3 CUANTIFICACIÓN

En la práctica de las matemáticas es usual, a partir de una operación binaria como la suma o la multiplicación, definir una notación que sirve para efectuar la operación sobre un conjunto de elementos. Eso de lugar a *sumatorias* y *multiplicatorias*, que son operaciones llamadas *de alto orden* derivadas de la suma y de la multiplicación de dos argumentos. No toda función binaria, sin embargo, da lugar a una operación de alto orden. Por ejemplo, ni la resta ni la división dan lugar a operaciones que las generalicen sobre una colección.

Dada una operación binaria $\bullet$, lo que hace que se pueda pensar en una operación de alto orden correspondiente es que, al aplicarla sobre una colección de argumentos, este proceso se pueda realizar paso a paso, operando resultados parciales con elementos de la colección aún no considerados, y en cualquier orden. Se verá también que va a ser importante que la operación tenga un elemento neutro, para empezar el proceso y saber cómo actúa la operación en el caso que la colección de argumentos sea vacía.

Para aclarar las ideas, se puede observar cómo la suma da lugar a las sumatorias. La suma sobre enteros es una operación de la forma

$$+ : \mathbf{int} \times \mathbf{int} \rightarrow \mathbf{int}$$

que recibe dos enteros y cuya evaluación tiene como resultado el entero igual a la suma de los argumentos. Esta operación tiene las siguientes propiedades:

```
(a + b) + c = a + (b + c)          // asociatividad
a + b = b + a                      // conmutatividad
a + 0 = a = 0 + a                  // 0 es neutro
```

Para evaluar una suma de varios argumentos, lo natural es comenzar con un valor `0` y agregar, paso a paso, argumentos aún no sumados, hasta terminar. El empezar con `0`, cuando no se ha sumado nada, hace pensar que es necesario saber que la suma de una colección vacía debe dar `0`, que es el elemento neutro de la operación `+`. Por la asociatividad, la suma de una secuencia de argumentos se puede calcular haciendo sumas simples -sobre dos argumentos- y sumando los resultados intermedios. La conmutatividad permite, además, hacer las sumas sin que importe el orden de los sumandos.

Una *sumatoria* es una suma sobre un conjunto de elementos. Es usual encontrar una notación tradicional como

$$\sum_{i=1}^{n} a_i$$

Sin embargo, también se usa algo como la llamada *notación en línea*:

$$(+i|\ 1 \le i \le n : a_i)$$

y el significado esperado es[1]

$$(a_i)[i:=1] + (a_i)[i:=2] + \ldots + (a_i)[i:=n]$$
$$=$$
$$a_1 + a_2 + \ldots + a_n$$

Nótese que se quiere que, por ejemplo:

---

[1] Los puntos suspensivos explican informalmente el significado. En realidad, éste se entenderá más claramente después.

```
   (+k | 1≤k≤n : k)
=
   1 + 2 + … + n
```

Y que no hay confusión si se suma en cualquier orden (de nuevo, porque la suma es asociativa y conmutativa). ¿Cuál deberá ser el valor de `(+k | 1≤k≤0 : k)`? Por convención, es `0`, el elemento neutro de la suma.

La notación en línea tiene ventajas sobre la tradicional. En realidad, se puede usar para denotar una sumatoria más general, como

```
(+i| Q : R)
```

donde `Q` es una condición sobre `i` y `R` es una expresión aritmética sobre `i`. Por ejemplo, las siguientes sumatorias resultan imposibles de escribir en la notación tradicional[2]:

```
(+i| 1≤i≤n ∧ par(i) : 2*i+1)
   (+i| 1≤2*i≤n  : 2*i+1)
```

pero tienen un significado que debería ser fácil de explicar en la notación en línea. En lo sucesivo se utilizará la notación en línea para denotar operaciones de alto orden. En el contexto de la lógica éstas van a corresponder a *operaciones de cuantificación*.

3.1 CUANTIFICACIONES

Una operación binaria ⊕ es *ACU* (asociativa, conmutativa y con unidad) si cumple que, para los elementos de su dominio, donde `x`, `y`, `z` son variables y u es una constante:

- `x ⊕ (y ⊕ z) = (x ⊕ y) ⊕ z`
- `x ⊕ y = y ⊕ x`
- `x ⊕ u  = u ⊕ x  =  x`

Como ejemplos de operaciones ACU se pueden mencionar:

+ : suma (en enteros, reales, …)
* : multiplicación (en enteros, reales, …)
∧ : conjunción lógica ∧
∨ : disyunción lógica

El elemento *neutro* toma su nombre de que, al operarlo con un elemento cualquiera `x`, se obtiene cono resultado `x`. Otros nombres para el elemento neutro son *cero*, *unidad* y *módulo*.

Para la operación ⊕ se define una operación de alto orden correspondiente, expresada con la notación en línea

```
   (⊕x | Q : E)
```

donde se reconocen las siguientes componentes:

`x`, la *variable* de la cuantificación, es un identificador de una variable;
`Q`, el *rango* de la cuantificación, es una condición sobre `x`;
`E`, el *cuerpo* de la cuantificación, es una expresión (del tipo[3] de las que opera ⊕).

---

[2] El problema es que la notación tradicional opera los elementos de un intervalo de enteros. La notación en línea permite como operandos colecciones que se describan con predicados y variables. Por supuesto, se puede extender la notación tradicional de alguna manera, aunque se termina con problemas tipográficos y usándose editores de fórmulas más bien complejos.

La idea es que

`(⊕x | Q : E) = "aplicar ⊕ a los E tales que Q"`

Como se ve, el orden en que se operen los argumentos no debería importar, por la asociatividad y la conmutatividad de ⊕ [4].

La notación para cuantificaciones puede destacar, si se requiere, el tipo de la variable de cuantificación. Si x es de tipo T y esto se quiere destacar, se escribe

`(⊕x:T | Q : E)`

Si `Q` ≡ `true`, se suele omitir, es decir:

`(⊕x |: E) = (⊕x | true : E)`

Para ciertos operadores, también es usual cambiar el símbolo de la operación por algún símbolo culturalmente más utilizado. Entonces, se pueden usar las siguientes notaciones y significados:

`(Σx | Q : E )`	`= (+x | Q : E )`	*// sumatorias*
`(Πx | Q : E )`	`= (*x | Q : E )`	*// multiplicatorias*
`(∀x | Q : E )`	`= (∧x | Q : E )`	*// "para todo", cuantificación universal*
`(∃x | Q : E )`	`= (∨x | Q : E )`	*// "existe", cuantificación existencial*

La semántica pretendida depende, en cada caso, de la operación involucrada:

`(Σx | Q : E )` // sumatoria de los `E` que cumplen `Q`
`(Πx | Q : E )` // multiplicatoria de los `E` que cumplen `Q`
`(∀x | Q : E )` // Para todo `x`, tal que `Q` valga, vale `E`
`(∃x | Q : E )` // Existe `x`, tal que `Q` vale, para el que vale `E`.

Ejercicios 3.1

Para las siguientes operaciones binarias, suponga definiciones y propiedades conocidas de manera intuitiva y determine si se puede o no construir una cuantificación que denote la operación de alto orden correspondiente. Cada operación `op` se describirá con una notación de la forma `x op y` seguida de una interpretación informal de la misma.

1 `a max b` ≈ "máximo entre los números naturales (enteros no negativos) a, b"

2 `a max b` ≈ "máximo entre los números reales a, b"

3 `a max b` ≈ "máximo entre los números reales no negativos a, b"

4 `a & b` ≈ "cadena de caracteres resultado de concatenar las cadenas a, b"

5 `a ≡ b` ≈ "a equivale a b" (valores booleanos)

---

[3] La noción de tipo de una variable se va a entender de manera intuitiva. Para quienes tiene alguna experiencia trabajando con lenguajes de programación, el concepto de tipo de las variables que allí se maneja debería ser suficiente para entender lo que aquí se va a utilizar.

[4] Por lo menos, esto es claro cuando el número de los `x` que cumplen `Q` es finito. Obsérvese que la notación permite hablar de rangos infinitos, v.gr., se puede pensar en una suma infinita, como `(+k| k≥0 : a(k))`. En este caso no cabe pensar que se van a operar todos los argumentos uno tras otro y en un orden cualquiera, puesto que esto sería un proceso de cálculo infinito.

3.2 REGLAS DE CUANTIFICACIÓN

Nótese que se quiere extender ahora el lenguaje formal, de manera que las cuantificaciones sean también fórmulas bien formadas. Esto da lugar a plantear que haya reglas de inferencia especiales que permitan calcular con las nuevas fórmulas. Como antes, se puede pensar en la corrección de estas nuevas reglas, y su justificación se puede argumentar teniendo en cuenta los ejemplos en los que se aplican (v.gr., las sumatorias y sus propiedades).

3.2.1 Reglas de inferencia para cuantificaciones

Hay 3 reglas de inferencia nuevas, una para sustitución de variables y dos para remplazo de iguales por iguales:

Sustitución:
$$\frac{\text{true}}{(\oplus x|\ E\ :\ S)[z:=P] = (\oplus x|\ E[z:=P]\ :\ S[z:=P])} \quad \text{z,P no dependen de x}$$

Leibniz:
$$\frac{P = Q}{(\oplus x\ |\ E[z:=P]\ :\ S) = (\oplus x\ |\ E[z:=Q]\ :\ S)}$$

Leibniz:
$$\frac{E \Rightarrow P = Q}{(\oplus x\ |\ E\ :\ S[z:=P]) = (\oplus x\ |\ E\ :\ S[z:=Q])}$$

Estas reglas permiten efectuar cambios de iguales por iguales en el rango y en el cuerpo de las cuantificaciones. Nótese que, para hacer sustituciones en el cuerpo de la cuantificación, es suficiente que las expresiones iguales lo sean para los elementos del rango.

Ejemplo A: Reglas de inferencia para cuantificaciones

Sustitución

```
   (+i| 1≤i<j : j+2)[j:= j+4]
=
   (+i| (1≤i<j)[j:= j+4] : (j+2)[j:= j+4])
=
   (+i| 1≤i<j+4 : j+4+2)
```

Se entiende que `i`, `j` son variables diferentes y, por tanto, se puede aplicar la regla de sustitución en vista de que `j` ni `j+4` dependen de `i`.

Remplazos en el rango:

```
   (+i:nat | 0≤i²<n²+2*n+1 : 3*i+2)
=     ⟨ n²+2*n+1  ≡  (n+1)² ⟩
   (+i:nat | 0≤i²<(n+1)² : 3*i+2)
=     ⟨ i≥0: 0≤i²<(n+1)² ≡  0≤i<n+1 ⟩
   (+i:nat | 0≤i<n+1 : 3*i+2)
```

En los dos remplazos señalados se cambia todo el rango por una expresión equivalente del mismo. En el segundo, obsérvese que la equivalencia depende de que la variable `i` es de tipo `nat` (números naturales, enteros no negativos).

Más remplazos en el rango:

```
   (∀i:nat | 0≤i²<n²+2*n+1 : 3*i+2>0)
=     ⟨ n²+2*n+1  ≡  (n+1)² ⟩
   (∀i:nat | 0≤i²<(n+1)² : 3*i+2>0)
=     ⟨ i≥0: 0≤i²<(n+1)² ≡  0≤i<n+1 ⟩
   (∀i:nat | 0≤i<n+1 : 3*i+2>0)
```

Las justificaciones de los remplazos en este ejemplo son exactamente las mismas que en el ejemplo anterior.

Remplazo en el cuerpo:

```
   (∀i | 0≤i<n : √(i²)≥0)
=     ⟨ 0≤i<n ⇒  (√(i²)≥0 ≡ i≥0) ⟩
   (∀i | 0≤i<n : i≥0)
```

La justificación requiere un conocimiento adicional del significado y de propiedades de los números reales.

§

3.2.2 Axiomas generales de cuantificación

Los axiomas siguientes permiten efectuar cálculos sobre cuantificaciones que, usualmente, redundan en resultados que simplifican las expresiones transformándolas en otras sin cuantificaciones o con cuantificaciones más simples (v.gr., con un rango más simple). Su uso repetido establece, en muchos casos, modos de evaluar las cuantificaciones en términos de valores primitivos (por ejemplo, al llegar a expresiones sin cuantificaciones).

1 Axioma: Rango vacío

```
   (⊕x | false : P) = u
```

∎

El axioma del rango vacío se refiere a operar con ⊕ todos los elementos que cumplan `false`. Como no hay ninguno, se define que el resultado es el elemento neutro de la operación.

Como justificación, parece natural afirmar que una suma vacía de números es `0`. Algo un poco más difícil de aceptar podría ser que una multiplicación vacía es `1`. Pero este es un axioma que, así parezca curioso para algunos ejemplos específicos, en la práctica ayuda a que el cálculo resulte fácil y coherente con la realidad.

Ejemplo A: Rango vacío

Sumatoria vacía:

```
   (+x:int | 1<x<0 : 5*x)
=     ⟨ 1<x<0 ≡ false ⟩
   (+x:int | false : 5*x)
=     ⟨ Rango vacío ⟩
   0
```

Multiplicatoria vacía:

```
   (*x:int | 1<x<0 : 5*x )
=     ⟨ 1<x<0 ≡ false ⟩
   (*x:int | false : 5*x )
=     ⟨ Rango vacío ⟩
```

```
1
```

Cuantificaciones universales y existenciales
Para los siguientes ejemplos, supóngase que la notación se refiere a elementos x de tipo `Animal`:

```
perro.x  ≈ "x es un perro"
gato.x   ≈ "x es un gato"
vuela.x  ≈ "x vuela"
pata4.x  ≈ "x es cuadrúpedo"
```

Nótese que no hay ningún x para el que

```
perro.x ∧ gato.x
```

sea verdadero.

Entonces, el axioma de rango vacío permite afirmar que

```
(∀x: Animal | perro.x ∧ gato.x : vuela.x ) ≡ true
(∃x: Animal | perro.x ∧ gato.x : vuela.x ) ≡ false
```

i.e., "todo perro que sea gato vuela" y "es falso que haya un animal que sea perro y gato, y que vuele".

En el caso del existencial también debe poderse afirmar que "es falso que haya un animal que sea perro y gato, y que sea cuadrúpedo", ya que el axioma de rango vacío permite asegurar que

```
(∃x: Animal | perro.x ∧ gato.x : pata4.x ) ≡ false
```

§

2 Axioma: Regla de un punto

Si E no depende de x:

```
(⊕x| x=E : P) = P[x:= E]
```

∎

La regla de un punto permite, como el axioma del rango vacío, liberarse de una cuantificación obteniendo como resultado una expresión más simple.

Como justificación, si se opera sobre un rango que solo tiene un elemento x, parece natural esperar que el resultado sea el valor de la expresión del cuerpo evaluada en x. También se puede explicar como el resultado de una operación ⊕ teniendo como argumentos el neutro u (i.e., el resultado de la operación sobre una colección vacía) y el valor de la expresión del cuerpo evaluado en el único elemento que está en el rango, i.e., `P[x:= E]`. Por la definición de elemento neutro, el resultado debe ser `P[x:= E]`.

La restricción de que E no dependa de x es importante, porque de otro modo puede desvirtuarse el hecho de que el rango se cumpla exactamente en un elemento.

Ejemplo B: Un punto

Sumatoria de un elemento

```
   (+x:int | 0<x<2 : 5*x )
=     ⟨ 0<x<2 ≡ x=1 ⟩
   (+x:int | x=1 : 5*x )
=     ⟨ Regla de un punto ⟩
   (5*x)[x:= 1]
=
   5*1
```

```
=
   5
```

Sumatoria incorrecta de un elemento

```
   (+x:int | x = x+1 : 5*x )
=     ⟨ Regla de un punto ⟩
    (5*x)[x:= x+1]
=
   5*(x+1)
```

Pero este cálculo es incorrecto[5]. En cambio, sí está bien calcular así:

```
   (+x:int | x = x+1 : 5*x )
=     ⟨ x=x+1 ≡ false ⟩
   (+x:int | false : 5*x )
=     ⟨ Rango vacío ⟩
   0
```

Cuantificaciones universales y existenciales

Para los siguientes ejemplos, supóngase la notación del Ejemplo A.

```
   (∀x: Animal | x=Pluto : vuela.x)
=     ⟨ Regla de un punto ⟩
   (vuela.x)[x:= Pluto]
=
   vuela.Pluto
```

Aquí se está afirmando que la cuantificación universal equivale a "`Pluto` es un animal que vuela"[6]. Obsérvese que, también:

```
   (∃x: Animal | x=Pluto : vuela.x)
=     ⟨ Regla de un punto ⟩
   (vuela.x)[x:= Pluto]
=
   vuela.Pluto
```

§

3 Axioma: Distributividad

Si las cuantificaciones están bien definidas:

```
   (⊕x| R : E1)  ⊕  (⊕x| R : E2) = (⊕x| R : E1 ⊕ E2)
```

∎

La exigencia de la buena definición de las cuantificaciones tiene que ver con que, en ocasiones, la notación puede no tener sentido al interpretarse como operar sobre todos los elementos que cumplan `R`. El problema puede estar en que el rango `R` sea infinito (hay infinitos elementos que están en el rango) o que, para algún elemento que esté en el rango `R`, alguna de las expresiones `E1`, `E2` o `E1⊕E2` no esté bien definida.

---

[5] Entre otras razones, se supone que `x` es una variable de la sumatoria, así que el resultado debería ser independiente de `x`. Un indicador más de que el cálculo tiene algo errado.

[6] Nótese que esto no quiere decir que sea verdad que "`Pluto` es un animal que vuela": solo que las dos afirmaciones son equivalentes.

El que el rango R sea infinito no es suficiente para afirmar que la notación no tenga sentido. En general, la buena definición depende del tipo de la variable de cuantificación y de la operación que se efectúa. Lo que sí puede afirmarse es que, si el rango es finito y los cuerpos de las cuantificaciones están bien definidos para cada elemento del rango, el axioma de distributividad es coherente con la realidad.

Ejemplo C: Distributividad en cuantificaciones

Partir una suma por los términos que se suman

$$
\begin{array}{ll}
 & (+k|\ 0<k<n\ :\ k+1) \\
= & \quad \langle\ \text{Distributividad}\ \rangle \\
 & (+k|\ 0<k<n\ :\ k) + (+k|\ 0<k<n\ :\ 1)
\end{array}
$$

Sumas infinitas bien definidas

$$
\begin{array}{ll}
 & (+k{:}\mathbf{int}|\ 0<k\ :\ k^{-2} + k^{-2}) \\
= & \quad \langle\ \text{Distributividad}\ \rangle \\
 & (+k{:}\mathbf{int}\ |\ 0<k\ :\ k^{-2}) + (+k|\ 0<k\ :\ k^{-2})
\end{array}
$$

La teoría de series permite afirmar que cada una de las sumas infinitas anteriores está bien definida (porque todas convergen) y, además, que el cálculo señalado es correcto.

Sumas infinitas mal definidas

$$
\begin{array}{ll}
 & (+k{:}\mathbf{int}|\ 0<k\ :\ k^{-1} + k^{-1}) \\
= & \quad \langle\ \text{Distributividad}\ \rangle \\
 & (+k{:}\mathbf{int}\ |\ 0<k\ :\ k^{-1}) + (+k|\ 0<k\ :\ k^{-1})
\end{array}
$$

También la teoría de series permite afirmar que al menos una de las sumas infinitas de este ejemplo no está bien definida (porque diverge). El axioma de distributividad está mal usado en este caso.

Cuantificaciones universales y existenciales

El significado pretendido de las cuantificaciones universales ("para todo") y existenciales ("hay uno") justifica que se pueda afirmar que estas cuantificaciones *siempre están bien definida*s. En otras palabras, el axioma de distributividad se puede aplicar siempre:

$$(\forall x|\ R\ :\ E1)\ \ \wedge\ \ (\forall x|\ R\ :\ E2)\ \ \equiv (\forall x|\ R\ :\ E1 \wedge E2)$$
$$(\exists x|\ R\ :\ E1)\ \ \vee\ \ (\exists x|\ R\ :\ E2)\ \ \equiv (\exists x|\ R\ :\ E1 \vee E2)$$

§

4 Axioma: Partir rango disyunto

Si las cuantificaciones están bien definidas y $R \wedge S \equiv \texttt{false}$:

$$(\oplus x|\ R \vee S\ :\ E)\ \ =\ \ (\oplus x|\ R\ :\ E) \oplus (\oplus x|\ S\ :\ E)$$

∎

Partir el rango tiene sentido para efectuar separadamente una operación de alto orden sobre una parte de los elementos del rango y sobre el resto. Si el rango consta de dos partes, R y S, todo elemento en él satisaface $R \vee S$. Si $R \wedge S \equiv \texttt{false}$, ningún elemento del rango cumple simultáneamente R y S. Entonces, si las cuantificaciones están bien definidas, es de esperarse que el axioma sea verdadero al interpretarlo en una realidad.

Ejemplo D: Rango disyunto

Partir rango en una suma

```
   (+k| 0<k<n : E(k))
=     ⟨ 0<k<n  ≡  (0<k<n ∧ par.k) ∨ (0<k<n ∧ ¬par.k) ⟩
   (+k| (0<k<n ∧ par.k) ∨ (0<k<n ∧ ¬par.k) : E(k))
=     ⟨ Partir rango ⟩
   (+k| 0<k<n ∧ par.k : k) + (+k| 0<k<n ∧ ¬par.k : E(k))
```

En este caso, puede ser conveniente expresar el rango que interesa (`0<k<n`) como dos partes disyuntas (los k pares, los k impares). Una vez hecho esto, la partición por rango disyunto es aplicable, siempre que n sea un valor entero bien definido, ya que entonces todas las sumas están bien definidas.

Partir rango mal aplicado

```
   (+k| (0<k ∧ par.k) ∨ (0<k ∧ ¬par.k) : k)
=     ⟨ Partir rango ⟩
   (+k| 0<k ∧ par.k : k) + (+k| 0<k ∧ ¬par.k : k)
```

En este caso ninguna de las sumas infinitas mencionadas converge. No tiene sentido hablar de aplicar el axioma de partir rango en este caso.

Cuantificación universal

Como se indicó arriba, las cuantificaciones universales siempre están bien definidas si los cuerpos lo están, i.e., siempre tiene sentido si los términos del cuerpo están bien definidos. Por ejemplo:

```
   (∀k| (0<k ∧ par.k) ∨ (0<k ∧ ¬par.k) : k>8)
=     ⟨ Partir rango ⟩
   (∀k| 0<k ∧ par.k : k>8) ∧ (∀k| 0<k ∧ ¬par.k : k>8)
```

§

Cuando se quiere partir el rango en dos partes que tienen elementos en común (i.e., `R ∧ S` puede no ser `false`), se puede plantear un axioma que remedie el hecho de incluir dos veces a los elementos que satisfacen `R ∧ S` al efectuar la operación de alto orden:

5 Axioma: Partir rango general

Si las cuantificaciones están bien definidas:

```
 (⊕x| R ∨ S : E) ⊕ (⊕x| R ∧ S : E)  =  (⊕x| R : E) ⊕ (⊕x| S : E)
```

∎

Ejemplo E: Partir rango general

Conteo de rango

La forma de contar los elementos de un rango es sumar 1 por cada elemento. Supóngase que en un grupo de personas unas hablan español, otras inglés y otras hablan los dos idiomas. Como notación, defínanse

`e.x` ≈ "`x` habla español"

`i.x` ≈ "`x` habla inglés"

Ahora, para contar cuánta gente hay en el grupo se puede calcular

```
   (+x| e.x ∨ i.x : 1)
```

El axioma de partir rango general permite afirmar que

```
   (+x| e.x ∨ i.x : 1) + (+x| e.x ∧ i.x : 1) = (+x| e.x : 1) + (+x| i.x : 1)
=     ⟨ aritmética ⟩
   (+x| e.x ∨ i.x : 1) = (+x| e.x : 1) + (+x| i.x : 1) - (+x| e.x ∧ i.x : 1)
```

Es decir: el número de personas en el grupo equivale al número de las que hablan español, sumado al de las que hablan inglés, menos las que hablan los dos idiomas.

§

El siguiente axioma justifica que, en el caso de las operaciones de lógica $\wedge$, $\vee$, no sea necesario evitar "operar dos veces" como se considera al partir rango general. Esto sucede porque las dos operaciones son idempotentes, i.e.,

```
x ∧ x ≡ x
x ∨ x ≡ x
```

Entonces, se plantea el axioma:

6 Axioma: Partir rango con idempotencia

Si las cuantificaciones están bien definidas, $\oplus$ idempotente ($x \oplus x = x$):

```
(⊕x | R ∨ S : E) =  (⊕x | R : E) ⊕ (⊕x | S : E)
```

∎

Ejemplo F: Partir rango con idempotencia

Considérese la situación planteada en el Ejemplo E y la afirmación

```
(∀x| e.x ∨ i.x : m.x)
```

donde
`m.x` ≈ "x es marinero".

El partir rango con idempotencia ($\alpha \wedge \alpha \equiv \alpha$) permite afirmar que

```
   (∀x| e.x ∨ i.x : m.x)
=     ⟨ Partir rango con idempotencia ⟩
   (∀x| e.x : m.x) ∧ (∀x| i.x : m.x)
```

Nótese que, como se está postulando también el Axioma de partir rango general, también debe ser cierto que

```
   (∀x| e.x ∨ i.x : m.x) ∧ (∀x| e.x ∧ i.x : m.x)
=     ⟨ Partir rango general ⟩
   (∀x| e.x : m.x) ∧ (∀x| i.x : m.x)
```

Y se puede concluir que

```
   (∀x| e.x ∨ i.x : m.x) ∧ (∀x| e.x ∧ i.x : m.x)
=     ⟨ ≡-Transitividad ⟩
   (∀x| e.x ∨ i.x : m.x)
```

§

El próximo axioma recuerda el intercambio del orden de suma en sumatorias anidadas. En estos casos es importante que el operar primero con respecto a una variable y luego con respecto a la otra sea posible, lo que requiere que las expresiones de los rangos no ganen dependencias que no se deberían tener al hacer los intercambios indicados.

7 Axioma: Intercambio de variables de cuantificación

Si las cuantificaciones están bien definidas, R no depende de y, Q no depende de x:

$(\oplus x \mid R : (\oplus y \mid Q : E)) = (\oplus y \mid Q : (\oplus x \mid R : E))$

∎

Ejemplo G: Intercambio de variables

Las aplicaciones de esta clase de axiomas son útiles en cálculos de sumatorias anidadas o incluso, de integrales en varias variables. Por ejemplo, para calcular la suma de los elementos de una matriz b[1..m,1..n] de números enteros, se puede sumar por filas o por columnas y el resultado debe ser el mismo, i.e.,

$(+i \mid 1 \le i \le m : (+j \mid 1 \le j \le n : b[i,j])$
$=$ ⟨ Intercambio de variables ⟩
$(+j \mid 1 \le j \le n : (+i \mid 1 \le i \le m : b[i,j])$

El axioma de intercambio no es aplicable cuando los rangos no son independientes entre sí. Por ejemplo:

$(+i \mid 1 \le i \le m : (+j \mid 1 \le j \le i : b[i,j])$

Nótese que, en este caso, al escribir algo como

$(+j \mid 1 \le j \le i : (+i \mid 1 \le i \le n : b[i,j])$

la variable i en el rango de la primera cuantificación es independiente de j. Es decir, habría que conocer un valor de i para poder calcular esta sumatoria.

Para la misma matriz se puede pensar en la afirmación de que exista una opareja de índices(i,j) para la que b[i,j]=0. Por el axioma de intercambio de variables, no debe importar si se trata de ubicar i primero que j o viceversa:

$(\exists i \mid 1 \le i \le m : (+j \mid 1 \le j \le n : b[i,j]=0)$
$=$ ⟨ Intercambio de variables ⟩
$(\exists j \mid 1 \le j \le n : (+i \mid 1 \le i \le m : b[i,j]=0)$

§

El siguiente axioma permite resumir cuantificaciones anidadas en una sola cuantificación con respecto a un conjunto de variables. Si se quiere, este axioma define la cuantificación "en más de una dimensión".

8 Axioma: Anidamiento

Si las cuantificaciones están bien definidas, R no depende de y:

$(\oplus x,y \mid R \wedge Q : E) = (\oplus x \mid R : (\oplus y \mid Q : E))$

∎

Ejemplo H: Anidamiento

Continuando con lo planteado en el Ejemplo G, la notación del Axioma de Anidamiento es una extensión de la idea de tener una sola variable en una operación de alto orden a tener una colección de variables. El axioma, de hecho, le da significado a la notación nueva, explicando que el prden de evaluación es, como es de esperarse, irrelevante para el cálculo de un resultado.

Entonces, se puede escribir:

$(+i,j \mid 1 \le i \le m \wedge 1 \le j \le n : b[i,j])$

```
=     ⟨ Anidamiento ⟩
   (+i| 1≤i≤m : (+j| 1≤j≤n : b[i,j])
```

Y también:

```
   (∃i,j| 1≤i≤m ∧ 1≤j≤n : b[i,j])
=     ⟨ Anidamiento ⟩
   (∃i| 1≤i≤m : (∃j| 1≤j≤n : b[i,j]=0)
```

§

Algo que se espera es que la evaluación de una cuantificación no dependa del nombre de la variable sobre la que la cuantificación se define. Por eso tiene sentido el siguiente axioma:

9 Axioma: Renombramiento de variable de cuantificación

Si R, E no dependen de y:

```
 (⊕x| R : E)  =  (⊕y| R[x:= y] : (E[x:= y])
```

∎

Claramente, se permite cambiar una variable por otra que, al renombrar, el valor de la expresión sea el mismo en cualquier caso. Si se contraviene alguna de las condiciones de independencia que el axioma exige, lo más seguro es que los dos lados de la ecuación del axioma correspondan a valores diferentes.

El siguiente teorema se puede demostrar dentro de la teoría esbozada. Generaliza el axioma de renombramiento de manera que no solo se cambie una variable por otra sino también por una expresión que denote una función que tenga inversa.

Teo: Cambio de variable de cuantificación

R no depende de y, E no depende de y, f es una expresión que tiene inversa:

```
 (⊕x | R : E)  =  (⊕y | R[x:= f.y] : E[x:= f.y])
```

Dem: Se omite.

∎

Ejemplo I: Cambio de variables

El cambio de variables es muy utilizado en cálculos numéricos, buscando la simplificación del cuerpo o del rango de la cuantificación. Por ejemplo:

```
   (+i| 0≤i<n : n-i)
=     ⟨ i:= n-j ⟩
   (+j| (0≤i<n)[i:= n-j] : (n-i)[i:= n-j])
=     ⟨ Sust ⟩
   (+j| 0≤n-j<n : n-(n-j))
=     ⟨ 0≤n-j<n ≡ 0<j≤n; aritmética ⟩
   (+j| 0<j≤n : j)
```

La condición de que la función f sea invertible es necesaria para que la nueva variable recorra tantos elementos como la variable original.

§

Ejercicios 3.2

1 Expanda las siguientes sustituciones textuales. De ser necesario, haga cambios de variables:

a $(\bullet x|\ 0<x+r<n : x+v)[v:= 1]$

b $(\bullet x|\ 0<x+r<n : x+v)[x:= 1]$

c $(\bullet x|\ 0<x+r<n : x+v)[n:= n-x]$

d $(\bullet x|\ 0<x<n : (\bullet y|\ 0<y<n : x+y+n))[n:= x+2*y]$

2 Decida si se puede usar el axioma de un punto para simplificar las siguientes expresiones (interprete la notación de acuerdo con el significado usual que se da en matemáticas). En caso afirmativo, aplique el axioma para calcular un resultado.

a $(+x{:}int\ |\ 0<x\leq 2 : x^2)$

b $(\wedge x{:}bool\ |\ x : true \Rightarrow x)$

c $(*x{:}nat\ |\ par.x : x+1)$

3 Explique si está o no bien usado el axioma de distributividad en cada uno de los siguientes casos.

a $(+x\ |\ 0<x\leq 5 : x+2x+3x) = (+x\ |\ 0<x\leq 5 : x) + (+x\ |\ 0<x\leq 5 : 2x + 3x)$

b $(+x\ |\ 0<x : x+(-x)) = (+x\ |\ 0<x : x) + (+x\ |\ 0<x : (-x))$

c $(+x\ |\ 0<x<10 : x+(-x)) = (+x\ |\ 0<x<10 : x) + (+x\ |\ 0<x<10 : (-x))$

4 Explique en cuales casos está bien utilizado el axioma de Renombramiento de variable de cuantificación. Indique cuál fue el renombramiento utilizado.

a $(+x|\ 0\leq x<n+1 : 2*x) = (+y|\ 0\leq y<n+1 : 2*y)$

b $(*x|\ 0\leq x<n+1 : x+3) = (*n|\ 0\leq n<n+1 : n+3)$

c $(\forall p|\ p \Rightarrow \neg p : \neg p) = (\forall r|\ r \Rightarrow \neg r : \neg r)$

d $(\exists p|\ p \Rightarrow (r \vee q) : p \Leftarrow q) = (\exists r|\ r \Rightarrow (r \vee q) : r \Leftarrow q)$

5 Verificar si las expresiones calculan el mismo valor; de no ser así explicar por qué.

a $(+x\ |\ 0\leq x\leq 2 : 2*x) = (+y\ |\ 0\leq y+5\leq 2 : 2*(y+5))$

b $(+x\ |\ 0\leq x\leq 2 : 2*x) = (+y\ |\ 0\leq y^2\leq 2 : 2*y^2)$

3.3 MANIPULACIÓN DE RANGOS

Los axiomas de partir rango tienen ejemplos importantes de aplicación si el rango se define como un intervalo de enteros. Más exactamente:

3.3.1 Teo: Rompimiento por término extremo

$(\oplus i\ |\ 0\leq i<n+1 : E) = (\oplus i\ |\ 0\leq i<n : E) \oplus E[i:= n]$

$(\oplus i\ |\ 0\leq i<n+1 : E) = E[i:= 0] \oplus (\oplus i\ |\ 0<i<n+1 : E)$

Dem: Se omite.

□

Cuando el rango es un intervalo, el partir rango repetidamente desde un extremo del intervalo hasta que solo quede un elemento en el rango y usar entonces el axioma de un punto, permite afirmar que:

$(\oplus i \mid 0 \leq i < n : E) = E[i:= 0] \oplus E[i:= 1] \oplus \ldots \oplus E[i:= n-1]$

que, si se vuelve a la motivación de las cuantificaciones es precisamente lo que se ha pretendido generalizar desde el caso de las sumatorias.

Ejercicios 3.3

1 Demostrar el Teorema de rompimiento por término extremo.