DD2452 Formal Methods

Examination Problems
12 March 2008, 14:00 - 19:00

Dilian Gurov
KTH CSC
tel: 08-790 8198

Give solutions in English or Swedish, each problem beginning on a new sheet. Write your name on all sheets. The maximal number of points is given for each problem. The course literature, the handouts, your own lecture notes, as well as reference material are admissible at the exam.

---

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

```
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.

   (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.

   (c) Identify and justify the resulting *proof obligations*.

---

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. $\boxed{4p}$

   (a) infinitely often *active*;

   (b) infinitely often *entry*;

   (c) one can always reach *entry*;

   (d) every *request* is eventually followed by a *response*;

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

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

Please Turn Over

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

```
$y1 = 1;$
$y2 = 1;$
$z = 0;$
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:

$$(\!| x = x_0 \land x > 0 |\!) \text{ CFACT } (\!| y = x_0! |\!)$$

(a) Present the proof as a proof tableau.

(b) Identify and justify the resulting proof obligations.

(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.

Hint1: In your proof, you can use the notation $mul(m, n)$ defined as $\prod_{m \leq i \leq n} i$ when $m \leq n$ and as 1 otherwise. Notice that the following equations hold:

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

Hint2: First, try to find appropriate assertions for the control points immediately before and after the **cobegin–coend** statement, capturing the (intermediate) values of $y1$ and $y2$, and how the limits z and x relate to each other and to the original bounds (0 and x_0, respectively). Then, apply the *Owicki-Gries* rule for **cobegin–coend** in a suitable fashion.

Please Turn Over

4. Consider the CCS processes P and Q defined by: $\boxed{7\text{p}}$

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

(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.

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

(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.

(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.

---

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.

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

(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): $\boxed{2\text{p}}$

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

Hint: Assume $\pi \models^{\mathcal{M}} \mathsf{GF}p \ \wedge\ \mathsf{FG}q$ for an arbitrary path π of an arbitrary model $\mathcal{M}$, and show $\pi \models^{\mathcal{M}} \mathsf{FG}\ (\mathsf{F}p \ \wedge\ q)$.

Good luck!