Aclaraciones para TP Redes Sociales

Taller de Álgebra

Primer Cuatrimestre 2019

1. Invariantes del TP

1.1. Invariantes de Representación de los Tipos

En el TP de este semestre, tabajamos con valores de tipo **Set**, **Usuario**, **Relacion**, **Publicacion**, **RedSocial**. Estos tipos están definidos como renombres, por lo que sus invariantes (y por lo tanto, los valores del tipo que no son válidos o *bien formados*) no son verificados por el motor de Haskell y queda como trabajo para el programador. Este documento explica las invariantes que determinan cuando los valores de estos tipos son válidos, y cuando dos valores de estos tipos son iguales.

1.1.1. Set

El tipo paramétrico **Set** ***a*** fue estudiado en clase. Su tipo es:

```
[a]
```

Invariantes de Representación:

- Todas las maneras posibles de ordenar la lista de los elementos de un conjunto representan a ese mismo conjunto.
- La lista que representa a un conjunto nunca tiene elementos repetidos.

Igualdad:
Dos valores c1 y c2 de **Set** son iguales cuando todos los elementos de c1 están en c2 y todos los elementos de c2 están en c1. No es posible usar el operador "==" para conjuntos porque da por distintos conjuntos que nosotros consideramos iguales. En clase vimos cómo definir la función iguales para conjuntos.

1.1.2. Usuarios

El tipo **Usuario** es:

```
(Integer, String)
```

Invariantes de Representación:

- El entero, que corresponde con el *id* del usuario, es un número positivo.
- El String, que corresponde con el *nombre* del usuario, es una cadena no vacía.

Igualdad:
Dos valores de **Usuario** son iguales cuando los números son iguales. Por Invariante del tipo RedSocial (Ver Sec. 1.1.5), establecemos que siempre que dos usuarios tengan el mismo id necesariamente van a tener el mismo nombre. Por lo tanto, es factible comparar la igualdad de dos usuarios usando el operador "==".

1.1.3. Relaciones

El tipo **Relacion** es:

```
(Usuario, Usuario)
```

Invariantes de Representación:

- Los Usuarios de la tupla son distintos.

Igualdad:
Dos valores de **Relacion** son iguales cuando asocian el mismo par de usuarios Es decir, la tupla (u1,u2) es igual a (u2,u1). Por lo tanto, no se los puede comparar con "==".

1.1.4. Publicaciones

El tipo **Publicacion** es:

```
(Usuario, String, Set Usuario)
```

Invariantes de Representación:

- El primer elemento de la tupla representa al usuario *autor* de la publicación.
- El segundo elemento de la tupla representa al *contenido* de la publicación, y es una cadena no vacía.
- El tercer elemento representa el conjunto de usuarios *que indicaron que les gustó* la publicación. Puede estar vacío. También es posible que el autor de la publicación aparezca en este conjunto.

Igualdad:
Dos valores de **Publicacion** son iguales cuando su autor es el mismo, su contenido es el mismo, y los conjuntos de usuarios a los que les gustó la publicación son iguales. Por Invariante del tipo RedSocial (Ver Sec. 1.1.5), establecemos que existe una única publicación con un determinado autor y contenido. Por lo tanto, en el contexto de una misma Red Social es factible comparar la igualdad de dos publicaciones usando el operador "==" (La publicación es necesariamente la misma o difiere en autor o contenido).

1.1.5. Redes Sociales

El tipo **RedSocial** es:

```
(Set Usuario, Set Relacion, Set Publicacion)
```

Invariantes de Representación:

- El conjunto de Usuarios, que representa a los *usuarios de la red*, tiene al menos un elemento (No hay redes sociales sin usuarios).
- Los usuarios que aparecen en alguna relación, los autores de una publicación y las personas a las que les gustó una publicación siempre están en el conjunto de usuarios de la red. Si en estos conjuntos aparecen múltiples valores del tipo Usuario con el mismo valor de id, entonces necesariamente tienen el mismo nombre. Notar que esto **no impide** que hayan dos usuarios con distinto id y el mismo nombre.
- El conjunto de Relaciones, que representa las *relaciones de amistad de la red*, consiste en relaciones entre usuarios (que existen entre los usuarios de la red, como fue mencionado). Si dos usuarios de la red u1 y u2 son amigos, entonces **exactamente una** de las relaciones (u1,u2) o (u2,u1) está en las relaciones de amistad de la red (No las dos, puesto que como las relaciones son iguales (Subsec. 1.1.3), si estuvieran ambas el conjunto de relaciones tendría elementos repetidos). No está predeterminado si la relación que aparecerá es (u1,u2) o (u2,u1).
- El conjunto de Publicaciones representa las publicaciones en la red. Puede estar vacío.
- No existen dos publicaciones en la red que tengan el mismo autor y el mismo contenido.

Igualdad:
Dos valores de **RedSocial** son iguales si los conjuntos de Usuarios son iguales, los conjuntos de Relaciones son iguales y los conjuntos de Publicaciones son iguales (Utilizando la igualdad de Conjuntos, por lo que no es válido utilizar "==").

2. Otras aclaraciones

- Al desarrollar una función que reduzca a uno de estos tipos, es importante que el valor obtenido cumpla con las invariantes especificadas. Es válido que funciones auxiliares intermedias o casos recursivos trabajen con valores que no respeten las invariantes en tanto se pueda garantizar que los resultados finales sí lo hagan.
- Notar que por Invariante del tipo **Publicacion** (Subsec. 1.1.4) un usuario u1 puede haber dado Like a su propia publicación, y en particular puede haberle dado like a *todas* sus propias publicaciones. Sin embargo, no alcanza para decir que u1 tiene un Seguidor Fiel, pues por enunciado se pide que haya *otro* usuario que haya dado Like a todas las publicaciones de u1.

- Respecto a los likes, notar también que un usuario puede haber dado like a la publicación de otro, sin que estos sean amigos. Lo único que es requerido es que ambos usuarios existan en la red social.
- Un usuario que no tiene publicaciones **no** tiene un Seguidor Fiel.
- Los casos de test para la función estaRobertoCarlos no requieren un caso donde se generen un millón de relaciones de amistad. Es admisible que se pruebe con una función que funcione de igual manera pidiendo una cantidad inferior. El código final de estaRobertoCarlos, de todos modos, debe exigir el millón de amistades.
- En una red social que no tiene definida ninguna relación de amistad, el usuario con más amigos es cualquiera de los que pertenecen a la red (Recordar que necesariamente siempre hay uno).
- Si hay dos usuarios en la red que no dieron like a ninguna publicación y se pregunta si dieron like a las mismas pubicaciones, la respuesta es **Verdadera**. También es verdadero que a un usuario le gustan las mismas publicaciones que a ese mismo usuario.
- Las funciones del TP que toman como uno de sus parámetros a una Red Social y como otro a un Usuario, tienen por dominio el conjunto de usuarios de la red. Es decir, si el usuario pasado no está en la red, el comportamiento no está definido. Se puede dar un error, fallar por *non-exhaustive patterns*, dar un valor inválido, etc.