Formal Software Verification

Ekta Patel

5009,corning ct,plainfield IL,60586

1st semester of master of computer science

Lewis university,Romiovilee

ektabenppatel@lewis.edu

Abstraction:- In this article I want to be focus on the formal verification methods of embedded system such as, model checking and theorem proving which is widely accepted as promising and contrary approaches. In this paper I want to be discuss various types of model checker. In the past, model base formal verification is quietly used in context of Embedded Systems ,but now model based formal Verification is well established in the context of Software development process. I want to be describe the embedded software verification using Testing, Debugging and Verification. Also this paper introduced model checkers for the validation of information system specifications, using the temporal logics , first-order logic and failure-divergence .In this paper I also want to be test model checking on the information system.

Keyword:

Formal verification method, modal checking, theorem proving, temporal logic, embedded software verification.

I. Introduction:

Embedded system software is a system in which the computer is totally encapsulated by the device it controls.the meaning of debugging is that the process of test the correction and make sure it works. Also we can describe it as a like, it is identifying a problem, determine the source of the problem, and then either correcting the problem or determining a way to work around it. verification means that the program satisfies a given specification by a correctly complies or by a prove. Testing is process in witch we identify deference between expected and actual results. Model checking is same as the testing but, it provides broader coverage than testing, it requires less human interaction than theorem proving, and it has the ability to easily deal with both safety and liveness properties. Information system is the very important part in the organization and the business.

II. Problem:

The main problems with the root of verification center between relationship that is, software design complexity, verification complexity, engineering resources and time as well as resource constraints. And if designed is increase then complexity is also going to be increased, verification effort also grown up. If design is double in size then conventional verification effort can easily quadruple through the rule of thumb.

III. Embedded Software Verification:

Since today there is so many proof of correctness about computer programs have been available. but if we show at academic development side then there is so many thing is still not solve like impracticability of the advance in research. so there is two main area of formal software verification.(1) theorem proving ,which is totally base on mathematical related ,means that proof of correctness is get there goal through derivation of theorem. But it is not mandatory for software too, like in the software verification we can proof correctness of the software without the mathematical concepts. (2) model checking, which is very very well popular in now a days for software development process. A deference between theorem proving and model checking is that when we use theorem proving then we do not need to visit the whole program's state space in order to verify certain properties.

Now I want to be discus this two area in more detail.

IV. Theorem Proving:

theorem proving is an iterative process. In theorem proving we can derive post conditions based on preconditions and the executable program. which means, if the precondition QPRE is holds before program P is executed, and RPOST holds after the execution of P". which is syntacally defined as:

{QPRE}P{RPOST}

Where, QPRE is precondition, RPOST is postcondition,and P is for entire program or single function call.

Theorem proving worked on two way , one is mathematics and another one is logic.Now,Implementation and specification are expressed as formulas in a formal logic. in theorem proving Properties are proved interactively. In which ,it implies that a user generally think about the proof step and after that apply it and after prover can execute this step and give us to one new proof state as a output and wait for next proof step .and again if there is any new proving step is available then again prover can tack that step and execute that step and give us to new output ,and wait for next one.. this process is continually executed , that is why theorem proving is time consuming.

V. MODEL CHECKING:

Model checking is straightforward process of the states of any system. The main concept of model checking is to prove that the, whether a given model is satisfies a certain requirement or not?. it requires less human interaction than theorem proving, and it has also help us to easily deal with both safety and liveness properties. In the model checking, From a logical point of view, the system is described by a semantically model. In model checking, we needs a modeling language which include the system description, a notation for the formulation of properties and algorithms through we can travel between state space. So for that in Model checking we use the temporal logic so we need to be put very less effort when traversing system states.

In 1980s when model checking is first time is introduced at that time it is only possible for few thousand of state , but now it is possible to check system with few million states. And this is possible for increasing computing power and by using more elaborated data structures for storing system states. In the model checking first we need to Define a formal model of the system in which it verify by creating a model of the system in a language which is fits the model checker's input language. And this language is : like the PROMELA language of the SPIN model checker. After that Provide a particular system property that should be proved by the model checker. And finally Invoke the model checking tool and receive a notification like whether the given system property was fulfiled it requirement or not.

A. LET'S EXECUTE THE SEMANTICALLY MODEL:

In model checking nondeterministic state machines are used to represent the behavior of the system. In which , Every single state is labeled with Boolean variables means yes or not, which are the used for evaluations of expressions in that state. And after whatever expressions we get that is correlate with particular system properties, e.g. Boolean expressions over variables or registers. In short it is syntactically :

$$M = (S,s0,R,L)$$

Where, M is semantically model ,S is stand for finite set of states, s0is initial state, R is transition relation and L is interpretation function. In which The transition relation specifies for each state whether and which successor states are possible. And The interpretation function labels each state with the set of atomic propositions that are true in that state. Lets tack one example:

- ➢ Coffee vending machine:

Coffee vending machine is the best example of semantically model.

- In which first user need to be enter the coin for buying coffee. After entering coing user need to be select the coffee type which he or she want to purchase.
- Now, coffee is only brewed if and only if when valid selection is made.
- After that The user is able to abort the procedure at any time.
- So in that every next step is only executed after the first step is completely finished and give the result. So we, can also say that next step is also base on previous step.

B. CTL:

Now I want to be discuss on the CTL. CTL is stand on Computation Tree Logic. CTL is a combination of a linear temporal logic and a branching-time logic .In a linear temporal logic there is so many operators are describe events with the same computation path. On the other hand, a branching-time logic provides operators over a set of states which is successors of a given state.

- ➢ Let's tack same example with the use of CTL:

- E[() selection)U(brew)]

  This means that coffee is only brewed if and only if when valid selection is made. But in CTL it's tack the meaning like As long as no selection was made, no coffee is brewed in any cases. After a selection is made, coffee is brewed for sure.

- AG[(brew)]

  This means A coffee is brewed in any case. And from the CTL point of view meaning of this would be Coffee will be brewed from now on, no matter what happens.

VI. MODEL CHECKING WITH INFORMATION SYSTEM:

In information system model checking is very interesting topic. In information system ,model checking is mainly focus on the model-driven engineering and generative programming. information system MDE specification has no any dedicated model checker. This make sense like for any team specialize ,the maintaining the existing tool in model cheking. So for that we have different type of model checker is available, which is SPIN,NUSMV, fdr2,etc,..

- ➢ For example I want to be take the library example for model checking for the formal verifiacation:

The library system only contains two types: books and members, where member is the person who has joined library membership. So, Members can Join and then Leave library membership also member can tack a new book from library without reserve it , renew a book for some time and after return a book. and on the other hand books can be Bring and then Discarded. Member can also reserved the book in some condition ,and if he/she don't want that book after reserving then cancel it or borrow the book.

Now I want to be discuss on the various type of model checking:

There is four type of model checking:

(1) Explicit state model checkers, which include the cadp,Spin and fdr2.
(2) Symbolic model checkers, which include NuSMV.
(3) Bounded model checkers, which include NuSMV and Alloy.
(4) Constraint satisfaction model checkers, which include ProB.

Now I want to be discuss some of them.

A SPIN:

Spin is The classical approach for on the-fly LTL model checking. In which Specifications are written in Promela and properties in LTL. Spin generates the C source code as a on-the-fly verifier. And if one time it is compile then this verifier is checks like if the property can satisfied by the model or not?

As we mention above , spin's specification are written in promela which is inspired from c source. Promela handle concurrent processes. In Promela we can use basic types like char, bit, int and arrays of these types. And Processes can be communicate with each other by writing and reading over a channel.

Spin use the LTL for a property concept. In which it provide so many operator like always, eventually , until, strong until, week until. Spin only considers states and there is no any notion for event on a transition.

B FDR2:

constrain satisfaction problem is used fdr2 as a explicit state checker.fdr2 is used for checking refinement, deadlocks, live locks and determinacy of process expressions. It generally used for building the state-transition graph. constrain satisfaction problem does not support state variables. In fdr2 we can use the basic data types like integer, Boolean, tuples, sets and sequences. There is a Models are described usinga variant of csp which is called CSPM.which is support the classical process algebra operators like prefix, choice. In fdr2 Expressions are dynamically typed. And if we want to defined the function of these type then we need to use lambda keyword.

In fdr2 properties are checked using process refinement. There is so many different types of refinement are available to checked various kind of property like if we want to check the safety properties then we use Trace refinement, while stable failure is used to check liveness properties. And Failure-divergence refinement is used to check live locks.

VII. CONCLUSION:

As discuss each and everything , personally I believe that CTL is more sufficient and use full for handle most common properties and also useful for temporal syntax in which properties are specified. LTL is useful, but in-sufficient. If we consider the large IS then there have typically hundred of entities and associations, but it seems quite reasonable to suppose that the verification of a property can be restricted to the entities and attributes involved. And in embedded software verification model checking is more preferable by system to test the system and describe the property for a large number of state, while theorem proving is the concepts of mathematical related . also theorem proving is quite difficult for large amount of state in the software because it is a very time consuming process .

References:

[1] Abrial,J.R.: TheB-Book: AssigningProgramstoMeanings.CambridgeUniversityPress,Cambridge, UK.

[2] Augusto, J.C., Ferreira, C., Gravell, A.M., Leuschel, M., Ng, K.M.Y.: The benefits of rapid modelling for e-business system development. ER Workshops pp. 1728.

[3] Aydal, E.G., Utting, M., Woodcock, J.: A comparisonof state-basedmodellingtools for model

validation. TOOLS-Europe08, Switzerland.

[4] [R08] T. Reinbacher, MCS-51 Simulator Integration into the [mc]square Model Checker, Technical Report, Institute of Embedded Systems, FH Technikum Wien.

[5] [CE81] E. Clarke, E. Emerson, Desing and synthesis of synchronisation skeletons using Branching Time Temporal Logic, In Proc. Workshop on Logics of Programs, Lecture Notes in ComputerScience, Vol. 131, Springer.

[6] [R07] F. Raimondi, Computational Tree Logic and Model Checking - A simple introduction, Validation and Verification Course Notes, University College London.

[7] [H97] G. Holzmann, The Model Checker SPIN, IEEE Transaction on Software Engineering, Vol. 23, No. 5, pp. 279-295.

[8] [H00] J. Holzmann, Logic Verification of ANSI-C Code with Spin, Proc. SPIN2000, Springer Verlag,

LNCS 1885, pp. 131-147