Si se parte de un diagrama de clase:

* Cada clase corresponde a un conjunto de objetos (átomos)
* Cada objeto es una entidad abstracta, atómica e inmodificable

En Alloy todo se construye a partir de átomos y relaciones

El **átomo** es la entidad primitiva. Es una entidad:

* Abstracta y Atómica: no puede dividirse en partes más pequeñas
* Inmodificable: sus propiedades no cambian con el tiempo
* Ininterpretable: no tiene propiedades propias

Una **relación** es una estructura que relaciona átomos.

* Define un conjunto de tuplas
* Cada tupla es una secuencia de átomo

Una biblioteca mantiene un mapeo de títulos de libros a autores

**Relaciones unarias**: el conjunto de bibliotecas, el conjunto de títulos y el conjunto de autores

Biblioteca = {(B0)}

Libro = {(L0), (L1), (L2)}

Autor = {(A0), (A1)}

Una **relacion binaria** de Libro a Autor

**escritoPor** = {(L0, A0), (L1, A1), (L2, A0)}

Una **relacion ternaria** de Biblioteca a Libro a Autor

**coleccion** = {(B0, L0, A0), (B0, L1, A1)}

**Tamaño de una relación:** el **número** de **tuplas** en la relación (equivalente a la cardinalidad de conjuntos, #)

**Aridad de una relación**: el **número** de **átomos** **en cada tupla** de la relación

**Ejemplos**:

Biblioteca = {(B0)} es una relación de tamaño 1 y aridad 1 (unaria)

coleccion = {(B0, L0, A0), (B0, L1, A1)} es una relación de tamaño 2 (ternaria) y aridad 3

**Signaturas (signatures)**

* Describen las entidades con las que se quiere razonar
* **Definen conjuntos fijos**

**Campos (fields)**

* Definen relaciones entre signaturas.
* Es la única manera de incorporar relaciones en Alloy

**Hechos (facts)**

**Aserciones (assertions)**

**Comandos y alcances (commands – scope)**

**Predicados y funciones (predicates – functions)**

Resticciones o constraints simples (multiplicidad sobre signaturas y/o relaciones)

**Una signatura define un conjunto de átomos** (que representa un conjunto base)

La declaración:

**sig A { }**

define un **conjunto de nombre A.**

Un conjunto puede introducirse como una extensión de otro, de la siguiente manera:

**sig A1 extends A { }**

de esta manera el conjunto **A1 es un subconjunto de A**, aunque con ciertas restricciones.

Ejemplo tenemos el siguiente Modelo:

sig **Biblioteca** {**colección**: Libro -> Autor}

sig **Libro** {**escritoPor**: set Autor}

sig **Autor** { }

sig **Novelista**, **Poeta**, **Periodista** extends **Autor** { }

Observación: Los nombres de color **naranja** representan **campos** o **fields**, mientras que los de color **azul** representan **signaturas**

Algunas instancias del modelo:

**Biblioteca** = {(B0)}

**Libro** = {(L0), (L1)}

**Autor** = {(A0), (A1)}

**Novelista** = { }

**Poeta** = { }

**Periodista** = { }

**coleccion** = { }

**escritoPor** = { }

**Biblioteca** = { (B0)}

**Libro** = {(L0), (L1), (L2)}

**Autor** = {(A0), (A1), (A2), (A3)}

**Novelista** = {(A0)}

**Poeta** = {(A1), (A2)}

**Periodista** = {(A3)}

**coleccion** = {(B0, L2, A3), (B0, L2, A0)}

**escritoPor** = {(L1, A2)}

Las **signaturas declaradas de manera independiente** son top-level signatures

IMPORTANTE:

* Las signaturas top-level **son disjuntas**
* **Las diferentes extensiones de una signatura son disjuntas**

Es posible definir signaturas abstractas:

**abstract** **sig A {}**

Es posible definir signaturas relacionadas por el concepto de subconjunto:

**sig B1 in** **B {}**

Signaturas abstractas: no poseen elementos propios, ajenos a las signaturas que las extienden, a menos que no tengan signaturas que las extiendan

**abstract sig A {}**

**abstract sig B { }**

**sig A1 extends A { }**

**sig A2 extends A { }**

**sig A3 in A { }**

**sig B1 in B { }**

B puede tener átomos propios (no pertenecientes a B1)

Las relaciones son declaradas como campos o fields de signaturas

Escribir

**sig A { f: e}**

introduce al modelo una **relación f** cuyo **dominio** **es** **o está incluido en A** y cuyo **rango** está dado por la expresión **e**.

Ejemplos:

sig A { f1 : A } f1 es un subconjunto de A x A

sig B { f2: A -> A} f2 es un subconjunto de ???

La **multiplicidad** **define** restricciones en cuanto al **tamaño** de los **conjuntos** (signaturas o relaciones).

Se escribe una **keyword** antes de la signatura, para restringir la cantidad de átomos que define esa signatura

**m** sig A {}

También pueden establecerse restricciones sobre los campos (fields)

sig A {f: m e}

sig A {f: e1 m -> n e}

Hay cuatro keywords de multiplicidad en Alloy:

1. **set**: cualquier número
2. **lone**: cero o uno
3. **some**: uno o más
4. **one**: exactamente uno (viene por defecto)

**Ejemplo:**

abstract sig Object { }

sig Directory extends Object {contents: set Objects }

sig File extends Object {}

sig Alias in File {to: **one** Object}

**Observación**: el keyword de multiplicidad por defecto (para relaciones binarias) es one, de manera que puede omitirse por resultar redundante.

Para **ejecutar una aserción** en **Alloy**, se utiliza el comando **check**. **Ejemplo**:

**check** nombre\_asercion **for 10**