Clausura (reflexo-)transitiva

- Cómo hacemos para especificar que una lista simplemente enlazada no tiene un ciclo?
  - $\forall n (*next(head,n) => !*next(next(n),n))$
- Sin utilizar clausura no es posible hacerlo de forma declarativa. Ahora… es la clausura expresable en primer orden?
- Si lo fuera, podemos escribir:
    - $\forall n (*suc(0,n))$.
    - Qué dice la fórmula de arriba?

Lógica Relacional

- Más poderosa que la lógica de primer orden (incluye clausura). Es la base para el lenguaje de modelado Alloy (Daniel Jackson, MIT).
- Se basa en el concepto de que todos los objetos son relaciones.
- El Analizador de Alloy permite chequear de forma automática si una aserción (fórmula) es consecuencia de fórmulas del modelo. Si no lo es, provee un contraejemplo.

Lógica Relacional: Lenguaje

- Por defecto el lenguaje incluye los siguientes operadores relacionales que permiten construir los términos de la lógica relacional.
  - + (unión de relaciones de la misma aridad),
  - - (resta de relaciones de la misma aridad),
  - . (composición de relaciones, o navegación),
  - * (clausura reflexo-transitiva de una relación binaria)
  - ^ (clausura transitiva de una relación binaria)
  - none : es una constante que representa la relación vacía (es unaria)
  - unit : es una constante que representa la relación universal que relaciona todos contra todos (es binaria)
  - iden: es la relación identidad binaria

Lógica Relacional: Lenguaje

- Además de lo que viene por defecto, es posible definir un conjunto de constantes $ {c_1,\ldots,c_k}$ que representan relaciones, cada una con su aridad.

El Lenguaje de Modelado Alloy

- Está basado en la lógica relacional.
- Lo presentaré a través de un ejemplo.

Un Modelo Alloy Simple

a field containing a binar

facts are a

mposition of relations

property to be verified

reflexive-transitive closure
restricted to set A

```
sig
one
fact refl
fact transi                                    Rel.r }

assert rEqualsItsClosure { Rel.r = A<:*(Rel.r)}
check rEqualsItsClosure for 5
```

gives instructions to the Alloy Analyzer
on the sizes of data domains

Alloy Analyzer: Limitaciones

- Dado que se realiza un chequeo acotado en el tamaño de los dominios de datos, el mismo es incompleto. Por lo tanto, cuando el Analizador de Alloy dice que no ha encontrado contraejemplos, podría haberlos (y encontrarlos) al considerar dominios de mayor tamaño.

Alloy Analyzer Demo

Desde /Applications/Alloy4.1

```
/Library/Internet\ Plug-Ins/JavaAppletPlugin.plugin/Contents/Home/bin/java -jar alloy4-1-10.jar
```

Alloy Analyzer: Cómo funciona?

- Dado que el dominio es finito, se podría hacer una prueba exhaustiva.
- Para scope 5, tenemos que considerar todas las relaciones posibles Rel entre A y A, con A teniendo hasta 5 elementos.
  - Si card(A) = 1: 2 relaciones solamente!
  - Si card(A) = 2: 16 relaciones solamente!
  - Si card(A) = 3: 512 relaciones solamente!
  - Si card(A) = 4: 65.536 relaciones solamente!
  - Si card(A) = 5: 2.199.023.255.552 relaciones **solamente?**

Alloy Analyzer: Cómo funciona?

- Se traducen los axiomas del modelo a una fórmula proposicional α.
- Se traduce la aserción a verificar a una fórmula proposicional β.
- Tomamos la fórmula $\alpha && !\beta$, y se la pasamos a un SAT-solver.
- Si el SAT-solver encuentra una valuación que hace verdadera a $\alpha && !\beta$, hemos encontrado un contraejemplo para β. Sinó, no existe un contraejemplo dentro de esos límites para el tamaño de los dominios.