EXAMINATION

Modeling and Analysis of Concurrent Systems 2

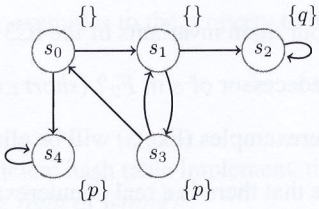
course code: 192135320
date: January 30, 2015
time: 13.45–16.45

General

- This is an 'open book' exam. Printed handouts (slides, articles, book chapters) may be used, but no handwritten notes, previous examinations, or their answers.
- This exam consists of 3 pages with 7 questions. In total 100 points can be earned:
  - 70 points with the material of the main lectures, (Question 1-4)
  - 30 points with the material from the research papers, (Question 5-7).

PART I: Basic Lectures

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) $\mathbf{A}\,\mathbf{FG}\neg q$ 3) $\mathbf{EF}\,\mathbf{AG}\, q$

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

B. Which of the formulas above hold in s_0, according to normal CTL* semantics? *(short explanation)*

C. Next, we consider a single fairness constraint p, i.e. $\{\{s_3, s_4\}\}$.
Indicate which of the formulas above hold in s_0, using the *fair* CTL* semantics. *(short explanation)*

Question 2 (18 points)

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

A. Rewrite the formula $\mathbf{AG}\,\neg q$ into the proper $\mathbf{EX}, \mathbf{EU}, \mathbf{EG}$ 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. Write down the precise fixed point expression, using least (μ) and/or greatest (ν) fixed point operators? *(It is not required to evaluate it.)*

Question 3 (18 points)

A. Provide the OBDD-representations of the following two formulas, using variable ordering $p < q < r$:

- $p \leftrightarrow q \leftrightarrow r$ and
- $(p \leftrightarrow q) \wedge r$.

B. The recursive `BddApplyAnd` function can be used to compute the conjuction ($\wedge$) of these two BDDs.

(i) Number the nodes in your BDDs and draw the call graph of `BddApplyAnd`.

(ii) Indicate clearly at which point(s) a result from the Operations Cache is reused.

C. Explain why the Operations Cache is required to compute `BddApplyAnd` in polynomial time.

Question 4 (18 points)

We consider a run of the IC3 algorithm for incremental inductive verification, with set of initial states I and set of bad states $\neg p$. After three iterations, we found $I = F_0 \subseteq F_1 \subseteq F_2 \subseteq F_3 \subseteq p$. Assume that we find some $s \in F_3$ with $s \rightarrow t$ but $t \notin p$. In that case IC3 tries to remove s from F_3 and to propagate this back by removing predecessors of s from F_2, etc.

A. Draw this situation in a Venn diagram.

B. Show that it follows from the four main invariants of the IC3 algorithm that $s \not\in F_2$.

C. Is it possible that there is no predecessor of s in F_2? (*short explanation*)

D. How does IC3 avoid that counterexamples (like s) will be eliminated one by one?

E. Explain when IC3 can conclude that there is a real counterexample to $AG\neg p$.

F. Explain when IC3 can conclude that the property $AG\neg p$ holds.

[SEE ALSO PAGE 3]

PART II: Student Lectures

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

mcr Multi-core Reachability (Laarman, van de Pol, Weber) [Lecture 3]

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

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

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

tpn Timed-Arc Petri Nets (Viesmose, Moesgaard etal.)

car Counterexample-Guided Abstraction Refinement (Clarke, Grumberg etal.)

smc Software Model Checking: the VeriSoft approach (Godefroid)

Question 5 (8 points)

Paper [lrn] uses the L^* learning algorithm in compositional verification.
Indicate if the following statements are *right* or *wrong*.

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

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

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

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

Question 6 (8 points)

Both papers [mcr] and [clt] discuss efficient hash table implementations.
Indicate if the following statements are *right* or *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, because this helps to avoid 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, as used in [mcr].

Question 7 (14 points)

Many algorithms apply forward reachability, based on breadth-first-search from the initial state. Indicate which of the following methods explore all concrete states. (*provide a short to-the-point explanation.*)

A. Inductive Incremental Verification (IC3)

B. SAT-based bounded model checking as in [bmc]

C. BDD-based symbolic model checking

D. The CEGAR approach as in [car]

E. The model checker TAPAAL for Timed-arc Petri Nets from [tpn]

F. The Software Model Checker VeriSoft [smc]

G. The multi-core model checker from [mcr] (in BFS mode)