Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

# Métodos Formales para Ingeniería de Software

Trabajo Práctico N° 1

**Conjuntos y Relaciones - Alloy**

Segundo Cuatrimestre de 2018

## Ejercicios

1. Indique el tamaño, aridad, dominio y rango de las siguientes relaciones:

   *a*) `Name = {(G0), (A0), (A1)}`

   *b*) `addressBook = {(G0,A0), (G0,A1), (A0,D0), (A1,D1)}`

   *c*) `r1 = {(A0,B0,C0), (A3,B0,C1), (A0,B0,D0), (A3,B1,C1), (A1,B2,C0)}`

   *d*) `r2 = {(A1,B3,C1), (A3,B1,C1), (A0,B0,C0), (A2,B2,D0)}`

   *e*) `r3 = {(A3,B1,C1)}`

2. Considere las siguientes relaciones y la definición de operadores de conjuntos para Alloy. Para cada inciso, indique si las operaciones debajo listadas son legales y, en caso de serlo, cuál es el resultado de realizar dichas operaciones.

   *a*)
   ```
   Target = {(G0), (A0), (A1), (D0), (D1), (D2)}
   Name = {(G0), (A0), (A1)}
   Alias = {(A0), (A1)}
   Group = {(G0)}
   Addr = {(D0), (D1), (D2)}
   addressBook = {(G0,A0), (G0,A1), (A0,D0), (A1,D1)}
   ```

   - `Alias + Group`
   - `Alias & Target`
   - `Name - Alias`
   - `Target - Addr`
   - `Target in Group`
   - `addressBook & Group`
   - `Alias in Name`
   - `Target = Group + Alias`

   *b*)
   ```
   A = {(A0), (A1), (A2), (A3)}
   B = {(B0), (B1), (B2)}
   C = {(C0), (C1)}
   D = {(D0)}
   r1 = {(A0,B0,C0), (A3,B0,C1), (A0,B0,D0), (A3,B1,C1), (A1,B2,C0)}
   r2 = {(A1,B3,C1), (A3,B1,C1), (A0,B0,C0), (A2,B2,D0)}
   r3 = {(A3,B1,C1)}
   ```

- `(A + B) & univ`
- `r2 - r1`
- `r1 & r2 = r2 & r1`
- `A + r1`
- `r3 in r2`
- `A - B`

3. Considere los siguientes conjuntos de átomos para las signaturas `A` y `B`:

   `A = {(A0), (A1), (A2), (A3)}`
   `B = {(B0), (B1), (B2)}`

   Para cada inciso, determine si la relación binaria $\mathtt{r}_i$ es *funcional*, *inyectiva*, *ambas*, o *ninguna*.

   *a*) `r1 = {(A2,B0), (A1,B1), (A3,B2), (A1,B2)}`

   *b*) `r2 = {(A2,B2), (A1,B1)}`

   *c*) `r3 = {(A0,B2), (A3,B0), (A0,B1)}`

   *d*) `r4 = {(A1,B1)), (A2,B1), (A3,B0), (A0, B0)}`

   *e*) `r5 = {(A2,B0), (A3,B1), (A0,B2)}`

4. Considere los operadores de unión (`+`), intersección (`&`) y diferencia (`-`) de conjuntos.

   *a*) Al aplicar dichos operadores sobre dos relaciones binarias que poseen la propiedad de ser *funcionales*, el resultado ¿preserva la característica de ser una relación *funcional* (con respecto a las relaciones originales)? Para aquellos casos en que su respuesta sea negativa, brinde un contraejemplo.

   *b*) Al aplicar dichos operadores sobre dos relaciones binarias que poseen la propiedad de ser *inyectivas*, el resultado ¿preserva la característica de ser una relación *inyectiva* (con respecto a las relaciones originales)? Para aquellos casos en que su respuesta sea negativa, brinde un contraejemplo.

   *c*) Al aplicar dichos operadores sobre dos relaciones binarias que poseen la propiedad de ser *funcionales* e *inyectivas*, el resultado ¿preserva la característica de ser una relación *funcional* e *inyectiva* (con respecto a las relaciones originales)? Para aquellos casos en que su respuesta sea negativa, brinde un contraejemplo.

5. Considere la siguiente definición de relaciones, las cuales proveen una representación para el manejo de múltiples libretas de direcciones:

```
Book = {(B0), (B1)}
Target = {(G0), (A0), (A1), (D0), (D1), (D2)}
Name = {(G0), (A0), (A1)}
Alias = {(A0), (A1)}
Group = {(G0)}
Addr = {(D0), (D1), (D2)}
address = {(B0,G0,A0), (B1,G0,A1), (B0,A0,D0),
           (B1,A1,D1), (B0,A1,D2),(B1,A0,D2)}
myAlias = {(A0)}
myAddr = {(D2)}
```

Para cada uno de los siguientes incisos, determine cuál es el resultado de evaluar la expresión allí listada.

$a$) `Alias -> Addr`

$b$) `Book -> Name -> Target`

$c$) `Name -> Addr + Name -> Name`

$d$) `address & (Book -> Name -> Addr)`

$e$) `Book.address`

$f$) `address.Alias`

$g$) `Book.address.myAlias`

$h$) `myAlias.(Book.address)`

$i$) `myAlias[address.myAddr]`

$j$) `Book.address[myAlias]`

$k$) `address[Book]`

$l$) `~address`

$m$) `Alias.~address`

$n$) `^(Book.address)`

$\tilde{n}$) `^(Name -> Target)`

$o$) `*(Alias.address)`

$p$) `Group <: (Book.address)`

$q$) `Addr <: Target`

$r$) `address :> myAlias`

$s$) `Name :> Group`

$t$) `iden ++ address[Book]`

$u$) `Book.address ++ (^(Book.address) - Book.address)`

6. Considere el siguiente modelo en Alloy:

```
abstract sig Target {}
abstract sig Name extends Target { addressBook: some Target}
sig Alias extends Name {}
sig Group extends Name {}
sig Addr extends Target {}
```

Dada la siguiente instancia del modelo:

```
Target = {(Addr0), (Addr1), (Addr2), (Alias0), (Alias1), (Group0)}
Name = {(Alias0), (Alias1), (Group0)}
Alias = {(Alias0), (Alias1)}
Group = {(Group0)}
Addr = {(Addr0), (Addr1), (Addr2)}
addressBook = {(Alias0,Addr0), (Alias0,Addr1), (Alias0,Addr2), (Alias0,Alias0),
               (Alias0,Alias1), (Alias0,Group0), (Alias1,Addr0), (Alias1,Addr1),
               (Alias1,Addr2), (Alias1,Alias0), (Alias1,Alias1), (Alias1,Group0),
               (Group0,Addr0), (Group0,Addr1), (Group0,Addr2), (Group0,Alias0),
               (Group0,Alias1), (Group0,Group0)}
```

*a*) Para cada una de las siguientes operaciones, indique el resultado de evaluar dicha operación sobre la instancia arriba listada:

- `Alias + Group`
- `Alias & Target`
- `Name - Alias`
- `Target in Group`
- `Alias in Name`
- `Target = Group + Alias`
- `Alias -> Addr`
- `Name -> Target`
- `Name -> Addr + Name -> Name`
- `addressBook & (Name -> Addr)`
- `addressBook.Alias`
- `addressBook[Name]`
- `Name.addressBook`
- `~addressBook`
- `Alias.~addressBook`
- `^addressBook`
- `*addressBook`
- `*addressBook - ^addressBook`
- `^(Name -> Target)`
- `Group <: addressBook`
- `Addr <: Target`
- `Name :> Group`
- `iden ++ addressBook`

*b*) Defina un comando `run` en Alloy con las restricciones necesarias para generar la instancia del modelo arriba listada.

*c*) Verifique los resultados obtenidos en el inciso *a*) de este ejercicio utilizando el *evaluador* de Alloy para determinar el resultado de las operaciones aplicadas sobre la instancia generada con la herramienta en el inciso *b*).

7. Para cada uno de los siguientes modelos y grupos de instancias, indique cuáles de ellas son instancias válidas del modelo. Para cada instancia inválida, indique los errores existentes. Se recomienda utilizar la herramienta Alloy para obtener una visualización gráfica del modelo.

   *a*)
   ```
   abstract sig Target {}
   sig Addr extends Target {}
   abstract sig Name extends Target {}
   sig Alias, Group extends Name {}
   ```

   - ```
     Target = {(D0), (A0), (A1)}
     Addr = {(D0)}
     Name = {(A0)}
     Alias = {(A0), (A1)}
     Group = {}
     ```

   - ```
     Target = {(D0), (D1) (A0), (A1), (A2), (G0)}
     Addr = {(D0), (D1)}
     Name = {(A0), (A1), (A2), (G0)}
     Alias = {(A0), (A1), (A2)}
     Group = {(G0)}
     ```

   - ```
     Target = {(D0), (A0), (G0)}
     Addr = {(D0)}
     Name = {(A0), (A1), (G0), (G1)}
     Alias = {(A0)}
     Group = {(G0)}
     ```

   - ```
     Target = {(A0), (A1), (A2), (G0), (G1)}
     Addr = {}
     Name = {(A0), (A1), (A2), (G0), (G1)}
     Alias = {(A0), (A1), (A2)}
     Group = {(G0), (G1)}
     ```

   - ```
     Target = {(D0), (D1), (A0), (A1)}
     Addr = {(D0), (D1)}
     Name = {(A0), (A1), (D0)}
     Alias = {(A0), (A1)}
     Group = {(D0)}
     ```
     ```

*b*)
```
abstract sig Person {siblings: Person}
sig Man extends Person {}
sig Woman extends Person {}
sig Married in Person {}
```

- Person = {(P0), (P1), (P2)}
  Man = {(P1), (P2)}
  Woman = {(P0)}
  Married = {}
  siblings = {(P0,P2), (P2,P0), (P1,P2)}

- Person = {(P0), (P1), (P2)}
  Man = {(P1), (P2)}
  Woman = {(P0), (P1)}
  Married = {(P1)}
  siblings = {}

- Person = {(P0), (P1)}
  Man = {(P0)}
  Woman = {(P1)}
  Married = {(P1)}
  siblings = {(P0,P0)}

- Person = {(P0), (P1)}
  Man = {(P0)}
  Woman = {(P1)}
  Married = {}
  siblings = {(P0,P2)}

- Person = {(P0), (P1), (P2)}
  Man = {(P0), (P2)}
  Woman = {(P1)}
  Married = {(P0), (P1)}
  siblings = {(P1,P2), (P2,P0), (P0,P0)}

- Person = {(P0), (P1), (P2)}
  Man = {(P2)}
  Woman = {(P0), (P1)}
  Married = {(P0), (P1)}
  siblings = {(P1,P2), (P1,P0), (P0,P1), (P2,P1)}

*c*)
```
sig PrimaryColor {}
sig SecondaryColor {}
one sig Red, Yellow, Blue extends PrimaryColor {}
```

- PrimaryColor = {(C1), (C2), (C3), (C4)}
  SecondaryColor = {(C4), (C5)}
  Red = {(C1)}
  Yellow = {(C2)}
  Blue = {(C3)}

- PrimaryColor = {(C1), (C2), (C3), (C4)}
  SecondaryColor = {(C5)}
  Red = {(C1)}
  Yellow = {}
  Blue = {(C3)}

- PrimaryColor = {(C1), (C2), (C3)}
  SecondaryColor = {}
  Red = {(C1)}
  Yellow = {(C3)}
  Blue = {(C2)}

- PrimaryColor = {(C1), (C2), (C3), (C4)}
  SecondaryColor = {(C5), (C6)}
  Red = {(C2)}
  Yellow = {(C3)}
  Blue = {(C3)}

*d*)
```
sig Name, Addr {}
sig Book {address: Name some -> lone Addr}
```

- Name = {(N0), (N1), (N2)}
  Addr = {(A0), (A1)}
  Book = {(B0)}
  address = {(B0, N1, A0), (B0, N2, A1), (B0, N0, A1)}

- Name = {(N0), (N1)}
  Addr = {(A0), (A1)}
  Book = {(B0)}
  address = {(B0, N0, A0)}

- Name = {(N0), (N1), (N2)}
  Addr = {(A0)}
  Book = {(B0), (B1)}
  address = {(B1, N0, A0), (B1, N2, A0)}

- Name = {(N0), (N1), (N2)}
  Addr = {(A0), (A1)}
  Book = {(B0), (B1)}
  address = {(B0, N2, A1), (B1, N0, A0), (B1, N2, A0), (B0, N1, A0)}

- Name = {(N0), (N1), (N2)}
  Addr = {(A0), (A1)}
  Book = {(B0), (B1)}
  address = {(B0, N2, A1), (B0, N0, A0), (B0, N2, A0)}

8. Para cada uno de los siguientes modelos, brinde al menos dos instancias válidas y, en caso de existir, dos instancias inválidas.

   *a*) `sig S1 {r1: lone T}`

   *b*) `sig S2 {r2: one T}`

   *c*) `sig S3 {r3: T -> one U}`

   *d*) `sig S4 {r4: T lone -> U}`

   *e*) `sig S5 {r5: some T}`

   *f*) `sig S6 {r6: set T}`

   *g*) `sig S7 {r7: T set -> set U}`

   *h*) `sig S8 {r8: T one -> U}`

9. Explique en lenguaje coloquial el significado de cada una de las siguientes declaraciones, a partir del modelo que se ilustra a continuación. Asimismo, para cada inciso, escriba en Alloy una declaración alternativa que posea el mismo significado coloquial.

```
abstract sig Target {}
sig Addr extends Target {}
abstract sig Name extends Target {}
sig Alias, Group extends Name {}
sig Book {address: Name -> Target}
```

   *a*) `all b: Book | some b.address`

   *b*) `all n: Name | some b: Book | lone b.address[n]`

   *c*) `#(Name.(b.address) - Name) > 0`

   *d*) `{n: Name | no (n.(Book.address) & Addr)}`

   *e*) `all b: Book | no n: Name | n in ^(b.address)[n]`

   *f*) `all n: Name | no disj t, t': Target | (t + t') in n.(Book.address)`

   *g*) `all b: Book | all n: Name| no disj t, t': Target |`
   `(t + t') in n.(b.address)`