International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 3, March 2013 ISSN 2319 - 4847

FORMAL SPECIFICATION AND VERIFICATION OF REACTIVE SYSTEMS

Dr. Kendar Pratap[1], Shelja[2], Manjeet Kaur Bedi[3]

[1]Lecturer, Govt. National P.G.College, Sirsa, Haryana (INDIA)
[2]Lecturer, Mata Sahib Kaur Khalsa College For Women, (Dhandowal) Shahkot, Jalandhar, Punjab(INDIA)
[3]Assistant Professor, Govt. National P.G.College, Sirsa, Haryana (INDIA)

ABSTRACT

A system is called reactive, if its role is to maintain an ongoing interaction with its environment. A reactive system changes its actions, outputs and status in response to the input it receives from outside. Reactive systems are used in performing several crucial tasks, where the input that is given to them is vast and, most importantly, unpredictable. In this paper, we propose the specification and verification of reactive systems. The approaches to specification and verification of reactive systems which are different from computational systems are discussed. We have presented the Kripke structure or fair transition system and temporal logic that are most well known techniques for specifying system their properties. Verification can be done through model checking or deductive verification technique depending on the type of reactive system.

Keywords: reactive systems, kripke structure, fair transition, temporal logic, model checking.

1. INTRODUCTION

A “reactive system” is a computer program that continuously interacts with its environment. The term reactive emphasizes the interactive character of the software – in the simplest case a reactive program interacts with its environment while more realistic systems the program is a collection of elements interacting with each other as well as their common environment. By environment in this context human users, physical environment, and other reactive systems that interact with the system under consideration. For example, a light consisting of a bulb and a switch is a reactive system, reacting to the user changing the switch position.

These systems are increasingly used in applications where failure is unacceptable: electronic commerce, high-speed communication networks, traffic control systems, avionics, automated manufacturing, etc.

Reactive systems are different from computation systems. Computational system program transforms an input into an output. It should terminate to produce output. Non-termination is not allowed for computation program. In case of termination, the result produced should be unique.

Table 1- Some differences between computational and reactive systems

Keyword	Computational system	Reactive system
Interaction with environment	No	Yes
No-termination	Undesirable	Often desirable
Unique result	Yes	No

On the other hand, reactive system computes by reacting to stimuli from its environment. Non-termination is good for such systems. The results produced by them do not have to be unique.

1.1 Types of Reactive systems

On the basis of system states, there are two types of reactive systems:

a) **Finite state reactive systems:** For finite state reactive systems, the states are given by a finite number of finite state variables. This includes, in particular, hardware systems with a fixed number of components. The verification of temporal properties for finite state systems is decidable: model checking algorithms can automatically decide if a temporal property holds for a finite-state system.

b) **Infinite state reactive systems:** Infinite-state systems feature variables with unbounded domains, typically found in software systems such as integers, lists, trees, and other data types. The verification problem for general infinite-state systems is not decidable, and model checking techniques are not directly applicable.

2. SPECIFICATION OF REACTIVE SYSTEMS

Specification of reactive system requires model and properties to be specified. Specification is independent of design. Thus reactive system specification involves:

2.1 Specification of reactive system model by Kripke Structure or Fair transition system:

Model of reactive system can be specified by different ways. But the most prominent way to specify the model of reactive systems is through Kripke structure or Fair transition system.

2.1.1 Specification with Kripke structure

The reactive system S: (∑, ϕ, R) is given by a set of states ∑, a set of initial states ϕ ⊆ ∑, and a transition relation R ⊆ ∑ × ∑ . If < s1; s2 > ε R, the system can move from s1 to s2. A system S can be identified with the corresponding Kripke structure, or state-space. This is directed graph whose vertices are the elements of ∑ and whose edges connect each state to its successor states. If ∑ is finite, S is said to be finite-state. Describing large or infinite state spaces explicitly is not feasible. Therefore, the state-space is implicitly represented in some other form: a hardware description, a program, or a ω–automation.

2.1.2 Specification with fair transition system

Fair transition systems are a convenient formalism for specifying both finite and infinite-state reactive systems. A fair transition system is given by a set of transition; each is a first order relation between the set of system variables and the next-state system variables, indicating how the system can move from one state to the next.

2.1.3 Reactive systems and their specification of properties by LTL, CTL, CTL*:

We use temporal logic to specify properties of reactive systems. The term temporal logic is used to describe any system of rules and symbolism for representing and reasoning about propositions qualified in terms of time.

Linear time temporal logic (LTL) describes sets of sequences of states, and can thus capture universal properties of systems, which are meant to hold for all computations. LTL provides following temporal operators:

- X - for next time.
- U - for until.
- G - for globally.
- F - for eventually.
- W- for weakly until.

Computational tree logic (CTL) is a branching time temporal logic. In this, properties are interpreted over computation tree of the kripke structure. CTL combines temporal operators and path quantifiers that are:

- A – for every path.
- E – there exists a path.

Full computation tree logic (CTL*) combines the expressive powers of LTL, and CTL. The logic CTL* includes both the branching-time CTL and LTL.

3. VERIFICATION OF REACTIVE SYSTEMS

Formally check a formal model of the system against a formal specification. Correct and highly dependable construction of such systems is particularly important and challenging. A very promising and increasingly attractive method for achieving this goal is using the approach of formal verification Formal verification of reactive systems can be divided into two basic categories:

3.1 Model checking:

It is a promising automatic technique. It checks design models against specification. Here the checking is automatic and bug traces easily. It is very effective for control-intensive designs and Protocols. Many Commercial and academic tools: Spin (Bell Labs.), Formal-Check (Cadence), VIS (UCB), SMV (CMU, Cadence), etc. In-house tools: Rule Base (IBM), Intel, SUN, Bingo (Fujitsu), etc

Reactive system S and a temporal property F, the verification problem is to establish whether S|= F. For finite-state systems, model checking answer this question by a systematic exploration of the state-space of S, based on the observation that checking that a formula is true in a particular model is generally easier than checking that is true in all models; kripke structure of S is the particular model in question, and F is the formula being checked.

Basic procedure of model checking consists of following:

a) Describe the system as finite state model: Kripke model are used to describe reactive systems such as communication protocols, operating systems, and hardware circuits.
b) Express properties in temporal logic: Properties of reactive systems are specified using temporal logic.

c) Formal verification by automatic exhaustive search over the static space (model checking algorithm): Model checking algorithm checks whether all the executions of the model satisfy the formula.

The two main model checking techniques to verify reactive systems are:

- **Explicit state model checking technique** constructs and explore state of the system, one at a time. It is especially useful for software, where complex data structures are hard to handle symbolically.
- **Symbolic model checking technique** combats the state-explosion problem by using specialized formalisms to represent sets of states. The main idea of this technique is to represent the behavior of the system in symbolic form. Symbolic model checking are of two types:
  - **BDD (Binary decision diagrams) symbolic model checking-** symbolic model checking with BDD is a canonical form of representing Boolean formulas.
  - **SAT (Satisfiability) symbolic model checking-** satisfiability checking is performed using specialized tools for checking satisfiability, called as SAT-solvers.

3.2 Theorem proving:

It is also known as deductive verification, an approach of formal verification in which verification is represented as a theorem in a formal theory. A formal theory consists of language in which formulas are written, a set of axioms and a set of inference rules. With these rules and axioms, a theorem can be proved. It is a most powerful technique. Specification and design are logical formulae. In this checking involves proving a theorem and it is semi-automatic.
Theorem proving can be used for both finite and infinite reactive system verification. High degree of human expertise required. It is mainly confined to academic. There are number of public domain tools: Nqthm, STeP, PVS, HOL, etc.

Basic procedure of deductive verification of reactive system consists of following steps:

- **Describe the system as fair transition system:**

The fairness of the system is verified. A fair transition system: $\phi= \{V, \theta, T, J, C\}$ consists of:

- **V-** A finite set f typed system variables. A state s is an interpretation of system variables. The set of all states is called state space.
- **θ-** An initial condition. A satisfiable assertion that characterizes the initial states.
- **T-** A transition relation.
- **J-** $\{J_1,....,J_k\}$ A set of justice(weak fairness) requirements. Ensure that a computation has infinitely many J_i states for each J_i, i=1, ..., k.
- **C-** {<p1, q1>, ... <pn, qn>} A set of compassion(strong fairness) requirements.

- **Express properties in temporal logic:** Temporal specifications are use to verify the system correctness. A temporal formula is constructed out of state formulas to which we apply Boolean operators ⌐ and v and the basic temporal operators:

○- Next **ʘ- Previous**
U- Until **S- Since**

- **Formal verification with the help of axioms and proof rules**: With this any property of the system can be proved. Proof rules of first order logic, higher order logic or proof by induction can be used in theorem proving verification.

4. Conclusion

The actual input for reactive systems is non-constant. It keeps changing as a result of events that are initiated by the environment and given as input to the system, and as a result of the responses of the system to this input. Hence, the primary goal of reactive systems is to provide an offline guarantee that the input constraints will be met at run-time, regardless of the actual input that may be given at any time to the system. Moreover, since many real-life systems are hard to analyze manually, we should like to have computer support for our verification tasks. Many subtle bugs have been caught in designs cleared by simulation or testing. Hence specifications and verifications are an essential element of rigorous system analysis.

References

[1] J. L Turner, T.L. MCCluskey, "The Construction of Formal Specification: An Introduction to the Model-Based and Algebric Approaches", McGraw-Hill, 1994.
[2] R.J. Weiringa, "Design methods of Reactive Systems", Morgan Kaufmann Publishers.
[3] E.M. Clarke, O. Grumberg and D. Peled, "Model Checking", MIT Press, 1999.

[4] Luca Aceto, Anna Ingolfsdottir, Kim G. Larsen, Jiri Srba, "Reactive Systems: Modelling, Specification and Verification", Cambridge University Press- 2007.
[5] Klaus Schneider, "Verification of Reactive Systems- Formal methods and algorithms", Springer-Verlag Berlin Heidelberg-2004.
[6] C. Kern and M.R. Greenstreet, "Formal Verification in Hardware Design: A Survey", ACM TODAES, 1999.
[7] R. Kurshan, "Computer - Aided Verification of Co-ordinating Processes", Princeton Univ.Press, 1994
[8] Z. Manna and A. Pnueli, "Temporal Specification and Verification of Reactive Systems", Vol. I and II, Springer 1995.
[9] K. L. McMillan, "Symbolic Model Checking", Kluwer 1993.
[10] Kirsten Mark Hansen, "Schematic Specification and Verification of Reactive Systems", Department of Computer Science, Technical University of Denmark, 1991.
[11] Nicholas Halbwachs, "Synchronous Programming of Reactive Systems", Kluwer Academic Publishers Ntherlands.