EXAM

Modeling and Analysis of Concurrent Systems 2

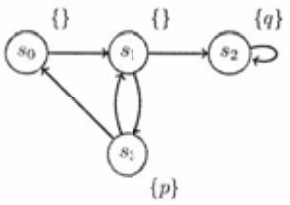
course code: 192135320
date: January 27, 2011
time: 8.45–12.15

General

- This is an 'open book' exam. Printed handouts (slides, articles, book chapters) may be used.
- This exam consists of 3 pages, 2 parts, and 8 questions
- Part I consists of open questions concerning the basic lectures (70 points)
- Part II consists of multiple choice questions concerning the research papers (30 points)

PART I: Basic Lectures - open questions

In Question 1 and 2, the following Kripke structure is considered:



Question 1 (16 points)

Consider the following formulas from CTL*:

1) $\mathbf{AG}\, q$ 2) $\mathbf{AFG}\neg q$ 3) $\mathbf{EF}\,\mathbf{AG}\, q$

A. Indicate which of the three formulas above are in LTL and which are in CTL. (*no explanation needed*)

B. Indicate which of the formulas above hold in s_0 above, using the standard CTL* semantics.

C. Next, we consider a single fairness constraint $\{p\}$.
Indicate which of the formulas above hold in s_0, using the fair CTL* semantics. Explain this answer.

Question 2 (20 points)

Next, we apply the symbolic model checking algorithm on the previous Kripke structure.

A. Rewrite the formula $\mathbf{AG}\,\neg q$ into the $\mathbf{EU}$ fragment.

B. Demonstrate the iterations of the symbolic model checking algorithm when computing the set of states in the previous Kripke structure that satisfy $\mathbf{AG}\neg q$. It is *not* required to present BDD notation.

C. The states that satisfy $\mathbf{EG}\,\neg q$ under fairness constraint $\{p\}$ can be computed by a nested fixed point computation. What is the precise nested fixed point expression? (It is *not* required to evaluate it.)

D. Explain why the generation of counter examples in symbolic model checking (as explained in *Model Checking* by Clarke etal.) requires considerably more memory than just symbolic model checking.

Question 3 (18 points)

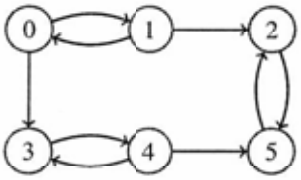
A. Provide the OBDD-representation of the formulas $p \wedge r$ and $(p \wedge q) \leftrightarrow r$, under the variable ordering $p < q < r$.

B. Demonstrate the `BDD-apply` function to compute the conjunction ($\wedge$) of the previous two BDDs. It is sufficient to show the call-graph of the recursive calls to `BDD-apply` and the final result.

C. What is the worst-case time-complexity of `BDD-apply` and how is this upper-bound ensured?

Question 4 (16 points)

We apply the distributed forward-backward reachability algorithm (FB) to compute the strongly connected components of the following graph:



A. Run the FB algorithm on this graph, choosing the pivot in such a way that the graph splits in as many independent parts as possible. Indicate clearly:

- which state(s) you choose as pivot
- what is the result of the intermediate forward and backward closures
- what are the resulting SCCs

B. Mention shortly the two different levels of parallelism introduced by the FB algorithm.

PART II: Research Papers - multiple choice questions

This part considers the following research papers studied during the course:

bmc Bounded Model Checking with SAT (Clarke, Biere, etal.)

abs The Software Model Checker BLAST (Beyer, Henzinger, etal.)

sym Better Verification through Symmetry (Ip, Dill)

lrn Learning to Divide and Conquer (Pasareanu, etal.)

mgc Memoised Garbage Collection for SMC (Nguyen, Ruys)

clt Efficient State storage with Cleary Tables (Dillinger, Maniolis)

mcr Multi-core Reachability (Laarman, van de Pol, Weber)

Question 5 (6 points)

Paper [abs] and [mgc] describe approaches to software model checking.
Which one of the following statements hold?

A. Both papers are based on extracting models from code (and checking the model)

B. Both papers are based on executing the code (in a non-standard way)

C. [abs] is based on model extraction, while [mgc] is based on code execution

D. [abs] is based on code execution, while [abs] is based on model extraction

Question 6 (6 points)

Paper [lrn] uses the L^* learning algorithm in compcsitional verification.
What is actually learnt in the paper?

A. The L^* algorithm learns assumptions on the component to be verified

B. The L^* algorithm learns assumptions on the environment of the component to be verified

C. The L^* algorithm learns the properties to be verified on the component

D. The L^* algorithm learns counter examples to the property to be verified

Question 7 (6 points)

Both papers [mcr] and [clt] discuss efficient hash table implementations.
Which one of the following statements is definitely *wrong*?

A. [clt] uses less memory than [mcr], because [clt] doesn't store full hash keys, while [mcr] does.

B. Both [clt] and [mcr] use some variant of linear probing, which is good for avoiding cache misses.

C. In [mcr], an inserted entry will always stay on the same position, while in [clt] inserted entries might move up or down by later insertions.

D. Although not mentioned explicitly in [clt], that implementation must contain some locking primitive, like the compare-and-swap operation used in [mcr].

Question 8 (12 points)

Model checking is an automatic but not always complete solution. The following anomalies could happen (among others):

1) the tool terminates successfully, but misses a potential violation of the property.

2) the tool loops forever without deciding the absence or presence of a bug.

3) the tool aborts because the needed memory is insufficient.

Indicate for each of the following statements if they hold or not.

A. In [sym] some individual states are not inspected, so 1) can happen.

B. In symbolic model checking with BDDs, 1) and 2) are not possible; only 3) may happen.

C. In [abs], 1) and 2) are not possible; only 3) may happen.

D. In [bmc] with a given fixed bound, 1) is possible but 2) cannot happen.

E. In [bmc] with an increasing bound, 1) is impossible and 2) still cannot happen.

F. With Cleary tables and Bloom filters as advocated in [clt], 1) can never happen.

G. A tool that suffers from 1) is more useful for bug hunting than for full verification