Alloy

Lab Manual-1

In Alloy, everything is built from **Atoms** and **Relations**.
An **atom** is a primitive entity that is-

- **Indivisible**: it cannot be broken down into smaller parts
- **Immutable**: its properties do not change over time
- **Uninterpreted**: it does not have any built-in property (the way numbers do for example)

A **relation** is a structure that **relates atoms**. It is a set of **tuples**, each tuple being a sequence of atoms. Relations can be many forms, such as

- Unary relations: a set of names, a set of addresses, and a set of books.
  - Name = {(N0),(N1),(N2)} // ***Atoms(3)***: N0, N1, N2; ***Tuples(3)***: N0, N1, N2
  - Addr = {(D0),(D1)} // ***Atoms(2)***: D0, D1; ***Tuples(2)***: D0, D1
  - Book = {(B0),(B1)} // ***Atoms(2)***: B0, B1; ***Tuples(2)***: B0, B1
- Binary relations:A binary relation from names to addresses
  - address = {(N0,D0),(N1,D1)} // ***Atoms(4)***: N0, D0, N1, D1; ***Tuples(2)***:(N0,D0), (N1,D1)
- Ternary relations: A ternary relation from books to name to addresses
  - addr = {(B0,N0,D0), (B0,N1,D1), (B1,N1,D2)} // ***Atoms(4)***: B0, N0, D0, N1, D1; ***Tuples(3)***: (B0,N0,D0), (B0,N1,D1), (B1,N1,D2)

There are two terms related to relations.

- **Size of a relation**: The number of tuples in the relation
- **Arity of a relation**: The number of atoms in each tuple of the relation

relation of **arity 1 and size 1**: ***myName = {(N0)}***
relation of **arity 2 and size 3**: ***address = {(N0,D0),(N1,D1),(N2,D1))***

Main components of Alloy model

1. Signatures: Describe classes of entities we want to reason about.
2. Fields: Define relations between signatures
3. Predicates
4. Functions
5. Facts
6. Assertions
7. Command and scopes

Signatures

A **signature** introduces a set of **atoms**. A signature named ***A*** can be declared as

sig A{}

Even, a set can be introduced as an extension of another; thus

sig A1 extends A{}

introduces a set ***A1*** that is a **subset** of ***A***.

Some variations of a signature:

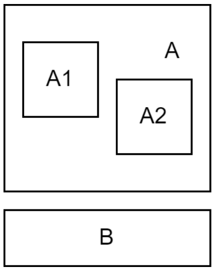
1.

sig A {}

sig B {}

sig A1 extends A {}

sig A2 extends A {}



Here, ***A1*** and ***A2*** are **extensions** of ***A***. Extensions of the same signature are **mutually disjoint**, as are top-level signatures.

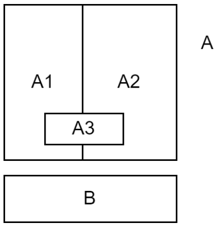
2.

abstract ***sig A {}***

sig B {}

sig A1 extends A {}

sig A2 extends A {}



An **abstract signature** has **no elements** except those belonging to its extensions or subsets. All extensions of an abstract signature ***A*** form a **partition** of A. A signature can be introduced as a subset of another.

sig A3 in A {}

Fields

Relations are declared as **fields** of signatures.

sig A { f: e}

It introduces a relation f of type ***A* x *e***, where ***e*** is an expression denoting a **product** of signatures. Some examples of ***signatures A, B, C*** are-

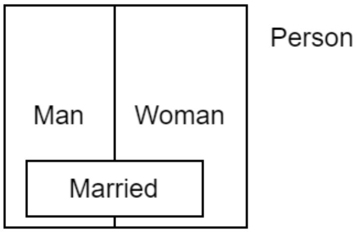
- Binary Relation: ***sig A { f1: B}*** // subset of A x B
- Ternary Relation: ***sig A { f2: B -> C }*** // subset of A x B x C

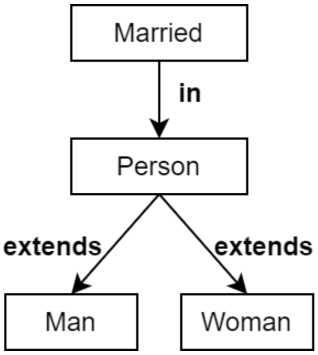
Real-life examples- Family Structure:

```
abstract sig Person {
        Children: Person,
        Siblings: Person
}

sig Man, Woman extends Person {}

sig Married in Person {
        spouse: Married
}
```





The Alloy Analyzer will generate instances of models so that we can see if they match our intentions.

AA allows us to constrain the size of sets. A multiplicity keyword placed before a signature declaration constraints the number of elements in the signature's set.

```
m sig A {}
```

We can also make multiplicities constraints on fields.

```
sig A { f: m e}
sig A { f: e1 m -> n e2}
```

The **default multiplicity** is one. There are four multiplicities

1. **set**: any number
2. **some**: one or more
3. **lone**: zero or one
4. **one**: exactly one