

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

- \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.
- \exists garantiza la disyunción : $p(1) \vee p(2) \vee p(3) \cdots \vee p(m)$. Con un caso true el cuantificador es true.

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.