GLOBAN'08 final exam

September 26, 2008

Name and family name: .....
.....

Each square aside of an answer should be filled with either Y (in case the answer is correct) or N (in case the answer is wrong). Here is an example:

0. Each prime number is

[Y] **i.** divisible only by 1 and itself,

[N] **ii.** divisible by 7,

[Y] **iii.** larger than -2.

Any combination of Y and N is possible in this test.

1. In an infinite Markov chain, the question whether a given set of states is reachable with positive probability, can be answered without considering the probabilities in the Markov chain.

   [N] **i.** True, since state s is reachable with probability > 0 iff there is a path to s.

   [Y] **ii.** False, since the probability to visit a state once may be positive, while visiting it infinitely often has likelihood zero.

   [N] **iii.** True, since the probability to eventually enter a BSCC is positive.

2. Any memoryless scheduler for a finite MDP M

   [Y] **i.** Always selects the same action in state s.

   [N] **ii.** Yields a maximal reachability probability in M that is at least as good as for any history-dependent scheduler.

   [Y] **iii.** Induces a finite-state Markov chain.

3. You download a program from an untrusted source. The program operates on variable h (containing high-secrecy information) and variable l (containing low-secrecy information). The value of l is available to the attacker before and after the program's execution. The program's timing behaviour may be observable by the attacker.

   Mark with Y the programs which are "pathologically" insecure, i.e., cannot be rewritten as secure programs, and with N other programs:

[N] **i.** `if` $h = l$ `then` $h := l$ `else` $h := -l$

[Y] **ii.** `if` $h = 0$ `then` $l := 0$ `else` $l := 1$

[N] **iii.** `if` $h = 0$ `then` (`while` $h < MaxInteger$ `do` $h := h + 1$)

4. Assume h and h' are high variables and l and l' are low ones. Assume the declassification policy allows leaking the initial value of h at the time of declassification at the earliest. Mark with Y the programs below which are instances of information "laundering" through declassification, mark with N other programs.

   [N] **i.** $l := h$; $l :=$`declassify`(h)

   [Y] **ii.** $h := h'$; $l :=$`declassify`(h)

   [N] **iii.** $h := h'$; $h := 0$; $l :=$`declassify`(h); $l := h'$

5. Recall *termination-insensitive* noninterference, which stipulates that if two runs of a program starting with two low-equivalent memories terminate then the resulting memories must be also low-equivalent (i.e., agree on the low variables). By the same token, consider declassification insensitive security: if two runs of a program starting with two low-equivalent memories terminate without encountering declassification statements, then the resulting memories must be also low-equivalent.

   Does this policy satisfy the following principles?

   [Y] **i.** Semantic consistency.

   [Y] **ii.** Monotonicity of release.

   [N] **iii.** Non-occlusion.

6. Consider the following class definition in FJEU

```
class C {
  C tl;
  C shallowcopy() {
     let res = new C in
     let _ = res.tl <- this.tl in res
  }
  C deepcopy() {
     let res = new C in
     if (this.tl == null)
       then res
       else let _ = res.tl <- this.tl.deepcopy() in res
  }
  C loopcopy() {
     if (this.tl == null)
       then this
       else let _ = this.tl <- this.tl.loopcopy() in this
  }
```

Which of the following is the *best* possible typing for each of the three methods in the type system for constant heap space.

N i.
`shallowcopy` : 1
`deepcopy` : 1
`loopcopy` : 0

N ii.
`shallowcopy` : 1
`deepcopy` : ∞
`loopcopy` : 1

Y iii.
`shallowcopy` : 1
`deepcopy` : ∞
`loopcopy` : 0

7. Consider the following class definition in FJEU

```
class C {
  void test(int a, int b) {
    let x = new SMS in
    let _ = auth(a) in
    loop(x,b)
  }
  void loop(SMS x, int b) {
    if (b > 0)
      then {
        send();loop(x,b-1)
      } else {
        ()
      }
  }
}
```

Which of the following are valid typings in the type system for block booking. The parameters a, b range over $\mathbb{D}$.

N i.
`test`$([0,a],[b,\infty]) : (0,0)$ where $a \geq b$
`loop`$([0,b]) : (0,0)$

Y ii.
`test`$([a,\infty],[0,b]) : (0,0)$ where $a \geq b$
`loop`$([0,b]) : (b,0)$

N iii.
`test`$([a,\infty],[b,\infty]) : (0,0)$ where $a \geq b$
`loop`$([0,b]) : (b,0)$

8. Consider the following FJEU class definitions

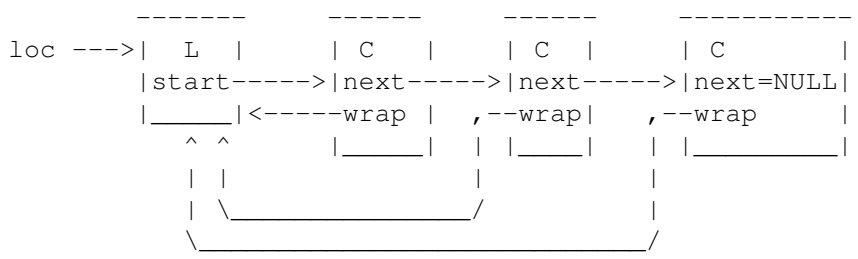
```
class L {
  C start;
}
class C {
  C next;
  L wrap;
}
```

and the following RAJA refinement:

```
views a, b;
class L implements a, b {
  a : pot = 0;
  b : pot = 0;
  a : C<a,a> start;
  b : C<b,b> start;
}

class C implements a, b {
  a : pot = 1;
  b : pot = 0;
  a : C<a,a> next;
  b : C<b,b> next;
  a : L<b,b> wrap;
  b : L<b,b> wrap;
}
```

Consider the heap represented by the following UML object diagram:



What is the potential $\phi_{\texttt{L<a>},h}(\texttt{loc})$?

N **i.** It is equal to 3.

N **ii.** It is equal to 4.

Y **iii.** It is equal to ∞.

9. A Flow Logic specification determines whether or not an analysis estimate correctly describes a given program. The Flow Logic specification may be recursively defined (for example due to recursive procedures or named process constants). When this is the case we intend to use:

[N] **i.** The inductive interpretation — corresponding to a least fixed point interpretation of the Flow Logic specification.

[Y] **ii.** The coinductive interpretation — corresponding to a greatest fixed point interpretation of the Flow Logic specification.

[N] **iii.** It does not matter: any interpretation satisfying the "iff" clauses will be equally good — corresponding to any fixed point interpretation of the Flow Logic Specification being equally good.

10. Given a Flow Logic specification and a program there may be many analysis estimates that correctly describe the program. When this is the case we intend to use:

    [Y] **i.** The least such analysis estimate.

    [N] **ii.** The greatest such analysis estimate.

    [N] **iii.** It does not matter: any analysis estimate that correctly describes the program will be equally good.

11. Consider an implementable Flow Logic specification in the sense of the notes. Given a program of size n it can be expanded to a formula of size $O(n)$. We may consider the time complexity $\mathsf{Tcheck}(n)$ of determining whether a given analysis estimate correctly describes a given program of size n. Similarly we may consider the time estimate $\mathsf{Tcompute}(n)$ for calculating the best analysis estimate that correctly describes a given program of size n. In all cases we are focusing on deterministic time complexity. Consider the ratio $\mathsf{Tgap}(n) = \mathsf{Tcompute}(n)/\mathsf{Tcheck}(n)$ indicating how much longer time it may take to compute the best analysis estimate as opposed to determining whether a given analysis estimate correctly describes the program. In general, the best upper bound of $\mathsf{Tgap}(n)$ will be:

    [N] **i.** An exponential function (in n).

    [N] **ii.** A polynomial (in n).

    [Y] **iii.** A constant.

12. If we try to apply rule <kill> to node $p_2 : P$ below, which are the possible outcomes?

    [Y] **i.** The application is blocked.

    [N] **ii.** The rule is applied, and p_2 is deleted from graph G.

    [Y] **iii.** The rule is applied, and edges r and l are deleted together with p_2 from graph G,

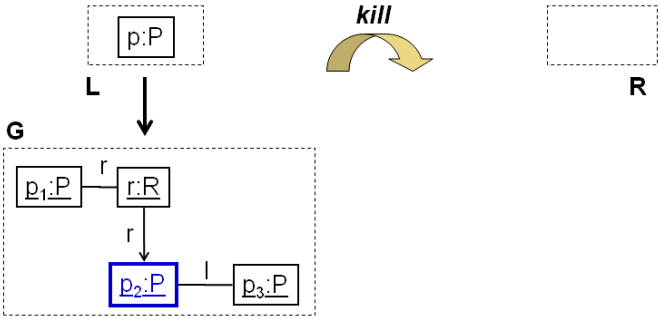


Figure for question 12

13. A Petri net can be seen as a graph transformation system with

    N **i.** places as nodes and transitions as edges,

    Y **ii.** tokens as nodes and transitions as rules,

    Y **iii.** places, transitions and tokens as nodes.

14. Where are there critical pairs between the rules below?

    Y **i.** Between <kill> and <moveGhost>.

    Y **ii.** Between <kill> and <movePacman>.

    N **iii.** Between <movePacman> and <moveGhost>.

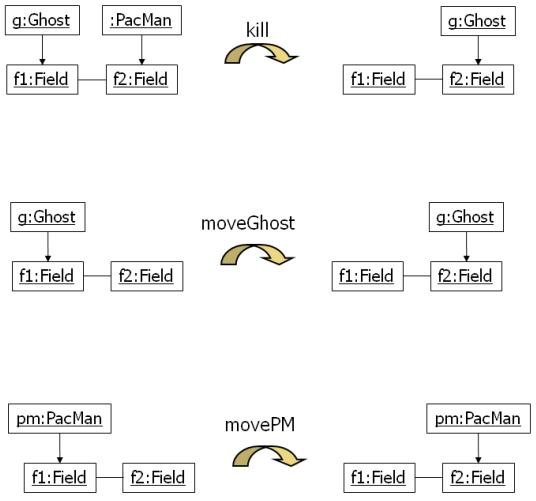


Figure for question 14

15. Assert if the given statement is true (Y) or false (N).

   [N] **i.** The first commercial object-oriented virtual machine was IBM's System/370, released in 1972.

   [N] **ii.** Virtualization of the x86 architecture relies on setting the processor to trap whenever a user mode instruction accesses privileged state.

   [Y] **iii.** A refinement type is a type coupled with a constraint that may be checked either statically or dynamically.

16. The following are instances of the reduction relation in RCF:

   [N] **i.** $a!\textbf{true} \Rsh a!\textbf{false} \rightarrow a!\textbf{false} \Rsh a!\textbf{true}$

   [Y] **ii.** $a!\textbf{true} \Rsh a? \rightarrow \textbf{true}$

   [N] **iii.** $a? \Rsh a!\textbf{true} \rightarrow \textbf{true}$

17. The following types of RCF are always of kind public, that is, functions belonging to these types may safely be passed to the opponent.

   [N] **i.** $\{x : \textbf{int} \mid \textbf{Odd}(x)\} \rightarrow \{y : \textbf{int} \mid \textbf{Even}(y)\}$

   [Y] **ii.** $\textbf{int} \rightarrow \{y : \textbf{int} \mid \textbf{Even}(y)\}$

   [N] **iii.** $(\textbf{int} \rightarrow \{x : \textbf{int} \mid \textbf{Odd}(x)\}) \rightarrow \textbf{int}$

18. The π-process $(\nu a\ \overline{b}a.P)|b(x).Q$

   [N] **i.** is stuck,

   [Y] **ii.** evolves in one step to $\nu a(P|Q')$, where $Q' = Q[a/x]$,

   [N] **iii.** evolves in one step to $P|(\nu a Q')$, where $Q' = Q[a/x]$.

19. Agree (Y) or disagree (N) with the answers. Can a CASPIS *service* interact with two *clients*?

   [Y] **i.** Yes.

   [N] **ii.** Yes, the service interacts with the clients by establishing a single session.

   [Y] **iii.** Yes, the service interacts with the clients through nested sessions.

20. MARCASPIS Process $s^5.P|s^7.Q|\overline{s}^4.Q$

   [Y] **i.** evolves with rate $\frac{5}{3}$ to $\nu r(r \triangleright P|s^7.Q|r \triangleright R)$,

   [Y] **ii.** evolves with rate $\frac{7}{3}$ to $s^5.P|\nu r(r \triangleright Q|r \triangleright R)$,

   [N] **iii.** evolves with rate 4 to $\nu r(r \triangleright (P|Q)|r \triangleright R)$.