

# Exercícios sobre Alloy

Alcino Cunha

17 de Outubro de 2013

1. Considere o seguinte modelo para representar os leilões em curso numa conhecida leiloeira online:

```
open util/ordering[Oferta]

sig Produto, Oferta {}
sig Leilao {
  produto : Produto
}
sig Cliente {
  leiloes : set Leilao,
  ofertas : Leilao -> Oferta
}
```

- (a) Especifique os seguintes invariantes:
  - Cada leilão pertence a um cliente.
  - Um cliente não pode licitar produtos que esteja a leiloar.
  - Não pode haver duas ofertas de igual valor no mesmo leilão.
- (b) Verifique que estes invariantes garantem que:
  - Um cliente não licita nos seus próprios leilões.
- (c) Refine o modelo por forma a capturar a mutabilidade das relações `Leilao`, `leiloes` e `ofertas`. Utilize o *local state idiom*.
- (d) Especifique a operação `Leiloar` que para um cliente `c` cria um leilão para o produto `p`. Verifique a sua consistência.
- (e) Especifique o predicado `Vencedor` que testa se um cliente `c` tem uma oferta vencedora num leilão de um produto `p`.
- (f) Especifique a operação `Licitar` que para um cliente `c` faz uma oferta vencedora num produto `p`. Verifique que essa operação é consistente e que realmente garante que `c` é `Vencedor` para `p`.
- (g) Refine o modelo por forma a garantir que as operações preservam os invariantes especificados (continuando a ser consistentes).

2. Considere o seguinte modelo do sistema de informação do mestrado em engenharia informática:

```
sig Aluno {}
sig Grupo {
  membros : some Aluno
}
sig Nota {}
abstract sig UCE {
  inscritos : set Aluno,
  grupos : set Grupo,
  notas : Aluno -> Nota
}
one sig MFES, CSSI, SD, EA extends UCE {}
```

- (a) Especifique os seguintes invariantes:

- Cada aluno só pode estar inscrito no máximo em duas UCEs.
- Todos os alunos dos grupos de uma UCE estão inscritos nessa UCE.
- Apenas os alunos inscritos têm (no máximo uma) nota em cada UCE.
- Cada aluno inscrito pertence apenas a um grupo em cada UCE.
- Todos os elementos de cada grupo têm a mesma nota.

- (b) Refine o modelo por forma a capturar a mutabilidade das relações `inscritos`, `grupos` e `notas`. Utilize o *local state idiom*.

- (c) Especifique a operação `LancaNota` que lança a nota de um aluno de uma UCE, por forma a garantir a sua consistência e a preservação dos invariantes.

- (d) Verifique que só é possível lançar notas em alunos sem grupo (ou que pertençam a um grupo só com um aluno). Que alteração sugere ao modelo para evitar esta anomalia?

3. Considere a seguinte especificação incompleta de um telemóvel. Para além da agenda telefónica, onde a cada nome estão associados vários números, o modelo guarda o conjunto de chamadas efectuadas e a hora actual.

```
open util/ordering[Hora]

sig Numero {}
sig Hora {}
one sig Actual in Hora {}
sig Nome {
  agenda : some Numero
}
```

```
sig Chamada {
  numero : one Numero,
  hora : one Hora
}
```

- (a) Especifique os seguintes invariantes:
    - Um número não pode pertencer a duas pessoas diferentes.
    - Todos os números chamados fazem parte da agenda.
    - Não podem existir chamadas simultâneas.
    - Todas as chamadas foram feitas antes da hora actual.
  - (b) Refine o modelo por forma a capturar a mutabilidade das relações `Nome`, `agenda`, `Actual` e `Chamada`. Utilize o *local state idiom*.
  - (c) Especifique as seguintes operações, garantindo a sua consistência e garantindo que o relógio avança:
    - `novo`: acrescentar um número à agenda.
    - `apaga`: eliminar um nome da agenda, apagando todos os números que lhe estão associados.
    - `chamar`: efectuar uma chamada para uma determinada pessoa - a chamada deve ficar registada com a hora actual.
  - (d) Refine o modelo por forma a garantir que as operações preservam os invariantes especificados (continuando a ser consistentes).
  - (e) Especifique as mesmas operações usando o *event idiom*: tente tirar partido da hierarquia para factorizar as propriedades comuns.
  - (f) Usando o *trace idiom* especifique sequências de eventos válidos partindo de um telefone sem qualquer informação. Especifique a seguinte propriedade sobre traços: só é registada a última chamada efectuada para cada número. Altere a especificação das suas operações por forma a garantir esta propriedade.
4. Considere o seguinte modelo de um gestor de memória. A assinatura `Free` contém o conjunto de endereços livres. A relação `block` representa os blocos alocados. Um bloco é essencialmente um conjunto de endereços.

```
open util/ordering[Addr]
```

```
sig Addr {
  block : lone Block
}
sig Free in Addr {}
sig Block {
  addr : set Addr
}
```

- Especifique os seguintes invariantes:

- Não existem endereços simultaneamente livres e ocupados.
  - Um endereço não pode estar alocado em dois blocos diferentes.
  - O apontador para um bloco faz parte dos endereços alocados nesse bloco.
  - Os endereços de um bloco são sequenciais e começam no apontador para o mesmo.
  - Verifique que estes invariantes garantem que:
    - Os blocos não podem ser vazios.
  - Refine o modelo por forma a capturar a mutabilidade das relações `Free`, `Block`, `block` e `addr`. Utilize o *global state idiom*.
  - Especifique as seguintes operações, garantindo a sua consistência:
    - `free`: dado um endereço liberta o bloco por ele apontado.
    - `alloc`: dado um inteiro estritamente positivo aloca um bloco com esse tamanho e devolve o endereço para o mesmo.
  - Refine o modelo por forma a garantir que as operações preservam os invariantes especificados (continuando a ser consistentes).
5. Especifique o problema da coloração de grafos em Alloy e utilize o Analyzer para descobrir uma coloração mínima do grafo de Peterson. Mais informação em:

[http://en.wikipedia.org/wiki/Graph\\_coloring](http://en.wikipedia.org/wiki/Graph_coloring)

6. Todos os artigos submetidos a uma conferência científica são revistos por um ou mais elementos da respectiva comissão de programa, que eventualmente decide quais deles devem ser aceites para apresentação e inclusão nas actas. Cada revisão inclui uma apreciação objectiva da qualidade do artigo sobre a forma de uma classificação (dentro de uma gama fixa de notas). O seguinte modelo Alloy especifica um *snapshot* deste processo.

```
open util/ordering[Nota]

sig Pessoa {}
some sig Comissao in Pessoa {}
sig Nota {}
sig Artigo {
  autores : some Pessoa,
  nota : Pessoa -> lone Nota
}
sig Submetido, Aceite in Artigo {}
```

- Especifique os seguintes invariantes:
  - As revisões só podem ser feitas por membros da comissão de programa a artigos submetidos.

- Um artigo não pode ser revisto pelos seus autores.
  - Todos os artigos aceites tem que ter pelo menos uma revisão.
  - Todos os artigos com uma nota máxima são automaticamente aceites.
- Verifique que estes invariantes garantem que:
    - Só são aceites artigos que foram submetidos.
  - Refine o modelo por forma a capturar a mutabilidade das relações `Submetido`, `Aceite` e `nota`. Utilize o *global state idiom*.
  - Especifique as seguintes operações, garantindo a sua consistência:
    - `submiter`: submeter um artigo.
    - `aceitar`: aceitar um artigo.
    - `rever`: classificar um artigo submetido.
  - Refine o modelo por forma a garantir que as operações preservam os invariantes especificados (continuando a ser consistentes).
  - Usando o *trace idiom* especifique sequências de eventos válidos. Especifique a seguinte propriedade sobre traços: não é possível rever as classificações de um artigo para valores mais baixos. Altere a especificação das suas operações por forma a garantir esta propriedade.
7. Uma empresa fabrica componentes numa linha de montagem organizada em fases de fabrico sucessivas. Um componente é fabricado numa das fases de fabrico usando materiais base e/ou sub-componentes fabricados previamente. Cada fase de fabrico deve ser suportada por máquinas que fazem a montagem dos componentes lá fabricados. O seguinte modelo Alloy (incompleto) especifica formalmente uma visão estática deste processo.

```

open util/ordering[Fase]

sig Fase {}

abstract sig Produto {}

sig Componente extends Produto {
  partes : set Produto,
  fase : one Fase
}

sig Material extends Produto {}

sig Maquina {
  fase : one Fase
}

```

- (a) Especifique invariantes que garantam as seguintes propriedades:

- Um componente não pode ser fabricado a partir do nada.
  - Um componente não pode ser um dos seus sub-componentes.
  - Todos os sub-componentes de um componente tem que ser fabricados em fases prévias.
  - Todas as fases envolvidas no fabrico de algum componente tem que ser suportadas por máquinas.
- (b) Altere o modelo por forma a permitir a modificação do catálogo de produtos fabricados, ou seja, permitir que as relações **Componente**, **partes**, e **fase** (do componente) sejam mutáveis, continuando a garantir os invariantes anteriores e potenciais restrições de integridade referencial. Utilize o *local-state idiom*, utilizando a assinatura **State** para denotar o estado.
- (c) Especifique uma operação que permita acrescentar um novo sub-componente ou material (um **Produto**) às partes necessárias para fabricar um componente, garantindo que essa operação preserva os invariantes anteriores.
- (d) Considere agora que existe um stock de materiais especificado da seguinte forma:

```
sig Material extends Produto {
  stock : Int -> State
}

fact {
  all s : State, m : Material {
    one m.(stock.s) and gte[m.(stock.s),0]
  }
}
```

Especifique um predicado que teste se um componente pode ser fabricado num determinado estado, ou seja, existe material suficiente em stock para o fabricar (incluindo todos os sub-componentes).

- (e) Usando o modelo apresentado no início do teste, apresente três exemplos de expressões que apresentem erros devido a *bounding types* vazios, *relevance types* vazios, e ambiguidade, respectivamente. Justifique formalmente a sua resposta.

8. Numa colónia de camaleões cada um dos camaleões pode, em cada momento, ter uma das seguintes cores: vermelho, azul ou verde. Sempre que dois camaleões de cores diferentes se encontram, ambos mudam para a cor que nenhum deles tinha. Caso contrário não mudam de cor.

```
sig Camaleao {
  cor : one Cor,
  encontra : lone Camaleao
```

```

}
abstract sig Cor {}
one sig Vermelho, Azul, Verde extends Cor {}

```

- (a) Especifique os seguintes invariantes:
    - Um camaleão não se encontra com ele próprio.
    - Os encontros são recíprocos.
  - (b) Refine o modelo por forma a capturar a mutabilidade das relações `cor` e `encontra`. Utilize o *local state idiom*.
  - (c) Utilizando o *trace idiom* especifique a dinâmica da colónia.
  - (d) Utilize o Alloy Analyzer para determinar a sequência de encontros necessários para levar a que uma dada colónia fique uniforme.
  - (e) Tente inferir qual a característica geral das colónias com potencial de ficarem uniformes.
9. O número de Erdős mede a “distância”, em termos de co-autoria de artigos, de um investigador ao prolífico matemático Paul Erdős. O número de Erdős de Paul Erdős é 0, o número de Erdős de todos os seus co-autores é 1, o dos co-autores dos seus co-autores é 2, e assim sucessivamente. Se não houver “caminho” entre uma pessoa e Paul Erdős o seu número de Erdős é indefinido. Mais informações sobre este número podem ser encontradas na respectiva página da Wikipedia:

[http://en.wikipedia.org/wiki/Erdos\\_number](http://en.wikipedia.org/wiki/Erdos_number)

O modelo do exercício anterior pode ser adaptado por forma a especificar este número. Basta modificar a assinatura `Pessoa` e acrescentar `Paul Erdős`:

```

open util/natural

sig Pessoa {
  erdos: lone Natural
}
one sig Erdos extends Pessoa {}

```

Defina um facto que garante que o número de Erdős de cada pessoa é correcto, de acordo com a especificação informal dada acima.

10. Considere o seguinte modelo dinâmico, com duas operações `X` e `Y` modeladas com eventos e *frame conditions* ao estilo de Reiter.

```

sig State {}

sig A {

```

```

    r : set State, s : set State, t : set State
  }

abstract sig Event {
  pre: State, pos: State,
  a : A
} {
  r.pos = r.pre + a
  s.pos != s.pre implies this in X
  t.pos != t.pre implies this in Y
}

sig X extends Event {} {
  s.pos = s.pre + a
}

sig Y extends Event {} {
  t.pos = t.pre + a
}

```

Defina um modelo equivalente especificando as operações usando predicados.

11. Modele um *snapshot* do site de partilha de fotografias Flickr em Alloy. Mais informação sobre o mesmo em:

<http://www.flickr.com/tour/>

12. Modele um repositório GIT em Alloy. Mais informação sobre o modelo de objectos do GIT em:

[http://home.cofares.net/GIT/book/1\\_the\\_git\\_object\\_model.html](http://home.cofares.net/GIT/book/1_the_git_object_model.html)