Trabajo Práctico

Combinatoria y recursión para la genética de plantas

Taller de Álgebra
Verano 2018

Fecha límite de entrega:
9 de marzo hasta las **23:59** hs.
Coloquio: Miércoles 14 de Marzo

Introducción

Si bien en Argentina y Brasil la semilla de arveja que se consume mayoritariamente es de color verde, en el mundo es mucho más popular el consumo de arveja amarilla. Preocupados por aumentar sus exportaciones, los productores de arvejas argentinos decidieron acudir al INTA (Instituto Nacional de Tecnología Agropecuaria) para que les explicara por qué a veces en sus campos crecían arvejas verdes y a veces amarillas.

Este trabajo consiste en ayudar al INTA a explicarle a los productores cómo funciona la genética de la planta a través de modelar la teoría de Gregor Mendel, quien en 1865 experimentó con arvejas para tratar de romper con la tradición de ese entonces que creía que el resultado de mezclar una planta de arvejas verdes con una de arvejas amarillas debía dar cómo resultado una planta de arvejas amarillo-verdosas.

Parte 1 - Modelo

Llamaremos **fenotipo** a cualquier rasgo o característica observable de la planta. Por ejemplo, el color de su semilla, el color de su flor, el tipo de tallo, etc.

Cada fenotipo está determinado por su **genotipo**, que es el par de variantes de un mismo gen que tiene la planta. A estas variantes se las conoce como **alelos**.

Por ejemplo para el gen que determina el color de la semilla, tenemos dos alelos: el alelo Y y el alelo y. El alelo Y produce semillas color amarillo, mientras que el alelo y produce semillas color verde.

Dijimos que cada planta tiene dos alelos de un mismo gen. Si la planta tiene dos versiones iguales del alelo, es decir su genotipo es YY o yy, entonces la planta tendrá semillas color amarillo o verde respectivamente. Ahora, si la planta tiene dos alelos distintos, Yy, entonces será visible el rasgo del **alelo** dominante. En el caso del color de la semilla, el alelo dominante es Y, por lo tanto la planta tendrá semillas de color amarillo.

Si bien la planta de la arveja tiene genotipos con dos alelos (se llama diploide), para ser más general, nuestro modelo deberá soportar genotipos de uno, dos o más alelos.

En general, se conoce qué alelo es el dominante sobre otro del mismo gen para distintos genes. En nuestro caso, diremos que un alelo domina a otro si su fuerza de dominación es mayor que la del otro alelo.

En Haskell, este modelo lo representaremos con los siguientes tipos:

```
type Gen = Char
type Rasgo = [Char]
type Dominacion = Integer
type Alelo = (Gen, Rasgo, Dominacion)
type Set a = [a]
type Genotipo = Set Alelo
```

Ejercicio 1

Dar el tipo y definir la función `crearAlelo` que dado un `Gen`, el `Rasgo` que determina y su factor de `Dominacion`, cree el `Alelo` correspondiente.

Ejercicio 2

Dar el tipo y definir la función `obtenerGen` que dado un `Alelo`, devuelve el `Gen` correspondiente (en nuestro caso siempre identificaremos un gen con una letra mayúscula, independientemente de a qué alelo corresponda)

Ejercicio 3

Dar el tipo y definir la función `obtenerRasgo` que dado un `Alelo`, devuelve el `Rasgo` que determina.

Ejercicio 4

Dar el tipo y definir la función `obtenerValorDominacion` que dado un `Alelo`, devuelve su factor de `Dominacion`.

Ejercicio 5

Dar el tipo y definir la función `alelosDelGen` que dada una lista de alelos, y un gen, devuelva todos los alelos correspondientes a ese gen.

Ejercicio 6

Dar el tipo y definir la función `rasgosPosiblesDelGen` que dada una lista de alelos, y un gen, devuelva todas las variantes de rasgos posibles que puede determinar ese gen.

Ejercicio 7

Dar el tipo y definir la función `genesPosibles` que dada una lista de alelos, devuelva todos los genes que contiene la lista, sin repeticiones.

Ejercicio 8

Dar el tipo y definir la función `crearGenotipo` que dados dos alelos, cree el genotipo correspondiente.

Ejercicio 9

Dar el tipo y definir la función `genDelGenotipo` que dado un genotipo, devuelva el gen asociado.

Ejercicio 10

Dar el tipo y definir la función `genotiposIguales` que dados dos genotipos, determine si son el mismo. Recordar que nuestro modelo debería poder soportar genotipos de uno, dos o más alelos.

Ejercicio 11

Dar el tipo y definir la función `aleloDominante` que dado un genotipo, devuelve el alelo que domina la relación.

Ejercicio 12

Dar el tipo y definir la función `rasgo` que dado un genotipo, devuelve el rasgo que finalmente es visible.

Parte 2 - Plantas

Cada planta, tiene un conjunto de genotipos (pero sólo uno de cada gen) y estará definida con el siguiente tipo:

```
type Planta = Set Genotipo
```

Ejercicio 13

Dar el tipo y definir la función `genesDePlanta` que dada una planta, devuelve el conjunto de genes de la planta

Ejercicio 14

Dar el tipo y definir la función `genotipo` que dada una planta y un gen, devuelve el genotipo asociado a ese gen en la planta.

Ejercicio 15

Dar el tipo y definir la función `rasgosDePlanta` que dada una planta, devuelve el conjunto de rasgos que son efectivamente visibles en la planta.

Ejercicio 16

Dar el tipo y definir la función `plantasIguales` que dadas dos plantas, determina si tienen los mismos genotipos.

Ejercicio 17

Dar el tipo y definir la función `plantasIgualesALaVista` que dadas dos plantas, determina si a la vista son indistinguibles, es decir, si tienen los mismos rasgos.

Parte 3 - Descendencia

A la hora de cruzar dos plantas madres para generar una planta hija, cada planta madre aporta uno de sus dos alelos para cada genotipo (cualquiera de los dos alelos puede ser heredado por la planta hija). De esa manera, la planta hija también contiene dos alelos para cada genotipo al igual que el resto de las plantas (uno heredado de cada planta madre).

A modo de ejemplo, para el gel que determina el color de la semilla, si las dos plantas madres tienen el genotipo Yy, los posibles genotipos descendientes son YY, yy o Yy (o yY que es equivalente dado que son conjuntos).

Es interesante notar que una planta con un genotipo Yy tiene un color de semilla amarilla. Por lo tanto, cruzar dos plantas con color de semilla amarilla (si su genotipo es Yy), puede resultar en una planta hija con semillas color verde, es decir, con genotipo yy. Este resultado fue lo que dio por tierra con las viejas teorías de que la descendencia heredaba un promedio de los colores de sus madres, y dio lugar a las leyes de Mendel.

Ejercicio 18

Dar el tipo y definir la función `puedeSerDescendienteDe` que dadas dos plantas, determina si la primera puede ser hija de la segunda, es decir, si para cada gen de la primera, al menos uno de los alelos correspondiente existen en la segunda.

Ejercicio 19

Dar el tipo y definir la función `crucesGenotipo` que dadas dos plantas madres y un gen, determina el conjunto de los posibles genotipos de su descendencia para dicho gen.

Pautas de Entrega

Para cada ejercicios deberá exhibir algunos ejemplos que muestren que la funcionalidad provista es correcta con respecto al enunciado.

Se debe enviar un e-mail conteniendo el código fuente en Haskell a la dirección algebra1-doc@dc.uba.ar. Dicho mail debe cumplir con el siguiente formato:

- El asunto debe ser "`[TPALGEBRA]`" seguido de los números de libreta de los integrantes del grupo[1] separados por guiones. Por ejemplo: `[TPALGEBRA]52/17-15/17`
- En el cuerpo del e-mail deben indicar: nombre, apellido, libreta (o DNI) y dirección de e-mail de cada integrante.
- El código Haskell debe estar adjunto al e-mail. El adjunto debe tener el nombre **tp.hs** y debe contener todas las funciones pedidas, **con el mismo nombre con que las presentamos**. El código debe poder ser ejecutado en GHCI sin ningún paquete especial.
- No es necesario entregar un informe sobre el trabajo.

Los objetivos a evaluar en el código de este trabajo prático son:

- Corrección.
- Declaratividad.
- Prolijidad: evitar repetir código innecesariamente y usar adecuadamente las funciones previamente definidas (tener en cuenta tanto las funciones definidas en el enunciado como las definidas por ustedes mismos).
- Uso adecuado de las técnicas vistas en clase como recursión o pattern matching.

Las sugerencias de los ejercicios pueden ayudar, pero no es obligatorio seguirlas.

Pueden definirse todas las funciones auxiliares que se requieran. Cada una debe tener un comentario indicando claramente qué hace.

Importante Se admitirá un único envío, sin excepción. Por favor planifiquen el trabajo para llegar a tiempo con la entrega.

[1] en caso de no tener libreta, el DNI

Referencias del lenguaje Haskell

- **The Haskell 2010 Language Report:** la última versión oficial del lenguaje Haskell a la fecha, disponible online en `http://www.haskell.org/onlinereport/haskell2010`.
- **Learn You a Haskell for Great Good!**: libro accesible, para todas las edades, cubriendo todos los aspectos del lenguaje, notoriamente ilustrado, disponible online en `http://learnyouahaskell.com` y en `http://aprendehaskell.es/` (en español).
- **Real World Haskell**: libro apuntado a zanjar la brecha de aplicación de Haskell, enfocándose principalmente en la utilización de estructuras de datos funcionales en la "vida real", disponible online en `http://book.realworldhaskell.org/read`.
- **Hoogle**: buscador que acepta tanto nombres de funciones y módulos, como signaturas y tipos *parciales*, online en `http://www.haskell.org/hoogle/`