Software Verification

ETH Zürich

14 December 2009

Surname, first name: ....

Student number: ....

I confirm with my signature, that I was able to take this exam under regular circumstances and that I have read and understood the directions below.

Signature: ....

Directions:

- Exam duration: 1 hour 45 minutes.
- Except for a dictionary you are not allowed to use any supplementary material.
- All solutions can be written directly on the exam sheets. If you need more space for your solution ask the supervisors for a sheet of official paper. You are **not** allowed to use other paper. Please write your student number on **each** additional sheet.
- Only one solution can be handed in per question. Invalid solutions need to be crossed out clearly.
- Please write legibly! We will only correct solutions that we can read.
- Manage your time carefully (take into account the number of points for each question).
- Please tell **immediately** the exam supervisors if you feel disturbed during the exam.

Good luck!

Question	Available points	Your points
1) Axiomatic semantics	12	
2) Separation logic	8	
3) Model checking	14	
4) Software model checking	14	
5) Program analysis	8	
6) Abstract interpretation	14	
Total	**70**	

1 Axiomatic semantics (12 points)

Consider the following Hoare triple (all variables of type NATURAL, assumed to describe mathematical natural numbers):

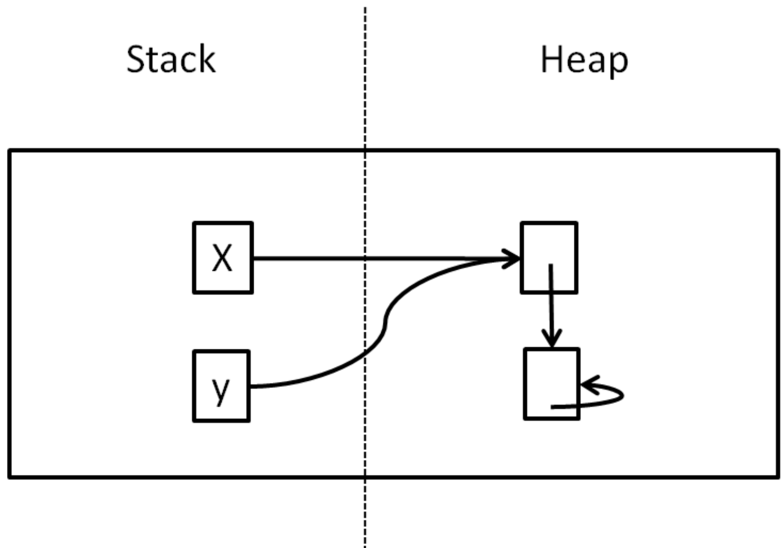
```
{ x = n }
from
    z := 0
until x < y do
    z := z + 1
    x := x − y
end
{ n = z ∗ y + x }
```

Prove that this triple is a theorem of Hoare's axiomatic system for partial correctness.

2 Separation logic (8 points)

2.1 (4 points)

Consider the following program state:



Indicate in the following table whether or not a given assertion is satisfied by this state. Indicate satisfaction with a T and non-satisfaction with an F.

	T or F
$\exists v \cdot x \mapsto v * v \mapsto v$	
$y \mapsto \_$	
$(x = y) \wedge (y \mapsto \_ * true)$	
$(x = y) * true$	

2.2 (4 points)

Do the following implications hold for any predicate P, Q and any heap? If an implication holds, explain why. If it does not hold, provide a counterexample.

(1) $(P) \;\Rightarrow\; (P * P)$

(2) $(P * Q) \;\Rightarrow\; [(P \wedge Q) * true]$

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

3 Model checking (14 points)

Let us recall the semantics of LTL over finite words with alphabet $\mathcal{P}$. For a word $w = w(1)w(2)\cdots w(n) \in (2^{\mathcal{P}})^*$ with $n \geq 0$ and a position $1 \leq i \leq n$ the satisfaction relation $\models$ is defined recursively as follows for $p, q \in \mathcal{P}$.

$w, i \models p$	iff	$p \in w(i)$
$w, i \models \neg\phi$	iff	$w, i \not\models \phi$
$w, i \models \phi_1 \wedge \phi_2$	iff	$w, i \models \phi_1$ and $w, i \models \phi_2$
$w, i \models \mathsf{X}\phi$	iff	$i < n$ and $w, i+1 \models \phi$
$w, i \models \phi_1 \mathsf{U} \phi_2$	iff	there exists $i \leq j \leq n$ such that: $w, j \models \phi_2$ and for all $i \leq k < j$ it is the case that $w, k \models \phi_1$
$w \models \phi$	iff	$w, 1 \models \phi$

Also recall the derived operators:

$\Diamond \phi$	defined as	$\mathbf{True} \, \mathsf{U} \, \phi$
$\Box \phi$	defined as	$\neg\Diamond\neg\phi$

3.1 Automata and LTL formulas (6 points)

Consider the automaton T in Figure 1, where A is the initial state and D is the accepting state.

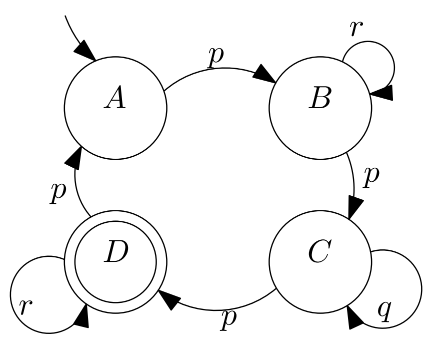


Figure 1: Automaton T.

For each of the following LTL formulas say whether every run of T satisfies the formula: if it does, demonstrate informally (but precisely) and briefly why this is the case; if it does not, provide a counterexample.

(1) $\Box\Diamond p$

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

(2) $\Diamond p$

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

(3) $\Box\Big((p \,\mathsf{U}\, q) \implies (p \,\mathsf{U}\, (q \wedge \Diamond p))\Big)$

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

3.2 Automata-based model checking (8 points)

Consider again the automaton T in Figure 1. Prove by the basic algorithm for automata-based model checking that the LTL formula $\psi \triangleq \Box(q \Longrightarrow \Diamond p)$ is a property of the automaton.

(1) Build an automaton $a(\neg\psi)$ for $\neg\psi$.

(2) Build the intersection automaton $T \times a(\neg\psi)$ and check that it has no reachable accepting state.

4 Software model checking (14 points)

Consider the following function that computes the product of two integers if they are both negative or both positive, and returns zero otherwise.

```
same_sign_product (x, y: INTEGER): NATURAL
  do
    if x > 0 then
      if y > 0 then
        Result := x * y
      else Result := 0 end
    else
      if x ≠ 0 then
        if y < 0 then
          Result := x * y
        else Result : = 0 end
      else Result := 0 end
    end
  ensure
    x*y > 0 ⟺ Result > 0
  end
```

4.1 Boolean abstractions (10 points)

Build the Boolean abstraction ssp_1 of same_sign_product with respect to the following predicates:

```
p = x > 0
q = y > 0
r = x*y > 0
s = Result > 0
t = x < 0
```

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

4.2 Abstract counterexamples (4 points)

Provide an annotated counterexample trace for the Boolean abstraction ssp_1. The counterexample should be in the form of a valid sequence of statements and branch conditions in ssp_1 which reaches the bottom of the function with a false postcondition. Each statement in the sequence must be preceded and followed by a complete description of the abstract program state in terms of values of the Boolean predicates p, q, r, s, t.

Also tell whether the counterexample trace is feasible in the original concrete function same_sign_product, briefly justifying your answer.

5 Program analysis (8 points)

Consider the following program fragment:

```
from
    x := 2
    y := 1
    z := x − 1
until z > 30 do
    if x < 5 then
        z := z ∗ x
    else
        z := z + y
    end
    x := x + 1
end
```

(1) Draw the control flow graph of the program fragment and label each elementary block.

(2) Annotate your control flow graph with the analysis result of a reaching definitions analysis of the program fragment.

6 Abstract interpretation (14 points)

Consider the language of integer arithmetic expressions $e \in \mathbf{Exp}$ defined by

$$e ::= n \mid -e \mid e + e \mid e * e$$

with the following concrete semantics $C : \mathbf{Exp} \to \mathbb{Z}$:

$$\begin{array}{rcl} C[n] & = & n \\ C[-e] & = & -C[e] \\ C[e+e] & = & C[e] + C[e] \\ C[e*e] & = & C[e] \cdot C[e] \end{array}$$

The goal of this exercise is to define an abstract interpretation to determine whether e is divisible by 5.

(1) Suggest a suitable abstract domain $\mathbf{D}$.

(2) Define the concretization function $\gamma : \mathbf{D} \to \wp(\mathbb{Z})$

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

(3) The abstract semantics is given by the function $A : \mathbf{Exp} \rightarrow \mathbf{D}$:

$$\begin{array}{rcl} A[n] & = & \ldots \\ A[-e] & = & \ominus A[e] \\ A[e+e] & = & A[e] \oplus A[e] \\ A[e*e] & = & A[e] \otimes A[e] \end{array}$$

Complete the specification of the function A by:

(a) defining $A[n]$, and

(b) defining the abstract operations $\ominus$, $\oplus$, $\otimes$.

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .