Análisis de Algoritmos

Natalia Baeza - Nadina Martínez Carod

30 de septiembre de 2024

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Verificación de programas con Dafny

1.1. Introducción

Dafny es un lenguaje de programación *verification-aware* que tiene soporte nativo para registrar especificaciones de código y está equipado con un verificador de programa estático. Combinando razonamiento automatizado sofisticado permite escribir código probablemente correcto (esto es, definiendo en el código las especificaciones de código). Tiene un entorno de desarrollo o se integra a entornos de desarrollo para trabajar de forma similar que con otros lenguajes de programación tales como Java o Python, lo que permite integrarlo de forma simple a tu workflow. Dafny hace verificación rigurosa como parte integral del desarrollo, reduciendo las incidencias en etapas tardías y costosas que pueden ser transparentes para el testing. [1] Además de tener una máquina de verificación para checkear las especificaciones al momento de la implementación, el ecosistema Dafny incluye varios compiladores, plugins para IDEs de desarrollo de software conocidos, un servidor de lenguaje basado en LSP, un formateador de código, un manual de referencia, tutoriales, tips para el usuario, libros, experiencias de profesores enseñando Dafny, y el expertise acumulado de los proyectos de la industria que usan Dafny.

Conceptualmente Dafny soporta aspectos como los citados a continuación:

1. operaciones matemáticas entre enteros y reales, vectores de bits, clases, iteradores, arreglos, tuplas, tipos genéricos, refinamiento y herencia.
2. tipos de datos inductivos que pueden tener métodos y son apropiados para el uso de patrones.
3. tipos de subconjunto
4. expresiones lambda
5. estructuras mutables e inmutables

Por otra parte, Dafny también ofrece una caja de herramientas para pruebas matemáticas relacionadas al software, incluyendo:

1. cuantificadores ligados y no ligados
2. pre- y post-condiciones, condiciones de terminación, loops, invariantes y especificaciones de lectura-escritura.
3. pruebas de cálculo y la habilidad de usar y probar lemmas.

Dafny es una herramienta que permite construir código involucrando a la verificación de software desde el primer momento. Dicha herramienta incluye un lenguaje de programación especializado en la verificación formal de código. El usuario escribe tanto la especificación como la implementación de los programas. Por otra parte soporta programación orientada a objetos, imperativa, secuencial, soporta clases genéricas y asignación dinámica y la escritura de algoritmos funcionales. La especificación de los algoritmos se construye incluyendo pre y post-condiciones y determinando una métrica de terminación. Los programas hechos con Dafny se verifican estáticamente. La verificación de un programa funciona compilando el programa Dafny a un lenguaje de verificación intermedio denominado **Boogie**, tal que si Boogie asegura la correctitud de un programa, también lo hace en Dafny. Boggie genera la verificación de las condiciones mediante su motor de razonamiento lógico.

Dafny es un lenguaje diseñado para hacer más fácil la escritura de código correcto. Esto refiere a correcto en el sentido de no tener errores en tiempo de ejecución, sino hacer el código correcto al momento en que el programador entiende qué es esto. Para lograrlo, Dafny descansa en sus anotaciones de alto nivel para razonar a partir de purebas de correctitud de código. El efecto de una pieza de código puede darse abstractamente, usando expresiones de alto nivel naturales del comportamiento deseado, el cual es más fácil y menos propenso a error para escribir. Dafny entonces genera una prueba donde el código debe cumplir las anotaciones (asumiendo que las anotaciones son correctas. Dafny deja lo complicado de escribir código libre de errores en que las anotaciones son libre de errores. Esto hace más fácil que escribir el código porque las anotaciones son cortas y más directas. Además de la correspondencia de la prueba, Dafny prueba que no hay errores en tiempo de ejecución, tales como indices fuera de los límites, referencias nulas, división por cero, etc. Esta garantía es poderosa, una ventaja de Dafny. Además provee terminación de código excepto en bucles especialmente diseñados.[2]

1.1.1. Características principales

Dafny tiene como objetivo principal facilitar la creación de código correcto y confiable. Para lograrlo, el lenguaje se basa en anotaciones de alto nivel que permiten *razonar* y demostrar la corrección del código permitiendo establecer en la declaración de métodos post-condiciones **ensures** que especifican las garantías que un método debe cumplir *después* de su ejecución mientras que las pre-condiciones **requires** establecen las condiciones que deben ser verdaderas *antes* de invocar el método. Además, Dafny incorpora aserciones **asserts** que son declaraciones

que deben ser *verdaderas* en un punto particular del programa e invariantes **invariant** que son condiciones que deben *mantenerse* a lo largo de la *ejecución* para reforzar la verificación del código. Estos elementos permiten garantizar la *integridad y correctitud* del código en cada etapa de su ejecución.

En la versión actual de Dafny, integrada en el editor de código *Visual Studio Code*, se utiliza una representación visual en el margen izquierdo para ilustrar el estado de verificación del código. Esta representación incluye:

- Una barra vertical delgada de color verde para indicar código que ha sido verificado correctamente.
- Un rectángulo rojo que señala errores en el código.
- Dos líneas amarillas verticales que delimitan el contexto del error, proporcionando una referencia visual clara.
- Un círculo con una marca de verificación para indicar aserciones parcialmente probadas.

1.1.2. Beneficios en su uso

Algunas de las ventajas destacadas del uso de Dafny o cualquier lenguaje de esta índole en la práctica de programación son:

- Verificación Formal de Programas: Dafny ofrece un enfoque de verificación formal que permite demostrar matemáticamente la ausencia de errores en tiempo de ejecución en un programa.
- Mejoras en la confiabilidad del código.
- Reducción de costos a largo plazo. La implementación requiere un esfuerzo inicial adicional, pero presenta un ahorro significativo al prevenir errores difíciles de corregir en etapas posteriores de desarrollo
- El verificador está integrado directamente con el lenguaje
- El lenguaje permite retornar más de una salida en los métodos.
- La sintaxis es similar a otros lenguajes imperativos como C, C o Java.
- Se pueden usar y combinar operadores matemáticos, lógicos y cuantificadores en las verificaciones.

1.1.3. Desafíos y limitaciones Potenciales

Algunos problemas que podrían encontrarse al utilizar Dafny son:

- Requiere práctica, destreza y costumbre a este enfoque de desarrollo, razonando las aserciones.
- Incrementa el tiempo de desarrollo, requiera más atención y detalle en comparación con métodos de desarrollo convencionales.
- El código se vuelve más complejo de interpretación y desarrollo.
- En la realidad se prioriza su uso en códigos complejos críticos funcionalmente en áreas financieras o de integridad de datos o vida humana.
- Hay escasa documentación, tutoriales, foros en comparación con otros lenguajes.
- Curva de Aprendizaje - Tiempo para capacitarse y ajustarse a las prácticas de desarrollo específicas de Dafny (verificación formal o con la sintaxis).
- Dificultad en la verificación de código existente, puede ser difícil y requerir esfuerzos considerables.

1.2. Ejemplos de implementación

1.2.1. Código 1

El cálculo del cuadruple de un número natural usando una función que calcula la suma de dos números naturales.

```
1   method Main ()
2  {
3   var nro : int :← 5;
4   var resultadoFinal : int :← sumaNaturalesConCero (nro, nro) * 2;
5   print resultadoFinal ;
6   assert resultadoFinal = nro * 4;
7  }
8   method sumaNaturalesConCero ( nro1 : int, nro2 : int) returns (
        resultado : int )
9   requires  nro1 = 0 || nro1 >0 // Pre - condicion
10  requires  nro2 = 0 || nro2 >0 // Pre - condicion
11  ensures resultado = nro1+nro2 // Post - condicion
12 {
13  if ( nro1 = 0 ){// nro1 es cero
14     resultado :← nro2 ;
15  } else if ( nro2 = 0 ){
16         resultado :← nro1;
```

```
17          } else {// nro1 > 0 and nro2 > 0
18                  resultado :← nro1 + nro2;
19                  }
20   }
21   }
```

1.2.2. Código 2

La serie sumatoria de n naturales La fórmula para la suma de los primeros n n´umeros naturales es:

$$\sum_{k=1}^{n} k = n \cdot (n+1)/2$$

```

 1 /* La formula para la suma de los primeros n numeros naturales
 2 denominada Serie aritmetica que es igual a n*(n +1) / 2 */
 3 /* Ejemplos :
 4 n:← 5, serie ← 15
 5 n:← 10 , serie ← 55
 6 n:← 100 , serie ← 5050 */
 7
 8 method Main ()
 9 {
10 var n : nat :← 100;
11 var resultadoSerie : nat :← CalculoSerieAritmetica (n);
12 print resultadoSerie ;
13 assert resultadoSerie = ( n *( n +1) ) /2;
14
15 }
16
17 method CalculoSerieAritmetica (n: nat ) returns ( sumatoria : nat )
18 requires n >:←1 // Pre - condicion
19 ensures sumatoria = (n *( n +1) ) /2 // Post - Condicion
20 {
21 sumatoria :← 0;
22
23 var i: nat :← 1;
24
25 while i <<:← n
26 invariant 1 <<:← i <<:← n +1
27 invariant sumatoria = i * (i -1) /2
28 decreases n - i
29 {
30 sumatoria :← sumatoria + i;
31 i :← i + 1;
32 }
33 }
```

Se muestran otros ejemplos aquí [3].

1.3. Conclusiones

Dafny es una herramienta que facilita la puesta en práctica de los conceptos teóricos de verificación formal. Como toda herramienta tiene beneficios y debilidades o desafíos en su uso. Es importante conocer ambos para su correcta aplicación y uso. Es importante desarrollar software entendiendo qué es lo que se pide y las especificaciones de código definidas formalmente ayuda al pensamiento computacional, a la claridad del desarrollo a reducir los errores en tiempo de ejecución y generar software de calidad. Es importante tomar el tiempo necesario para esto y esto denota una curva de aprendizaje de crecimiento lento. Pensar a nivel de predicados y desarrollar confiando en la especificación es la premisa del desarrollo de software de calidad. El ambiente interactivo facilita el uso de la herramienta y permite llevar adelante una programación con verificación estática durante el dinamismo de la implementación de la solución.

Respecto al uso de la herramienta en ambientes productivos de software requiere un tiempo de aprendizaje que debe ser justificado. El uso de la misma en código Legacy puede no mostrar los beneficios en tiempo y forma perjudicando a las estimaciones de desarrollo de producto. Por otra parte la comunidad Dafny es pequeña en el mundo del software y tiene uso a nivel académico principalmente.

1.4. Referencias

[1]Referencia oficial de Dufny - Manual de usuario
[2] Getting Started with Dafny: A Guide
[3]Especificación y verificación de Código - Parte 0 - K. Rustan M. Leino