Práctica 1. Especificación de predicados en Dafny.

EDA. Grupos E y F.
Profesor: Isabel Pita.

Comenzamos practicando la definición de predicados, que luego se utilizarán para escribir las especificaciones.

1. Dafny está instalado en el entorno Visual Studio 2013. Busca la aplicación Visual Studio 2013 (debe ser esta version) y ábrela.

2. Crea un proyecto vacío de C++.

3. Obtén del CV los ficheros `plantillaPredicados.dfy` y `specFunctionsEDA.dfy`.

   a) En el primero se encuentran las plantillas de los predicados que vamos a escribir durante la práctica. Añádelo al proyecto. Observa que se encuentra todo comentado excepto el primer predicado. Esto es debido a que las plantillas sin rellenar son incorrectas para Dafny y te producirían errores. Según vayas avanzando en la práctica ve moviendo el comienzo del comentario (`/*`) a los predicados o métodos siguientes

   b) En el segundo se encuentran definidos predicados y lemas que van a permitir comprobar la corrección de tus predicados. Este archivo se supone oculto, no necesitas consultarlo. Si copias los predicados de este archivo, habrás conseguido tenerlos todos correctos, pero no habrás aprendido nada.

   Añade este archivo al proyecto, ya que se necesita para realizar las pruebas, pero no lo abras en el editor, no necesitas consultarlo.

4. Observa que la extensión de los ficheros que se utilizan en Dafny es `.dfy`.

5. Una vez añadidos los ficheros observa que en el menú de Visual Studio ha aparecido una nueva opción: DAFNY.

6. El primer predicado que vamos a definir indica si un número es par.

   a) Escribe el predicado donde se encuentran los puntos suspensivos.

   ```
   predicate par (n : int)
   requires n > 0
   ensures par(n) == ( ........ )
   ```

   b) Dafny nos indica si el predicado está bien escrito, esto es, si la sintaxis empleada es correcta. Para ello, una vez escrito el predicado, sálvalo (Ctrl+s) y Dafny mostrará una línea verde a la izquierda del código. Si existe algún error de sintáxis, Dafny subraya la zona del error, y en la ventana de `Lista de errores` muestra un comentario indicando el error y la línea en que se produce. Prueba a escribir algo incorrecto, por ejemplo poniendo el símbolo = en lugar del símbolo de igualdad (==). Veras como Dafny subraya la zona incorrecta y en la ventana `Lista de errores` se muestra el error `rbrace expected`.

   c) Una forma de probar que el predicado es correcto, es decir, que expresa lo que queremos, es probar que es equivalente a uno que sepamos que es correcto. Si Dafny es capaz de hacer la prueba entonces nuestro predicado está bien. Sin embargo, normalmente no es capaz de probar automáticamente la equivalencia y habría que escribir lemas para ayudar a la prueba, lo que se encuentra fuera del ámbito del curso. El profesor proporciona en el fichero `specFunctionsEDA.dfy` algunos predicados correctos para hacer la equivalencia. Estos predicados se llaman siempre con la palabra `Referencia` o `Ref`, por ejemplo `parReferencia`. Por ello no debes utilizar estas palabras en tus predicados.

   Para hacer la prueba se utiliza el método `TestPar` del archivo `plantillaPredicados.dfy`.

d) La clausula `assert` que aparece en el método `TestPar`, prueba que el predicado que se escribe a continuación es correcto. En el primer caso prueba que para todo valor de `n` positivo, el predicado `parReferencia` es equivalente al predicado `par` que has escrito.

e) Descomenta el primer `assert` y salva el archivo. Comprueba que se muestra la línea verde a la izquierda y que no aparecen errores en la ventana de *Lista de errores*. Tu predicado es correcto.

f) Vamos a ver que haríamos cuando la prueba de equivalencia con el predicado propuesto por el profesor no sea válida Esto no significa que el predicado sea incorrecto, puede ocurrir que el sistema no sea capaz de probar la equivalencia. Daremos algunos casos de prueba utilizando valores que cumplen el predicado y otros que no lo cumplen. Esto nos dará alguna pista sobre la corrección del predicado. Con estas pruebas no aseguramos que el predicado exprese realmente lo que se espera, pero podemos aproximarnos a ello buscando casos de prueba significativos.

g) Ve poniendo comentarios y descomentando cada uno de los `assert` que se encuentran en el método `TestPar` y comprueba para cada uno de ellos si Dafny lo da como correcto (raya verde a la izquierda) o si aparece un punto rojo. En este caso mira la descripción del error en la ventana *Lista de errores* para familiarizarte con Dafny.

```
assert par(2);  // Es correcto
assert par(3);  // Error: assertion violation
assert par(0);  // Error: possible violation of function precondition
assert par(-2); // Error: possible violation of function precondition
assert par(-5);  // Doble error
assert par(100000);  // Correcto
```

7. Ahora vamos a escribir un predicado que indique que los elementos de una secuencia son todos positivos.

   a) Descomenta del fichero `plantillaPredicados.dfy` la plantilla del predicado `positivo`. Deja comentado el resto del fichero que todavía no hemos utilizado.

   b) Escribe el predicado correspondiente

   c) Descomenta el método `TestPositivo` para intentar comprobar la equivalencia de tu predicado con alguno de los propuestos por el profesor. Si la prueba no sale correcta realiza los pasos siguientes.

   d) Descomenta el método `TestPositivoSeq`. Ve probando los `assert` de uno en uno como hiciste en el caso del predicado `par`.

   e) Si falla algún `assert` cuyo caso sea correcto, debes corregir el predicado.

8. Escribir un predicado que indique que los elementos de una secuencia son todos iguales.

   a) Descomenta del fichero `plantillaPredicados.dfy` la plantilla del predicado `iguales`. Deja comentado el resto del fichero que todavía no hemos utilizado.

   b) Escribe el predicado correspondiente

   c) Descomenta el método `TestIguales` para intentar comprobar la equivalencia de tu predicado con alguno de los propuestos por el profesor. Si la prueba no sale correcta realiza los pasos siguientes.

   d) Descomenta el método `TestIgualesSeq`. Ve probando los `assert` de uno en uno como hiciste en el caso del predicado `par`.

   e) Si falla algún `assert` cuyo caso sea correcto, debes corregir el predicado.

9. A continuación se explican varias formas de escribir el predicado iguales para que el alumno aprenda diversas formas de escribir predicados. Esta sección no debe consultarse hasta no haber resuelto la anterior.

   a) Definición del predicado `iguales` utilizando dos variables ligadas. Cada una de ellas indica una posición del vector.

```
predicate iguales (a : seq<int>)
ensures iguales(a) ==  forall u, w :: 0 <= u<|a| && 0<=w<|a| ==> a[u]==a[w]
```

b) Podemos también definir el predicado limitando los valores de las variables ligadas `u` y `v`.

```
predicate iguales (a : seq<int>)
ensures iguales(a) == forall u, w :: 0 <= u < w < |a| ==> a[u] == a[w]
```

c) O utilizando una única variable ligada. En este caso hay que tener cuidado para que la variable ligada no se salga del rango. Observad que no se puede utilizar la expresión `u+1` directamente en la secuencia, como si se tratara de un vector. Hay que tomar la posición `u+1` de la secuencia entera.

```
predicate iguales (a : seq<int>)
ensures iguales(a) == forall u :: 0 <= u < |a| -1 ==> a[u] == a[..][u+1]
```

d) También podemos limitar la variable por el comienzo del intervalo

```
predicate iguales (a : seq<int>)
ensures iguales(a) == forall u :: 0 < u < |a| ==> a[..][u-1] == a[u]
```

10. Escribir un predicado que indique que los elementos de una secuencia son todos distintos.

    a) Descomenta las plantillas correspondientes al predicado `distintos`.

    b) Escribe el predicado y pruébalo primero con el método `TestDistintos` y si no obtienes la respuesta correcto, utiliza el método de prueba `TestDistintosSeq`.

11. Siguiendo esta misma metodología escribe los siguientes predicados y pruébalos.

12. Escribir un predicado que indique que no existen dos elementos consecutivos iguales en una secuencia.

13. Escribir un predicado que indique que una secuencia es estrictamente creciente. Este predicado se puede escribir utilizando dos variables ligadas o solo una. Intenta resolverlo de las dos formas.

14. Escribir un predicado que indique que una secuencia está ordenada. (Es creciente pero no estrictamente creciente)

15. Escribir un predicado que indique que un número `p` es primo. Un número es primo cuando no tiene divisores menores que él. En este caso Dafny puede indicar un warning indicando *No terms found to trigger on.* Se desconoce el motivo por el que aparece el warning, pero no afecta al predicado.