Software Verification

Monday, January 13th 2020, 3 hours

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

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

1 Bounded Model-Checking (14pts)

This section will browse a few concepts that have been seen in the first part of the course (Bounded Model-Checking).

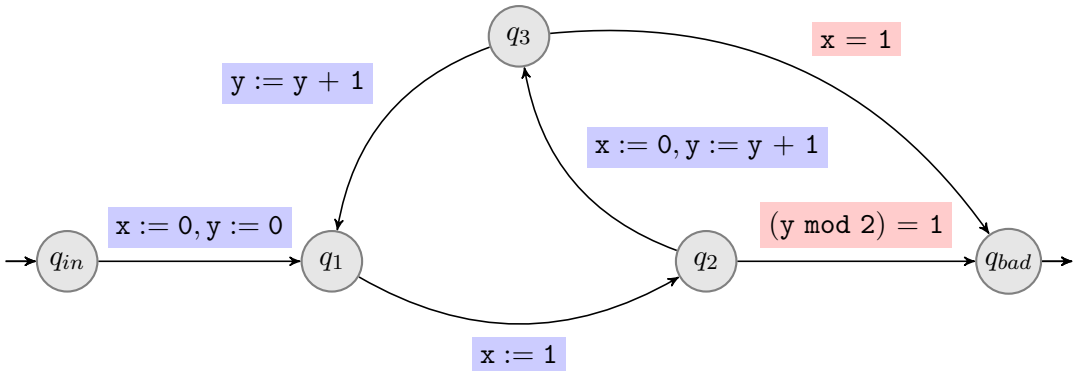
1.1 Principles

We start with some questions to check your understandings of the Bounded Model-Checking algorithms. We recall that given a transition system $\mathcal{T}$, the Bounded Model-Checking at depth n explore all executions of size at most n and check if one of them reach an undesirable state (such an execution is called *faulty*).

Question 1 *A Bounded Model-Checking search can return three answers: Counter-example found, No counter-example found (exhaustive), No counter-example found (non-exhaustive).*

For each possible answer, explain what the answer means about the presence of a faulty execution, its size, and the size of all executions of the model.

Question 2 *We have seen in course two algorithm : Depth-First Search, and Global. Recall briefly how the executions are explored in the two algorithms and what guarantee they give on the faulty execution they return if they return one.*



Question 3 *Apply the bounded model-checking algorithm 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 unsatisfiable ones, and give the result of the algorithm.*

Is there a bound on which the algorithm would give a different answer? Justify.

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".

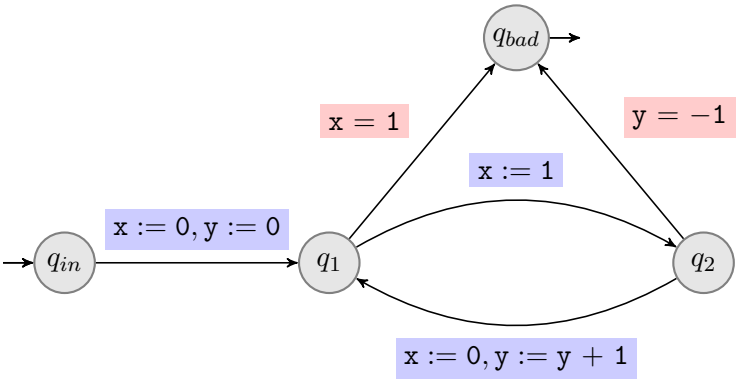
Question 4 *Prove that, for any transition system $\mathcal{T}$, and set* Bad*, if there is a n such that both n-`Init` and n-`GoodInd` hold, then the set* Bad *is not reachable in $\mathcal{T}$.*

Question 5 *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 6 *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 7 *Prove that the system of question 3 is correct using the procedure described earlier, supposing* Bad *is the set of all configurations containing* q_{bad}.



Question 8 *Will the process described earlier prove the previous system, where* Bad *is the set of configurations containing* q_{bad}*, to be correct? Why (we expect at least a detailed explanation, or a proof)?*

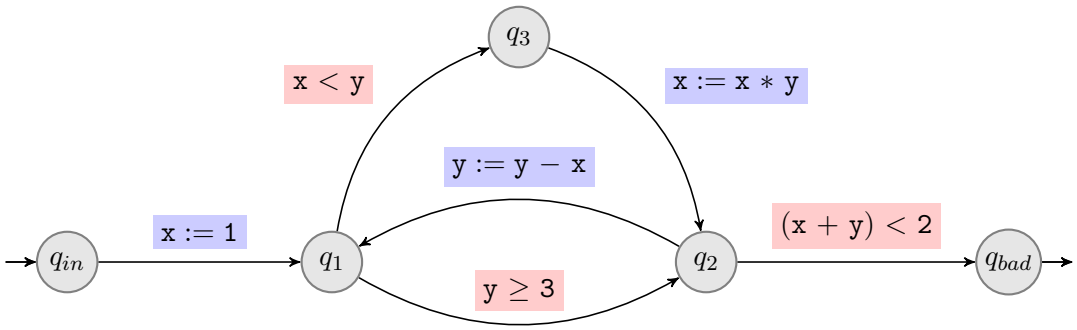
Question 9 *Can you find a subset of* Good *which would satisfy* n-`Init` *and would be* n*-inductive for some* n*? Does that suffices to prove the system to be correct?*

2 Abstract Interpretation (6pts)

This section deals with abstract interpretation, and is divided into two independent subsections. The first subsection applies range analysis on an example and the second subsection proves a property (mentioned in the course) on least fixpoint approximation.

2.1 Range analysis

We perform range analysis — this analysis was presented in the course — on the control-flow automaton depicted below. This control-flow automaton has two variables `x` and `y`, both ranging over integers. The initial location is q_{in} and the bad location is q_{bad}. Recall that range analysis uses the *abstract domain of intervals*.



Like in the course, an analysis will be called *successful* when the abstract value obtained for q_{bad} is $\bot$. Round-robin iteration shall use the following order on locations: $q_{in}, q_1, q_2, q_3, q_{bad}$.

Question 10 *Apply the round-robin algorithm with widening. Do* not *use narrowing. Is the analysis successful?*

Question 11 *Starting from the result of the previous question, perform a decreasing iteration with narrowing. Is the analysis successful?*

2.2 Least fixpoint approximation

We first recall some notions and notations from the course. For the remainder of this subsection, we consider a complete lattice $(L, \sqsubseteq)$, with greatest lower bound written $\sqcap$ and least upper bound written $\sqcup$. A function $f : L \to L$ is called *non-decreasing* when it satisfies $(x \sqsubseteq y \implies f(x) \sqsubseteq f(y))$ for every $x, y \in L$. The *least fixpoint* of a function $f : L \to L$, if it exists, is denoted by $\mathrm{lfp}(f)$.

By the theorem of Knaster-Tarski, every non-decreasing function $f : L \to L$ on a complete lattice $(L, \sqsubseteq)$ has a least fixpoint that satisfies $\mathrm{lfp}(f) = \bigsqcap\{x \in L \mid f(x) \sqsubseteq x\}$.

Question 12 *Prove that for every subsets $X \subseteq L$ and $Y \subseteq L$,*

$$\textit{if} \quad X \subseteq Y \quad \textit{then} \quad (\bigsqcap X) \sqsupseteq (\bigsqcap Y) \ .$$

Question 13 *Prove that for every non-decreasing functions $f : L \to L$ and $g : L \to L$,*

$$\textit{if} \quad \forall x \in L : f(x) \sqsubseteq g(x) \quad \textit{then} \quad \mathrm{lfp}(f) \sqsubseteq \mathrm{lfp}(g) \ .$$

3 TWO-SAT Interpolation (10pts)

A TWO-SAT formula is a SAT formula given as a conjunction of clauses that contain only two literals. For instance $(x_1 \vee \neg x_2)$ is a TWO-SAT with literals x_1 and $\neg x_2$, and $(x_1 \vee \neg x_2 \vee x_3)$ is not a TWO-SAT since this last clause involves three literals. In this section, if l is a literal of the form $\neg x$ for some variable x, then we identify $\neg l$ with x. Intuitively, $\neg\neg x = x$.

Question 14 *Provide a resolution tree proving that the TWO-SAT formula $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (x_2 \vee \neg x_3)$ is unsat.*

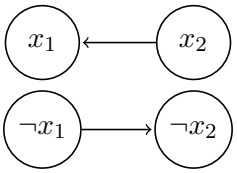
Question 15 *From the previous resolution tree, provide an interpolant for the decomposition of the formula into the left part $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$ and the right part $(x_2 \vee x_3) \wedge (x_2 \vee \neg x_3)$.*

We associate to a TWO-SAT formula ϕ the directed graph $G_\phi = (V_\phi, \to_\phi)$ defined as follows:

- V_ϕ is the set of literals x or $\neg x$ where x ranges over the variables occuring in ϕ, and

- $\rightarrow_\phi$ is a binary relation over V_ϕ defined by $l \rightarrow_\phi l'$ if $\neg l \vee l'$ is a clause of ϕ.

Example 1 *The directed graph associated to the formula* $(x_1 \vee \neg x_2)$ *is the following one:*



Question 16 *Draw the directed graph associated to* $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (x_2 \vee \neg x_3)$.

We introduce the binary relation $\xrightarrow{*}_\phi$ over the literals in V_ϕ defined by $l \xrightarrow{*}_\phi l'$ if there exists a directed path in G_ϕ from l to l'.

Question 17 *Prove that if* $l \xrightarrow{*}_\phi l'$ *then* ϕ *implies* $\neg l \vee l'$.

Question 18 *Prove that if a cycle of* G_ϕ *contains a variable and its negation then* ϕ *is unsat.*

Now, let ϕ_L and ϕ_R be two TWO-SAT formulas, and let us introduce $\phi = \phi_L \wedge \phi_R$. Variables that occur both in ϕ_L and ϕ_R are called *global variables*. A literal x or $\neg x$ where x is a global variable is called a *global literal*. We assume that there exists a cycle in G_ϕ that contains a variable and its negation. We introduce the TWO-SAT formula ϕ_I obtained as the conjunction of the clauses $\neg l \vee l'$ for every pair (l, l') of global literals such that $l \xrightarrow{*}_{\phi_L} l'$.

Question 19 *Prove that* ϕ_L *implies* ϕ_I.

Question 20 *Prove that* $\phi_I \wedge \phi_R$ *is unsat.*

Question 21 *Deduce that* ϕ_I *is an interpolant for* (ϕ_L, ϕ_R).

Question 22 *The formula* ϕ_I *can be quadratic in size in the worst case. Provide some intuitions on how to modify the construction of* ϕ_I *in order to obtain formulas of linear sizes.*