

# Algoritmos y Estructuras de Datos II

Tomás Agustín Hernández



# 1. Especificación

## Consideraciones importantes / Reminders

- Utilizar operadores luego: Si estoy en LPO (Lógica de Primer Orden) utilizar los operadores luego si vemos que hay una posible indefinición como una división, o ingresar a una lista a un índice. Recordar que el para todo y un existe, aunque esté acotado por un rango, los cuantificadores predicen IGUAL para todos los valores. Entonces, aunque diga que  $x$  es positivo, también probará dividir inclusive por 0 y estallará.
- Recordar las condiciones bidireccionales
  - Si por algún motivo tengo que armar una “lista”, como, por ejemplo, los divisores de un número  $x$  tengo que indicar que, si el número divide a  $x$ , entonces ese número está en res, pero además todos los valores que están en res DIVIDEN a  $x$ . Es una condición bidireccional.
  - Otro ejemplo puede ser que tenga que considerar el máximo de una lista, si todos los valores  $y$  que están en la lista son menores que res entonces significa que res también pertenece a esa lista original.
- Recordar el significado de los cuantificadores con dos variables al mismo tiempo: En la lógica se ejecutan todos de uno a la vez. Es decir, si tengo que poner un para todo adentro de un para todo entonces hago un para todo solo con dos variables y listo.
- Recordar que cuando en un procedimiento llamo a un predicado y ese predicado devuelve algo de un para todo, existe (básicamente un valor de verdad) tengo que castear ese valor en el procedimiento porque son dos mundos distintos. Ej: asegura:  $\text{res} = \text{True} \iff \text{predicado}$
- Los predicados y funciones auxiliares no describen problemas. Son herramientas sintácticas para descomponer predicados.
  - Los procedimientos pueden llamar a funciones auxiliares o predicados. Un procedimiento no puede llamar a otro procedimiento.
  - Los predicados pueden llamar a predicados o auxiliares.
  - Las auxiliares solo pueden llamar auxiliares.
- No usamos nunca  $==$  en especificación, usamos siempre  $=$  y estamos comparando, no asignando.
- No existe el guardar o asignar en el mundo de la lógica. No puedo guardar en una lista en un índice específico porque si un valor. Para esto solemos usar que  $x$  valor pertenecerá a esta lista, por ejemplo.
- Si tengo un algoritmo que cumple una funcionalidad específica con un require más débil, puedo poner el require más restrictivo y va a funcionar igual pero NO al revés.

## Fórmulas compuestas

Decimos que una fórmula es compuesta a una fórmula que tiene más de una operación y esa operación necesita realizarse antes de conocer su valor.

- $(p \wedge q) \vee m$
- $((p \wedge q) \vee m) \implies n$

## Fórmula atómica

Decimos que una fórmula es atómica si se puede inferir su valor con una, o ninguna operación. Es irreducible.

- $p$
- $p \wedge q$

## Fórmulas bien definidas

Decimos que una fórmula está bien definida cuando el orden que hay que hacer las operaciones es clara. Es decir, cuando cada operación toma dos variables proposicionales, y al realizar la operación termina siendo una fórmula atómica.

- $p \wedge q \vee r$  está mal formada. No se especifica si primero se realiza el  $\wedge$  o el  $\vee$ .
- $(p \wedge q) \vee r$  está bien formada.
- $p \wedge q \wedge r \wedge m$  está bien formada porque son todas conjunciones.
- $p \vee q \vee r \vee m$  está bien formada porque son todas disyunciones.

## Cuantificadores

- Para todo:  $\forall$ 
  - *Garantiza la conjunción* :  $p(1) \wedge p(2) \wedge p(3) \cdots \wedge p(m)$ . Todos los casos deben ser true para que el cuantificador sea true.
  - Se acompaña por un  $\longrightarrow$  a la hora de predicar sobre los elementos.
  - $(\forall i : \mathbb{Z})(0 \leq i < |s| \longrightarrow_L s[i] \bmod 2 = 0)$ . Todos los elementos de la lista son divisibles por 2.
  - Estructura:  $\forall + \text{rango} + \longrightarrow_L$
- Existe:  $\exists$ 
  - *Garantiza la disyunción* :  $p(1) \vee p(2) \vee p(3) \cdots \vee p(m)$ . Con un caso true el cuantificador es true.
  - Se acompaña por un  $\wedge$  a la hora de predicar sobre los elementos.
  - $(\exists i : \mathbb{Z})(0 \leq i < |s| \wedge_L s[i] \geq 0)$ . Existe algún elemento en la lista que es mayor o igual a 0.
  - $\exists + \text{rango} + \wedge_L$

## Equivalencias entre fórmulas

Decimos que dos fórmulas son equivalentes  $\iff$  los valores de la tabla de verdad al aplicar la operación arroja el mismo resultado.

## Valuaciones

Las valuaciones surgen en base a la tabla de verdad. Las valuaciones serian darle valor a las variables proposicionales y ver el resultado de la operación. Solo hacen referencias a fórmulas atómicas.

## Tautologías, contradicciones y contingencias

- Una fórmula es tautología  $\iff$  el resultado de la operación en cada fila arroja siempre V.
- Una fórmula es contradicción  $\iff$  el resultado de la operación en cada fila arroja siempre F.
- Una fórmula es contradicción  $\iff$  el resultado de la operación en cada fila arroja siempre V y F.

## Relaciones de fuerza entre fórmulas

Decimos que una fórmula es más fuerte que la otra  $\iff$  una fórmula es más restrictiva que la otra, o está incluida en la otra.

En el mundo de la lógica, decimos que A es más fuerte que B  $\iff A \implies B$

- Si  $(A \implies B)$  y  $(B \implies A)$  son tautologías, entonces A y B son equivalentes.
- Si  $(A \implies B)$  es tautología y  $(B \not\implies A)$  no es tautología, entonces decimos que A es más fuerte que B.
- Si  $(A \not\implies B)$  y  $(B \not\implies A)$  son contingencias, entonces no existe relación de fuerza entre A y B.

Algunos ejemplos:

- $|s| = 0 \implies |s| \geq 0$ . En este caso vemos que  $|s| = 0$  es más fuerte que  $|s| \geq 0$  pues  $|s| = 0$  está incluido en  $|s| \geq 0$ . Por lo tanto,  $A \implies B$
- $|s| = 0 \implies |s| \geq 3$ . En este caso vemos que  $|s| = 0$  no es más fuerte que  $|s| \geq 3$  pues  $|s| = 0$  no está incluido en  $|s| \geq 3$ . Por lo tanto,  $A \not\implies B$
- $2 \leq i < |s| \implies 1 \leq i < |s|$ . En este caso  $A \implies B$ , pues  $i = 2$  está incluido en el rango de B. Por lo tanto,  $A \implies B$
- $0 \leq i < |s| \implies 1 \leq i < |s|$ . En este caso  $A \not\implies B$ , pues el 0 de A no es parte de B. Por lo tanto,  $A \not\implies B$

## Lógica trivaluada

También llamada lógica secuencial porque se procesa de izquierda a derecha; Nos introduce los conceptos de  $\wedge_L \vee_L \longrightarrow_L$  y el valor de indefinido  $\perp$ .

Se termina de evaluar una expresión cuando se puede deducir el valor de verdad.

Considere  $x = \text{true} \wedge y = \perp \wedge z = \text{false}$

- $x \vee_L y$ : Como el  $\vee_L$  necesita uno solo para ser verdadero, entonces como  $x$  ya es true entonces toda la fórmula es verdadera.
- $x \wedge_L y$ : Como el  $\wedge_L$  necesita que ambas variables sean verdaderas, evalúa indefinido y el programa estalla.
- $\neg x \longrightarrow_L y$ : Como el  $\longrightarrow_L$  solo es falso si el antecedente es true y el consecuente false, como en este caso el antecedente ya es falso, toda la implicación es verdadera.
- $(x \wedge z) \wedge_L y$ : Como el  $\wedge_L$  necesita que ambas fórmulas sean true, en este caso, como  $(x \wedge z)$  es falso, entonces ya toda la fórmula es falsa. Nótese que el  $\wedge$  de la condición interna no contiene el luego porque jamás se indefinirá.
- $(\forall i : \mathbb{Z})(0 \leq i < |s| \longrightarrow_L s[i] \geq 0)$  Nótese que aquí usamos un  $\longrightarrow_L$  porque podría ser que la lista esté indefinida o no exista el valor en  $s[i]$

## Predicados

- Viven en el mundo de la lógica.
- Nos sirven para poder modularizar nuestras especificaciones.
- Solamente devuelven valores de verdad True y False y es necesario castearlos en caso de querer devolver true como tipo de dato.
- Los predicados pueden llamar a otros predicados o funciones auxiliares.

Ejemplo cuando tenemos que transformar el valor de verdad a tipo de dato:

```
pred divisiblePorDos (n:  $\mathbb{Z}$ ) {  
     $n \bmod 2 = 0$   
}  
proc esMultiploDeDos (in n:  $\mathbb{Z}$ ) : Bool  
    requiere {true}  
    asegura { $res = \text{true} \iff \text{divisiblePorDos}(n)$ }
```

Ejemplo usando un predicado sin necesidad de transformar el valor de verdad a tipo de dato:

```
pred todosSonPares (l:  $\text{seq}(\mathbb{Z})$ ) {  
     $(\forall i : \mathbb{Z})(0 \leq i < |l| \longrightarrow_L l[i] \bmod 2 = 0)$   
}  
proc todosPares (in l:  $\text{seq}(\mathbb{Z})$ ) : Bool  
    requiere { $\text{todosSonPares}(l)$ }
```

## Variables Ligadas y Libres

Las variables son ligadas  $\iff$  están dentro de un cuantificador mientras que son libres cuando no lo están.

- $(\forall i : \mathbb{Z})(0 \leq i < |s| \longrightarrow_L n \geq s[i])$  i es una variable ligada mientras que n y s son variables libres.
- $(\exists j : \mathbb{Z})(0 \leq j < |s| \wedge_L n \geq s[j])$  j es una variable ligada mientras que n y s son variables libres.
- $(\forall i : \mathbb{Z})(0 \leq i < |s| \longrightarrow_L n \geq s[i]) \wedge P(i)$  Ojo acá. i es una variable ligada, pero la i que está fuera del cuantificador  $P(i)$  no está ligada. Esta última debería ser renombrada para no tener problemas y confusiones.

Cuando tenemos variables ligadas **no** podemos hacer nada sobre ellas, entre esas cosas, no podemos reemplazarlas porque no dependen de nosotros sino de los cuantificadores.

## Cuantificadores anidados

Anidamos cuantificadores cuando el rango de las variables es exactamente el mismo.

- $(\forall i, j : \mathbb{Z})(0 \leq i, j < |s| \longrightarrow_L n \geq s[i][j]) \equiv (\forall i : \mathbb{Z})((0 \leq i < |s| \longrightarrow_L (\forall j : \mathbb{Z})(0 \leq j < |s| \longrightarrow_L n \geq s[i][j])))$