)

General Comments

The thesis addresses the formal verification of client-server systems, particularly, systems with a single server and multiple clients, potentially unboundedly many. Petri-nets are chosen as the suitable modeling language to capture the concurrency and the unbounded nature of such systems. To model systems with distinguishable clients, v-nets are proposed, which are a particular type of Petri-net. In the later part of the thesis, Elementary Object Systems (EoS) are chosen as a representation of single server-multiple client systems, which provides a hierarchical petri-net representation, where tokens of the higher level Petri-net can be Petri-nets. The central modeling idea is to model the server as the top level net and the clients as the nets within the top level server net. In terms of expressing the verification properties of interest, the thesis takes into consideration two types of logic. The first is an extension of Linear Temporal logic to Linear Integer Arithmetic (LTL_LIA) and the second is the First Order Temporal Logic with monodic restriction (FOTL_1).

The first contribution of the thesis is a bounded model checking algorithm and supporting tool for Petri-nets with properties expressed as LTL_LIA. The SMT encoding of the net and the LTL_LIA specification is discussed. A 2-dimensional bounded model checking algorithm is discussed which essentially runs two bounding loops, one over the number of clients and the other on the length of the trace. The architecture of the tool DCModelchecker is presented thoroughly, together with experimental results on benchmarks from the MCC. Two variants of the 2-D model checking algorithm have been implemented, namely DCModelchecker1.0 and DCModelchecker2.0. The former is based on unfolding the net with interleaving semantics whereas the later is based on unfolding with true concurrent semantics.

The encoding of the net and the LTL_LIA specification into SMT formulation is the primary technical contribution of this chapter. The implementation in a tool adds to a systems engineering contribution. Empirical evaluation of the tools is sufficient.

The second contribution of the thesis is a bounded model checking algorithm and tool for Petri-nets with properties expressed as FOTL_1. Chapter 6 argues that certain verification properties are not expressible in LTL_LIA and thus requires more expressible logic such as FOTL_1. Examples of such properties are provided in a model of Automatic Parking System. To address this limitation, the chapter provides an SMT encoding of bounded semantics of specification in FOTL_1. This is the main contribution of this chapter.

The final contribution (Chapter 7) is the study of reachability and coverability problem of EoS under three types of lossiness, namely system level, object level and combined lossiness. The decidability status of lossy problems are examined and reported. The chapter starts with an introduction to EoS and how lossiness may sometimes ease a computational problem in hand. The chapter then introduces the definition of l-reachability/coverability problem in the presence of three types of lossiness and presents lemmas, theorems and proofs to conclude the decidability status reported in Table 7.1. The tool Telingo, to verify lossy PNs and EOS is presented. In all, the chapter has a commendable contribution.

Overall, I have no reservations that the contributions in the thesis, both quantity and quality wise are acceptable.

Clarifications/Amendments (if any)

The writing needs considerable improvement in my honest opinion. The following are chapterwise suggestions for amendments:

Chapter-1:

- Page 3, Para 1: Mention the other ways of addressing the state-space explosion problem such as state abstraction.
- Fig 1.3: Try to place the figure close to the text where it is referred.
- Page 3, Section 1.1: A definition of Buchi automaton is required before Defn 1.1.2.
- Why do you introduce the BMC problem on Buchi Automaton as the model? Why not introduce Petri nets right away?
- Page 4: Para 1, Line 1: G set is undefined here.
- Page 5, para 1: Honestly, I did not understand the example. What is the "bound" here? Because you mention "üsing the principle of bounded model checking".
- Page 6, Para 1, Line 5: Generally, requests are made by the clients and responded by the server. It is written the opposite.

Chapter-2

- Page 10, last line: The PN of Figure 2.1 doesn't show the transition weights. The PN after firing t_1 is the same PN of Fig 2.1? Please check.
- Page 12, Fig 2.3: Please specify the F(p,t) and F(t,p) values, without which it is hard to follow the example. P_3 should not have a token. Either the marking in the caption is incorrect or the figure.
- Fig 2.4: Need a better/clear figure here. The markings in P_3 is not clear.
- Figure 2.5: Can transitions that share tokens like t_1, t_2, fire concurrently? What if firing of one disables the other? Please clarify.
- Page 14, Sect 2.2 Para 1:
    - An introduction to "ünfolding a Petri-net" is crucial here. An example to demonstrate unfolding with bounded depth would help the reader. This is shown much later in the thesis, but is required right here.
    - Unfolding a PN with true concurrency will need smaller bound to cover markings. What is the relation between bound K of analysis in interleaving semantics vs the bound of analysis in concurrent semantics? A discussion is necessary.
- Page 18: What is the definition of an unbounded PN?
- Fig 2.6: What is APS? It is said much later in the thesis. Say it here.
- Page 19, Defn 2.2.1: What is v?
- Page 20, Para 1, last line: I do not see the mode of the transition in Fig 2.6, as claimed!

Chapter 3:

- Page 33, Sect 3.2: The state names do not match with the states of Fig 2.6.

Chapter 4:

- Sect 4.1: Say what is the distinction between Algo1 and Algo2.

Chapter 5:

- Sect 5.4: What is MCC?
- Page 71: Is ITS a BMC tool? What might be the cause of the discrepancy between ITS tools and DCModelchecker? Why is the tool confidence not 1? Need more discussion.
- Page 72, Para 3: "The execution times of DCMC 2.0 and DCMC 1.0 are comparable". Why so? Given that DCM 1.0 one works with true concurrent semantics, should it not be faster? Need further discussion here.

- Page 77: DCMC1.0 and DCMC 2.0 are BMC tools. How does evaluation of unbounded PNs in BMC tools going to be meaningful/significant? Verification of unbounded client server systems with bounded model checking reads like an oxymoron. Please clarify.
- In Table 5.2 experiments, what was the bound used?

Chapter 6:

- Page 80, Fig 2.6: The figure number should be corrected.
- Page 81, Para 2: These states are not shown in the v-net of Figure 2.6. Similarly t_{cp} transition is missing. This is disappointing given that the example of this figure serves as the running example.
- Page 81, para 2, last 3 lines: This is a repetition of sentences in Page 80, para 1. Please avoid repetition.
- Page 82: The examples are already described in Page 38. Please avoid repetition.
- Page 88: There is no discussion on the experimental results of the tool on running MC instances with FOTL_1 properties. This must be added!!
- I suggest that please add a limitations section to clarify the shortcomings of the tool, if any.

Section B.3

(Use additional sheets if required)

Questions for Viva-Voce

1. Why is the work on DCModelchecker1.0 and DCModelchecker 2.0 not published? What could you do to make it publishable?
2. Did you participate in the Model Checking Competition? What was your standing?