

Software Verification – Exam

ETH Zürich

17 December 2012

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 **immediately** tell the exam supervisors if you feel disturbed during the exam.

Good luck!

Question	Available points	Your points
1) Axiomatic semantics	18	
2) Separation Logic	15	
3) Data flow analysis	10	
4) Model checking	15	
5) Real time verification	12	
Total	70	

[This page is intentionally left blank.]

1 Axiomatic semantics (18 points)

Consider the following annotated program, where A is an array indexed from 1 of element type G , n is an integer variable storing A 's size, k is another integer variable, v is a variable of type G initialized to some fixed value, $found$ is a Boolean variable.

```

1  {  $n \geq 0$  }
2  from
3     $k := n$ 
4     $found := \mathbf{False}$ 
5  until  $found$  or  $k < 1$  loop
6    if  $A[k] = v$  then
7       $found := \mathbf{True}$ 
8    else
9       $k := k - 1$ 
10   end
    {  $(found \implies 1 \leq k \leq n \wedge A[k] = v) \wedge (\neg found \implies k < 1)$  }
```

1.1 Program semantics (2 points)

Characterize, in plain English, which value of k the program computes from the inputs A , n , and v . In other words: what does the program do?

Solution:

As apparent from the postcondition, the program searches for an element with value v in A between positions 1 and n . If it finds the element, k stores the element's position; if it does not find the element, k is out of array's bounds.

1.2 Partial correctness (15 points)

Prove that the triple (precondition, program, postcondition) is a theorem of Hoare's axiomatic system for partial correctness.

Solution:

```

1  {  $n \geq 0$  }
2  from
3     $k := n$ 
4     $found := \mathbf{False}$ 
5  {  $0 \leq k \leq n \wedge (found \implies 1 \leq k \leq n \wedge A[k] = v)$  }
6  until  $found$  or  $k < 1$  loop
7  {  $1 \leq k \leq n \wedge \neg found \wedge (found \implies 1 \leq k \leq n \wedge A[k] = v)$  }
8  if  $A[k] = v$  then
9    {  $A[k] = v \wedge 1 \leq k \leq n \wedge \neg found$  }
10   {  $0 \leq k \leq n \wedge 1 \leq k \leq n \wedge A[k] = v$  }
```

```

11   found := True
12   { 0 ≤ k ≤ n ∧ (found ⇒ 1 ≤ k ≤ n ∧ A[k] = v) }
13   else
14   { A[k] ≠ v ∧ 1 ≤ k ≤ n ∧ ¬found }
15   { 1 ≤ k ≤ n + 1 ∧ (found ⇒ 2 ≤ k ≤ n + 1 ∧ A[k - 1] = v) }
16   k := k - 1
17   { 0 ≤ k ≤ n ∧ (found ⇒ 1 ≤ k ≤ n ∧ A[k] = v) }
18   end
19   { 0 ≤ k ≤ n ∧ (found ⇒ 1 ≤ k ≤ n ∧ A[k] = v) }
20   end
21   { (found ∧ 1 ≤ k ≤ n ∧ A[k] = v) ∨ (¬found ∧ k = 0) }
22 { (found ⇒ 1 ≤ k ≤ n ∧ A[k] = v) ∧ (¬found ⇒ k < 1) }

```

1.3 Termination (1 point)

Find a suitable *variant* function V to prove termination. V must be such that it decreases along all branches of the loop body, and it is nonnegative after every iteration of the loop. You do *not* have to prove termination, just write a suitable variant and informally argue why it is a suitable variant.

Solution:

The variant $V \triangleq k + [found = \mathbf{False}]$, where the square brackets denote Iverson bracket (equal to 1 if its arguments holds, and equal to 0 otherwise), is always nonnegative and decrease in both branches (k decreases along the **else** branch, and $[found = \mathbf{True}]$ decreases along the **then** branch). Hence, V can be used to build a proof of termination for the loop.

2 Separation Logic (15 points)

A well-formed binary tree t is given by the grammar:

$$t \stackrel{\text{def}}{=} n \mid (t_1, t_2)$$

So a tree value t can be either a leaf, which is a single number n , or an internal node with a left subtree t_1 and a right subtree t_2 .

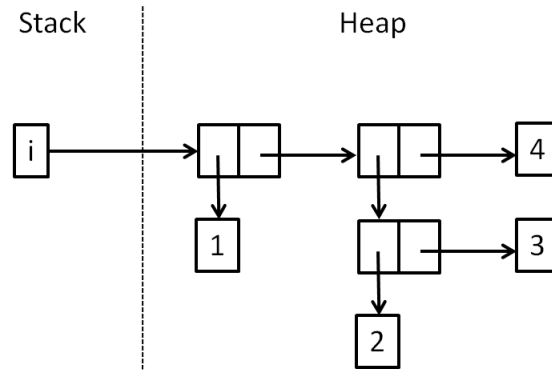
Consider the definition of the recursive predicate $tree\ t\ i$ which asserts that i is a pointer to a well-formed binary tree t :

$$\begin{aligned} tree\ n\ i &\stackrel{\text{def}}{=} i \mapsto n \\ tree\ (t_1, t_2)\ i &\stackrel{\text{def}}{=} \exists l, r. i \mapsto l, r * tree\ t_1\ l * tree\ t_2\ r \end{aligned}$$

With these definitions in mind, answer the following questions.

2.1 Predicate Satisfaction (6 points)

Consider the following program state:



Indicate in the 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
$\exists z. i \mapsto z$	F
$\exists x, y. i \mapsto 1, x * x \mapsto y, 4 * true$	F
$\exists j. tree\ (2, 3)\ j * true$	T
$\exists j, t. i \mapsto j * j \mapsto 1 * tree\ t\ (i + 1)$	F
$\exists t_1, t_2. tree\ (t_1, (t_2, 4))\ i$	T
$\exists j, k. i \mapsto j * j \mapsto 1 * (i + 1) \mapsto k * tree\ ((2, 3), 4)\ k$	T

2.2 Code Verification (9 points)

Give a brief proof outline of the following triple. There must be at least one assertion between every two sub-commands.

$$\begin{aligned} &\{tree\ (1, t)\ i\} \\ &x := [i]; [i] := 2; y := [i + 1]; \mathbf{dispose}\ i; \mathbf{dispose}\ x; \mathbf{dispose}\ (i + 1) \\ &\{tree\ t\ y\} \end{aligned}$$

```

{tree (1, t) i}
{∃l, r · i ↦ l, r * tree 1 l * tree t r}
  x := [i];
{∃r · i ↦ x, r * tree 1 x * tree t r}
  [i] := 2;
{∃r · i ↦ 2, r * tree 1 x * tree t r}
  y := [i + 1];
{i ↦ 2, y * tree 1 x * tree t y}
  dispose i;
{(i + 1) ↦ y * tree 1 x * tree t y}
{(i + 1) ↦ y * x ↦ 1 * tree t y}
  dispose x;
{(i + 1) ↦ y * tree t y}
  dispose (i + 1);
{tree t y}

```

3 Data flow analysis (10 points)

Consider the following program fragment (all variables are of type *INTEGER*):

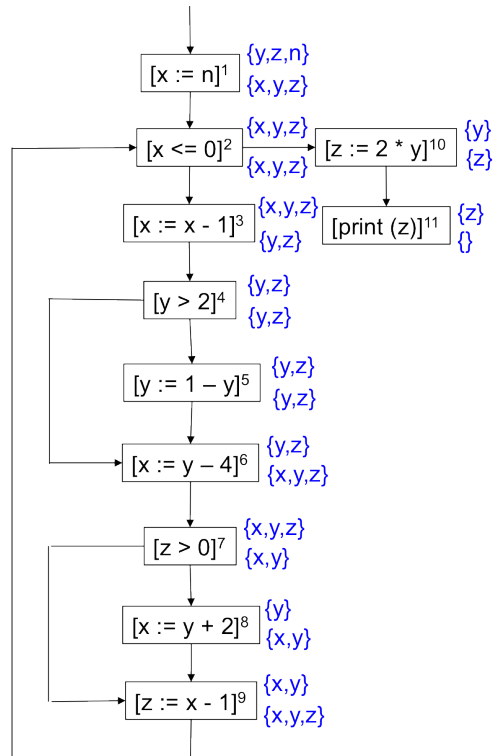
```

1  from
2   $x := n$ 
3  until  $x \leq 0$  do
4   $x := x - 1$ 
5  if  $y > 2$  then
6   $y := 1 - y$ 
7  end
8   $x := y - 4$ 
9  if  $z > 0$  then
10  $x := y + 2$ 
11 end
12  $z := x - 1$ 
13 end
14  $z := 2 * y$ 
15 print ( $z$ )

```

- (1) (3 points) Draw the *control flow graph* of the program fragment and label each elementary block.
- (2) (5 points) Annotate your control flow graph with the analysis result of a *live variables analysis* of the program fragment.

Solution:



- (3) (2 points) Explain how the live variables analysis can be used for dead code elimination, and apply this technique to the program fragment; it suffices to state which statement(s), if any, would be removed as dead code.

Solution:

An assignment $[x := a]^\ell$ is dead if the value of x is not used before it is redefined. The live variables analysis gives, for each program point, the variables that may be live at the exit from the point. Hence if $x \notin L\mathit{Vexit}(\ell)$, the assignment is dead.

This is the case for the block at program point 3, which is the only statement to be removed.

4 Model Checking (15 points)

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

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

4.1 Automata and LTL formulas (7 points)

Consider the automaton \mathcal{A} (with states A, B, C) in Figure 1, over the alphabet $\{p, q\}$. Notice that A is the initial state and B is final.

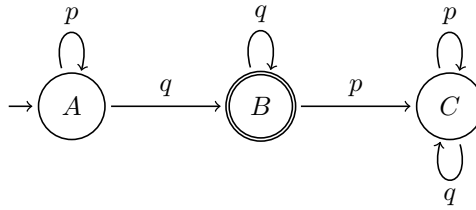


Figure 1: Automaton \mathcal{A} over alphabet $\{p, q\}$.

For each of the following LTL formulas say whether every accepting run of \mathcal{A} satisfies the formula. If it does, argue informally (but precisely) why this is the case; if it does not, provide a counterexample.

(1) $\mathcal{A} \models \diamond p$

No: the word q is accepted by \mathcal{A} but does not satisfy $\diamond p$.

(2) $\mathcal{A} \models \diamond q$

Yes: every accepting run must reach state B , and hence it must include q somewhere.

(3) $\mathcal{A} \models \neg p \vee \mathbf{X}(\neg q)$

No: the word pq does not satisfy $\neg p \vee \mathbf{X}(\neg q)$, because p holds in the first position and q holds in the second position; however, it is accepted by \mathcal{A} .

(4) $\mathcal{A} \models \square(q \implies \square q)$

Yes: after the first q is read, \mathcal{A} is in state B , which no accepting run ever leaves; and the only transition possible in B is reading q .

(5) $\mathcal{A} \models \Box(p \cup q)$

Yes: when \mathcal{A} is in state A , q must occur in the future (to reach the accepting state B) and only p can occur before it; hence $\mathcal{A} \models p \cup q$. After \mathcal{A} enters state B , it does not leave it on any accepting run; the only transition possible in B is reading q , which trivially satisfies $p \cup q$ with q occurring at the evaluation instant.

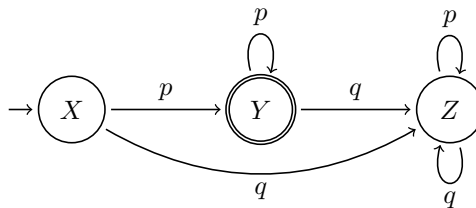
4.2 Automata-based model checking (8 points)

Show that $\mathcal{A} \models p \implies \Diamond q$ using automata-based model checking as follows.

Property automaton (4 points). Construct an automaton \mathcal{F} that accepts *precisely* the words that satisfy $\neg(p \implies \Diamond q)$, that is, the *complement* of the property we want to verify.

Solution:

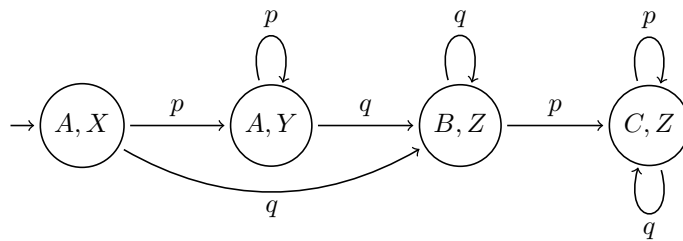
$\neg(p \implies \Diamond q)$ is accepted by the automaton:



Intersection automaton (4 points). Construct the intersection automaton $\mathcal{A} \times \mathcal{F}$ that accepts precisely the words accepted by both \mathcal{A} and \mathcal{F} and show that $\mathcal{A} \times \mathcal{F}$ does not accept any words.

Solution:

No accepting state is connected.



5 Real Time Verification (12 points)

Recall the semantics of (a subset of) MTL over finite timed words with alphabet \mathcal{P} and time domain \mathbb{T} . For a timed word

$$w = [\sigma(1), t(1)][\sigma(2), t(2)] \cdots [\sigma(n), t(n)] \in (\mathcal{P} \times \mathbb{T})^*$$

with $n \geq 0$ and a position $1 \leq i \leq n$ the satisfaction relation \models is defined recursively as follows for $p \in \mathcal{P}$ and J an interval of \mathbb{T} with integer endpoints.

$$\begin{array}{ll} w, i \models p & \text{iff } p = \sigma(i) \\ w, i \models \neg\phi & \text{iff } w, i \not\models \phi \\ w, i \models \phi_1 \wedge \phi_2 & \text{iff } w, i \models \phi_1 \text{ and } w, i \models \phi_2 \\ w, i \models \diamond_J \phi & \text{iff there exists } i \leq j \leq n \text{ such that: } t(j) - t(i) \in J \text{ and } w, j \models \phi \\ w, i \models \square_J \phi & \text{iff for all } i \leq j \leq n: \text{ if } t(j) - t(i) \in J \text{ then } w, j \models \phi \\ w \models \phi & \text{iff } w, 1 \models \phi \end{array}$$

As time domain \mathbb{T} , we will consider either the natural numbers $\mathbb{N} = \{0, 1, 2, \dots\}$ or the nonnegative real numbers $\mathbb{R}_{\geq 0}$.

5.1 MTL semantics (4 points)

(1) Are the MTL formulas ψ_1 and ψ_2 :

$$\begin{aligned} \psi_1 &\triangleq \diamond_{[1,1]} \left(\diamond_{[1,1]}(p) \right) \\ \psi_2 &\triangleq \diamond_{[2,2]}(p) \end{aligned}$$

equivalent over time domain \mathbb{N} ? If they are, show that their semantics imply each other; if they are not, provide a timed word which is satisfied by one formula and not satisfied by the other.

Solution:

They are *not* equivalent: the word $\mu = [p, 1][p, 3]$ is satisfied by ψ_2 but not by ψ_1 which requires at least *three* positions in a word to satisfy it.

(2) Does the answer to the previous question (1) change over the time domain $\mathbb{R}_{\geq 0}$? Explain why it does or does not change.

Solution:

The answer is still negative, and the very same word μ works for the time domain $\mathbb{R}_{\geq 0}$.

5.2 Timed automata and MTL formulas (8 points)

In this section, we take $\mathbb{R}_{\geq 0}$ as the time domain \mathbb{T} . Consider the timed automaton \mathcal{T} (with locations A, B, C) in Figure 2, over the alphabet $\{p, q\}$ and with clocks x and y . Notice that A is the initial location and C is final.

For each of the following MTL formulas say whether every accepting run of \mathcal{T} satisfies the formula. If it does, argue informally (but precisely) why this is the case; if it does not, provide a counterexample.

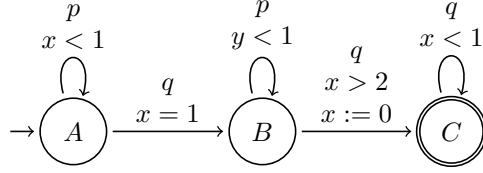


Figure 2: Timed automaton \mathcal{T} over alphabet $\{p, q\}$ with clocks x and y .

(1) $\mathcal{T} \models \diamond_{(0,2)} q$

No: the timed word $[q, 1][q, 10]$ is accepted by \mathcal{T} but it does not satisfy $\diamond_{(0,2)} q$ because q does not occur in the interval $(0, 2)$ relative to the first position where the formula is evaluated (according to the semantics given above).

(2) $\mathcal{T} \models \square_{[0,\infty)} (\diamond_{(1,\infty)} p)$

No: the timed word $[q, 1][q, 2.2]$ is accepted by \mathcal{T} but it does not satisfy $\square_{[0,\infty)} (\diamond_{(1,\infty)} p)$ because p never occurs..

(3) $\mathcal{T} \models \square_{[0,\infty)} (q \implies \square_{[0,\infty)} (\neg p))$

Yes: after the first occurrence of q at time 1 (which appears in every accepted word), p cannot happen anymore, thus trivially satisfying $\square_{[0,\infty)} (\neg p)$. In fact, clock y has the value 1 when entering location B , since it is synchronized with x ; therefore, the self loop on location B cannot be taken. Finally, there are no accessible other transitions where p may occur.

(4) $\mathcal{T} \models \diamond_{(2,\infty)}(q) \wedge \square_{(3,\infty)}(\neg q)$

No: the timed word $[q, 1][q, 10][q, 10.3]$ is accepted by \mathcal{T} but it does not satisfy $\square_{(3,\infty)}(\neg q)$, and hence neither the whole MTL formula $\diamond_{(2,\infty)}(q) \wedge \square_{(3,\infty)}(\neg q)$, even if it satisfies its first conjunct.