfCharging < 20)  is true
NuSMV >
```

4- The charge of the battery can not go more than the capacity of battery which it is 100. The formal language for that is F(stateOfCharging>100) is false



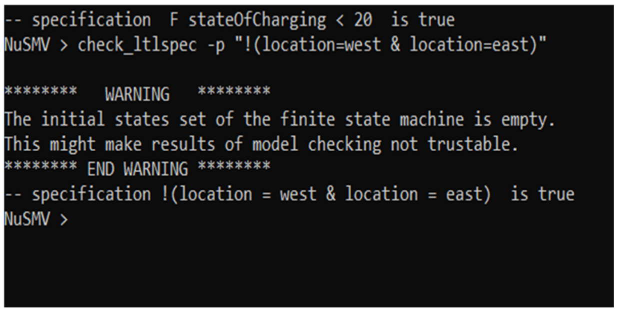
```
-- specification !(location = west & location = east)  is true
NuSMV > check_ltlspec -p "F(stateOfCharging>100)"
-- specification  F stateOfCharging > 100  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -> State: 2.1 <-
    location = west
    status = charged
    stateOfCharging = 100
  -> State: 2.2 <-
```

5- We can not move anywhere until we get charged check_ltlspec -p"!(status!=charged U status=arrived)“ Is true



```
  status = arrived
NuSMV > check_ltlspec -p"!(status!=charged U status=arrived)"
-- specification !(status != charged U status = arrived)  is true
NuSMV >
```

6- The bus cannot be in two stations at the same time. !(location==west and location==east)



```
-- specification  F stateOfCharging < 20  is true
NuSMV > check_ltlspec -p "!(location=west & location=east)"

********   WARNING   ********
The initial states set of the finite state machine is empty.
This might make results of model checking not trustable.
******** END WARNING ********
-- specification !(location = west & location = east)  is true
NuSMV >
```

For the rest of requirement we can use the order syntax Simulate -i -k 10 -v

And then we can track the states. This process could continue for ever and when ever the charge of battery becomes less than 20 after charging it goes back to arriving and the it goes charging till the charge becomes more than 20. So it can continue travelling.

V. Summary

In this paper we made a model for the main algorithm in bus electrification software by NUSMV and We also tried to use UPPAL model checker. Then, we checked our specifications by which we could validate our software.

VI. ACKNOWLEDGMENT (HEADING 5)

I have used Société de transport de Montréal (STM) Route 211 as a transit electrification example. Please note that the scenario mentioned in this requirement for proposal is for investigation purpose only. I have no knowledge nor was told by anyone about STM's future electrification plans.

VII. REFRENCE:

1- Robert T. Futrell, Donald F. Shafer, Linda I. Shafer."Quality Software Project Management" Prentice Hall, 2014.

2- Christel Baier and Joost-Pieter Katoen." Principles of Model Checking" MIT Press, 2008.

3- Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (Editors)" Handbook of Model Checking" Springer, 2018.

4- https://people.eecs.berkeley.edu/~sseshia/219c/spr07/lectures/Lec5.pdf

5- https://www.uio.no/studier/emner/matnat/ifi/INF5140/v15/slides/obdd.pdf

6- https://en.wikipedia.org/wiki/Model_checkinghttps://www.youtube.com/watch?v=GIrOek9sGyQ

7- https://en.wikipedia.org/wiki/NuSMV

8- Roberto Cavada, Alessandro Cimatti, Gavin Keighren, Emanuele Olivetti, Marco Pistore and Marco Roveri "NuSMV 2.6 Tutorial" FBK-irst - Via Sommarive 18, 38055 Povo (Trento) – Italy.

9- Huth, Ryan" Logic in Computer Science: Modelling and Reasoning about Systems" ,2nd edition,2004.