**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**
**Mestrado Integrado em Engenharia Informática e Computação**

# Métodos Formais em Eng.ª de Software

## EXERCÍCIOS

**1.** **[2 valores]** Considere as relações A={1->2, 2->3} e B={2->1,3->2,3->1}. Qual das seguintes afirmações é correta?

- ☐ ^A = ~B
- ☐ #(A.B) = #(B.~A)
- ☐ ^B = ~A
- ☐ Todas as alíneas anteriores estão corretas
- ☐ Nenhuma alínea está correta

**2.** **[2 valores]** Considere as relações A={(2,3),(3,2),(3,3)} e B={(3),(2)}. Qual das seguintes afirmações é correta?

- ☐ (B <: A) in ((A :> B) – iden)
- ☐ A.B & B.A = none
- ☐ Todas as alíneas anteriores estão corretas
- ☐ Nenhuma alínea está correta

**3.** **[2 valores]** Considere a relação r:A->A. Qual das seguintes relações descreve uma relação simétrica?

- ☐ iden in r
- ☐ r.A = r[A]
- ☐ ~r in r
- ☐ Todas as alíneas anteriores estão corretas
- ☐ Nenhuma alínea está correta

**4.** **[2 valores]** Qual das seguintes relações torna a expressão "**r:A->B | #(A.r) = #(r.B)**" verdadeira?

- ☐ r: A one -> one B
- ☐ r: A lone -> some B
- ☐ r: A set -> set B
- ☐ Duas das alíneas anteriores estão corretas
- ☐ Nenhuma alínea está correta

**5.** [12 valores] Considere a seguinte formalização em Alloy para representar relações entre diversas entidades numa Universidade:

```
abstract sig Person {}
sig Teacher extends Person {}
abstract sig Student extends Person {}
sig Graduate, Undergrad extends Student {}
sig Instructor in Person {}
sig Course {
      taughtby: one Instructor,
      enrolled: some Student,
      waitlist: set Student
}
```

**a)** Formalize o facto "todos os *Instructor* são *Teacher* ou *Graduate*".

**b)** Formalize o facto, "Existem pelo menos 2 *Course* e 5 *Undergrad*".

**c)** Formalize a seguinte asserção: "um *Graduate* não pode ser *Instructor* de um *Course* para o qual está inscrito (*enrolled*) ou em lista de espera (*waitlist*)".

**d)** Especifique uma função que devolva os estudantes (*Student*) que estão a frequentar (*enrolled*) ou em lista de espera (*waitlist*) de disciplinas lecionadas por um *Instructor* (a ser passado por argumento).

**e)** Especifique um predicado que remova um docente (*Instructor*) da Universidade. Assuma que todas as disciplinas lecionadas (*Course*) pelo docente (*Instructor*) deixam de ser oferecidas, ou seja, são também removidas.

**6.** Considere a seguinte formalização em Alloy de um sistema que guarda informação sobre o índice dos livros de uma biblioteca:

```
abstract sig Object {
  t: one Title
}
sig Title {}
sig Section extends Object {
   subsection : set Section
}
sig Books {
  name : Title,
  index : Title -> Section
}
```

**a)** Formalize o facto "não existe nenhum livro com menos do que 3 secções".

**b)** Formalize o facto "todas as secções (e subsecções) pertencem a livros".

**c)** Escreva uma operação que retire um livro e todas as suas secções e subsecções da biblioteca.

**d)** Escreva um predicado ou função que indique o título do livro que tem o índice com maior número de secções e subsecções.

**7.** Dardos é um jogo entre duas pessoas, onde cada jogador arremessa três dardos por jogada. Uma das versões mais populares é a 501. Basicamente cada jogador começa com 501 pontos. A cada jogada são subtraídos os pontos obtidos nos três dardos e a pontuação atual é guardada em *actualpoints*. Para terminar o jogo, o jogador tem que atingir exatamente 0 pontos e acertar no centro do tabuleiro numa jogada. Se numa jogada, o jogador não conseguir exatamente os pontos necessários para atingir o valor zero, a jogada é anulada e a vez passa para o outro jogador.

```
abstract sig Tipo {}
sig SIMPLES, CENTRO, ANULADA extends Tipo {}
abstract sig Jogador {}
sig A, B extends Jogador {}
sig jogada {
      type : one Tipo,
      value : some Int, // guarda os valores atingidos pelos 3 dardos
      player : one Jogador,
      actualpoints : one Int,
      next : lone jogada // jogada seguinte
}
one sig primeira extends jogada {}
-- fact1: em cada jogada, o jogador lança 3 dardos
-- fact2: a sequência de jogada tem alternadamente jogadas dos jogadores <A> e <B>
```

**a)** Formalize o facto *fact1*.

**b)** Formalize o facto *fact2*.

**c)** Escreva uma operação, *fimJogo*, que verifique se a última jogada é do tipo CENTRO e se tem um somatório de pontos atingidos pelos dardos igual ao *actualpoints* da jogada anterior do mesmo jogador. Assuma que existe uma função *fun soma[val: set Int]:Int* que retorna o somatório de um conjunto de inteiros (*val*).

**d)** Especifique um predicado que verifique se as jogadas estão estruturadas numa sequência, isto é, não pode haver jogadas em que a jogada seguinte (*next*) já ocorreu anteriormente, nem jogadas que não estejam ligadas à sequência.