DD2452 Formal Methods

Examination Problems
with partial solutions
12 March 2008

Dilian Gurov
KTH CSC
tel: 08-790 8198

---

1. Consider the following program Euclid for computing the greatest common divisor $gcd(m, n)$ of two positive integers m and n: 4p

```
while (x != y) {
    if (x < y) {
        y = y - x;
    } else {
        x = x - y;
    }
}
```

   (a) Specify the program for *total correctness* by means of a pre- and post-condition. The specification should meaningfully express the purpose of the program without knowing its text.
   Solution: Pre-condition $x \geq 1 \wedge y \geq 1 \wedge x = x_0 \wedge y = y_0$, post-condition $x = gcd(x_0, y_0)$.

   (b) Verify that the program meets its specification. Present the proof as a proof tableau. Clearly identify the *invariant* and the *variant* of the while loop.
   Solution: With invariant $x \geq 1 \wedge y \geq 1 \wedge gcd(x, y) = gcd(x_0, y_0)$ and variant $x + y$ it is straightforward to complete the annotation.

   (c) Identify and justify the resulting *proof obligations*.
   Solution: Straightforward; justifications use simple properties of $gcd(m, n)$, most importantly:
   $m \geq 1 \wedge n \geq 1 \wedge m < n \;\rightarrow\; gcd(m, n) = gcd(m, n - m)$ and
   $m \geq 1 \wedge n \geq 1 \wedge m = n \;\rightarrow\; gcd(m, n) = m$.

---

2. Let $Atoms = \{entry, active, request, response\}$ be a set of atomic propositions, and let $\mathcal{M} = (S, \rightarrow, L)$ be a model over $Atoms$ defined by states $S = \{s_0, s_1, s_2, s_3\}$, transitions $\rightarrow = \{(s_0, s_1), (s_1, s_0), (s_1, s_2), (s_2, s_3), (s_3, s_1)\}$ and labelling function $L = \{(s_0, \{entry\}), (s_1, \{active\}), (s_2, \{active, request\}), (s_3, \{active, response\})\}$. This could be seen as a rudimentory model of a bank teller machine. For every property listed below, suggest a formalisation in LTL (or argue why there cannot be such), and determine its validity on state s_0 by referring to the formal semantics of LTL formulas (see handouts). For formulas that do not hold, provide a *counter-example* by means of an infinite path not satisfying the formula. 4p

   (a) infinitely often *active*;
   Solution: $\mathsf{GF}\ active$, valid on s_0.

   (b) infinitely often *entry*;
   Solution: $\mathsf{GF}\ entry$, not valid on s_0, counter-example: $s_0 s_1 s_2 s_3 s_1 s_2 s_3 \ldots$

   (c) one can always reach *entry*;
   Solution: this property is not expressible in LTL, since it quantifies existentially over paths.

   (d) every *request* is eventually followed by a *response*;
   Solution: $\mathsf{G}\ (request \rightarrow \mathsf{F}\ response)$, valid on s_0.

(e) if from some point on never *request*, then infinitely often *entry*;

Solution: $\mathsf{FG}\ \neg request \ \rightarrow\ \mathsf{GF}\ entry$, or $\mathsf{G}\ (\mathsf{G}\ \neg request \ \rightarrow\ \mathsf{F}\ entry)$, valid on s_0.

(f) *response* only if *request* some time before.

Solution: $(\mathsf{G}\ \neg response) \vee (\neg response\ \mathsf{U}\ request)$, valid on s_0.

---

3. Consider the following concurrent program CFact for computing the factorial $m!$ of a positive integer m: 9p

```
y1 = 1;
y2 = 1;
z = 0;
cobegin
   while (z < x − 1) {
      z = z + 1;
      y1 = y1 ∗ z;
   }
|| while (x > z + 1) {
      y2 = y2 ∗ x;
      x = x − 1;
   }
coend;
if (z < x) {
   z = z + 1;
   y1 = y1 ∗ z;
} else {
   skip;
};
y = y1 ∗ y2;
```

The *idea* of the algorithm is that the factorial of a number can be computed independently (and thus concurrently) "from below" and "from above", until the two limits (here z and x) meet. However, the limits are not guaranteed to meet exactly, so an additional test is needed at the end. The final value is then the product of the two partial results (here $y1$ and $y2$).

Now, verify that the program meets the specification:

$$(\!|x = x_0 \wedge x > 0|\!)\ \text{CFact}\ (\!|y = x_0!|\!)$$

(a) Present the proof as a proof tableau.

Solution: We use the notation $mul(m, n)$ defined as $\prod_{m\le i\le n} i$ when $m \le n$ and as 1 otherwise. Appropriate assertions for the control points immediately before and after the **cobegin–coend** statement are, respectively:
$0 \le z \wedge z \le x \wedge x \le x_0 \wedge y1 = mul(1, z) \wedge y2 = mul(x + 1, x_0)$ and
$0 \le z \wedge (z = x - 1 \vee z = x) \wedge x \le x_0 \wedge y1 = mul(1, z) \wedge y2 = mul(x + 1, x_0)$
We can now apply the *Owicki-Gries* rule for **cobegin–coend** with pre- and post-condition to the first parallel branch respectively:
$0 \le z \wedge z \le x \wedge y1 = mul(1, z)$ and
$0 \le z \wedge (z = x - 1 \vee z = x) \wedge y1 = mul(1, z)$
and with pre- and post-condition to the second parallel branch respectively:
$z \le x \wedge x \le x_0 \wedge y2 = mul(x + 1, x_0)$ and

$(z = x - 1 \vee z = x) \wedge x \leq x_0 \wedge y2 = mul(x + 1, x_0)$
Notice that the two pre-conditions are also suitable loop invariants. Completing the annotation is then straightforward.

(b) Identify and justify the resulting proof obligations.

Solution: Straightforward; justifications use some simple properties of $mul(m, n)$ and factorial:

$$\begin{aligned} mul(1, m) * mul(m + 1, n) &= n! && \text{whenever } 0 \leq m \leq n \\ mul(1, m + 1) &= mul(1, m) * (m + 1) && \text{whenever } 0 \leq m \\ mul(m, n) &= m * mul(m + 1, n) && \text{whenever } 0 \leq m \leq n \end{aligned}$$

(c) Identify all *critical formulas*, and show one case of non-interference: pick a critical formula from the first parallel command and the assignment statement $x = x - 1$ from the second parallel command, and show that the statement does not interfere with the formula.

Solution: There are 4 critical formulas in each parallel branch. Pick for instance the first critical formula $0 \leq z \wedge z \leq x \wedge y1 = mul(1, z)$ from the first parallel command. We need to proof validity of the Hoare triple:

$(\!|0 \leq z \wedge z \leq x \wedge y1 = mul(1, z) \wedge z \leq x - 1 \wedge x - 1 \leq x_0 \wedge y2 = mul((x - 1) + 1, x_0)|\!)$
$x = x - 1$
$(\!|0 \leq z \wedge z \leq x \wedge y1 = mul(1, z)|\!)$

which is straightforward.

---

4. Consider the CCS processes P and Q defined by: 7p

$$\begin{aligned} S &\triangleq p.\overline{v}.S \\ A &\triangleq \overline{p}.(v.A + a.c.\mathbf{0}) \\ B &\triangleq \overline{p}.(v.B + b.d.\mathbf{0}) \\ P &\triangleq (A \mid S \mid B)\backslash\{p, v\} \\ Q &\triangleq a.c.\mathbf{0} + b.d.\mathbf{0} \end{aligned}$$

(a) Derive formally the immediate transitions of process P by referring explicitly to the CCS transition rules (see handouts). Don't forget to annotate your derivation(s) with rule names.

Solution: There are two immediate transitions of process P, namely:
$P \xrightarrow{\tau} ((v.A + a.c.\mathbf{0}) \mid \overline{v}.S \mid B)\backslash\{p, v\}$ and
$P \xrightarrow{\tau} (A \mid \overline{v}.S \mid (v.B + b.d.\mathbf{0}))\backslash\{p, v\}$ (derivations omitted).

(b) Explore the whole state space of P, and draw the graph of the labelled transition system induced by P.

Solution: The state space of P contains 8 process terms (omitted).

(c) The execution of process P will not reach itself again, but rather its defining term $(A \mid S \mid B)\backslash\{p, v\}$. But conceptually, we would like to identify the latter term with the initial state (that is, process P). Suggest a meaningful rule that remedies this and allows P to be re-visited.

Solution: We could add the rule:

$$\mathsf{Def2}\ \frac{E \xrightarrow{\alpha} F}{E \xrightarrow{\alpha} A}\ A \stackrel{\mathsf{def}}{=} F$$

(d) Draw the graph of the labelled transition system induced by process Q. Prove $P \approx Q$ by exposing a suitable relation R for which you show that it is a weak bisimulation.

(e) Process P can be seen as providing a means of achieving the effect of sequential choice "+" between two concurrent behaviours, here represented by processes $a.c.\mathbf{0}$ and $b.d.\mathbf{0}$, by means of a semafor S. Do you see any drawbacks of such a solution, as compared to sequential choice? Could there potentially be a better solution, if the task is to synchronize concurrent behaviours? Give an intuitive justification for your answers.

Solution: This form of choice suffers from the drawback of containing livelock behaviours. Even worse, it contains a livelock behaviour that never offers action a to the environment, and another one that never offers action b. On the other hand, there is no livelock-free solution to the problem. Still, it could be a better solution to offer actions a and b alternatingly:

$$\begin{aligned}
S_2 &\triangleq p_A.\overline{v_A}.p_B.\overline{v_B}.S_2 \\
A &\triangleq \overline{p_A}.(v_A.A + a.c.\mathbf{0}) \\
B &\triangleq \overline{p_B}.(v_B.B + b.d.\mathbf{0}) \\
P &\triangleq (A \mid S_2 \mid B)\backslash\{p_A, v_A, p_B, v_B\}
\end{aligned}$$

---

5. Consider the labelled transition system $\mathcal{T} = (\mathcal{S}, Act, \rightarrow)$ with states $\mathcal{S} = \{s_0, s_1\}$, actions $Act = \{a, b\}$, and transition relation $\rightarrow = \{(s_0, a, s_0), (s_0, b, s_1), (s_1, b, s_0)\}$, and consider the modal μ-calculus formula $\Phi = \mu Z.\ [a]\,\mathbf{ff} \vee (\langle b\rangle\,\mathbf{tt} \wedge [b]\,Z)$. (See handouts.) $\boxed{4\text{p}}$

   (a) Compute the first three fixed–point approximants of Φ. Simplify these as much as possible.

   Solution: The first three fixed–point approximants of Φ are:

$$\begin{aligned}
\mu Z^1.\ [a]\,\mathbf{ff} \vee (\langle b\rangle\,\mathbf{tt} \wedge [b]\,Z) &= [a]\,\mathbf{ff} \vee (\langle b\rangle\,\mathbf{tt} \wedge [b]\,\mathbf{ff}) \\
&= [a]\,\mathbf{ff} \\
\mu Z^2.\ [a]\,\mathbf{ff} \vee (\langle b\rangle\,\mathbf{tt} \wedge [b]\,Z) &= [a]\,\mathbf{ff} \vee (\langle b\rangle\,\mathbf{tt} \wedge [b]\,[a]\,\mathbf{ff}) \\
\mu Z^3.\ [a]\,\mathbf{ff} \vee (\langle b\rangle\,\mathbf{tt} \wedge [b]\,Z) &= [a]\,\mathbf{ff} \vee (\langle b\rangle\,\mathbf{tt} \wedge [b]\,([a]\,\mathbf{ff} \vee (\langle b\rangle\,\mathbf{tt} \wedge [b]\,[a]\,\mathbf{ff})))
\end{aligned}$$

   (b) Based on the formal semantics of the modal μ-calculus, explain the intuitive meaning of the formula.

   Solution: The formula expresses the property "on all b-paths, eventually a is not enabled".

   (c) Use the proof system for the modal μ-calculus to prove $s_0 \vdash^{\mathcal{T}} \Phi$. In your proof, clearly identify the rule applied at each step.

---

6. Prove the following implication on LTL formulas by referring to the formal semantics of LTL formulas (see handouts): $\boxed{2\text{p}}$

$$\mathsf{GF}p \ \wedge\ \mathsf{FG}q \ \rightarrow\ \mathsf{FG}\ (\mathsf{F}p \ \wedge\ q)$$

Solution: (Sketch) We assume $\pi \models^{\mathcal{M}} \mathsf{GF}p \ \wedge\ \mathsf{FG}q$ for an arbitrary path π of an arbitrary model $\mathcal{M}$, and then show $\pi \models^{\mathcal{M}} \mathsf{FG}\ (\mathsf{F}p \ \wedge\ q)$ by referring to the formal semantics of LTL formulas and by simple logic.
