Software Verification

Wednesday, January 14th 2026, 3 hours

This assignment contains three independent parts: the first part deals with bounded model-checking, the second part is about interpolation, and the last part addresses abstract interpretation.

All documents are authorized during the examination
Answers can be written in French

1 Bounded Model-Checking (8pts)

We recall that given a transition system $\mathcal{T}$, the Bounded Model-Checking at depth n explores all executions of length at most n and checks if one of them reaches an undesirable state (such an execution is called *faulty*).

1.1 Application to an example

We consider the following program:

```
x := max(1,x);
p := max(2,p);
q := max(p+1,q);
while(x < p * q){
  assert(x mod p != 0 || x mod q != 0);
  x := x+1;
}
```

We consider that `max` is a basic instruction of the language. We also consider that we can perform boolean operation in the guards (you will also allow it in your automaton).

Question 1 *Give the control-flow automaton corresponding to that code. If you have skip edges of the form $q \xrightarrow{\text{skip}} q'$ such that q has no other outgoing edge or q' has no other incoming edge, merge q and q' (in order to get a smaller automaton).*

In the same spirit, merge the three initial affectation in one single transition.

Question 2 *Apply the bounded model-checking algorithm in forward to the previous system up to bound 7: draw the exploration tree displaying the formulæ added (only the added one not the complete formula) at each step in the Depth-First Search algorithm, highlight the unreachable nodes, and give the result of the algorithm.*

Is there a bound for which the answer would be different? Justify your answer.

Question 3 *If we replace the variables p and q by constant values, what constraint on p and q do we need to have a system that cannot violate the assert?*

Question 4 *Prove that, if we replace p and q by constant values, there exist a bound for which the forward BMC cannot answer Non-Exhaustive.*

1.2 Proof of correctness

Bounded Model Checking is intrinsically a semi-algorithm, as the Model-Checking is undecidable in general. In fact, except for simple systems, it is unlikely such a procedure asserts a program is correct. The goal of this section is to provide a way to prove that for some program, we have a bound b such that if the BMC didn't find a bug at depth b, then there is no bug. It will rely on the determination of an inductive invariant of the program.

We fix a transition system $\mathcal{T} = (\mathcal{S}, \Delta, I, \text{Bad})$, where $\mathcal{S}$ is the set of configuration, Δ its successor relation, which we suppose total, I its initial set of configurations, and Bad the set of bad configurations (in the course, Bad was always the set of configurations with the bad state, but the algorithm would work for any set).

We recall that a sequence of configurations $s_0, \cdots, s_k$ is an execution if and only if the following formula is true:

$$s_0 \in I \wedge \bigwedge_{i=0}^{k-1} \Delta(s_i, s_{i+1})$$

and that it is a faulty execution if and only if the following formula is true:

$$s_0 \in I \wedge \bigwedge_{i=0}^{k-1} \Delta(s_i, s_{i+1}) \wedge s_k \in \text{Bad}$$

Given an integer n and a set of configurations $\mathcal{C}$, we say that $\mathcal{C}$ is n-inductive for $\mathcal{T}$ if, for every $s_0, s_1, \cdots, s_{n-1} \in \mathcal{C}$, and $s_n \in \mathcal{S}$, if $s_0 \to s_1 \to \cdots \to s_n$ is a path of $\mathcal{T}$, then s_n is in $\mathcal{C}$. Informally, that means that there is no path in $\mathcal{T}$ such that the first n elements are in $\mathcal{C}$ and the n^{th} is not in $\mathcal{C}$.

We call Good the complement of Bad.

We call n-`GoodInd` the property "Good is n-inductive", and n-`Init` the property "all configurations reachable in at most n steps from a configuration of I are in Good".

Question 5 *Prove that, for any transition system $\mathcal{T}$, and set* Bad*, if there is a n such that both n-`Init` and n-`GoodInd` hold, then the set* Bad *is not reachable in $\mathcal{T}$.*

Question 6 *What formula(s) can you give to a SMT-solver to check if a transition system $\mathcal{T}$ satisfy n-`Init` and n-`GoodInd` for a given n. You will suppose that Δ,* Bad *and I are predicates over configurations you can directly use in the formula, and that you quantify over configurations of $\mathcal{T}$. Do not hesitate to cut your formulæ in sub-formulæ if needed.*

Question 7 *Deduce a variant of Bounded Model Checking which, given a transition system $\mathcal{T}$ and a bound n checks if it is possible to prove the system to be correct using induction.*

Is it closer to the Depth-First Search algorithm or the Global algorithm?

Question 8 *Given a transition system, suppose we have proven the existence of a subset of* Good *which is both n-inductive, and that contains all configurations reachable in at most n steps.*

What can you say of this system? Why?

2 PDR (7pts)

2.1 Minimal clauses

We fix a sequence $v_1, \ldots, v_n$ of boolean variables, i.e. variables ranging over the domain of boolean values $\mathbb{B} = \{\text{true}, \text{false}\}$. We identify valuations of those variables with bivectors in $\mathbb{B}^n$ the expected way. In particular, we associate with a propositional formula ϕ over those variables the set $[\![\phi]\!] \subseteq \mathbb{B}^n$ of bivectors satisfying ϕ. A *literal* is a variable v_i or its negation $\neg v_i$ for some $i \in \{1, \ldots, n\}$. A *clause* is a finite set of *literals*. We introduce the partial order $\sqsubseteq$ over the clauses defined by $c \sqsubseteq c'$ if every literal of c occurs in c'.

Example 1 *With $n = 2$, and $\phi = v_1 \vee \neg v_2$ then $[\![\phi]\!] = \{(1,0),(1,1),(0,0)\}$. Notice that ϕ is a clause and $v_1 \sqsubseteq \phi$.*

Given a set $B \subseteq \mathbb{B}^n$, a clause c is a B-clause if $B \subseteq [\![c]\!]$. A B-clause c is said to be *minimal* if for every B-clause $c' \sqsubseteq c$ we have $c' = c$.

Question 9 *Provide an instance for $n \geq 1$, $B \subseteq \mathbb{B}^n$, and a B-clause c such that there exist two minimal B-clauses $c_1, c_2 \sqsubseteq c$ such that $c_1 \neq c_2$.*

Question 10 *Prove that for every $B \subseteq \mathbb{B}^n$, and for every B-clause c, there exists a minimal B-clause $c' \sqsubseteq c$.*

Question 11 *Provide an algorithm based on a SAT solver that takes as input a propositional formula ϕ denoting a set $B = [\![\phi]\!] \subseteq \mathbb{B}^n$ and a clause c and check if c is a B-clause.*

Question 12 *Provide an algorithm, based on a SAT solver that takes as input a formula ϕ denoting a set $B = [\![\phi]\!] \subseteq \mathbb{B}^n$, a B-clause c where $B = [\![\phi]\!]$ and returns a minimal B-clause $c' \sqsubseteq c$.*

2.2 Frames

A *(finite) transition system* is a triple $(S, \to, s_{init}, s_{bad})$ where S is a non-empty finite set of *states*, $\to$ is a binary relation on S, $s_{init} \in S$ is the *initial state*, and $s_{bad} \in S$ is the *final (bad) state*. An *execution* is a sequence $s_0, \ldots, s_k$ of states such that $s_{j-1} \to s_j$ for every $j \in \{1, \ldots, k\}$, and such that $s_0 = s_{init}$. The number $k \in \mathbb{N}$ is called the *length* of the execution. Such an execution is said to be *good* if $s_j \neq s_{bad}$ for every $j \in \{0, \ldots, k\}$. A *frame* is a finite sequence $F_0 \subseteq \cdots \subseteq F_k$ of subsets of S such that $s_{init} \in F_0$ and such that for every $j \in \{1, \ldots, k\}$ for every $x \in F_{j-1}$ and for every $y \in S$, if $x \to y$ then $y \in F_j$. The number k is called the *length* of the frame. Such a frame is said to be *safe* if $s_{bad} \notin F_k$.

Question 13 *Prove that there exists a safe frame of length k iff every execution of length bounded by k is good.*

A *frame* is said to be *inductive* if there exists $j \in \{1, \ldots, k\}$ such that $F_{j-1} = F_j$.

Question 14 *Provide a bound on the maximal length of a frame that is not inductive. You must justify your result.*

Question 15 *Prove that there exists an inductive safe frame iff every execution is good.*

3 Abstract Interpretation (10pts)

The main objective of this section is to show that abstract interpretation, which was presented using Galois connections in the course, can also be phrased in terms of so-called closure operators.

3.1 Abstract interpretation with closure operators

We start by recalling some basic notions. A *complete lattice* is a partially-ordered set $(L, \sqsubseteq)$ such that every subset $X \subseteq L$ admits a greatest lower bound and a least upper bound. A function $f : L \to L$ on a complete lattice $(L, \sqsubseteq)$ is called

- *monotonic* if $(x \sqsubseteq y \implies f(x) \sqsubseteq f(y))$ for every $x, y \in L$,
- *reductive* if $f(x) \sqsubseteq x$ for every $x \in L$,
- *extensive* if $f(x) \sqsupseteq x$ for every $x \in L$,
- *idempotent* if $f(x) = f(f(x))$ for every $x \in L$.

We also recall the characterization of Galois connections that was given in the course. Given two complete lattices $(C, \sqsubseteq)$ and $(A, \preceq)$, a pair of functions $\alpha : C \to A$ and $\gamma : A \to C$ forms a *Galois connection*, written $(C, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (A, \preceq)$, if, and only if, α and γ are monotonic, $\alpha \circ \gamma$ is reductive, and $\gamma \circ \alpha$ is extensive.

We now introduce the notion of closure operator and show that every Galois connection induces a closure operator. A *closure operator* on a complete lattice $(L, \sqsubseteq)$ is a function $\sigma : L \to L$ that is monotonic, extensive and idempotent.

Question 16 *Assume that $(C, \sqsubseteq)$ and $(A, \preceq)$ are two complete lattices. Prove that, for every Galois connection $(C, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (A, \preceq)$, the function $\gamma \circ \alpha$ is a closure operator.*

The proof that every closure operator induces a Galois connection will require more work. For the remainder of this subsection, we fix a complete lattice $(C, \sqsubseteq)$ and a closure operator σ on C. The greatest lower bound and least upper bound of $(C, \sqsubseteq)$ are written $\sqcap$ and $\sqcup$, respectively.

An element $c \in C$ is called *closed* if $\sigma(c) = c$. Let A denote the set of all closed elements of C. Observe that, by idempotence, we have $A = \{\sigma(c) \mid c \in C\}$. Let $\preceq$ denote the restriction to A of the partial order $\sqsubseteq$ on C. Formally, $\preceq$ is the intersection of $\sqsubseteq$ and $(A \times A)$. Observe that $\preceq$ is a partial order on A. Our first objective is to show that the partially-ordered set $(A, \preceq)$ is a complete lattice.

Question 17 *Let $X \subseteq A$. Prove that $\sigma(\sqcup X)$ is the least upper bound of X in $(A, \preceq)$.*

Question 18 *Let $X \subseteq A$. Prove that $\sqcap X$ is closed (i.e., $\sigma(\sqcap X) = \sqcap X$). Deduce that $\sqcap X$ is the greatest lower bound of X in $(A, \preceq)$.*

We have shown that $(A, \preceq)$ is a complete lattice with greatest lower bound $\wedge$ and least upper bound $\vee$ verifying $\wedge X = \sqcap X$ and $\vee X = \sigma(\sqcup X)$, for every subset $X \subseteq A$.

Question 19 *Prove that $(C, \sqsubseteq) \underset{\sigma}{\overset{id}{\leftrightarrows}} (A, \preceq)$ is a Galois connection.*[1]

[1] The function $id : A \to C$ is the *identity* function (i.e., $id(a) = a$ for all $a \in A$).

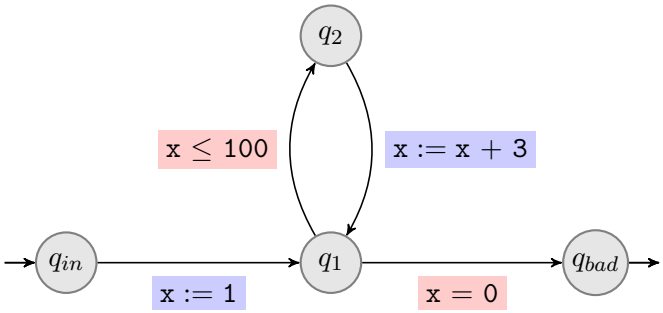


Figure 1: Control-flow automaton of Question 22.

3.2 Application to program analysis

We now apply the results of the previous subsection to program analysis by abstract interpretation. Assume a fixed parameter $t \in \mathbb{N}$ which shall serve as a threshold. Consider the function $\theta : \mathcal{P}(\mathbb{Z}) \to \mathcal{P}(\mathbb{Z})$ defined by

$$\theta(S) \;=\; \begin{cases} S & \text{if S is finite and } |S| \leq t \\ \mathbb{Z} & \text{otherwise} \end{cases}$$

In the above definition, $|S|$ denotes the cardinal of S. Note that $\theta(\emptyset) = \emptyset$.

Example 2 *Assuming that the threshold t is 3, we have firstly $\theta(\{2\}) = \{2\}$, secondly $\theta(\{-1, 2, 5\}) = \{-1, 2, 5\}$, and thirdly $\theta(\{-1, 2, 5, 11\}) = \mathbb{Z}$.*

Question 20 *Prove that θ is a closure operator on the complete lattice $(\mathcal{P}(\mathbb{Z}), \subseteq)$.*

As in subsection 3.1, we let $A = \{\theta(S) \mid S \subseteq \mathbb{Z}\}$ denote the set of closed elements of $\mathcal{P}(\mathbb{Z})$, and we write $\preceq$ the restriction to A of the partial order $\subseteq$ on $\mathcal{P}(\mathbb{Z})$. The results of subsection 3.1 ensure that $(A, \preceq)$ is a complete lattice and that $(\mathcal{P}(\mathbb{Z}), \subseteq) \underset{\theta}{\overset{id}{\leftrightarrows}} (A, \preceq)$ is a Galois connection.

Question 21 *Does the abstract lattice $(A, \preceq)$ satisfy the ascending chain condition? Justify your answer.*

Consider the control-flow automaton depicted in Figure 1. This control-flow automaton has a single variable `x`, which ranges over integers. The initial location is q_{in} and the bad location is q_{bad}. Recall that '`:=`' denotes an assignment and that '`=`' denotes a condition. Like in the course, an analysis will be called *successful* when the abstract value obtained for q_{bad} has an empty concretization. Round-robin iteration shall use the following order on locations: $q_{in}, q_1, q_2, q_{bad}$.

Question 22 *Apply the round-robin algorithm to analyze the control-flow automaton depicted in Figure 1 using the Galois connection $(\mathcal{P}(\mathbb{Z}), \subseteq) \underset{\theta}{\overset{id}{\leftrightarrows}} (A, \preceq)$, assuming that the threshold t is 3. Is the analysis successful?*