U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO | 4º ANO
EIC0039 | MÉTODOS FORMAIS EM ENGENHARIA DE SOFTWARE | 2014-15 – 1º SEMESTRE

Com consulta restrita ao "Alloy quick reference". Duração: 50 minutos.

Nome do estudante:____**Nº**____________

1. **[11.2 valores]** Para cada uma das perguntas abaixo, assinale com uma cruz a resposta verdadeira. Cada resposta **correta** vale **1.6 valores**. Cada resposta **errada** vale **-0.4 valores**.

a) A cor dos semáforos (`Light`) pode ser verde, amarelo, ou vermelho. Como traduzir em Alloy?

□ `sig Light{}    sig Green, Yellow, Red extends Light{}`

☑ `enum Light {Green, Yellow, Red}`

□ `sig Light{}   sig Green, Yellow, Red in Light{}`

□ `Todas as anteriores são corretas`

b) Num exame, cada professor tem de ser alocado a uma única sala e cada sala tem de ter pelo menos um professor alocado. Como traduzir em Alloy?

□ `sig Exam{rooms: set Room, teachers: set Teacher, alloc: rooms some -> one teachers}`

□ `sig Exam{rooms: set Room, teachers: set Teacher, alloc: teachers one -> some rooms}`

☑ `sig Exam{rooms: set Room, teachers: set Teacher, alloc: teachers some -> one rooms}`

□ `sig Exam{rooms: set Room, teachers: set Teacher, alloc: teachers 1..* -> 1 rooms}`

c) Qual é o fecho transitivo (`^R`) da relação binária `R = {(a,b),(b,c),(c,b)}`?

□ `^R = {(a,b),(a,c),(b,c),(c,b)}`

☑ `^R = {(a,b),(a,c),(b,b),(b,c),(c,b),(c,c)}`

□ `^R = {(a,a),(a,b),(a,c),(b,a),(b,b),(b,c),(c,a),(c,b),(c,c)}`

□ `^R = {(a,b),(b,c),(c,b)}`

d) Dados `R1={(a,a),(a,b),(b,c)}` e `R2={(a,a),(a,c)}` qual é o valor da junção `R1.R2`?

□ `R1.R2 = {(a,a),(a,b),(a,c),(b,c)}`

□ `R1.R2 = {(a,a,a),(a,a,c)}`

□ `R1.R2 = {(a,a),(a,b)}`

☑ `R1.R2 = {(a,a),(a,c)}`

e) Dados `R1={(a,b),(b,b),(c,b)}`, `R2={(a,a)}` e `R3={(b)}` qual é o valor de (`R1 ++ R2) :> R3`?

☑ `{(b,b),(c,b)}`

□ `{(a,b),(b,b),(c,b)}`

□ `{(b,b)}`

□ `Nenhuma das anteriores está correta`

f) Dado `sig Task{precendences: set Task},` como garantir que não há precedências circulares?

□ `fact acyclic {no t: Task | t in t.^precedences}`

□ `fact acyclic {no t: Task | t->t in ^precedences}`

□ `fact acyclic {no ^precedences & iden}`

☑ `Todas as anteriores são corretas`

g) Dado `sig Exam{grades:Student->lone Int}`, como obter os pares (exame, nota) de um estudante?

□ `fun results[s: Student]: Exam->Int { all e: Exam, g: Int | e->s->g in grades }`

☑ `fun results[s: Student]: Exam->Int { {e:Exam, g: e.grades[s]} }`

□ `fun results[s: Student]: Exam->Int { grades[s] }`

□ `Todas as anteriores são corretas`

2. [**8.8 valores**] Preencher os blocos em branco.

```
    sig Medicin {
     incompatibilities: set Medicin-this // other medicins incompatible with this one
    }
    fact incompatibilities_symmetry {
     -- if m1 is incompatible with m2, then the opposite also holds
     incompatibilities = ~incompatibilities
0.8
    }
    sig Doctor { }
    sig Patient {
0.4  doctors: some Doctor, -- doctors (1 or more) of this patient (only them can prescribe medicins)
0.4  alergies: set Medicin, -- medicins (0 or more) that this patient is alergic to
0.8  prescriptions: Doctor lone->set Medicin -- current (active) prescriptions, as a set
      -- of pairs (doctor, medicin prescribed), with each medicin prescribed by at most one doctor
    }
    fun medicins[p: Patient] : set Medicin {
     p.prescriptions[Doctor]
0.8
    }
    pred safety_invariants[p: Patient] {
     -- a patient cannot be prescribed a medicin to which he/she is alergic
     no medicins[p] & p.alergies
0.8
     -- a patient cannot be prescribed mutually incompatible medicins
     no m1, m2: medicins[p] | m1 in m2.incompatibilities
0.8
     -- medicins can be prescribed only by the patient's doctors
     p.prescriptions.Medicin in p.doctors
0.8
    }
    -- doctor d prescribes medicin m to patient p, resulting in a new patient state p'
    pred prescribe[d: Doctor, m: Medicin, p, p': Patient] {
     -- pre-conditions (don't use predicate safety_invariants!)
     d in p.doctors   (can be removed by using +d in post-condition)
     not m in p.alergies + medicins[p].incompatibilities
     not m in medicins[p]   (can be removed using -Doctor->m in post-condition)
1.6
    -- post-conditions (don't use predicate safety_invariants!)
     p'.doctors = p.doctors   (or: + d)
     p'.alergies = p.alergies
     p'.prescriptions = p.prescriptions + d ->m
     (or: p'.prescriptions = (p.prescriptions - Doctor->m) + d->m
1.6 }
    assert prescribe_preserves_safety_invariants {
      all d: Doctor, m: Medicin, p, p': Patient |
        safety_invariants[p] and prescribe[d,m,p,p'] => safety_invariants[p']
    }
    check prescribe_preserves_safety_invariants
```

Boa sorte!