

Introducción a los Algoritmos - 1er. cuatrimestre 2015

Guía 1: Expresiones y funciones. Satisfactibilidad y validez

Objetivos

El objetivo de esta guía es brindar una introducción al formalismo básico que se utilizará en la materia. Introduciremos los conceptos principales involucrados apoyándonos en un terreno conocido como es la aritmética. Definimos expresiones y fórmulas aritméticas, funciones; y trabajamos con los conceptos de precedencia, tipado, validez y satisfactibilidad.

1. Analizá con tus compañeros las siguientes oraciones:

Ea oaió o iee ooae iea ue a iuiee o iee oae.

St rcn n tn vcls mntrs q l ntrr n tn cnsnnts.

- a) ¿Las comprendés?
- b) ¿Una computadora los podría comprender?
- c) ¿Cuál es la diferencia entre una persona y una computadora?
- d) ¿Qué características debería cumplir una expresión para ser “comprendida” por una computadora?

Comenzando con la aritmética

Comenzaremos trabajando con los conceptos matemáticos con los que más familiarizado estás: los números y la aritmética, pero vamos a estudiarlos desde una perspectiva formal. En primer lugar, presentaremos la **sintaxis** que nos permitirá construir expresiones. La sintaxis se ocupa de definir qué símbolos podremos utilizar y de la forma correcta en que deben estar dispuestos para armar expresiones.

Nuestras **expresiones** podrán tener la siguiente estructura:

- Un número, por ejemplo: 1, 35, 8. A estas expresiones les llamaremos **constantes**.
- Una letra, por ejemplo: x , y , z . A estas expresiones les llamaremos **variables**.
- Combinando dos expresiones con un **operador**, y encerrando la expresión resultante entre paréntesis, podemos construir expresiones más complicadas. Por el momento los operadores que utilizaremos serán $+$, $-$, $*$, $/$, $^$ (este último símbolo corresponde a la potenciación). Por ejemplo, combinando la variable z con la constante 6 a través del operador $-$ se puede formar la expresión: $(z - 6)$. Podremos eliminar los paréntesis superfluos según las reglas que analizaremos más adelante.
- A su vez, cada una de estas expresiones puede volver a combinarse con otras formando expresiones más complejas, por ejemplo: $((x + y)/(2^x)) + (6/3)$.

Para poder expresar propiedades como $2 + 1$ es igual a la expresión 3, o que x es menor a $x + 1$ agregaremos símbolos de **relación**.

- Cuando combinamos dos expresiones usando un símbolo de relación decimos que construimos una **fórmula**. Una fórmula es un tipo particular de expresión. Los símbolos de relaciones que consideraremos por el momento serán: $=$, \neq , $<$, \leq , $>$ y \geq . Por ejemplo, si tomamos las expresiones $2 * 6$ y $x + 7$ y el símbolo de relación $=$, podemos construir la fórmula $2 + 6 = x + 7$.

2. Construí dos expresiones con al menos dos operadores. ¿Cómo podrías justificar que tus expresiones están bien construidas?
3. Presentá dos expresiones que estén mal construidas y da razones para justificar porqué considerás que están mal construidas.

4. Construí una fórmula de acuerdo con las reglas precedentes. ¿Cómo podrías justificar que está bien construida?

Precedencia y tipado

La **precedencia** de los operadores nos permite escribir fórmulas grandes de manera simple y más legible. Cuando un operador tiene mayor precedencia que otro, podemos escribir una fórmula que involucra a ambos sin necesidad de poner paréntesis, y a pesar de esto, la fórmula tiene un sentido único. Un ejemplo conocido por todos es el caso de la suma respecto a la multiplicación. El operador $*$ tiene mayor precedencia que $+$ y por ello es que normalmente interpretamos la expresión $2 + 4 * 3$ como $2 + (4 * 3)$. Esta regla es la que nos permitía en la primaria “separar en términos” una expresión algebraica. Al mismo tiempo, y por la misma regla, sabemos que no es lo mismo la expresión $(2 + 4) * 3$ que $2 + 4 * 3$. No siempre los paréntesis pueden sacarse indiscriminadamente sin cambiar el sentido de la expresión.

De la misma manera, cuando un operador **asociativo** se aplica múltiples veces es posible eliminar paréntesis, ya que el orden en que se efectúa la operación no altera el resultado. Por ejemplo $(2+4)+7$ es igual a $2+(4+7)$ y por lo tanto podemos escribir simplemente $2 + 4 + 7$.

Los operadores $+$, $*$ son asociativos.

tabla de precedencia	más arriba → mayor precedencia
\wedge	potencia
$*, /$	producto y división
$+, -$	suma y resta
$=, <, \leq, >, \geq$	operadores de comparación

Por otro lado, la noción de **tipado** nos permite distinguir expresiones que están “bien escritas”, es decir que tienen sentido, que representan algún valor. El tipo de un operador o función nos dice cuál es la clase de valores que toma y cuál es la clase del valor que devuelve. En las expresiones algebraicas en general todos los valores son de tipo número, que denotamos con *Num* (y los **subtipos** *Nat*, *Int*, ...). Como ejemplo tomemos el caso de la suma: tanto a la izquierda del símbolo “ $+$ ” como a la derecha debe haber expresiones que representen números, y el resultado total también es un número.

Además del tipo *Num* utilizaremos el tipo de los valores booleanos, que denotamos con *Bool*. Este tipo incluye las constantes *True* y *False* que representan las nociones de verdadero y falso respectivamente. Los valores booleanos son importantes porque son el resultado de los operadores de comparación $=, <, \leq, >, \geq$.

Las nociones de precedencia y tipado son importantes para asegurar que escribimos expresiones que tienen un sentido único y bien definido. Los siguientes ejercicios tratan sobre estas nociones en el marco de las expresiones algebraicas.

5. Sacá todos los paréntesis que sean *superfluos* según las reglas de precedencia y asociatividad de los operadores aritméticos. Por ejemplo:

$$\begin{aligned} & (8 - 6) * x = (6 * (x^2)) + 3 \\ \equiv & \{ \text{precedencia de } * \text{ por sobre } + \} \\ & (8 - 6) * x = 6 * (x^2) + 3 \\ \equiv & \{ \text{precedencia de } ^2 \text{ sobre } * \} \\ & (8 - 6) * x = 6 * x^2 + 3 \end{aligned}$$

$$\begin{aligned} a) & -(5 + x) + ((3 * 6)/(4 * 5)) * (8 * 5) \\ b) & (((2)^2 + 5) - (4 * 2)/(4)) + 1 = ((5 * x)/8) + (3 * 5)^3 \end{aligned}$$

6. Introducí paréntesis para hacer *explícita* la precedencia.

$$\begin{aligned} a) & 5 * 3 + 4 \geq 7 - 7 + 3 \\ b) & 3 + 4 * x = 4 \end{aligned}$$

7. Escribí el *tipo* de los siguientes operadores y funciones: $*, /, ^2, \leq, \geq, =$. ¿Cuál es el tipo de esos mismos operadores en `haskell`? ¿Qué sucede con el operador $-$?

Como ejemplo presentamos el caso del operador $+$. Este operador toma dos números y devuelve otro número. Podemos escribir su tipo en *notación funcional* listando el tipo de los parámetros y a continuación el tipo resultado:

$$+ : Num \rightarrow Num \rightarrow Num$$

Otra forma de escribir el tipo de este operador, útil para armar el árbol de tipado de una expresión, es en *notación de árbol*:

$$\frac{Num + Num}{Num}$$

En `haskell` el operador (+) tiene el mismo tipo. En `ghci` podemos corroborarlo de la siguiente manera:

```
Main> :t (+)
(+) :: Num a => a -> a -> a
```

Esta respuesta se interpreta de la siguiente manera. El tipo del operador (+) es lo que está a la derecha de =>, es decir $a \rightarrow a \rightarrow a$, donde la variable a indica cualquier tipo. Por otro lado, lo que está a la izquierda de => nos indica que el tipo a es un subtipo de Num , es decir, Nat , Int .

Validez y satisfactibilidad

Dos cualidades asociadas a la semántica de una fórmula, es decir, a la interpretación o significado que le damos, son los conceptos de **satisfactibilidad** y **validez**. Estas propiedades ganan relevancia cuando se trata de fórmulas con variables. Decimos que una fórmula es **satisfactible** cuando es verdadera para algunos valores posibles de las variables. Decimos que es **válida** cuando es verdadera para todos los valores posibles de las variables.

Nuestro principal interés es distinguir cuándo una fórmula es válida y cuándo no. Como la validez es independiente de los valores de las variables involucradas, si queremos demostrar la validez de una fórmula no podemos hacer ninguna suposición sobre los valores de las variables. Por el contrario, cuando queremos demostrar que una fórmula es **no válida** es suficiente con encontrar al menos un valor que haga que la fórmula sea falsa: esto es un contraejemplo. Algunas veces además es posible demostrar directamente que la fórmula es falsa para todos los valores de las variables. En este caso decimos que la fórmula es **no satisfactible**, porque no existe ningún valor posible que haga que sea verdadera.

Importante: Cuando una fórmula es siempre verdadera la llamamos **válida**, cuando es verdadera para algunos valores la llamamos **satisfactible**, cuando es falsa para algunos valores la llamamos **no válida** y cuando es falsa para todos los valores la llamamos **no satisfactible**.

8. ¿Qué valor de la variable x *satisface* las siguientes ecuaciones, es decir, qué valor hace que las fórmulas sean verdaderas? Por ejemplo para la fórmula $6 * x + 8 = x + 3$:

$$\begin{aligned} & 6 * x + 8 = x + 3 \\ \equiv & \{ \text{restar } x \text{ en ambos miembros} \} \\ & \underline{6 * x} + 8 \underline{-x} = \underline{x} + 3 \underline{-x} \\ \equiv & \{ \text{aritmética} \} \\ & 5 * x + 8 = 3 \\ \equiv & \{ \text{restar } 8 \text{ en ambos miembros} \} \\ & 5 * x + \underline{8 - 8} = \underline{3 - 8} \\ \equiv & \{ \text{aritmética} \} \\ & 5 * x = -5 \\ \equiv & \{ \text{dividir por } 5 \text{ en ambos miembros, aritmética} \} \\ & x = -1 \end{aligned}$$

Luego la respuesta es que solamente $x = -1$ hace verdadera la fórmula $6 * x + 8 = x + 3$. En otros casos, puede haber más de un valor para x que satisfaga la fórmula; en ese caso hay que encontrarlos a todos. *Notá que cuando realizamos dos pasos a la vez, utilizamos un subrayado diferente para cada subexpresión involucrada.*

- a) $2 * x + 3 = 5$
 b) $x = x + 1$
 c) $x + x = 2 * x$

Preguntas:

- a) ¿Cómo se interpreta cada resultado?
- b) ¿qué relación hay entre estos ejercicios y los conceptos de **validez** y **satisfactibilidad**? es decir ¿cómo le llamaríamos a cada fórmula?

9. Demostrá que las siguientes ecuaciones son *válidas*, es decir que las fórmulas son verdaderas para cualquier valor que tome la variable x . Por ejemplo:

$$\begin{aligned} & 4 * x + 14 = 2 * (2 * x + 5) + 4 \\ \equiv & \{ \text{distributividad de } * \text{ con } +, \text{ def. de } * \} \\ & 4 * x + 14 = 4 * x + \underline{10 + 4} \\ \equiv & \{ \text{def. de } + \} \\ & 4 * x + 14 = 4 * x + 14 \\ \equiv & \{ \text{restar } 4 * x + 14 \text{ en ambos miembros } \} \\ & \underline{4 * x + 14 - (4 * x + 14) = 4 * x + 14 - (4 * x + 14)} \\ \equiv & \{ \text{reflexividad de } = (x = x) \} \\ & \text{True} \end{aligned}$$

- a) $(x - 1) * (x + 1) = x^2 - 1^2$
- b) $(a + b)^2 = a^2 + 2 * a * b + b^2$

Preguntas

- a) ¿identificás estas fórmulas?
- b) ¿tienen nombre?
- c) ¿qué expresan?

10. Las siguientes ecuaciones no son válidas, es decir, para al menos algún valor de la variable x cada fórmula es falsa. Justificá dando un *contraejemplo*. Además, indicá cuáles son satisfactibles y qué valores las satisfacen.

Por ejemplo para la ecuación $3 * x + 1 = 3 * (x + 1)$.

Contraejemplo: $x = 5$ *falsifica* la fórmula ya que

$$\begin{aligned} & 3 * x + 1 = \underline{3 * (x + 1)} \\ \Rightarrow & \{ \text{tomando } x = 5 \} \\ & \underline{3 * 5 + 1 = 3 * 5 + 3} \\ \equiv & \{ \text{def. de } * \} \\ & \underline{15 + 1 = 15 + 3} \\ \equiv & \{ \text{def. de } + \} \\ & 16 = 18 \\ \equiv & \{ \text{igualdad de números } \} \\ & \text{False} \end{aligned}$$

- a) $x + (y * z) = (x + y) * (x + z)$
- b) $x + 100 = x$

11. Decidí si son *válidas* o *no válidas* y *satisfactibles* o *no satisfactibles* las siguientes (in)ecuaciones. Justificá en cada caso.

- a) $x^2 + y^2 = (x + y)^2$
- b) $(a - b) * (a + b) * ((a + b)^2 - 2 * a * b) = a^4 - b^4$
- c) $x^2 + 2 * x + 4 = 0$

Preguntas:

- a) ¿Qué diferencia existe entre dar un ejemplo y demostrar la validez de una fórmula?

- b) ¿Qué diferencia existe entre dar un contraejemplo y demostrar que una fórmula es equivalente a Falso? ¿Cuándo se puede utilizar cada *estrategia*?

12. Da ejemplos y una justificación apropiada de una fórmula:

- a) válida (y por lo tanto satisfactible).
- b) satisfactible pero no válida.
- c) no satisfactible (y por lo tanto no válida).

Evaluación

Cuando trabajamos con expresiones y fórmulas, muchas veces **simplificamos** las expresiones para que sean más simples. Este proceso de simplificación está asociado a la semántica o significado que le damos a la expresión. Por ejemplo, si tenemos la expresión $7+1$ la reemplazamos por 8 pues sabemos que ambas expresiones representen el mismo número. La noción de **evaluación** está más relacionada con las características sintácticas (asociado a los símbolos) de la expresión cumpliendo una función similar.

13. Simplificá las siguientes expresiones

- a) $70 * 47 / (3 + 2)$
- b) $(3 + 1) * 10 = 25$
- c) $2 * x + y = 1 + 2 * y$

14. Evaluá en `haske11` las expresiones del ejercicio anterior. ¿Qué sucede con la última expresión?

Funciones

En matemática, se dice que una magnitud o cantidad es función de otra si el valor de la primera depende exclusivamente del valor de la segunda. Por ejemplo, supongamos que queremos viajar desde Córdoba a Buenos Aires en auto. La duración del viaje T dependerá de la distancia d entre Córdoba y Buenos Aires y de la velocidad v que lleve el auto. Del mismo modo, el área A de un círculo es función de su radio r . Estas magnitudes a veces pueden vincularse a través de la proporcionalidad directa o ser inversamente proporcionales. Así, el área A de un círculo es proporcional al cuadrado de su radio, lo que se expresa con una fórmula del tipo $A = \pi * r^2$ y la duración de un viaje T es inversamente proporcional a la velocidad del vehículo ($T = d/v$). A la variable que se encuentra a la izquierda de estas fórmulas (el área, la duración) se la denomina variable dependiente, y a las variables de las que depende (el radio, la velocidad, la distancia) se las llama variables independiente.

De manera más abstracta, el concepto general de función se refiere a una regla que asigna a cada elemento de un primer conjunto un único elemento de un segundo conjunto. Por ejemplo, cada número entero posee un único cuadrado, que resulta ser un número natural (incluyendo el cero):

$$\begin{array}{ccccccc} \dots & -2 & -1 & 0 & 1 & 2 & 3 & \dots \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\ & +4 & +1 & 0 & +1 & +4 & +9 & \end{array}$$

Esta asignación constituye una función entre el conjunto de los números enteros y el conjunto de los naturales.

Diremos que una **función** le “asigna” a los elementos de un conjunto, llamado **dominio**, elementos de otro conjunto, llamado **codominio**. La notación:

$$f : A \rightarrow B$$

indica que f es una función con dominio A y codominio B . La definición de una función tiene la forma

$$f.x \doteq \langle \text{expresión que depende de } x \rangle$$

donde f es el nombre de la función y x es/son la/s variable/s independiente/s.

15. En las siguientes definiciones identificá las variables, las constantes y el nombre de la función

- a) $f.x \doteq 5 * x$
- b) $duplica.a \doteq a + a$
- c) $por2.y \doteq 2 * y$
- d) $multiplicar.zz.tt \doteq zz * tt$

16. Escribí una función que dados dos valores, calcule su promedio.
17. Tomando las definiciones del punto 15 evaluá las siguientes expresiones. Justificá cada paso utilizando la notación aprendida. Luego, controlá los resultados en `Haskell`.
- a) $(multiplicar.(f.5).2) + 1$
 - b) $por2.(duplica.(3 + 5))$
18. Tomando las definiciones en el punto 2 demostrá que *duplica* y *por2*, si son aplicadas al mismo valor, dan siempre el mismo resultado. En otras palabras, la expresión $duplica.x = por2.x$ es **válida**.

Tipado de funciones

Tomemos la función *g* definida así:

$$g.x.y \doteq 3x - x * y > 0$$

Esta función toma dos argumentos de tipo *Num* y devuelve un valor de tipo *Bool*. Así, el tipo de *g* se declara de la siguiente forma:

$$g :: Num \rightarrow Num \rightarrow Bool$$

Es decir, los tipos de los argumentos se listan primero, siguiendo el orden en el que serán llamados por la función, y en último lugar se coloca el tipo del resultado de evaluar la función.

19. Dar el tipo de las funciones del ejercicio 15 y el ejercicio 16.
20. Dar el tipo de las siguientes funciones:

- a) $g.y \doteq 8 * y$
- b) $h.z.w \doteq z + w$
- c) $j.x \doteq x \leq 0$

Definiciones de Funciones por casos

Hay ocasiones en las que una sola fórmula no alcanza para definir una función. Así, existen funciones que para un conjunto de argumentos requieren una definición y para otro conjunto de argumentos necesitan de otra definición diferente. Es el caso, por ejemplo, de la función *espar* que dado un natural devuelve *true* si el número es par o *false* si el número es impar.

Una **definición por casos** de una función tendrá la siguiente forma general:

$$f.x \doteq \left(\begin{array}{l} B_0 \rightarrow f_0 \\ \square B_1 \rightarrow f_1 \\ \vdots \\ \square B_n \rightarrow f_n \end{array} \right)$$

donde las B_i son expresiones de tipo booleano, llamadas **guardas** y las f_i son expresiones del mismo tipo que el resultado de *f*. Para un argumento dado el valor de la función se corresponde con la expresión cuya guarda es verdadera para ese argumento.

Al trabajar con expresiones booleanas se hace necesario incorporar a nuestro formalismo los operadores \wedge, \vee, \neg que pueden ser utilizados para combinar dos fórmulas para obtener una nueva fórmula. En `haske11` estos operadores se escriben `&&`, `||` y `not`, respectivamente.

21. Definí las funciones que describimos a continuación, luego implementalas en `haskell`. Por ejemplo:

Enunciado: $signo : Int \rightarrow Int$, que dado un entero retorna su signo, de la siguiente forma: retorna 1 si x es positivo, -1 si es negativo y 0 en cualquier otro caso.

Solución:

$$\text{signo}.x \doteq \left(\begin{array}{l} x > 0 \rightarrow 1 \\ \square \quad x < 0 \rightarrow -1 \\ \square \quad x = 0 \rightarrow 0 \end{array} \right)$$

En `haskell` se escribe así:

```
sgn :: Int -> Int
sgn x | x>0  = 1
      | x<0  = -1
      | x==0 = 0
```

- a) $entre0y9 : Int \rightarrow Bool$, que dado un entero devuelve *True* si el entero se encuentra entre 0 y 9.
- b) $rangoPrecio : Int \rightarrow String$, que dado un número que representa el precio de una computadora, retorne “muy barato” si el precio es menor a 2000, “demasiado caro” si el precio es mayor que 5000, “hay que verlo bien” si el precio está entre 2000 y 5000, y “esto no puede ser!” si el precio es negativo.
- c) $absoluto : Int \rightarrow Int$, que dado un entero retorne su valor absoluto.
- d) $esMultiplo2 : Int \rightarrow Bool$, que dado un entero n devuelve *True* si n es múltiplo de 2.
Ayuda: usar `mod`, el operador que devuelve el resto de la división.