ALLOY

1. **set**: o conjunto pode ter **qualquer número** de elementos (inclusive nenhum).
2. **lone**: o conjunto pode ter **zero ou um** elemento.
3. **some**: o conjunto **deve ter pelo menos um** elemento.
4. **one**: o conjunto **deve ter exatamente um** elemento
5. **~**: Representa o **inverso** de uma relação. Se succ é a relação de sucessores, então ~succ é a relação de predecessores.
6. **^**: Representa o **fecho transitivo** de uma relação. Se succ é a relação de sucessores, então ^succ é a relação de todas as estações de trabalho que podem ser alcançadas a partir de uma estação através de sucessores sucessivos.
7. **+ (União de Conjuntos)**: O operador + é usado para unir dois conjuntos. Exemplo: A + B significa "a união de A e B".
8. **& (Interseção de Conjuntos)**: O operador & é usado para obter a interseção de dois conjuntos. Exemplo: A & B significa "a interseção de A e B".
9. **- (Diferença de Conjuntos)**: O operador - é usado para calcular a diferença entre dois conjuntos. Exemplo: A - B significa "os elementos de A que não estão em B".
10. **' (apóstrofo):** Usado para denotar o **próximo estado** ou o **estado futuro**

$\neg\phi$	$\texttt{not}\ \phi$
$\phi \wedge \psi$	$\phi\ \texttt{and}\ \psi$
$\phi \vee \psi$	$\phi\ \texttt{or}\ \psi$
$\phi \rightarrow \psi$	$\phi\ \texttt{implies}\ \psi$
$(\phi \wedge \psi) \vee (\neg\phi \wedge \theta)$	$\phi\ \texttt{implies}\ \psi\ \texttt{else}\ \theta$
$\phi \leftrightarrow \psi$	$\phi\ \texttt{iff}\ \psi$

11.

$x = y$	$x = y$
$A(x)$	x **in** A
$R(x, y)$	$x\text{->}y$ **in** R
$\forall x \,.\, A(x) \rightarrow \phi$	**all** x **:** A | ϕ
$\exists x \,.\, A(x) \wedge \phi$	**some** x **:** A | ϕ

run {} for 4 but exactly 2 Dir, 3 Name

- For 4 - pode criar até 4 instâncias para **cada assinatura não abstrata**
- Exactly 2 - Especifica que haverá **exatamente 2 instâncias**

Modulos - util/ordering

1. `lt` (less than / "menor que"):
   - Representa a comparação **menor que**.
   - Exemplo: `x lt y` significa que **x é menor que y**.
2. `lte` (less than or equal / "menor ou igual"):
   - Representa a comparação **menor ou igual a**.
   - Exemplo: `x lte y` significa que **x é menor ou igual a y**.
3. `gt` (greater than / "maior que"):
   - Representa a comparação **maior que**.
   - Exemplo: `x gt y` significa que **x é maior que y**.
4. `gte` (greater than or equal / "maior ou igual"):
   - Representa a comparação **maior ou igual a**.
   - Exemplo: `x gte y` significa que **x é maior ou igual a y**.

- always φ -> φ is true in all future states
- eventually φ -> φ is true in some future state
- after φ -> φ will be true in the next state
- ψ until φ -> φ will eventually be true and ψ is true until then
- φ releases ψ -> ψ só pode deixar de ser verdade depois de φ
- once φ -> φ já foi verdade
- Historically φ -> φ foi sempre verdadeiro

- Before φ -> φ foi verdadeiro no estado anterior
- ψ Since φ-> φ já foi verdade e depois disso o ψ foi verdadeiro

```
sig User {
	follows : set User,
	sees : set Photo,
	posts : set Photo,
	suggested : set User
}

sig Influencer extends User {}

sig Photo {
	date : one Day
}
sig Ad extends Photo {}

sig Day {}

// Specify the following properties.
// You can check their correctness with the different commands and
// when specifying each property you can assume all the previous ones to be
true.

pred inv1 {
	// Every image is posted by one user.
	//all y : Photo | one x : User | x -> y in posts
	all y : Photo | one x : User | x in posts.y
}

pred inv2 {
	// Users cannot follow themselves.
	//all x : User | not x in x.follows
	no p : User | p in p.follows
}

pred inv3 {
	// Users can see ads posted by everyone,
	// but only see non ads posted by followed users.
	all u : User, a : Photo | u -> a in sees implies a in Ad or some u1 : User | u1
-> a in posts and u -> u1 in follows
	// all u : User, a : u.sees-Ad | some posts.a & u.follows
}

pred inv4 {
	// If a user posts an ad then all its posts should be labeled as ads.
	all x : User | (some y : Photo | y in x.posts & Ad)
```

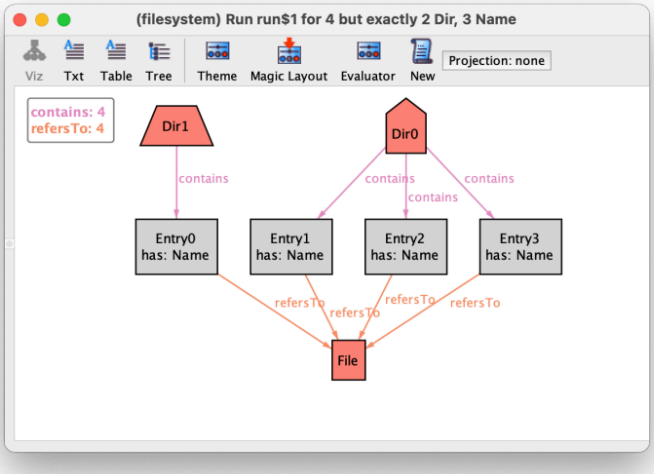
```
implies (all z : Photo | x -> z in posts implies z in Ad)
}

pred inv5 {
        // Influencers are followed by everyone else.
        all i : Influencer, u : User | i != u implies u -> i in follows
}

pred inv6 {
        // Influencers post every day.
        all i : Influencer, d : Day | some p : Photo | i in posts.p and p.date = d
}

pred inv7 {
  all u,s : User |
    (s in u.follows.follows)
    and (not u->s in follows)
    and (u != s)
    iff s in u.suggested
}

pred inv8 {
        // A user only sees ads from followed or suggested users.
        all u1, u2 : User, a : Ad |
        (u2 in posts.a and u1 in sees.a) implies (u1 in follows.u2 or u1 in
suggested.u2)
}
```



Additional requirements

```
fact {
  // All directories are referred to in at most one entry
  all x : Dir, y,z : Entry | y->x in refersTo and z->x in refersTo implies y = z

  // The root is not referred in any entry
  all x : Entry, y : Root | x->y not in refersTo

  // All objects except the root are referred to in at least one entry
  all x : Object | x not in Root implies some y : Entry | y->x in refersTo

  // Different entries in a directory must have different names
  all x : Dir, y,z : Entry, w : Name {
    x->y in contains and x->z in contains and y->w in has and z->w in has implies y = z
  }
}
```

Create item

```
pred create [i : Item] {
  // guard
  i not in Accessible + Trashed
  // effect
  Accessible' = Accessible + i
  // frame condition
  Trashed' = Trashed
}
```