TRABAJO DE GRADO

"Forks without philosophers" o de cómo la cuantificación universal perdió una batalla y de las ventajas que ello reportó.

Pablo E. Martínez López

Departamento de Informática
Universidad Nacional de La Plata

1996

A mis viejos, que lo hicieron posible,

y a Lorena y Aylen, que aguantaron el período de gestación.

Contenidos

Apéndices

Agradecimientos

A mi director, Gabriel Baum, por su ayuda y apoyo constantes, y por la paciencia que me tuvo (tiene). Al Lic. Marcelo Frias (el Oso) por su ayuda y por permitirme incluir el ejemplo del Mínimo Ancestro Común, que desarrollamos juntos. Al Dr. Gunther Schmidt, por darme las pistas que me permitieron demostrar con éxito algunos de los teoremas. Al Lic. Pedro D'Argenio por la cuidadosa lectura de buena parte del material y por las sugerencias que hizo para mejorarlo, así como por facilitarme el prefacio de su tesis.

Al LIFIA (Laboratorio de Investigación y Formación en Informática Avanzada) por el sustento laboral y de infraestructura que me proveyó, y por la oportunidad de enfrentar un desafío difícil en Argentina, como es tratar de hacer investigación de nivel internacional.

A la ESLAI (Escuela Superior LatinoAmericana de Informática) pues, a pesar de su temprana muerte en las manos de los sicarios de la dependencia, aportó un ingrediente fundamental en mi formación, tanto académica como humana, y alcanzó a enseñarme que para hacer bien las cosas no basta con encerrarse en una burbuja y reconcentrar mediocridad.

A los miembros del Piet–Forum, y en especial a Gonzalo (del que tomé prestada algunas ideas para este agradecimiento), por el espacio (electrónico) de solaz que significan, agregando esa cuota de humor necesaria para que la lucha no sea tan aburrida.

A ese gigante que es el aparato burocrático–administrativo de la UNLP, por la resistencia que opuso a que yo llegara hasta aquí (y quién sabe si no me reserva aún algún otro golpe), pues se transformó en verdadero desafío y permitió que transcurriera suficiente tiempo (y quizás más del necesario) como para que yo aprendiera un buen número de esas cosas que no se aprenden en los libros y que contribuyen a que este texto sea mucho mejor.

Finalmente quiero reiterar el agradecimiento a mis viejos, por sentar los cimientos que sostienen esta estructura y por su apoyo en los (por suerte, ya lejanos) momentos difíciles, y a mis amadísimas Lorena y Aylen, por darle sentido a la vida y por la tolerancia y paciencia con que soportaron el largo período que llevó la gestación de esta tesis.

Capítulo 1

Introducción

"...you will get a crushed-flies-stained room and the doubt whether some flies have escaped."

Pedro D'Argenio, Tesis de Grado.

En sentido general, este trabajo trata sobre la construcción formal de programas. La necesidad de construir los programas formalmente ha sido ampliamente discutida a lo largo de los últimos años (ver, por ejemplo, [Gri81, DF88, BW88, NPS89, Par90]), y ha adquirido una importancia cada vez mayor. En los comienzos, la programación se realizaba de una forma intuitiva, casi se podría decir artística; con el correr de los años se comprobó que los métodos utilizados eran inadecuados, por lo que se desarrollaron nuevas técnicas, mediante las cuales un programa debía ser diseñado al mismo tiempo que la prueba de su corrección. Quizás la mejor metáfora para describir la necesidad de formalidad es la que D'Argenio describe en el prefacio de su tesis de grado ([D'A], ver también Ap. D), donde compara a los programas con moscas, y a los métodos formales con una máquina mata-moscas; si los programadores utilizasen sus zapatos para aplastar las moscas (no utilizando los métodos formales, sino la intuición), sucedería lo que se cita en el epígrafe. Es por ello que se hace tanto hincapié en la investigación de métodos que permitan un tratamiento formal del proceso de desarrollo de software, o bien para la construcción, o bien para la especificación y verificación. En esta tesis utilizaremos un método de construcción que consiste en realizar transformaciones sobre una especificación hasta obtener un programa que la satisfaga.

En sentido particular, este trabajo investiga algunos mecanismos para tratar formalmente especificaciones que involucran cuantificación universal. Las expresiones que involucran cuantificación universal no son computacionalmente eficientes (o incluso computables), debido a que requieren testear un número grande (o infinito) de casos en el rango de la cuantificación; es nuestro interés obtener reglas que permitan transformar estas especificaciones en términos algorítmicos de igual significado. Dado que el caso general de cuantificación universal no tiene solución general, deben estudiarse formas particulares; la forma elegida para su estudio en esta tesis es $(\forall z)(x \ R \ z \ \rightarrow \ y \ S \ z)$, donde la cuantificación universal discurre sobre una implicación.

La elección no es casual, puesto que este patrón aparece en un buen número de ejemplos, tales como problemas de ordenamiento, de minimización, etc. Esta forma de cuantificación puede ser transformada (bajo ciertas condiciones) en una forma recursiva, logrando de esta manera el objetivo deseado.

El mecanismo formal que utilizaremos para el desarrollo de programas se conoce con el nombre de síntesis de programas (comprendido dentro del área que se conoce como programación transformacional, [Par90]), y consiste en comenzar con una especificación obviamente correcta, aunque tal vez ineficiente, de un problema, y mejorarla a través de la aplicación de transformaciones que preserven la correctitud. Para que este método sea sencillo de utilizar se debe poseer un lenguaje simple y poderoso, donde se puedan expresar tanto especificaciones como programas; además, la notación debe expresar convenientemente los *conceptos*, facilitando el razonamiento, pero debe permitir también las *transformación mecánica* entre expresiones. El lenguaje propuesto en este trabajo es un cálculo algebraico, basado en un álgebra a la que llamamos *fork álgebra*. Las fork álgebras son una extensión de las álgebras relacionales introducidas por Tarski en [Tar41], a las que se les agrega una operación llamada *fork*, la cual aumenta su expresividad hasta hacerlas útiles para el cálculo de programas; surgieron en el campo de la informática, como un cálculo ecuacional para especificación y diseño formal de programas en un marco relacional [HV91, VHB92, BHSV93, FAN93, HBS93, FA94], pero mostraron tener injerencia también en los campos del álgebra y de la lógica, como se puede ver en [MSS92, FBHV93, SN94, FHVB94, VHF94, FHV95].

El cálculo relacional se obtiene al considerar los términos, ecuaciones e inecuaciones del álgebra como especificaciones de programas, y a los axiomas o teoremas válidos en los modelos como reglas de derivación que preservan la semántica de dichas especificaciones. De esta manera las operaciones del álgebra pueden representar las distintas maneras en que los programas pueden combinarse entre sí (por ejemplo, el producto relativo (;) puede usarse para representar la composición secuencial de dos programas). El formalismo así definido permite expresar especificaciones y programas ([HV91, HBS93]), así como estrategias de derivación ([HV91]).

Los motivos para elegir las fork álgebras como base para el formalismo son varios. Por un lado provee un lenguaje de especificación con expresividad suficiente como para que los problemas puedan ser especificados en el nivel de abstracción deseado, sin darles, necesariamente, un estilo algorítmico; por otro lado los términos sin operaciones no–algorítmicas pueden ser interpretados como especificaciones ejecutables (programas), en el mismo estilo que en los cálculos funcionales; adicionalmente el no–determinismo puede expresarse de manera natural, a diferencia de los cálculos funcionales que imponen el determinismo como condición. La combinación de estas características hace que el lenguaje de especificación y el de programación sean el mismo, facilitando la tarea de derivar un programa a partir de una especificación, pero sin por ello limitar el lenguaje de especificación, como sucede con los cálculos funcionales (ver, por ejemplo, [BW88] y [Mac90]).

Para poder considerar la expresividad como una ventaja se deben tener en cuenta dos problemas

que surgen al definir un cálculo de este tipo; por una parte, la cuestión de si se pueden expresar todas las clases de problemas deseadas, y por otro, que no haya modelos "raros", esto es, que no hayan sido tenidos en mente al dar la axiomatización. Estos problemas son conocidos como el problema de la expresividad y el problema de la representabilidad.

El problema de la expresividad se puede enunciar de la siguiente manera: "¿Podemos expresar cualquier fórmula de la lógica de primer orden como un término del álgebra?". En el caso de las álgebras relacionales de Tarski, la respuesta es negativa y fue dada por el mismo Tarski en [Tar41]. Para el caso de las fork álgebras la respuesta es positiva y fue dada por Haeberer y Veloso en [HV91].

El problema de la representabilidad consiste en saber si todo modelo del álgebra es isomorfo a un modelo estándar. Para el caso de las álgebras relacionales, la respuesta a este problema es negativa y fue dada por Roger Lyndon en [Lyn50], donde presenta un álgebra relacional simple, finita y no trivial que no es representable; posteriormente Schmidt en [SS93] demuestra que toda álgebra relacional que cumple con un axioma adicional (al que llama *point axiom*) es representable. Para el caso de las fork álgebras la respuesta es positiva, aunque la correcta elección de los modelos estándar haya llevado a ciertas discusiones académicas (ver [MSS92, FBHV93, BHSV93, FHVB94]) que llegaron a hacer creer que no podía obtenerse una respuesta positiva; finalmente se demostró que eran representables en [FBHV95].

Dado que las fork álgebras tienen suficiente poder expresivo y son representables, podemos estar seguros de que las ventajas mencionadas no son un mero deseo sino un hecho demostrado. Ahora bien, ¿cómo expresamos propiedades de programas con fork álgebras? Por ejemplo se puede escribir $(R ; R^{\smile} \preceq 1)$ para expresar que R es una relación funcional o función, $(R^{\smile} ; R \preceq 1)$ para expresar que R es inyectiva, $(P + Q)$ para expresar la elección no determinística entre P y Q, si entendemos P y Q como relaciones que denotan programas, etcétera. La clave para entender la manera de expresar propiedades está en el teorema de representabilidad, ya que gracias a él, cada término tiene una interpretación conjuntista estándar. En la Secc. 2.3 volveremos sobre este tema.

El enfoque relacional que proveen las fork álgebras no es el único que se ha utilizado como base para construcción formal de programas. Podemos mencionar, entre otros, al cálculo para esquemas de programas recursivos de de Bakker y de Roever ([dBdR72]), el enfoque de D. Smith ([Smi83]), el cálculo de la preespecificación más débil de Hoare ([HH86a, HH86b]), y las álgebras relacionales heterogéneas de Gunther Schmidt ([SS93]). Este último es bastante similar al enfoque de fork álgebras; las diferencias entre ambos se pueden encontrar en [BHSV93].

Para expresar las fórmulas cuya forma es $(\forall z)(x \; R \; z \; \rightarrow \; y \; S \; z)$ definiremos una operación derivada en las fork álgebras, a la que llamaremos *implicación relacional*. Esta operación fue presentada por primera vez en [HBS93] junto con un ejemplo muy simple. La caracterización de las fórmulas de este tipo y la enunciación de reglas de transformación que permitan manipularlas son de gran importancia pues una gran cantidad de problemas se expresan naturalmente en términos

de este tipo de fórmulas. A pesar de su utilidad, la implicación relacional está expresada en función de complementos y por lo tanto no es fácil el uso de la intuición al utilizarla, como tampoco es eficiente su computación tal cual está definida. El principal aporte de esta tesis es enunciar un conjunto de reglas que permiten transformar una especificación expresada mediante la implicación relacional (o sea con la forma previamente mencionada), en un término algebraico que pueda considerarse como un algoritmo. El método utilizado para obtener el conjunto de reglas consistió en el estudio de varios casos de problemas conocidos: mínimo de una lista, ocho reinas, clausura transitiva de una relación y mínimo ancestro común, principalmente. Estos problemas fueron especificados utilizando la implicación relacional, y luego se realizaron derivaciones hasta obtener formas algorítmicas de los mismos, extrayendo en el proceso las propiedades de la implicación que permitieron su computación. Con el fin de mostrar el uso de las reglas, presentaremos el ejemplo del mínimo ancestro común, por ser el más completo de ellos.

El presente trabajo se divide en tres partes principales. En el Cap. 2 presentaremos las fork álgebras, haciendo especial hincapié en la relación que existe entre los modelos concretos (conjuntistas) y los abstractos, y explicaremos su uso como mecanismo formal de desarrollo de programas. En el Cap. 3 presentaremos la implicación relacional, daremos propiedades sobre ella y mostraremos su uso en un estudio de caso: el problema de encontrar el mínimo ancestro común de dos elementos en un árbol binario. En el Cap. 4 presentamos las demostraciones de los teoremas y propiedades utilizados en los capítulos anteriores; por razones de legibilidad no fueron incluídas como parte del texto y por ello se presentan en forma independiente. El Cap. 5 contiene las conclusiones del trabajo.

Adicionalmente presentamos cuatro apéndices con información relevante: la definición de las fork álgebras (Ap. A), algunas propiedades de las fork álgebras utilizadas en la derivación (Ap. B), las reglas sobre la implicación relacional aportadas por esta tesis (Ap. C), y, por considerarlo de interés, el prefacio de la tesis de [D'A] (Ap. D).

Capítulo 2

Cálculo de programas en un marco algebraico

"Las relaciones son los modos por los que mi mente percibe los vínculos entre los entes singulares, pero ¿qué garantiza la universalidad y la estabilidad de esos modos?"

Umberto Eco, El Nombre de la Rosa.

2.1 Introducción

La técnica de síntesis de programas consiste en aplicar transformaciones sintácticas a una especificación correcta, pero quizás ineficiente, hasta obtener un programa que realiza la misma tarea, pero con mayor eficiencia. Esta técnica es similar a los cálculos algebraicos de las matemáticas, lo que sugiere que para aplicar esta técnica podemos buscar un modelo algebraico de programas y operaciones entre ellas. Una forma de entender un programa secuencial es a través de la relación que el mismo define entre los datos de entrada y los de salida. Es natural, entonces, pensar un programa como una relación binaria entre datos, y a las formas de combinar programas como operaciones entre relaciones binarias. La caracterización algebraica de esta interpretación se llama álgebra de relaciones binarias (también llamada álgebra relacional propia), y es un caso particular de los campos de conjuntos (también llamados subálgebras de partes de un conjunto).

Las álgebras de relaciones binarias son el modelo conjuntista de una estructura algebraica abstracta: las álgebras relacionales[1]. La ventaja de estas últimas sobre las primeras es que sólo utilizan variables para representar relaciones (las álgebras de relaciones binarias utilizan también variables para elementos) y permiten por ello expresiones más concisas, claras y manipulables sintácticamente.

[1] En realidad, de las álgebras relacionales que cumplen con un axioma adicional, el *point axiom* (Def. 2.2.6).

El estudio de la relación que existe entre un modelo conjuntista y una estructura algebraica abstracta se conoce con el nombre de *problema de representabilidad*, y consiste en establecer si los modelos del álgebra abstracta son exactamente los del álgebra concreta (conjuntista). Esta relación entre álgebras abstractas y modelos estándar se repite en la mayoría de las estructuras algebraicas estudiadas por el álgebra moderna, pues las versiones abstractas surgen como generalizaciones de modelos específicos. Determinar si el problema de representabilidad tiene una respuesta afirmativa para una clase determinada de álgebras es muy importante, pues de esa manera se establece que la versión abstracta es equivalente a la concreta, y por lo tanto es posible aprovechar la sencillez y claridad de la versión abstracta sin modificar la intuición que brindan los modelos concretos.

Una posible elección para el lenguaje en el que se expresarán los programas y especificaciones es, entonces, el álgebra relacional. Lamentablemente el poder expresivo de estas álgebras no abarca al de la lógica de primer orden, y por lo tanto existen relaciones que se pueden definir en la lógica, pero no en el álgebra. El *problema de la expresividad* se puede enunciar de la siguiente manera: "¿Podemos expresar cualquier fórmula de la lógica de primer orden como un término del álgebra?". Este problema es de gran importancia, pues si la respuesta fuera negativa no estaríamos seguros de poder expresar en este marco propiedades o relaciones que pueden expresarse en lógica de primer orden, lo cual no es deseable. Entonces necesitamos otro modelo algebraico, con suficiente poder expresivo.

La propuesta de Haeberer y Veloso ([HV91]) para solucionar el problema de expresividad de las álgebras relacionales es la de introducir un nuevo operador (al que llamaron *fork*, y que dió origen a las fork álgebras). En el trabajo citado demuestran que el poder expresivo de estas álgebras comprende al de la lógica de primer orden. Respecto de la representabilidad de las mismas, se mantuvo una larga discusión teórica acerca de si eran o no representables, (ver [MSS92, FBHV93, BHSV93, FHVB94]) hasta que en [FBHV95] se demuestra que lo son. Estas propiedades permiten su utilización como base para el desarrollo formal de programas.

A pesar de que existen otras soluciones al problema de expresividad de las álgebras relacionales (como las álgebras *Cilíndricas* [HMT71, HMT85] y las álgebras *Poliádicas* [Hal62], por ejemplo), las mismas poseen desventajas para realizar especificación y desarrollo de programas cuando se las compara con las fork álgebras, pues tienen signaturas infinitas (cantidad infinita de cilindrificaciones y elementos diagonales, en el caso de las álgebras cilíndricas, o transformaciones, en el caso de las álgebras poliádicas).

Lo que resta de la sección se organiza de la siguiente manera. Comenzamos por definir los campos de conjuntos y las álgebras de Boole, y explicar el problema de representabilidad de las mismas, ya que resultan suficientemente familiares y por lo tanto proveen de intuición para comprender dicho problema en las demás álgebras (para un tratamiento introductorio de este tema ver [OZ94]). Presentamos luego las álgebras relacionales (y su versión concreta, las álgebras de relaciones, también llamadas álgebras relacionales propias en algunos trabajos), caracterizándolas

como extensión de las álgebras de Boole (se las llama también álgebras de Boole con operadores, ver [JT51, JT52]); explicamos por qué estas álgebras podrían resultar útiles para desarrollo formal de programas pero mostramos que su poder expresivo no es suficiente para esta tarea. Presentamos, entonces, las fork álgebras, extensión de las álgebras relacionales, como solución al problema de la expresividad. Enunciamos los teoremas de representabilidad y expresividad de estas álgebras, y explicamos su relevancia en el cálculo de programas. Finalmente mostramos como utilizar este formalismo para expresar propiedades y para calcular programas.

2.2 De las Álgebras de Boole a las Fork Álgebras

2.2.1 Campos de Conjuntos y Álgebras de Boole

Toda persona que haya incursionado en álgebra sabe que un subconjunto del conjunto de partes de un conjunto dado, cerrado por las operaciones de unión, intersección y complemento conforma una estructura algebraica; a dicha estructura la llamaremos un campo de conjuntos.

Definición 2.2.1 *Un* Campo de Conjuntos *sobre un conjunto no vacío U es una estructura*

$$\mathsf{SF} = < \mathcal{R}, \cup, \cap, ', \emptyset, U >$$

tal que:

- $\mathcal{U} \subseteq \mathcal{P}(U)$, *cerrado para todas las operaciones,*
- $\cup$ *es la unión de conjuntos,*
- $\cap$ *es la intersección de conjuntos,*
- $'$ *es el complemento respecto de* U,
- $\emptyset$ *es el conjunto vacío, y*
- U *es llamado conjunto universal.*

Se utilizará el símbolo clásico ($\subseteq$) *para la inclusión de conjuntos.* ■

Teniendo a los campos de conjuntos en mente, se busca dar una caracterización algebraica (abstracta) de ellos, o sea una estructura algebraica que cumple ciertos axiomas y tal que las propiedades que valen en un campo de conjuntos sean exactamente las que valen para ella.

Definición 2.2.2 *Un* Álgebra de Boole *es una estructura*

$$\mathsf{BA} = < A, \vee, \wedge, ', 0, 1 >$$

donde

- $\vee$ *y* $\wedge$ *son operaciones binarias sobre* A *(llamadas* supremo *e* ínfimo, *respectivamente),*

- ' *una operación unaria sobre A (llamada* complemento*),*
- 0 *y* 1 *dos elementos de A (llamados* primer *y* último elemento, *respectivamente),*

y tal que para toda terna a, b, c, de elementos de A:

1. $a \vee a = a \quad y \quad a \wedge a = a$ *(idempotencia)*
2. $a \vee b = b \vee a \quad y \quad a \wedge b = b \wedge a$ *(conmutatividad)*
3. $a \vee (b \vee c) = (a \vee b) \vee c \quad y \quad a \wedge (b \wedge c) = (a \wedge b) \wedge c$ *(asociatividad)*
4. $a \vee (a \wedge b) = a \quad y \quad a \wedge (a \vee b) = a$ *(absorción)*
5. $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \quad y \quad a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ *(distributividad)*
6. $0 \neq 1$
7. $a \vee 0 = a \quad y \quad a \wedge 1 = a$
8. $a \vee a' = 1 \quad y \quad a \wedge a' = 0.$

El símbolo $\leq$ *se utilizará para el orden subyacente, y queda definido por* $a \leq b \leftrightarrow a \vee b = b$ *(o, equivalentemente,* $a \wedge b = a$*).* ■

El problema de la representabilidad de las álgebras de Boole es un resultado conocido en álgebra, y establece que toda álgebra de Boole es isomorfa a un campo de conjuntos ([Sto36]). Así, toda propiedad que vale en un álgebra de Boole, vale también en un campo de conjuntos, y por lo tanto las álgebras de Boole son la descripción abstracta de los campos de conjuntos.

Otro resultado conocido para las álgebras de Boole es que tienen un poder expresivo equivalente al del cálculo proposicional (todo cálculo proposicional determina un álgebra de Boole conocida como *álgebra proposicional de Lindenbaum*, [BM77]).

2.2.2 Álgebras de Relaciones y Álgebras Relacionales

Un caso particular de campo de conjuntos es el de un conjunto de relaciones binarias sobre un conjunto A, cerrado por las operaciones de unión, intersección y complemento respecto de la relación universal ([CT51, TG87]). Sobre este campo de conjuntos se pueden definir, entre otras, las siguientes operaciones:

- $R|S = \{< x,y >/(\exists z)(< x,z > \in R \wedge < z,y > \in S)\}$ (producto relacional)
- $R^{\sim} = \{< x,y >/< y,x > \in R\}$ (conversa)
- $\Delta = \{< x,x >/x \in A\}$ (relación identidad)

La estructura resultante de agregar estas operaciones al campo de conjuntos de relaciones binarias se conoce con el nombre de Álgebra de Relaciones Binarias, o Álgebra Relacional Propia.

Definición 2.2.3 *Un* Álgebra Relacional Propia *sobre un conjunto U es una estructura*

$$\mathsf{PRA} = < \mathcal{R}, \cup, \cap, |, ', ^{\sim}, \emptyset, V, \Delta_{Dom(V)} >$$

tal que:

- $\mathcal{R} \subseteq \mathcal{P}(U \times U)$,
- $< \mathcal{R}, \cup, \cap, ', \emptyset, V >$ *es un campo de conjuntos sobre* $\mathcal{P}(U \times U)$,
- $|$ *denota el* producto relacional,
- $^{\sim}$ *denota la operación* conversa *y*
- $\Delta_{Dom(V)}$ *denota la relación* identidad *sobre el dominio (o equivalentemente el rango) de la relación* V.

El símbolo $\subseteq$ *se utiliza para la inclusión usual de relaciones.* ■

La particularidad de las álgebras relacionales propias es que los elementos que la componen (relaciones) poseen siempre una estructura interna (los elementos de un campo de conjuntos (conjuntos) no necesariamente poseen estructura); las operaciones adicionales tienen que ver con esta estructura interna.

Al igual que en el caso de los campos de conjuntos tratamos de dar una estructura abstracta que describa mediante axiomas a las álgebras de relaciones binarias. La estructura propuesta para esto son las álgebras relacionales.

Definición 2.2.4 *Un* Álgebra Relacional (Abstracta) *es una estructura*

$$\mathsf{ARA} = < A, +, \bullet, ^{-}, ;, ^{\smile}, 0, \infty, 1 >$$

tal que:

1. $< A, +, \bullet, ^{-}, 0, \infty >$ *es un álgebra de Boole, completa y atómica, donde*
   - $+$ *denota la operación supremo,*
   - $\bullet$ *denota la operación ínfimo,*
   - $^{-}$ *denota la operación complemento,*
   - 0 *denota el elemento* cero,
   - ∞ *denota el elemento* unidad, *y*
   - $\subset$ *(llamada* inclusión*) denota el orden subyacente.*
2. $< A, ;, 1 >$ *es un semigrupo, es decir,*
   - $(R;S);T = R;(S;T)$, *y*

- $R;1 = 1;R = R$.

3. $R;S \subset T \Leftrightarrow R^{\smile};\overline{T} \subset \overline{S} \Leftrightarrow \overline{T};S^{\smile} \subset \overline{R}$, *(regla de Schröder).*

4. $R \neq 0 \Rightarrow \infty;R;\infty = \infty$, *(regla de Tarski)* ■

Esta versión de las álgebras relacionales no es representable. Dicho resultado fue presentado por Roger Lyndon en [Lyn50], donde construyó un álgebra relacional simple y no trivial que no es representable. La condición de representabilidad para las álgebras relacionales fue establecida por Maddux y Tarski en [MT76] y presentada por Gunther Schmidt en [SS93] bajo el nombre de *point axiom*.

Definición 2.2.5 *Dado* R *un elemento de un álgebra relacional, definimos las nociones de vector y de punto como:*

- R *es un* vector *sii* $R = R;\infty$
- R *es un* punto *sii* R *es un vector,* $R \neq 0$, *y* $R;R^{\smile} \preceq 1$. ■

Definición 2.2.6 *Dada un álgebra relacional* A, *diremos que* A *satisface el* point axiom *si, para toda relación* $R \neq 0$, *se cumple que* $(\exists P)(\exists Q)(P, Q$ *son puntos* $\wedge\ P;Q^{\smile} \preceq R)$ ■

La propiedad presentada por Schmidt establece que toda álgebra relacional que satisface el *point axiom* es isomorfa al álgebra de relaciones binarias sobre el conjunto de sus puntos, o sea, es representable. Por lo tanto, podremos decir que las álgebras relacionales son la versión abstracta de las álgebras de relaciones binarias, siempre y cuando nos mantengamos restringidos a aquellas que satisfacen el *point axiom*.

La particularidad de las álgebras de relaciones binarias y su versión abstracta, las álgebras relacionales, es la que nos abre las puertas para que podamos representar programas mediante los términos de estas álgebras: las relaciones binarias servirán para representar la relación de entrada-salida de un programa. Entonces surge la pregunta ¿es suficientemente expresivo el lenguaje de las álgebras relacionales?, o sea ¿cualquier programa se puede representar mediante un término de un álgebra relacional?

Para contestar esta pregunta, recordemos que la relación de entrada salida de un programa puede representarse mediante una fórmula de la lógica de primer orden. Entonces podemos re-escribir la pregunta: ¿cualquier fórmula de la lógica de primer orden se puede representar mediante un término de un álgebra relacional? La respuesta, por ahora, es no. Para demostrarlo, empezaremos por ver como se puede representar una fórmula de la lógica mediante un término del álgebra; este ejemplo, así como el que sigue, fueron dados por Tarski en [Tar41].

Dada la fórmula

$$(\forall x)(\forall y)(\exists u)(R(x,u) \wedge R(y,u)) \tag{2.1}$$

queremos representarla como un término del álgebra relacional. Para ello, comenzamos por notar que 2.1 es equivalente a

$$(\forall x)(\forall y)(\exists u)(\mathsf{R}(x,u) \wedge \mathsf{R}^{\smile}(u,y))$$

y entonces, basándonos en el teorema de representabilidad, y en la definición de producto relacional propio, podemos decir que el término $\mathsf{R};\mathsf{R}^{\smile}$ representa a dicha ecuación.

Ahora veamos la siguiente fórmula

$$(\forall x)(\forall y)(\forall z)(\exists u)(\mathsf{R}(x,u) \wedge \mathsf{R}(y,u) \wedge \mathsf{R}(z,u)) \tag{2.2}$$

Si aplicásemos la misma idea que para la Ecu. 2.1, sólo podríamos expresar de manera abstracta a lo sumo dos de las tres partes de la conjunción. Tarski, en el trabajo citado, demuestra que no existe un término del álgebra relacional que exprese esta fórmula. Posteriormente se demostró que el poder expresivo de las álgebras relacionales es equivalente al de una lógica de tres variables cuyas fórmulas tienen a lo sumo dos variables libres.

2.2.3 Fork Álgebras Propias y Abstractas

Las fork álgebras propias son álgebras de relaciones binarias extendidas con un operador nuevo, llamado *fork*, diseñado para tratar con objetos de estructura compleja (similar a la de los árboles binarios).

Definición 2.2.7 *Una* $\star$ Fork Álgebra Propia *sobre un conjunto* U *es una estructura*

$$\star\,\mathsf{PFA} = < \mathcal{R}, U, \cup, \cap, |, \underline{\nabla}, ', ^{\smile}, \emptyset, V, \Delta_{Dom(V)}, \star >$$

tal que:

1. $< \mathcal{R}, \cup, \cap, |, ', ^{\smile}, \emptyset, V, \Delta_{Dom(V)} >$ *es un álgebra relacional propia.*

2. $\star$ *es una operación binaria inyectiva de* V *en* U.

3. $\mathcal{R}$ *es cerrada para la operación* $\underline{\nabla}$ *definida por*

$$\mathsf{R}\,\underline{\nabla}\,\mathsf{S} = \{< x, (y \star z) > / < x, y > \in \mathsf{R},\ < x, z > \in \mathsf{S}\ \ y < y, z > \in V\}$$

■

La operación $\underline{\nabla}$ abstrae la operación $\star$, permitiendo manejar, en el nivel de las relaciones, las estructuras arbóreas generadas por esta última. Es importante destacar que estos "árboles" pueden representar órdenes no bien fundados, y tener entonces ramas infinitas. A pesar de que a primera vista puede parecer una desventaja, esta propiedad puede ser útil para modelizar procesos o computaciones infinitas, como se hizo en [FAN93]. Además, estas estructuras arbóreas permiten manejar tantas variables como sea necesario al representar fórmulas de primer orden mediante términos del álgebra.

Para definir las fork álgebras propias, ocultamos la operación $\star$ de la definición anterior.

Definición 2.2.8 *Una* Fork Álgebra Propia *sobre un conjunto U es el reducto*

$$\mathsf{PFA} = < \mathcal{R}, \cup, \cap, |, \underline{\nabla}, ', ^{\sim}, \emptyset, V, \Delta_{Dom(V)} >$$

de una $\star$ fork álgebra propia. ■

La estructura algebraica abstracta que describe a las fork álgebras propias es la Fork Álgebra Abstracta, cuya definición es:

Definición 2.2.9 *Una* Fork Álgebra (Abstracta) *es una estructura*

$$\mathsf{AFA} = < A, +, \bullet, ;, \nabla, ^{-}, ^{\smile}, 0, \infty, 1 >$$

(donde la operación ∇ es llamada fork*) tal que:*

1. $< A, +, \bullet, ;, ^{-}, ^{\smile}, 0, \infty, 1 >$ *es un álgebra relacional, con* 0 *su elemento* cero, ∞ *su elemento* unidad *y $\preceq$ (llamada* inclusión relacional*) el orden subyacente.*
2. $\mathsf{R} \nabla \mathsf{S} = (\mathsf{R}; (1 \nabla \infty)) \bullet (\mathsf{S}; (\infty \nabla 1))$
3. $(\mathsf{R} \nabla \mathsf{S}); (\mathsf{T} \nabla \mathsf{Q})^{\smile} = (\mathsf{R}; \mathsf{T}^{\smile}) \bullet (\mathsf{S}; \mathsf{Q}^{\smile})$
4. $(1 \nabla \infty)^{\smile} \nabla (\infty \nabla 1)^{\smile} \preceq 1$ ■

El teorema de representabilidad de las fork álgebras establece que toda fork álgebra es isomorfa a una fork álgebra propia y fue demostrado por primera vez en [FBHV95].

Teorema 2.2.10 *Toda fork álgebra abstracta $\mathcal{A}$ es isomorfa a alguna fork álgebra propia.*

El teorema de expresividad de las fork álgebras (demostrado en [VH91]) afirma que el poder expresivo de las fork álgebras propias abarca al del cálculo de predicados de primer orden.

Teorema 2.2.11 *Dado un lenguaje de primer orden $\mathcal{L}$ existe una función $\mathcal{T}$ que asigna a cada fórmula n-aria ϕ de $\mathcal{L}$ un fork-término* T*, tal que* T *es la internalización del conjunto $\{x/\phi(x)\}$.*

Ambos resultados establecen que las fork álgebras son un marco formal adecuado para sustentar la tarea de construir programas formalmente (ver Secc. 2.3). Esto es así debido a que el teorema de expresividad nos asegura que todo programa tendrá un término que lo represente, y el teorema de representabilidad nos asegura que los términos abstractos son una manera más adecuada a nuestros propósitos de escribir las relaciones binarias de entrada-salida.

Podemos ver en la Def. 2.2.9 que hay ciertas expresiones que aparecen reiteradas veces. Estas expresiones tienen significación en el cálculo de programas (ver Secc. 2.3), por lo que simplificar la notación es importante; para ello se introducen operaciones derivadas (construídas a partir de las operaciones básicas). Las definiciones se dan en forma abstracta, pero describimos a continuación cual es la intención de cada una de estas operaciones desde el punto de vista de los modelos estándar.

Definición 2.2.12 *Sean* π, ρ, $\otimes$, 2, $Dom()$ *y* $Ran()$ *las operaciones definidas por*

1. $\pi = (1 \nabla \infty)^{\smile}$ *(primera proyección)*
2. $\rho = (\infty \nabla 1)^{\smile}$ *(segunda proyección)*
3. $R \otimes S = (\pi ; R) \nabla (\rho ; S)$ *(producto)*
4. $2 = 1 \nabla 1$ *(duplicación de datos)*
5. $Dom(R) = (R ; R^{\smile}) \bullet 1$ *(dominio de R)*
6. $Ran(R) = (R^{\smile} ; R) \bullet 1$ *(rango de R)* ■

Las operaciones π y ρ representan a las relaciones de proyección respecto de la operación subyacente $\star$. Dado que la relación *1* representa a la relación identidad, la relación *2* representa a la operación de duplicación de sus datos de entrada, y $2^{\smile}$ resulta ser un filtro de igualdad sobre $< x \star y >$ tal que $x{=}y$. $Dom(R)$ y $Ran(R)$ son expresiones relacionales que caracterizan el *dominio* y el *rango* de la relación R.

Estas operaciones serán usadas en las derivaciones de programas, y, además, con ellas, los axiomas de la Def. 2.2.9 pueden reescribirse como:

- $R \nabla S = \left(R ; \breve{\pi}\right) \bullet \left(S ; \breve{\rho}\right)$ Def. 2.2.9-2
- $\pi \nabla \rho \preceq 1$ Def. 2.2.9-4

2.3 Fork Álgebras e Informática

En esta sección trataremos el tema de la utilización de las Fork Álgebras como base para la tarea de construcción formal de programas. Primero mostraremos como se expresarán los programas y las reglas de transformación con el lenguaje algebraico mediante algunos ejemplos, y luego mostraremos como se utilizará el lenguaje algebraico para expresar propiedades matemáticas de los programas.

2.3.1 Cálculo de Programas

El lenguaje de las fork álgebras se compone de términos, formados mediante las operaciones aplicadas sobre relaciones elementales (los elementos del conjunto A de la Def. 2.2.9). Cada uno de estos términos puede entenderse como la versión abstracta de una relación binaria (en el sentido conjuntista) que represente la relación de entrada-salida del programa que se quiere escribir. Por ejemplo, si contamos con un término relacional abstracto llamado mult que representa al programa que dados dos números naturales devuelve el resultado de multiplicarlos (mult sería la versión abstracta de $\{< (x \star y), x \times y > / x, y \in Nat\}$) podemos construir un término que represente al programa que dado un número lo eleva al cuadrado de la siguiente forma: $sqr = 2 ; mult$.

El uso de la relación *2* es la manera abstracta de expresar la replicación del dato de entrada, para poder multiplicarlo luego por sí mismo.

De la misma manera, las proyecciones π y ρ y la operación $\otimes$, permiten la manipulación abstracta de estructura de producto cartesiano. Estas estructuras, elementos complejos similares a los árboles binarios (cf. Secc. 2.2.3) se usarán en la representación conjuntista de los programas (las fork álgebras propias), para representar la entrada de un programa. Para poder operar sobre alguno de los componentes de este elemento complejo, necesitamos de programas que puedan expresar la 'extracción' de este componente; dichos programas pueden ser construídos de manera sistemática mediante el uso de las proyecciones π y ρ combinadas apropiadamente ([VHB92]), y se los suele llamar "ruido formal", ya que sólo tienen que ver con el reacomodamiento de los elementos abstraídos. Por ejemplo, para escribir un programa que dado un par de valores devuelve el mismo par, pero con los valores cambiados, escribiremos $((\rho \nabla \pi))$, y lo llamaremos swap; si queremos un programa que dado el elemento $< < x, y >, < z, w > >$ obtenga el elemento z, escribimos $\rho ; \pi$; si queremos un programa que dado $< < x, y >, < z, w > >$ devuelva $< y, z >$, escribimos $((\pi ; \rho) \nabla (\rho ; \pi))$.

Al considerar los términos como programas, las ecuaciones válidas en el álgebra (igualdades entre términos) se pueden interpretar como reglas que afirman que dos programas distintos tienen el mismo comportamiento de entrada-salida, y pueden, por lo tanto, ser utilizadas como reglas de transformación, que permiten reemplazar a uno de los programas por el otro, sin alterar el significado. Por ejemplo, por definición de $\otimes$, vale que $(\pi \nabla \rho) = (1 \otimes 1)$, lo cual puede ser utilizado en una derivación para eliminar proyecciones con el fin de simplificar la lectura del término (y eventualmente su eficiencia, según la complejidad de las proyecciones). Además, esta ecuación refleja la idea expuesta en el párrafo anterior, donde tomar la primera y segunda componentes de un par, y armar un nuevo par con ellas es la identidad sobre pares.

Por eso, las ecuaciones válidas en las fork álgebras son importantes, ya que son las que nos permitirán realizar las transformaciones de programas, y por eso el objetivo de esta tesis es encontrar ecuaciones válidas que permitan eliminar la implicación relacional en favor de ecuaciones recursivas.

En el Ap. B presentamos algunas de estas ecuaciones elementales, las cuales se utilizarán en las demostraciones de los distintos pasos de derivación del ejemplo del Cap. 3.

2.3.2 Expresando Propiedades Relacionalmente

¿Cómo expresamos propiedades de un programa mediante el lenguaje algebraico? Determinadas ecuaciones e inecuaciones permiten expresar propiedades, como por ejemplo

Definición 2.3.1 *Una relación* P *se dice*

1. funcional *si, y sólo si,* $\mathsf{P}^{\smile} ; \mathsf{P} \preceq 1$,

2. inyectiva *si, y sólo si,* $\mathsf{P} ; \mathsf{P}^{\smile} \preceq 1$, *y*

3. constante *si, y sólo si,* $\mathsf{P} \neq 0$, $\mathsf{P} = \infty;\mathsf{P}$, *y* $\mathsf{P}^{\smile};\mathsf{P} \preceq 1$ ■

La intuición de estas definiciones se obtiene interpretando las operaciones abstractas en el álgebra propia. Por ejemplo, la Def. 2.3.1-1 se puede interpretar de manera conjuntista como $\mathsf{P}^{\sim}|\mathsf{P} \subseteq \Delta$, que dice que no existe z tal que $(x\ \mathsf{P}^{\sim}\ z)$, $(z\ \mathsf{P}\ y)$ y $x \neq y$, o sea, que P es una relación funcional.

Otra forma de expresar propiedades es a través de la representación de conjuntos mediante términos del álgebra (llamada internalización de conjuntos). Existen varias maneras de internalizar conjuntos en un marco relacional (ver [VHB92]); por ejemplo, para representar al conjunto X:

- Hoare y He en [HH86a] usaron *vectores*, relaciones que cumplen que $\mathsf{R} = \mathsf{R};\infty$, y que representan de manera abstracta a la relación $\{< x, y > / x \in X\}$;
- Haeberer y Veloso en [VHB92] usaron identidades parciales, relaciones incluídas en 1, que se anotan como 1_X para el conjunto X, y que representan de manera abstracta a la relación $\{< x, x > / x \in X\}$.

En este trabajo utilizaremos la segunda opción, ya que puede ser interpretado como un programa que testee la pertenencia al conjunto que se internaliza (el programa acepta como entradas únicamente a los elementos del conjunto, y los devuelve sin cambio). Esta interpretación nos permite escribir términos que representen a las sentencias del estilo *if-then-else*, de la siguiente manera:

$(1_B;\mathsf{P} + 1_{\neg B};\mathsf{Q})$ puede representar al programa (*if* B *then* P *else* Q)

También se puede expresar tipos de datos; por ejemplo, el tipo de las listas se representará como $1_{\mathcal{L}^*}$. En [MB95a, MB95b] se muestra como estas identidades parciales pueden ser definidas en términos de operaciones primitivas de los tipos. En esta tesis los tipos de datos (y en particular el tipo de los árboles binarios) se asumirán conocidos; esto significa que tendremos relaciones que simbolicen a las operaciones elementales sobre el tipo en cuestión, junto con sus propiedades.

Las identidades parciales se pueden utilizar también para expresar correctitud parcial, pre y poscondiciones y terminación. Supongamos que P es un término que denota a un programa, S uno que representa una especificación y W un conjunto; entonces podemos escribir $(1_W;\mathsf{P} \preceq \mathsf{S})$ para expresar que el programa P es parcialmente correcto respecto de S, o que W es una precondición para P y podemos escribir $(1_W \preceq \mathsf{P};\mathsf{P}^{\smile})$ para expresar que con entrada en el conjunto W, P termina.

Una forma particular de definición de relaciones es mediante el uso de ecuaciones recursivas. Dichas ecuaciones definirán la menor relación (en el sentido de inclusión relacional) que sea solución de la ecuación. Dado que, salvo el complemento, todas las demás operaciones son continuas, esto es correcto, siempre y cuando no utilicemos expresiones complementadas como parte de la ecuación recursiva. Una salvedad a esta restricción es que podemos utilizar un número par de operaciones

de complemento, ya que el complemento es una operación antitónica (ver [SS93] para más detalles sobre las operaciones antitónicas).

¿Cuáles son los pasos que se deben realizar para conseguir un programa eficiente a partir de una especificación? El primer paso consiste en escribir la especificación utilizando el cálculo de predicados de primer orden para definir la relación de entrada-salida del programa deseado. Luego, utilizando el teorema de expresividad, encontramos un fork-término que represente al término de primer orden. Finalmente aplicamos reglas de derivación hasta obtener un fork-término que sólo utilice operaciones algorítmicas, el cual será interpretado como un programa. Este es el proceso que se utiliza en el ejemplo de la Secc. 3.3.

Capítulo 3

Tratamiento de problemas que involucran cuantificación universal

"Y que bastaba decir pez para nombrar al pez, sin ocultar su concepto con sonidos engañosos."

Umberto Eco, El Nombre de la Rosa.

3.1 Generalidades

Una construcción de la lógica que es de gran utilidad a la hora de especificar programas es la cuantificación universal. Utilizando expresiones que involucran cuantificación universal podemos hablar de un elemento que sea especial entre todos los de su misma "especie" (por ejemplo, el elemento mínimo o máximo de un conjunto, el ínfimo o supremo de otros dos, etc.), podemos pedir que todos los elementos de una estructura compuesta cumplan cierta propiedad (por ejemplo, todos los elementos del subárbol izquierdo son menores que la raíz, etc.), y muchas otras propiedades. Por ello consideramos a la cuantificación universal como una "herramienta" poderosa para realizar especificaciones. La desventaja de esta herramienta de especificación es que las expresiones que la utilizan no sugieren directamente algoritmos, ya que habitualmente requieren testear un número grande (o infinito) de casos en el rango de la cuantificación. Ahora bien, ¿cómo obtenemos un algoritmo a partir de una especificación que involucra cuantificación universal? Claramente, necesitamos un conjunto de reglas de derivación que nos permitan manipular este tipo de expresiones y "eliminar" la cuantificación. Pero, dado que el caso general de cuantificación universal no tiene una solución general (en el sentido de obtener una expresión algorítmica equivalente), hemos elegido para su estudio una forma especial de fórmulas, donde la cuantificación discurre sobre una implicación:

$$(\forall z)(x \ \mathsf{R} \ z \ \rightarrow \ y \ \mathsf{S} \ z)$$

En este capítulo presentaremos la *implicación relacional*, una operación derivada en las fork álgebras; esta operación será la contrapartida algebraica de las fórmulas cuya forma sea la mencionada, permitiéndonos expresar cuantificación universal en las especificaciones hechas con el lenguaje de las fork álgebras. Daremos un conjunto de propiedades que la implicación relacional cumple, y que serán de gran utilidad como reglas de derivación, lo cual será mostrado mediante un caso de estudio que es representativo de un buen número de problemas expresables mediante cuantificación universal. Las reglas obtenidas nos permitirán obtener un algoritmo recursivo a partir de la especificación original.

3.2 La Implicación Relacional

3.2.1 Definición y Explicación Intuitiva

La cuantificación universal es una herramienta poderosa para realizar especificaciones. La fórmula

$$\phi(x, y) \ = \ (\forall z)(x \ \mathsf{R} \ z \ \rightarrow \ y \ \mathsf{S} \ z)$$

es un caso particular de cuantificación universal, donde la misma discurre sobre una implicación.

Una forma equivalente de ϕ es

$$\phi(x, y) \quad sii \quad (\forall z)(\neg(x \ \mathsf{R} \ z) \ \vee \ z \ \mathsf{S}^{\smile} \ y)$$

Nuestro interés es definir una operación abstracta "$\rightarrow$", que describa esta fórmula en el álgebra. En [Tar41] se definió la suma relativa:

$$x \ (\mathsf{R} \oplus \mathsf{S}) \ y \quad sii \quad (\forall z)(x \ \mathsf{R} \ z \ \vee \ z \ \mathsf{S} \ y)$$

llevándonos a la equivalencia

$$\phi(x, y) \quad sii \quad x \ (\mathsf{R}' \oplus \mathsf{S}^{\smile}) \ y$$

En el mismo artículo se dió una caracterización abstracta de la suma relativa, que se basa en la equivalencia lógica entre $(\forall x)(\mathsf{P}(x))$ y $\neg(\exists x)(\neg \mathsf{P}(x))$. Así, $(\forall z)(x \ \mathsf{R} \ z \ \vee \ z \ \mathsf{S} \ y)$ es equivalente a $\neg(\exists z)(\neg(x \ \mathsf{R} \ z) \ \wedge \ \neg(z \ \mathsf{S} \ y))$ y entonces se puede escribir

$$\mathsf{R} \oplus \mathsf{S} \ = \ \overline{\overline{\mathsf{R}}; \overline{\mathsf{S}}}$$

lo cual establece que $\oplus$ no es una operación fundamental en un álgebra relacional; de las fórmulas anteriores obtenemos la descripción abstracta de la implicación.

Definición 3.2.1 *La implicación relacional entre* R *y* S *se define como*

$$(\mathsf{R} \rightarrow \mathsf{S}) \ = \ \overline{\mathsf{R}; \overline{\mathsf{S}}^{\smile}}$$

■

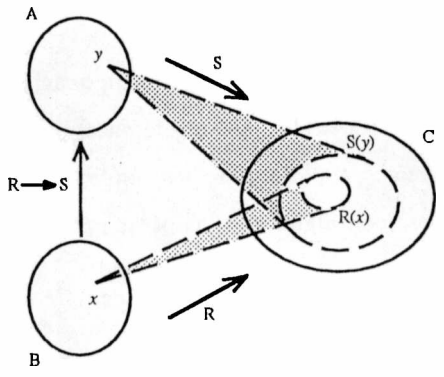


Figura 3.1: Representación gráfica de la implicación relacional $(R \rightarrow S)$

En los modelos estándar, podemos entender a la implicación relacional como: "x se relaciona con y vía $(R \rightarrow S)$ si, y sólo si, la imagen de x por R está contenida en la imagen de y por S" (ver Fig. 3.1).

3.2.2 Propiedades

Puesto que la caracterización algebraica de "$\rightarrow$" es incómoda para manipulación algebraica (ya que el complemento no tiene un comportamiento agradable sobre la composición), presentaremos algunas propiedades de "$\rightarrow$" que nos permitirán evitar el uso de su definición en las derivaciones. Estas propiedades son la principal contribución de este trabajo.

La primera de las propiedades comprende dos consecuencias inmediatas de la definición.

Proposición 3.2.2 *Sean* P, Q *y* R *relaciones cualesquiera. Se cumple que*

1. $((P+Q)\rightarrow R) = (P\rightarrow R) \bullet (Q\rightarrow R)$
2. $(P\rightarrow(Q\bullet R)) = (P\rightarrow Q) \bullet (P\rightarrow R)$

Dem. Ver Prop. 4.1.1. ■

La segunda propiedad establece que la cuantificación universal está contenida en una cuantificación existencial.

Proposición 3.2.3 *Se cumple que* $Dom(R);(R\rightarrow S) \preceq R;S^{\smile}$.

Dem. Ver Prop. 4.1.3. ■

La tercera propiedad presentada establece lo que sucede cuando la implicación está fuera del dominio del antecedente; en los modelos estándar, la interpretación de esta propiedad es intuitiva, ya que al estar fuera del dominio, la imagen por P es vacía.

Proposición 3.2.4 *Siendo* P *y* Q *relaciones cualesquiera, se cumple que*

$$\left(\overline{Dom(P)}\bullet 1\right);(P\rightarrow Q) = \left(\overline{Dom(P)}\bullet 1\right);\infty$$

Dem. Ver Prop. 4.1.2. ∎

La siguiente establece una manera general de simplificar la implicación relacional cuando el antecedente es una suma relacional y sólo interesa lo que sucede en el dominio de dicha suma. A primera vista la expresión resultante es más compleja, pero las implicaciones tienen componentes más simples y por lo tanto pueden tratarse con alguna de las otras propiedades.

Proposición 3.2.5 *Sean* P, Q *y* R *relaciones cualesquiera. Se cumple que*

$$\begin{aligned} &Dom(P+Q);((P+Q)\rightarrow R) = \\ &Dom(P);\Big((P\rightarrow R)\bullet\Big(Dom(Q);(Q\rightarrow R) + \Big(\overline{Dom(Q)}\bullet 1\Big);\infty\Big)\Big) \\ &+ \\ &Dom(Q);\Big((Q\rightarrow R)\bullet\Big(Dom(P);(P\rightarrow R) + \Big(\overline{Dom(P)}\bullet 1\Big);\infty\Big)\Big) \end{aligned}$$

Dem. Ver Prop. 4.1.4. ∎

Las dos proposiciones siguientes sirven para "sacar" relaciones funcionales fuera de la implicación relacional. El primero de los casos elimina totalmente la implicación, y el segundo la simplifica.

Proposición 3.2.6 *Sea* A *una relación funcional. Entonces* $Dom(A);(A\rightarrow P) = A;P^{\smile}$.

Dem. Ver Prop. 4.1.5. ∎

Proposición 3.2.7 *Sea* B *una relación funcional. Entonces se cumple que*

$$Dom(B);(B;P\rightarrow Q) = B;(P\rightarrow Q)$$

Dem. Ver Prop. 4.1.6. ∎

Por último, daremos dos propiedades que permiten simplificar la implicación relacional; esta simplificación se obtiene trasladando la definición recursiva del antecedente a una especificación recursiva para la implicación relacional.

Proposición 3.2.8 *Sean* A *y* B *relaciones funcionales y sean* $P = A+(B;P)$ *y* $T = P\rightarrow Q$, *cumpliendo que*

- $Dom(P) = Dom(A)+Dom(B)$, *y*
- $Dom(A)\bullet Dom(B) = 0$.

Entonces $Dom(P);T = A;Q^{\smile} + B;T$.

Dem. Ver Prop. 4.1.7. ∎

Proposición 3.2.9 *Sean* A, B *y* C *relaciones funcionales, y sean* $T = (P\rightarrow Q)$ *y* $P = A+(B;P)+(C;P)$ *cumpliendo que*

- $Dom(P) = Dom(A)+Dom(B)+Dom(C)$,

- $Dom(\mathsf{A}) \bullet Dom(\mathsf{B}) = 0$,
- $Dom(\mathsf{A}) \bullet Dom(\mathsf{C}) = 0$, *y*
- $Dom(\mathsf{B}) \bullet Dom(\mathsf{C}) = 0$.

Entonces $Dom(\mathsf{P}); \mathsf{T} = \mathsf{A}; \mathsf{Q}^{\smile} + \mathsf{B}; \mathsf{T} + \mathsf{C}; \mathsf{T}$.

Dem. Ver Prop. 4.1.8. ∎

3.3 Un Estudio de Caso: Mínimo Ancestro Común

En estas sección presentaremos un problema que se especifica fácilmente utilizando lógica de primer orden, y cuya especificación relacional se construye usando la implicación relacional. Luego mostraremos una derivación que nos permitirá obtener una expresión recursiva para el problema planteado. Las propiedades de la implicación relacional presentadas en la Secc. 3.2 serán utilizadas en la derivación, mostrando así su utilidad en la tarea de la construcción formal de programas. Las demostraciones de las propiedades particulares utilizadas en la derivación se presentan en la Secc. 4.2, con el objetivo de no oscurecer el ejemplo con los detalles técnicos. Solamente mostraremos las manipulaciones formales cuando se utilicen las propiedades de la implicación relacional cuya aplicación queremos ejemplificar.

3.3.1 Especificación del Problema en Lógica de Primer Orden

El problema que presentamos puede ser enunciado de la siguiente manera:

> "Dados un Arbol Binario T sin nodos repetidos y dos elementos x e y pertenecientes a T, encontrar el *Mínimo Ancestro Común* de x e y. El *Mínimo Ancestro Común* (MAC) de dos nodos es el ancestro común a dichos nodos que cumple que todo otro ancestro común a ellos es a su vez ancestro de él".

Las relaciones que se utilizan a lo largo de esta sección se interpretan como programas que, cuando reciben una entrada, producen un resultado que se retorna como salida. Con esta idea en mente, sea Anc (abreviatura de Ancestros) una relación tal que, dados un árbol T y un elemento x de T, retorne los ancestros de x en T. Con esta relación podemos construir una especificación formal de la relación AC (abreviatura de Ancestro_Comun), que toma un árbol T y dos elementos x e y de él, y retorna los ancestros comunes de x e y.

$$< T, < x, y > > \ \mathsf{AC} \ \ a \ \ \textit{sii} \ \ < T, x > \ \mathsf{Anc} \ \ a \ \ \wedge \ \ < T, y > \ \mathsf{Anc} \ \ a$$

Con AC construímos la especificación formal de la relación MAC, que captura el problema en el lenguaje de la lógica de primer orden.

$$< T, < x, y > > \ \mathsf{MAC} \ \ a \ \ \textit{sii} \ \ < T, < x, y > > \ \mathsf{AC} \ \ a \ \ \wedge \ (\forall z)(< T, < x, y > > \ \mathsf{AC} \ \ z \ \ \rightarrow \ \ < T, a > \ \mathsf{Anc} \ \ z)$$

Faltan dar fórmulas que expresen la relación Anc y las relaciones complejas del tipo de datos árbol. La relación Anc queda especificada por:

$$< T, x > \text{ Anc } a \quad sii \quad (\exists T')(T' \sqsubseteq T \ \wedge \ T' \text{ raíz } a \ \wedge \ T' \text{ en } x),$$

La relación $\sqsubseteq$ retorna todos los árboles que contienen a un árbol dado T' como subárbol; usaremos también la conversa de $\sqsubseteq$, denotada por $\sqsupseteq$.

$$T' \sqsubseteq T \quad sii \quad (T' = T) \ \vee \ (\exists T'')((T \text{ der } T'' \ \vee \ T \text{ izq } T'') \ \wedge \ T' \sqsubseteq T'')$$

La relación en dado un árbol, retorna sus elementos.

$$T \text{ en } x \quad sii \quad (x \text{ raíz } T) \ \vee \ (\exists T')((T \text{ der } T' \ \vee \ T \text{ izq } T') \ \wedge \ T' \text{ en } x)$$

Las relaciones raíz, izq y der son las relaciones básicas del tipo de datos subyacente, árbol binario, y se asumen conocidas, así como sus propiedades (algunas de las cuales se listan en la Secc. 4.2.1). Las tres reciben a un árbol como entrada y retornan la raíz, el subárbol derecho y el subárbol izquierdo, respectivamente.

3.3.2 Especificación del Problema en Fork Álgebra Abstracta

Cada una de las fórmulas de primer orden que aparecen en la especificación se traducen a un término de AFA, por el teorema de expresividad. La fórmula que define a la relación MAC se traduce a $\mathsf{MAC} = \mathsf{MAC}^{\circ}; \rho$, donde

$$\mathsf{MAC}^{\circ} = \begin{pmatrix} \pi \\ \nabla \\ \mathsf{AC} \end{pmatrix} \bullet (\mathsf{AC} \rightarrow \mathsf{Anc}) \tag{3.1}$$

La proyección π en el primer subtérmino de la Ecu. 3.1 se utiliza para conservar una copia del árbol binario de entrada, con el fin de compararlo con el obtenido por el segundo subtérmino. La interpretación de este término es casi directa al compararlo con la fórmula de la que proviene.

La fórmula que define a AC se traduce a

$$\mathsf{AC} = ((\pi \nabla \rho; \pi); \mathsf{Anc}) \bullet ((\pi \nabla \rho; \rho); \mathsf{Anc}) \tag{3.2}$$

Las proyecciones se utilizan para preparar la entrada correcta a la relación Anc; este mecanismo se explicó en la Secc. 2.3. Nuevamente la interpretación es directa.

También tendremos un término relacional para la relación Anc:

$$\mathsf{Anc} = \begin{pmatrix} \sqsupseteq \\ \otimes \\ 1 \end{pmatrix}; \begin{pmatrix} \pi; \text{raíz} \\ \nabla \\ (\text{en} \otimes 1); \breve{2} \end{pmatrix}; \pi \tag{3.3}$$

Este término se obtiene a partir de la fórmula de la sección previa, pero utiliza al fork como expresión condicional (recordar la Prop. B.5-1 y la representación del *if-then-else* vista en la Secc. 2.3).

El subtérmino $(\sqsupseteq \otimes 1)$ elige un subárbol del árbol de entrada. El siguiente subtérmino es la expresión condicional: $(\pi ; \text{raíz})$ selecciona la raíz del subárbol, y la misma es retornada si el elemento de entrada pertenece al subárbol, lo cual es evaluado por el subtérmino $(\text{en} \otimes 1) ; 2^{\smile}$. Recordemos que la operación $2^{\smile}$ funciona como filtro de igualdad; entonces $(\text{en} \otimes 1)$ elige un elemento del subárbol y lo compara por igualdad con el elemento de entrada, lo cual es una manera de preguntar por pertenencia.

Los términos para las relaciones $\sqsupseteq$ y en se obtienen directamente a partir de sus fórmulas, considerando a la igualdad como la relación identidad del tipo dado:

$$\sqsupseteq \;=\; 1_{Arbol} \;+\; (\text{der} + \text{izq}) ; \sqsupseteq \tag{3.4}$$

$$\text{en} \;=\; \text{raíz} \;+\; (\text{der} + \text{izq}) ; \text{raíz} \tag{3.5}$$

Es interesante observar que con estas definiciones se cumple que $\text{en} = \sqsupseteq ; \text{raíz}$ (cf. Prop. 4.2.6).

3.3.3 Derivación de una especificación recursiva para MAC

Para derivar un fork–término recursivo a partir de la especificación de MAC dada más arriba, seguiremos varios pasos. El primero de ellos consiste en derivar un fork–término recursivo para la relación Anc y con éste, un fork–término recursivo para la relación AC. Finalmente, utilizando las propiedades de la implicación relacional introducidas en la Secc. 3.2.2 y el término obtenido para AC, derivaremos el término buscado para MAC.

Las demostraciones de los pasos de derivación se dan en el Cap. 4, con el fin de simplificar la lectura de esta sección omitiendo los detalles técnicos. Sólo se muestra la porción de la derivación donde se utilizan las proposiciones de eliminación de cuantificación universal.

Como primer punto, debemos notar que el tipo de datos de los árboles binarios puede particionarse en tres subconjuntos: el que contiene sólo al árbol vacío, el que contiene a todos los árboles que se componen únicamente de una hoja, y el que contiene a todos los árboles con al menos un subárbol no vacío, como queda expresado en la Prop. 4.2.1.

Como el dominio de la relación Anc es $(1_{Arbol} \otimes 1)$, aplicando dicha propiedad, obtenemos

$$\text{Anc} \;=\; (1_{Nil} \otimes 1) ; \text{Anc} \;+\; (1_{T_1} \otimes 1) ; \text{Anc} \;+\; (1_{T>1} \otimes 1) ; \text{Anc}$$

que expresa un análisis de casos sobre la entrada de la relación Anc.

Analizaremos cada uno de los tres términos independientemente.

La Prop. 4.2.9 establece que el primero se reduce a 0; intuitivamente es correcto, ya que el árbol vacío no tiene elementos.

Las Props. 4.2.10 y 4.2.11 establecen el caso base y el caso recursivo para el segundo y el tercero, respectivamente.

$$(1_{T_1} \otimes 1); \mathsf{Anc} = ((1_{T_1}; \mathsf{raíz}) \otimes 1); \breve{2}$$

y

$$(1_{T>1} \otimes 1); \mathsf{Anc} = \begin{pmatrix} \pi; 1_{T>1}; \mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1); \breve{2} \end{pmatrix}; \pi + \begin{pmatrix} \mathsf{der} + \mathsf{izq} \\ \otimes \\ 1 \end{pmatrix}; \mathsf{Anc}$$

La intuición de la Prop. 4.2.10 es que si un árbol se compone sólo de un nodo, el único ancestro es el elemento en ese nodo (si el elemento pertenece al árbol). La intuición de la Prop. 4.2.11 es que, si el elemento pertenece al árbol, la raíz es su ancestro, y el resto de los ancestros se obtienen recursivamente de los subárboles.

Poniendo juntas estas expresiones obtenemos la siguiente expresión recursiva de Anc:

$$\mathsf{Anc} = ((1_{T_1}; \mathsf{raíz}) \otimes 1); \breve{2} + \begin{pmatrix} \pi; 1_{T>1}; \mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1); \breve{2} \end{pmatrix}; \pi + \begin{pmatrix} \mathsf{der} + \mathsf{izq} \\ \otimes \\ 1 \end{pmatrix}; \mathsf{Anc} \qquad (3.6)$$

La derivación continúa con la relación AC. La Prop. 4.2.12 establece una forma recursiva para ella, obtenida realizando el unfold de la Ecu. 3.6 en la Ecu. 3.2 y algunas manipulaciones algebraicas:

$$\begin{aligned} \mathsf{AC} = {} & \left(1_{T_1}; \mathsf{raíz} \otimes \breve{2}\right); \breve{2} \\ & + (1_{T>1} \otimes 1); \underbrace{\left(Dom\left((\mathsf{en} \otimes \pi); \breve{2}\right) \bullet Dom\left((\mathsf{en} \otimes \rho); \breve{2}\right)\right)}_{\phi}; \pi; \mathsf{raíz} \\ & + ((\mathsf{der} \otimes 1) + (\mathsf{izq} \otimes 1)); \mathsf{AC} \end{aligned}$$

Llamaremos A al primer subtérmino, B al segundo, y $\mathsf{C} = (\mathsf{C}_1 + \mathsf{C}_2)$ al primer factor del tercer subtérmino; con lo cual la ecuación anterior queda:

$$\mathsf{AC} = \mathsf{A} + \mathsf{B} + \mathsf{C}; \mathsf{AC} \qquad (3.7)$$

El subtérmino A establece que si un árbol está compuesto sólo por un nodo, el único ancestro común a los elementos de entrada puede ser la raíz, y sólo en el caso en que ambos elementos sean iguales a ella. El subtérmino B establece que si ambos elementos pertenecen al árbol, la raíz del mismo es un ancestro común a ellos. El tercer subtérmino establece que los demás ancestros comunes se obtienen recursivamente de los subárboles izquierdo y derecho.

Ahora, haciendo unfold de la Ecu. 3.7 en la Ecu. 3.1, tenemos que $\mathsf{MAC}^\circ =$

$$\begin{pmatrix} \textit{aplicando unfold sobre Ecu. 3.7 y} \\ \textit{distribuyendo } \nabla \textit{ sobre } + \end{pmatrix}$$

$$= ((\pi \nabla \mathsf{A}) + (\pi \nabla \mathsf{B}) + (\pi \nabla (\mathsf{C}; \mathsf{AC}))) \bullet ((\mathsf{A} + \mathsf{B} + \mathsf{C}; \mathsf{AC}) \rightarrow \mathsf{Anc}) =$$

$$\left(\textit{por Prop. 4.2.13} \right)$$

$$= (\pi \nabla A) \bullet (A ; Anc^{\smile}) + ((\pi \nabla B) + (\pi \nabla (C ; AC))) \bullet ((B + C ; AC) \rightarrow Anc) =$$

$$\left(\textit{por Prop. 4.2.14} \right)$$

$$\begin{aligned} = \; & (\pi \nabla A) \bullet (A ; Anc^{\smile}) \\ & + (\pi \nabla B) \bullet (B ; Anc^{\smile}) \bullet (Dom(C_1 ; AC) ; C_1 ; (AC \rightarrow Anc)) \\ & + (\pi \nabla B) \bullet (B ; Anc^{\smile}) \bullet (Dom(C_2 ; AC) ; C_2 ; (AC \rightarrow Anc)) \\ & + (\pi \nabla B) \bullet (B ; Anc^{\smile}) \bullet \left(Dom(B) ; \left(\overline{Dom(C ; AC)} \bullet 1\right) ; \infty\right) \\ & + (\pi \nabla (C ; AC)) \bullet (B ; Anc^{\smile}) \bullet (Dom(C_1 ; AC) ; C_1 ; (AC \rightarrow Anc)) \\ & + (\pi \nabla (C ; AC)) \bullet (B ; Anc^{\smile}) \bullet (Dom(C_2 ; AC) ; C_2 ; (AC \rightarrow Anc)) \\ & + (\pi \nabla (C ; AC)) \bullet (B ; Anc^{\smile}) \bullet \left(Dom(B) ; \left(\overline{Dom(C ; AC)} \bullet 1\right) ; \infty\right) = \end{aligned}$$

$$\left(\textit{por Prop. 4.2.15} \right)$$

$$\begin{aligned} = \; & (\pi \nabla A) \\ & + (\pi \nabla B) \bullet (Dom(C_1 ; AC) ; C_1 ; (AC \rightarrow Anc)) && \text{(I)} \\ & + (\pi \nabla B) \bullet (Dom(C_2 ; AC) ; C_2 ; (AC \rightarrow Anc)) && \text{(II)} \\ & + (\pi \nabla B) \bullet \left(Dom(B) ; \left(\overline{Dom(C ; AC)} \bullet 1\right) ; \infty\right) && \text{(III)} \\ & + (\pi \nabla (C ; AC)) \bullet (Dom(C_1 ; AC) ; C_1 ; (AC \rightarrow Anc)) && \text{(IV)} \\ & + (\pi \nabla (C ; AC)) \bullet (Dom(C_2 ; AC) ; C_2 ; (AC \rightarrow Anc)) && \text{(V)} \\ & + (\pi \nabla (C ; AC)) \bullet \left(Dom(B) ; \left(\overline{Dom(C ; AC)} \bullet 1\right) ; \infty\right) && \text{(VI)} \end{aligned}$$

A esta altura de la derivación comenzamos a ver aparecer un algoritmo.

El subtérmino $(\pi \nabla A)$ es el mismo que para el caso de AC, ya que el árbol tiene sólo un nodo.

La Prop. 4.2.16 establece que I, II y VI son iguales a 0. La intuición para los dos primeros es que, mientras $(\pi \nabla B)$ selecciona la raíz del árbol de entrada, el otro intersecando elige un elemento perteneciente a alguno de los subárboles, con lo que, dado que el árbol tiene todos los elementos distintos, no pueden ser el mismo elemento, y entonces la intersección da 0; para VI es muy sencillo ver que da 0, pues los intersecandos tienen dominios disjuntos.

La Prop. 4.2.17 establece el caso base,

$$III = Dom \left(\begin{array}{c} \overline{Dom((der \otimes 1) ; AC)} \bullet 1 \\ \nabla \\ \overline{Dom((izq \otimes 1) ; AC)} \bullet 1 \end{array} \right) ; \phi ; \pi ; \left(\begin{array}{c} 1 \\ \nabla \\ raíz \end{array} \right)$$

que dice que si ambos elementos están en subárboles distintos, entonces la raíz es el mínimo ancestro común.

La Prop. 4.2.20 establece los casos recursivos

$$IV = Dom((der \otimes 1) ; AC) ; (\pi \nabla ((der \otimes 1) ; MAC))$$

y

$$\mathsf{V} \;=\; \mathit{Dom}((\mathsf{izq}\otimes \mathit{1});\mathsf{AC});(\pi\,\nabla((\mathsf{izq}\otimes \mathit{1});\mathsf{MAC}))$$

que dicen que si ambos elementos están en el mismo subárbol, entonces el mínimo ancestro común se obtiene recursivamente, aplicando MAC al subárbol correspondiente.

Juntando todas las expresiones previas no nulas, obtenemos la especificación para MAC:

$$\begin{aligned}\mathsf{MAC} \;=\; \mathsf{MAC}^{\circ};\rho \;=\;\; & (\pi\,\nabla\,\mathsf{A});\rho \\ & +\; \mathit{Dom}\begin{pmatrix} \overline{\mathit{Dom}((\mathsf{der}\otimes \mathit{1});\mathsf{AC})\bullet \mathit{1}} \\ \nabla \\ \overline{\mathit{Dom}((\mathsf{izq}\otimes \mathit{1});\mathsf{AC})\bullet \mathit{1}}\end{pmatrix};\phi;\pi;\begin{pmatrix} \mathit{1} \\ \nabla \\ \mathsf{raíz}\end{pmatrix};\rho \\ & +\; \mathit{Dom}((\mathsf{der}\otimes \mathit{1});\mathsf{AC});(\pi\,\nabla((\mathsf{der}\otimes \mathit{1});\mathsf{MAC}));\rho \\ & +\; \mathit{Dom}((\mathsf{izq}\otimes \mathit{1});\mathsf{AC});(\pi\,\nabla((\mathsf{izq}\otimes \mathit{1});\mathsf{MAC}));\rho\end{aligned}$$

la cual, extendiendo las abreviaturas y usando las Props. B.5 y B.13, MAC queda igual a

$$\begin{aligned}& \begin{pmatrix} \mathit{1}_{T_1};\mathsf{raíz} \\ \otimes \\ \breve{2}\end{pmatrix};\breve{2} \\ & +\; \mathit{Dom}\begin{pmatrix} \overline{\mathit{Dom}((\mathsf{der}\otimes \mathit{1});\mathsf{AC})\bullet \mathit{1}} \\ \nabla \\ \overline{\mathit{Dom}((\mathsf{izq}\otimes \mathit{1});\mathsf{AC})\bullet \mathit{1}}\end{pmatrix};\Big(\mathit{Dom}\Big((\mathsf{en}\otimes\pi);\breve{2}\Big)\bullet \mathit{Dom}\Big((\mathsf{en}\otimes\rho);\breve{2}\Big)\Big);\pi;\mathsf{raíz} \\ & +\; \mathit{Dom}((\mathsf{der}\otimes \mathit{1});\mathsf{AC});(\mathsf{der}\otimes \mathit{1});\mathsf{MAC} \\ & +\; \mathit{Dom}((\mathsf{izq}\otimes \mathit{1});\mathsf{AC});(\mathsf{izq}\otimes \mathit{1});\mathsf{MAC}\end{aligned} \tag{3.8}$$

Como se mencionó en la Secc. 2.3, los términos que involucran dominios pueden interpretarse como tests para sentencias condicionales. Teniendo esto en cuenta, la Ecu. 3.8 sugiere la siguiente función como algoritmo:

$$\begin{aligned}& \mathsf{MAC}(t,\,x,\,y) \\ & \quad = \mathsf{raíz}(t) && \text{, si } \{\text{“ } t \textit{ es sólo una hoja, y } x \textit{ e } y \textit{ son iguales a } \mathsf{raíz}(t) \text{ ”}\} \\ & \quad = \mathsf{raíz}(t) && \text{, si } \{\text{“ } x \textit{ e } y \textit{ no están en el mismo subárbol ”}\} \\ & \quad = \mathsf{MAC}(\mathsf{der}(t),\,x,\,y) && \text{, si } \{\text{“ } x \textit{ e } y \textit{ están en el subárbol derecho ”}\} \\ & \quad = \mathsf{MAC}(\mathsf{izq}(t),\,x,\,y) && \text{, si } \{\text{“ } x \textit{ e } y \textit{ están en el subárbol izquierdo ”}\}\end{aligned}$$

La especificación de MAC establece que dicha relación es funcional, y por ello podemos traducirla a una función. En los casos en que las relaciones no son funcionales la traducción debe utilizar, o bien algún lenguaje que posea no-determinismo, o alguna representación del mismo, como por ejemplo el método de lista de éxitos ([Wad85]).

De esta manera concluye el ejemplo.

Capítulo 4

Demostraciones

"...Si lees esta escritura y das su interpretación, llevarás al cuello collar de oro y serás el tercero en el reino."

Daniel, 5, 16.

Este capítulo recopila todas las demostraciones de las proposiciones utilizadas en el Cap. 3. Si bien podría haberse colocado como apéndice, consideramos que las demostraciones son una parte importante de esta tesis y por ello merecen formar parte de la estructura principal del trabajo. De todas maneras, este capítulo puede ser salteado en caso de que no se tenga interés en las demostraciones.

4.1 Propiedades de la implicación relacional

Proposición 4.1.1 *[Prop. 3.2.2] Sean* P, Q *y* R *relaciones cualesquiera. Se cumple que*

1. $((\mathsf{P}+\mathsf{Q})\rightarrow\mathsf{R}) = (\mathsf{P}\rightarrow\mathsf{R}) \bullet (\mathsf{Q}\rightarrow\mathsf{R})$
2. $(\mathsf{P}\rightarrow(\mathsf{Q}\bullet\mathsf{R})) = (\mathsf{P}\rightarrow\mathsf{Q}) \bullet (\mathsf{P}\rightarrow\mathsf{R})$

Dem. Para demostrar el primer ítem, hacemos:

$((\mathsf{P}+\mathsf{Q})\rightarrow\mathsf{R}) =$

(*por Def. 3.2.1*)

$= \overline{\overline{(\mathsf{P}+\mathsf{Q})};\overline{\mathsf{R}^{\smile}}} =$

(*distribuyendo ; sobre +*)

$= \overline{\overline{\mathsf{P};\overline{\mathsf{R}^{\smile}}+\mathsf{Q};\overline{\mathsf{R}^{\smile}}}} =$

(*por Ley de DeMorgan*)

$= \left(\overline{\overline{\mathsf{P};\overline{\mathsf{R}^{\smile}}}}\right)\bullet\left(\overline{\overline{\mathsf{Q};\overline{\mathsf{R}^{\smile}}}}\right) =$

(*por Def. 3.2.1*)

$= (P \rightarrow R) \bullet (Q \rightarrow R)$

El segundo ítem se demuestra de manera análoga. ■

Proposición 4.1.2 *[Prop. 3.2.4] Siendo* P *y* Q *relaciones cualesquiera, se cumple que*

$$\left(\overline{Dom(P)} \bullet 1\right);(P \rightarrow Q) = \left(\overline{Dom(P)} \bullet 1\right);\infty$$

Dem. La desigualdad $\preceq$ es trivial, ya que $(P \rightarrow Q) \preceq \infty$.

Para la desigualdad $\succeq$, vemos que

$\left(\overline{Dom(P)} \bullet 1\right);(P \rightarrow Q) =$

(*aplicando unfold sobre Def. 3.2.1*)

$= \left(\overline{Dom(P)} \bullet 1\right);\left(\overline{P;\overline{Q^{\smile}}}\right) =$

(*por Prop. B.13-1*)

$= \left(\overline{Dom(P)} \bullet 1\right);\left(\overline{Dom(P);P;\overline{Q^{\smile}}}\right) \succeq$

$\succeq \left(\overline{Dom(P)} \bullet 1\right);\left(\overline{Dom(P);\infty}\right) =$

(*por Prop. B.20*)

$= \left(\overline{Dom(P)} \bullet 1\right);\left(\overline{Dom(P)} \bullet 1\right);\infty =$

(*por Prop. B.17*)

$= \left(\overline{Dom(P)} \bullet 1\right);\infty$

■

Proposición 4.1.3 *[Prop. 3.2.3] Se cumple que* $Dom(R);(R \rightarrow S) \preceq R;S^{\smile}$.

Dem. Para demostrar esta propiedad veremos que la intersección entre el primer término y el complemento del segundo da *0*. O sea

$(Dom(R);(R \rightarrow S)) \bullet \left(\overline{R;S^{\smile}}\right) =$

(*aplicando unfold sobre Def. 3.2.1*)

$= \left(Dom(R);\overline{\overline{R;S^{\smile}}}\right) \bullet \left(\overline{R;S^{\smile}}\right) =$

(*por Prop. B.14*)

$= Dom(R);\left(\overline{R;S^{\smile}} \bullet \overline{R;S^{\smile}}\right) =$

(*por Ley de DeMorgan*)

$= Dom(R);\left(\overline{(R;S^{\smile}) + (R;S^{\smile})}\right) =$

$$\Big(\ \textit{distribuyendo ; sobre +}\ \Big)$$

$$= Dom(R);\Big(\overline{R;\big(\overline{S^{\smile}} + S^{\smile}\big)}\Big) =$$

$$= Dom(R);\big(\overline{R;\infty}\big) =$$

$$\Big(\ \textit{por Prop. B.19}\ \Big)$$

$$= Dom(R);\big(\overline{Dom(R);\infty}\big) =$$

$$\Big(\ \textit{por Prop. B.20}\ \Big)$$

$$= \underbrace{Dom(R);\big(\overline{Dom(R)\bullet 1}\big)}_{=\ 0};\infty =$$

$$= 0$$

Por lo tanto la proposición vale. ■

Proposición 4.1.4 *[Prop. 3.2.5] Sean* P, Q *y* R *relaciones cualesquiera. Se cumple que*

$$\begin{aligned}
&Dom(P+Q);((P+Q)\rightarrow R) \ = \\
&Dom(P);\Big((P\rightarrow R)\bullet\Big(Dom(Q);(Q\rightarrow R) + \Big(\overline{Dom(Q)}\bullet 1\Big);\infty\Big)\Big) \\
&+ \\
&Dom(Q);\Big((Q\rightarrow R)\bullet\Big(Dom(P);(P\rightarrow R) + \Big(\overline{Dom(P)}\bullet 1\Big);\infty\Big)\Big)
\end{aligned}$$

Dem. $Dom(P+Q);((P+Q)\rightarrow R) =$

$$\left(\begin{array}{l}\textit{por Prop. B.15-2 y}\\ \textit{distribuyendo ; sobre +}\end{array}\right)$$

$$= Dom(P);((P+Q)\rightarrow R) + Dom(Q);((P+Q)\rightarrow R) =$$

$$\Big(\ \textit{por Prop. 4.1.1}\ \Big)$$

$$= Dom(P);((P\rightarrow R)\bullet(Q\rightarrow R)) + Dom(Q);((P\rightarrow R)\bullet(Q\rightarrow R))$$

Pero, dado que $1 = Dom(P)+\Big(\overline{Dom(P)}\bullet 1\Big) = Dom(Q)+\Big(\overline{Dom(Q)}\bullet 1\Big)$, la última expresión es igual a

$$\begin{aligned}
&Dom(P);\Big((P\rightarrow R)\bullet\Big[Dom(Q)+\Big(\overline{Dom(Q)}\bullet 1\Big);(Q\rightarrow R)\Big]\Big) + \quad = \\
&Dom(Q);\Big(\Big[Dom(P)+\Big(\overline{Dom(P)}\bullet 1\Big);(P\rightarrow R)\Big]\bullet(Q\rightarrow R)\Big)
\end{aligned}$$

$$\left(\begin{array}{l}\textit{distribuyendo ; sobre + y}\\ \textit{por Prop. 4.1.2}\end{array}\right)$$

$$\begin{aligned}
= \quad & Dom(P);\Big((P\rightarrow R)\bullet\Big(Dom(Q);(Q\rightarrow R) + \Big(\overline{Dom(Q)}\bullet 1\Big);\infty\Big)\Big) + \\
& Dom(Q);\Big((Q\rightarrow R)\bullet\Big(Dom(P);(P\rightarrow R) + \Big(\overline{Dom(P)}\bullet 1\Big);\infty\Big)\Big)
\end{aligned}$$

■

Proposición 4.1.5 *[Prop. 3.2.6] Sea* A *una relación funcional. Entonces* $Dom(A);(A \rightarrow P) = A;P^{\smile}$.

Dem. Si A es una relación total, la prueba se puede encontrar en [SS93] p. 57. En el caso de que A sea parcial, demostraremos ambas desigualdades para probar la proposición.

$\succeq$) Esta desigualdad es consecuencia de [SS93] p. 57.

$\preceq$) Sea la relación A' definida por $A' = \left(\overline{Dom(A)} \bullet 1\right) + A$.

La relación A' es la totalización funcional de la relación A. Entonces

$\overline{A';\overline{P^{\smile}}} =$

(*aplicando unfold sobre definición de* A' *y distribuyendo* ; *sobre* $+$)

$= \overline{\left(\overline{Dom(A)} \bullet 1\right);\overline{P^{\smile}} + A;\overline{P^{\smile}}} \succeq$

$\succeq \overline{\left(\overline{Dom(A)} \bullet 1\right);\infty + A;\overline{P^{\smile}}} =$

(*por Ley de DeMorgan*)

$= \left(\overline{\left(\overline{Dom(A)} \bullet 1\right);\infty}\right) \bullet \left(\overline{A;\overline{P^{\smile}}}\right) =$

(*por Prop. B.20*)

$= (Dom(A);\infty) \bullet \left(\overline{A;\overline{P^{\smile}}}\right) =$

(*por Prop. B.21*)

$= \left(Dom(A) \nabla \overline{A;\overline{P^{\smile}}}\right);\rho =$

(*por Prop. B.5-2 y por Prop. B.13-2*)

$= Dom(A);\overline{A;\overline{P^{\smile}}} =$

(*aplicando fold sobre Def. 3.2.1*)

$= Dom(A);(A \rightarrow P)$

Por lo tanto,

$Dom(A);(A \rightarrow P) \preceq Dom(A);\overline{A';\overline{P^{\smile}}} =$

(*por Props. B.15 y B.17*)

$= Dom(A);Dom(A');\overline{A';\overline{P^{\smile}}} =$

(*por [SS93] p. 57, pues* A' *es total*)

$= Dom(A);A';\overline{\overline{P^{\smile}}} =$

(*por definición de* A', *y por ser* $^{-}$ *es una convolución*).

$= A;P^{\smile}$

■

Proposición 4.1.6 *[Prop. 3.2.7] Sea* B *una relación funcional. Entonces se cumple que*

$$Dom(B);(B;P\rightarrow Q) = B;(P\rightarrow Q)$$

Dem. Para el caso en que B es una relación total, la prueba se puede encontrar en [SS93] p. 57. Si B es parcial, deben demostrarse ambas desigualdades.

Que $Dom(B);(B;P\rightarrow Q) \succeq B;(P\rightarrow Q)$ se cumple es consecuencia de [SS93] p. 57. Para la segunda desigualdad, sea B' definida por $B' = \left(\overline{Dom(B)}\bullet 1\right)+B$. Entonces

$\overline{\overline{B';P;\overline{Q^{\smile}}}} =$

(*aplicando unfold sobre definición de* B' *y distribuyendo* ; *sobre* +)

$= \overline{\overline{\left(\overline{Dom(B)}\bullet 1\right);P;\overline{Q^{\smile}}} + B;P;\overline{Q^{\smile}}} \succeq$

$\succeq \overline{\overline{\left(\overline{Dom(B)}\bullet 1\right);\infty} + B;P;\overline{Q^{\smile}}} =$

(*por Ley de DeMorgan y por Prop. B.20*)

$= (Dom(B);\infty)\bullet\left(\overline{\overline{B;P;\overline{Q^{\smile}}}}\right) =$

(*por Prop. B.21*)

$= \left(Dom(B)\nabla\overline{\overline{B;P;\overline{Q^{\smile}}}}\right);\rho =$

(*por Props. B.5 y B.13-2*)

$= Dom(B);\overline{\overline{B;P;\overline{Q^{\smile}}}} =$

(*aplicando fold sobre Def. 3.2.1*)

$= Dom(B);((B;P)\rightarrow Q)$

Para terminar, de la derivación anterior obtenemos

$Dom(B);((B;P)\rightarrow Q) \preceq Dom(B);\left(\overline{\overline{B';P;\overline{Q^{\smile}}}}\right) =$

(*por Prop. B.17*)

$= Dom(B);Dom(B');\left(\overline{\overline{B';P;\overline{Q^{\smile}}}}\right) =$

(*por [SS93] p. 57, pues* B' *es total*)

$= Dom(B);B';\overline{\overline{P;\overline{Q^{\smile}}}} =$

(*por definición de* B')

$= B;\overline{\overline{P;\overline{Q^{\smile}}}} =$

(*aplicando fold sobre Def. 3.2.1*)

$= B;(P\rightarrow Q)$

■

Proposición 4.1.7 *[Prop. 3.2.8] Sean* A *y* B *relaciones funcionales y sean* $P = A + (B;P)$ *y* $T = P \rightarrow Q$, *cumpliendo que*

- $Dom(P) = Dom(A) + Dom(B)$, *y*
- $Dom(A) \bullet Dom(B) = 0$.

Entonces $Dom(P);T = A;Q^{\smile} + B;T$.

Dem. Se demuestra utilizando las Props. 4.1.4 y 4.1.6. ■

Proposición 4.1.8 *[Prop. 3.2.9] Sean* A, B *y* C *relaciones funcionales, y sean* $T = (P \rightarrow Q)$ *y* $P = A + (B;P) + (C;P)$ *cumpliendo que*

- $Dom(P) = Dom(A) + Dom(B) + Dom(C)$,
- $Dom(A) \bullet Dom(B) = 0$,
- $Dom(A) \bullet Dom(C) = 0$, *y*
- $Dom(B) \bullet Dom(C) = 0$.

Entonces $Dom(P);T = A;Q^{\smile} + B;T + C;T$.

Dem. Se obtiene como consecuencia de la Prop. 4.1.7. ■

4.2 Propiedades usadas en la derivación de MAC

Todas las propiedades y lemas de esta sección utilizan las relaciones y abreviaturas introducidas en la Secc. 3.3.

4.2.1 Propiedades de los árboles

Las propiedades de los árboles que involucran a las operaciones básicas del tipo se asumen conocidas, por lo que no se ofrece demostración de ellas.

Proposición 4.2.1 $1_{Arbol} = 1_{Nil} + 1_{T_1} + 1_{T>1}$ ■

Proposición 4.2.2 $1_{Nil};\mathsf{raíz} = 1_{Nil};\mathsf{der} = 1_{Nil};\mathsf{izq} = 0$ ■

Proposición 4.2.3 $1_{T_1};\mathsf{der} = 1_{T_1};\mathsf{izq} = 0$ ■

Proposición 4.2.4 $1_{T_1};\mathsf{en} = 1_{T_1};\mathsf{raíz}$ ■

Proposición 4.2.5 $Dom(\mathsf{der}) + Dom(\mathsf{izq}) = 1_{T>1}$ ■

Proposición 4.2.6 $\mathsf{en} = \sqsupseteq;\mathsf{raíz}$ ■

Proposición 4.2.7 $\mathsf{Anc} \preceq \pi;\mathsf{en}$ ■

Proposición 4.2.8 *Un árbol binario T, no tiene elementos repetidos, si, y sólo si se cumple que:*

1. $\mathsf{raíz} \bullet (\mathsf{der};\mathsf{en}) = 0$
2. $\mathsf{raíz} \bullet (\mathsf{izq};\mathsf{en}) = 0$
3. $(\mathsf{der};\mathsf{en}) \bullet (\mathsf{izq};\mathsf{en}) = 0$ ■

4.2.2 Pasos de Derivación

Estas propiedades son utilizadas en la derivación de la Secc. 3.3.

Proposición 4.2.9 *Se cumple que* $(1_{Nil} \otimes 1);\mathsf{Anc} = 0$

Dem. Se realiza unfold de las especificaciones de Anc y de $\sqsupseteq$, y luego se aplican propiedades de los árboles.

$(1_{Nil} \otimes 1);\mathsf{Anc} =$

$$\left(\textit{aplicando unfold sobre Ecu. 3.3} \right)$$

$$= \begin{pmatrix} 1_{Nil} \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \sqsupseteq \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \pi;\mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1);\breve{2} \end{pmatrix};\pi =$$

$$\left(\textit{por Prop. B.9} \right)$$

$$= \begin{pmatrix} 1_{Nil};\sqsupseteq \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \pi;\mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1);\breve{2} \end{pmatrix};\pi =$$

$$\left(\textit{aplicando unfold sobre Ecu. 3.4} \right)$$

$$= \begin{pmatrix} 1_{Nil};(1_{Arbol} + (\mathsf{der} + \mathsf{izq});\sqsupseteq) \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \pi;\mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1);\breve{2} \end{pmatrix};\pi =$$

$$\left(\begin{array}{l} \textit{distribuyendo ; sobre + y} \\ \textit{por Prop. 4.2.2} \end{array} \right)$$

$$= \begin{pmatrix} 1_{Nil} \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \pi;\mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1);\breve{2} \end{pmatrix};\pi =$$

$$\left(\begin{array}{l} (1_{Nil} \otimes 1) \textit{ es funcional y} \\ \textit{por Prop. B.11} \end{array} \right)$$

$$= \begin{pmatrix} (1_{Nil} \otimes 1);\pi;\mathsf{raíz} \\ \nabla \\ (1_{Nil} \otimes 1);(\mathsf{en} \otimes 1);\breve{2} \end{pmatrix};\pi =$$

$$\left(\begin{array}{l} \textit{por Prop. B.6-1} \\ \textit{Dom}(\rho;1) = \textit{Dom}(\rho) = \textit{Dom}(\pi) \textit{ y} \\ \textit{por Prop. B.13-1} \end{array} \right)$$

$$= \left(\begin{array}{c} \pi ; 1_{Nil} ; \text{raíz} \\ \nabla \\ (1_{Nil} \otimes 1) ; (\text{en} \otimes 1) ; \breve{2} \end{array} \right) ; \pi =$$

(*por Prop. 4.2.2*)

$$= 0$$

■

Proposición 4.2.10 *Se cumple que* $(1_{T_1} \otimes 1) ; \text{Anc} = ((1_{T_1} ; \text{raíz}) \otimes 1) ; \breve{2}$

Dem. Como primer paso realizamos unfold de las especificaciones de Anc y de $\sqsupseteq$, y luego aplicamos propiedades de los árboles.

$(1_{T_1} \otimes 1) ; \text{Anc} =$

(*aplicando unfold sobre Ecu. 3.3*)

$$= \left(\begin{array}{c} 1_{T_1} \\ \otimes \\ 1 \end{array} \right) ; \left(\begin{array}{c} \sqsupseteq \\ \otimes \\ 1 \end{array} \right) ; \left(\begin{array}{c} \pi ; \text{raíz} \\ \nabla \\ (\text{en} \otimes 1) ; \breve{2} \end{array} \right) ; \pi =$$

(*por Prop. B.9*)

$$= \left(\begin{array}{c} 1_{T_1} ; \sqsupseteq \\ \otimes \\ 1 \end{array} \right) ; \left(\begin{array}{c} \pi ; \text{raíz} \\ \nabla \\ (\text{en} \otimes 1) ; \breve{2} \end{array} \right) ; \pi =$$

(*aplicando unfold sobre Ecu. 3.4*)

$$= \left(\begin{array}{c} 1_{T_1} ; (1_{Arbol} + (\text{der} + \text{izq}) ; \sqsupseteq) \\ \otimes \\ 1 \end{array} \right) ; \left(\begin{array}{c} \pi ; \text{raíz} \\ \nabla \\ (\text{en} \otimes 1) ; \breve{2} \end{array} \right) ; \pi =$$

(*distribuyendo ; sobre +*)

$$= \left(\begin{array}{c} 1_{T_1} ; 1_{Arbol} + ((1_{T_1} ; \text{der}) + (1_{T_1} ; \text{izq})) ; \sqsupseteq \\ \otimes \\ 1 \end{array} \right) ; \left(\begin{array}{c} \pi ; \text{raíz} \\ \nabla \\ (\text{en} \otimes 1) ; \breve{2} \end{array} \right) ; \pi =$$

(*por Prop. 4.2.3*)

$$= \left(\begin{array}{c} 1_{T_1} ; 1_{Arbol} \\ \otimes \\ 1 \end{array} \right) ; \left(\begin{array}{c} \pi ; \text{raíz} \\ \nabla \\ (\text{en} \otimes 1) ; \breve{2} \end{array} \right) ; \pi =$$

(*por Props. B.17 y B.11*)

$$= \left(\begin{array}{c} (1_{T_1} \otimes 1) ; \pi ; \text{raíz} \\ \nabla \\ (1_{T_1} \otimes 1) ; (\text{en} \otimes 1) ; \breve{2} \end{array} \right) ; \pi =$$

$$\left(\begin{array}{l} \textit{por Prop. B.6-1 y} \\ Dom(\rho;1) = Dom(\rho) = Dom(\pi) \end{array} \right)$$

$$= \left(\begin{array}{c} \pi ; 1_{T_1} ; \mathsf{raíz} \\ \nabla \\ (1_{T_1} \otimes 1) ; (\mathsf{en} \otimes 1) ; \breve{2} \end{array} \right) ; \pi =$$

$$\left(\textit{por Prop. B.9} \right)$$

$$= \left(\begin{array}{c} \pi ; 1_{T_1} ; \mathsf{raíz} \\ \nabla \\ ((1_{T_1} ; \mathsf{en}) \otimes 1) ; \breve{2} \end{array} \right) ; \pi =$$

$$\left(\textit{por Prop. 4.2.4} \right)$$

$$= \left(\begin{array}{c} \pi ; 1_{T_1} ; \mathsf{raíz} \\ \nabla \\ ((1_{T_1} ; \mathsf{raíz}) \otimes 1) ; \breve{2} \end{array} \right) ; \pi =$$

$$\left(\textit{por Prop. B.5-1} \right)$$

$$= \left(\begin{array}{c} ((1_{T_1} ; \mathsf{raíz}) \otimes 1) ; \pi \\ \nabla \\ ((1_{T_1} ; \mathsf{raíz}) \otimes 1) ; \breve{2} \end{array} \right) ; \pi =$$

$$\left(\textit{por Prop. B.11} \right)$$

$$= \left(\begin{array}{c} 1_{T_1} ; \mathsf{raíz} \\ \otimes \\ 1 \end{array} \right) ; \left(\begin{array}{c} \pi \\ \nabla \\ \breve{2} \end{array} \right) ; \pi =$$

$$\left(\textit{por Prop. B.5-1 y } \breve{2} \textit{ es funcional} \right)$$

$$= \left(\begin{array}{c} 1_{T_1} ; \mathsf{raíz} \\ \otimes \\ 1 \end{array} \right) ; \breve{2} ; 2 ; \pi =$$

$$\left(\textit{por Prop. B.5-1} \right)$$

$$= \left(\begin{array}{c} 1_{T_1} ; \mathsf{raíz} \\ \otimes \\ 1 \end{array} \right) ; \breve{2}$$

■

Proposición 4.2.11 *Se cumple que*

$$\left(\begin{array}{c} 1_{T>1} \\ \otimes \\ 1 \end{array} \right) ; \mathsf{Anc} = \left(\begin{array}{c} \pi ; 1_{T>1} ; \mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1) ; \breve{2} \end{array} \right) ; \pi + \left(\begin{array}{c} \mathsf{der} + \mathsf{izq} \\ \otimes \\ 1 \end{array} \right) ; \mathsf{Anc}$$

Dem. Como primer paso realizamos unfold de las especificaciones de Anc y de $\sqsupseteq$, y luego de algunas manipulaciones algebraicas, realizamos nuevamente fold de Anc, obteniendo la ecuación recursiva buscada.

$(\,1_{T>1} \otimes 1\,)\,;\mathsf{Anc} =$

$$\left(\textit{aplicando unfold sobre Ecu. 3.3} \right)$$

$$= \begin{pmatrix} 1_{T>1} \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \sqsupseteq \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \pi\,;\mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1)\,;\,\breve{2} \end{pmatrix};\pi =$$

$$\left(\textit{aplicando unfold sobre Ecu. 3.4} \right)$$

$$= \begin{pmatrix} 1_{T>1} \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} 1_{Arbol} + (\mathsf{der}+\mathsf{izq})\,;\sqsupseteq \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \pi\,;\mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1)\,;\,\breve{2} \end{pmatrix};\pi =$$

$$\left(\textit{distribuyendo } \otimes \textit{ sobre } + \textit{ y } ; \textit{ sobre } + \right)$$

$$= \begin{pmatrix} 1_{T>1} \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} 1_{Arbol} \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \pi\,;\mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1)\,;\,\breve{2} \end{pmatrix};\pi \;+$$
$$\begin{pmatrix} 1_{T>1} \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} (\mathsf{der}+\mathsf{izq})\,;\sqsupseteq \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \pi\,;\mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1)\,;\,\breve{2} \end{pmatrix};\pi =$$

$$\left(\textit{por Prop. B.9} \right)$$

$$= \begin{pmatrix} 1_{T>1}\,;1_{Arbol} \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \pi\,;\mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1)\,;\,\breve{2} \end{pmatrix};\pi \;+$$
$$\begin{pmatrix} 1_{T>1}\,;(\mathsf{der}+\mathsf{izq}) \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \sqsupseteq \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \pi\,;\mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1)\,;\,\breve{2} \end{pmatrix};\pi =$$

$$\left(\textit{aplicando fold sobre } \mathsf{Anc} \right)$$

$$= \begin{pmatrix} 1_{T>1} \\ \otimes \\ 1 \end{pmatrix};\begin{pmatrix} \pi\,;\mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1)\,;\,\breve{2} \end{pmatrix};\pi \;+\; \begin{pmatrix} 1_{T>1}\,;(\mathsf{der}+\mathsf{izq}) \\ \otimes \\ 1 \end{pmatrix};\mathsf{Anc} =$$

$$\left(\begin{array}{l} \textit{por Props. B.12 y B.5-1 y} \\ \textit{por Prop. 4.2.5} \end{array} \right)$$

$$= \begin{pmatrix} \pi\,;1_{T>1}\,;\mathsf{raíz} \\ \nabla \\ (\mathsf{en} \otimes 1)\,;\,\breve{2} \end{pmatrix};\pi \;+\; \begin{pmatrix} (\mathsf{der}+\mathsf{izq}) \\ \otimes \\ 1 \end{pmatrix};\mathsf{Anc}$$

■

Proposición 4.2.12 *Se cumple que* AC =

$$\Big(1_{T_1}\,;\mathsf{raíz}\otimes\breve{2}\Big)\,;\,\breve{2}$$
$$+\ (1_{T>1}\otimes 1)\,;\underbrace{\Big(Dom\Big((\mathsf{en}\otimes\pi)\,;\,\breve{2}\Big)\ \bullet\ Dom\Big((\mathsf{en}\otimes\rho)\,;\,\breve{2}\Big)\Big)}_{\phi}\,;\pi\,;\mathsf{raíz}$$
$$+\ ((\mathsf{der}\otimes 1)+(\mathsf{izq}\otimes 1))\,;\mathsf{AC}$$

Dem. Primero realizamos unfold de la Ecu. 3.6 en la Ecu. 3.2, luego algunas manipulacione algebraicas y, finalmente realizamos fold de AC.

$$\mathsf{AC} =$$

$$\Big(\ \textit{aplicando unfold sobre Ecu. 3.2}\ \Big)$$

$$= ((\pi\,\nabla\,\rho\,;\pi)\,;\mathsf{Anc})\ \bullet\ ((\pi\,\nabla\,\rho\,;\rho)\,;\mathsf{Anc}) =$$

$$\Big(\ \textit{por Prop. B.2}\ \Big)$$

$$= ((1\otimes\pi)\,;\mathsf{Anc})\ \bullet\ ((1\otimes\rho)\,;\mathsf{Anc}) =$$

$$\Big(\ \textit{aplicando unfold sobre Ecu. 3.6}\ \Big)$$

$$= ((1\otimes\pi)\,;\mathsf{Anc})\ \bullet\ \left((1\otimes\rho)\,;\left(\underbrace{\begin{pmatrix}1_{T_1}\,;\mathsf{raíz}\\ \otimes\\ 1\end{pmatrix}\,;\breve{2}}_{\mathsf{AC}_1}\ +\ \underbrace{\begin{pmatrix}\pi\,;1_{T>1}\,;\mathsf{raíz}\\ \nabla\\ (\mathsf{en}\otimes 1)\,;\,\breve{2}\end{pmatrix}\,;\pi}_{\mathsf{AC}_2}\ +\ \underbrace{\begin{pmatrix}\mathsf{der}+\mathsf{izq}\\ \otimes\\ 1\end{pmatrix}\,;\mathsf{Anc}}_{\mathsf{AC}_3}\right)\right) =$$

$$\Big(\ \textit{distribuyendo}\ ;\ \textit{sobre}\ +\ \Big)$$

$$= ((1\otimes\pi)\,;\mathsf{AC}_1\ +\ (1\otimes\pi)\,;\mathsf{AC}_2\ +\ (1\otimes\pi)\,;\mathsf{AC}_3)\ \bullet$$
$$((1\otimes\rho)\,;\mathsf{AC}_1\ +\ (1\otimes\rho)\,;\mathsf{AC}_2\ +\ (1\otimes\rho)\,;\mathsf{AC}_3) =$$

$$\left(\begin{array}{l}\textit{distribuyendo}\ \bullet\ \textit{sobre}\ +\ \textit{y}\\ \textit{anulando sumandos iguales}\end{array}\right)$$

$$= \underbrace{((1\otimes\pi)\,;\mathsf{AC}_1)\ \bullet\ ((1\otimes\rho)\,;\mathsf{AC}_1)}_{\mathsf{AC}_{11}}\ +\ \underbrace{((1\otimes\pi)\,;\mathsf{AC}_1)\ \bullet\ ((1\otimes\rho)\,;\mathsf{AC}_2)}_{\mathsf{AC}_{12}}\ +$$
$$\underbrace{((1\otimes\pi)\,;\mathsf{AC}_1)\ \bullet\ ((1\otimes\rho)\,;\mathsf{AC}_3)}_{\mathsf{AC}_{13}}\ +\ \underbrace{((1\otimes\pi)\,;\mathsf{AC}_2)\ \bullet\ ((1\otimes\rho)\,;\mathsf{AC}_2)}_{\mathsf{AC}_{22}}\ +$$
$$\underbrace{((1\otimes\pi)\,;\mathsf{AC}_2)\ \bullet\ ((1\otimes\rho)\,;\mathsf{AC}_3)}_{\mathsf{AC}_{23}}\ +\ \underbrace{((1\otimes\pi)\,;\mathsf{AC}_3)\ \bullet\ ((1\otimes\rho)\,;\mathsf{AC}_3)}_{\mathsf{AC}_{33}}$$

Procederemos independientemente con cada uno de los subtérminos.

Los subtérminos AC_{12}, AC_{13}, AC_{23} se demuestran iguales a 0.

Para demostrar que $\mathsf{AC}_{12} = 0$, primero utilizamos las Props. B.6, B.12 y B.9 sobre el segundo intersecando, y luego la Prop. B.14 para obtener $((1_{T_1}\bullet 1_{T>1})\otimes 1)$ como antecedente de la intersección; como este término es igual a 0, queda demostrado.

De manera similar se procede con AC_{13}, utilizando las Props. B.13 y B.9 antes de la Prop. B.14.

Para demostrar que $\mathsf{AC}_{23} = 0$, se realiza el unfold de Anc, y mediante la Prop. B.9 se obtiene (der ; en) en el segundo intersecando, que luego, al intersecarse con raíz, da 0 por Prop. 4.2.8.

Para el subtérmino AC_{11} tenemos que:

$((1 \otimes \pi); AC_1) \bullet ((1 \otimes \rho); AC_1) =$

(*aplicando unfold sobre definición de* AC_1)

$$= \left((1 \otimes \pi); \left(\left(1_{T_1}; \text{raíz}\right) \otimes 1\right); \breve{2}\right) \bullet \left((1 \otimes \rho); \left(\left(1_{T_1}; \text{raíz}\right) \otimes 1\right); \breve{2}\right) =$$

(*por Prop. B.9*)

$$= \left(\left(\left(1_{T_1}; \text{raíz}\right) \otimes \pi\right); \breve{2}\right) \bullet \left(\left(\left(1_{T_1}; \text{raíz}\right) \otimes \rho\right); \breve{2}\right) =$$

($\breve{2}$ *es inyectiva y subdistributividad de* ; *sobre* •)

$$= \left(\begin{pmatrix} 1_{T_1}; \text{raíz} \\ \otimes \\ \pi \end{pmatrix} \bullet \begin{pmatrix} 1_{T_1}; \text{raíz} \\ \otimes \\ \rho \end{pmatrix}\right); \breve{2} =$$

(*por Prop. B.10*)

$$= \begin{pmatrix} 1_{T_1}; \text{raíz} \\ \otimes \\ \pi \bullet \rho \end{pmatrix}; \breve{2} =$$

(*por Prop. B.3*)

$$= \begin{pmatrix} 1_{T_1}; \text{raíz} \\ \otimes \\ \breve{2} \end{pmatrix}; \breve{2}$$

El subtérmino AC_{22} se procesa de la siguiente manera:

$((1 \otimes \pi); AC_2) \bullet ((1 \otimes \rho); AC_2) =$

(*aplicando unfold sobre definición de* AC_2)

$$= \left((1 \otimes \pi); \begin{pmatrix} \pi; 1_{T>1}; \text{raíz} \\ \nabla \\ (\text{en} \otimes 1); \breve{2} \end{pmatrix}; \pi\right) \bullet \left((1 \otimes \rho); \begin{pmatrix} \pi; 1_{T>1}; \text{raíz} \\ \nabla \\ (\text{en} \otimes 1); \breve{2} \end{pmatrix}; \pi\right) =$$

(*por Prop. B.11*)

$$= \left(\begin{pmatrix} (1 \otimes \pi); \pi; 1_{T>1}; \text{raíz} \\ \nabla \\ (1 \otimes \pi); (\text{en} \otimes 1); \breve{2} \end{pmatrix}; \pi\right) \bullet \left(\begin{pmatrix} (1 \otimes \rho); \pi; 1_{T>1}; \text{raíz} \\ \nabla \\ (1 \otimes \rho); (\text{en} \otimes 1); \breve{2} \end{pmatrix}; \pi\right) =$$

(*por Props. B.6 y B.9*)

$$= \left(\begin{pmatrix} \pi; 1_{T>1}; \text{raíz} \\ \nabla \\ (\text{en} \otimes \pi); \breve{2} \end{pmatrix}; \pi\right) \bullet \left(\begin{pmatrix} \pi; 1_{T>1}; \text{raíz} \\ \nabla \\ (\text{en} \otimes \rho); \breve{2} \end{pmatrix}; \pi\right) =$$

(*por Prop. B.5-1*)

$$= \left(Dom\left((\mathsf{en}\otimes\pi);\breve{2}\right);\pi;1_{T>1};\mathsf{raíz}\right)\bullet\left(Dom\left((\mathsf{en}\otimes\rho);\breve{2}\right);\pi;1_{T>1};\mathsf{raíz}\right) =$$

$$\left(\text{por Prop. B.5-1}\right)$$

$$= \left(Dom\left((\mathsf{en}\otimes\pi);\breve{2}\right);(1_{T>1}\otimes 1);\pi;\mathsf{raíz}\right)\bullet\left(Dom\left((\mathsf{en}\otimes\rho);\breve{2}\right);(1_{T>1}\otimes 1);\pi;\mathsf{raíz}\right) =$$

$$\left(\text{por Prop. B.17, dos veces}\right)$$

$$= \left((1_{T>1}\otimes 1);Dom\left((\mathsf{en}\otimes\pi);\breve{2}\right);\pi;\mathsf{raíz}\right)\bullet\left((1_{T>1}\otimes 1);Dom\left((\mathsf{en}\otimes\rho);\breve{2}\right);\pi;\mathsf{raíz}\right) =$$

$$\left(\text{por Prop. B.14}\right)$$

$$= (1_{T>1}\otimes 1);\left(\left(Dom\left((\mathsf{en}\otimes\pi);\breve{2}\right);\pi;\mathsf{raíz}\right)\bullet\left(Dom\left((\mathsf{en}\otimes\rho);\breve{2}\right);\pi;\mathsf{raíz}\right)\right) =$$

$$\left(\text{por Prop. B.14}\right)$$

$$= (1_{T>1}\otimes 1);\left(Dom\left((\mathsf{en}\otimes\pi);\breve{2}\right)\bullet Dom\left((\mathsf{en}\otimes\rho);\breve{2}\right)\right);\pi;\mathsf{raíz}$$

Por último, para AC_{33}:

$$((1\otimes\pi);\mathsf{AC}_3)\bullet((1\otimes\rho);\mathsf{AC}_3) =$$

$$\left(\text{aplicando unfold sobre definición de } \mathsf{AC}_3\right)$$

$$= \left(\begin{pmatrix}1\\\otimes\\\pi\end{pmatrix};\begin{pmatrix}\mathsf{der}+\mathsf{izq}\\\otimes\\1\end{pmatrix};\mathsf{Anc}\right)\bullet\left(\begin{pmatrix}1\\\otimes\\\rho\end{pmatrix};\begin{pmatrix}\mathsf{der}+\mathsf{izq}\\\otimes\\1\end{pmatrix};\mathsf{Anc}\right) =$$

$$\left(\text{por Prop. B.9}\right)$$

$$= \left(\begin{pmatrix}\mathsf{der}+\mathsf{izq}\\\otimes\\1\end{pmatrix};\begin{pmatrix}1\\\otimes\\\pi\end{pmatrix};\mathsf{Anc}\right)\bullet\left(\begin{pmatrix}\mathsf{der}+\mathsf{izq}\\\otimes\\1\end{pmatrix};\begin{pmatrix}1\\\otimes\\\rho\end{pmatrix};\mathsf{Anc}\right) =$$

$$\left(\text{por Prop. 4.2.8}\right)$$

$$= \begin{pmatrix}\mathsf{der}+\mathsf{izq}\\\otimes\\1\end{pmatrix};(((1\otimes\pi);\mathsf{Anc})\bullet((1\otimes\rho);\mathsf{Anc})) =$$

$$\left(\text{distribuyendo } \otimes \text{ sobre } + \text{ y aplicando fold sobre Ecu. 3.2}\right)$$

$$= ((\mathsf{der}\otimes 1)+(\mathsf{izq}\otimes 1));\mathsf{AC}$$

Juntando los resultados de cada subtérmino, obtenemos el resultado buscado. ■

Proposición 4.2.13 *Se cumple que*

$$((\pi\nabla\mathsf{A})+(\pi\nabla\mathsf{B})+(\pi\nabla(\mathsf{C};\mathsf{AC})))\bullet((\mathsf{A}+\mathsf{B}+\mathsf{C};\mathsf{AC})\rightarrow\mathsf{Anc})$$

$$=$$

$$(\pi\nabla\mathsf{A})\bullet(\mathsf{A};\mathsf{Anc}^{\smile})$$

$$+$$

$$((\pi\nabla\mathsf{B})+(\pi\nabla(\mathsf{C};\mathsf{AC})))\bullet((\mathsf{B}+\mathsf{C};\mathsf{AC})\rightarrow\mathsf{Anc})$$

Dem. Para la demostración se utilizan primero la Prop. 3.2.2, y después las Props. 3.2.4 y 3.2.6.

$$((\pi \nabla A) + (\pi \nabla B) + (\pi \nabla (C;AC))) \bullet ((A + B + C;AC) \rightarrow Anc) =$$

$$\left(\textit{por Prop. 3.2.2} \right)$$

$$= ((\pi \nabla A) + (\pi \nabla B) + (\pi \nabla (C;AC))) \bullet (A \rightarrow Anc) \bullet ((B + C;AC) \rightarrow Anc) =$$

$$\left(\begin{array}{l} \textit{distribuyendo } \bullet \textit{ sobre } + \textit{ y} \\ \textit{por Prop. B.13-1} \end{array} \right)$$

$$\begin{aligned} = \; & ((Dom(A);(\pi \nabla A)) \bullet (A \rightarrow Anc) \bullet ((B + C;AC) \rightarrow Anc)) + \\ & ((Dom(B);((\pi \nabla B) + (\pi \nabla (C;AC)))) \bullet (A \rightarrow Anc) \bullet ((B + C;AC) \rightarrow Anc)) = \end{aligned}$$

$$\left(\textit{por Props. B.14 y B.13-1} \right)$$

$$\begin{aligned} = \; & ((\pi \nabla A) \bullet (Dom(A);(A \rightarrow Anc)) \bullet (Dom(A);((B + C;AC) \rightarrow Anc))) + \\ & (((\pi \nabla B) + (\pi \nabla (C;AC))) \bullet (Dom(B);(A \rightarrow Anc)) \bullet (((B + C;AC) \rightarrow Anc))) = \end{aligned}$$

$$\left(\textit{por Props. 3.2.4 y 3.2.6} \right)$$

$$\begin{aligned} = \; & ((\pi \nabla A) \bullet (A;Anc^{\smile}) \bullet (Dom(A);\infty)) + \\ & (((\pi \nabla B) + (\pi \nabla (C;AC))) \bullet (Dom(B);\infty) \bullet (((B + C;AC) \rightarrow Anc))) = \end{aligned}$$

$$\left(\textit{para todo } R,\ R \preceq \infty \right)$$

$$= ((\pi \nabla A) \bullet (A;Anc^{\smile})) + (((\pi \nabla B) + (\pi \nabla (C;AC))) \bullet (((B + C;AC) \rightarrow Anc)))$$

■

Proposición 4.2.14 *Se cumple que*

$$\begin{aligned} & ((\pi \nabla B) + (\pi \nabla (C;AC))) \bullet ((B + C;AC) \rightarrow Anc) \\ = \; & \\ & (\pi \nabla B) \bullet (B;Anc^{\smile}) \bullet (Dom(C_1;AC);C_1;(AC \rightarrow Anc)) \; + \\ & (\pi \nabla B) \bullet (B;Anc^{\smile}) \bullet (Dom(C_2;AC);C_2;(AC \rightarrow Anc)) \; + \\ & (\pi \nabla B) \bullet (B;Anc^{\smile}) \bullet \left(Dom(B);\left(\overline{Dom(C;AC)} \bullet 1\right);\infty\right) \; + \\ & (\pi \nabla (C;AC)) \bullet (B;Anc^{\smile}) \bullet (Dom(C_1;AC);C_1;(AC \rightarrow Anc)) \; + \\ & (\pi \nabla (C;AC)) \bullet (B;Anc^{\smile}) \bullet (Dom(C_2;AC);C_2;(AC \rightarrow Anc)) \; + \\ & (\pi \nabla (C;AC)) \bullet (B;Anc^{\smile}) \bullet \left(Dom(B);\left(\overline{Dom(C;AC)} \bullet 1\right);\infty\right) \end{aligned}$$

Dem. El primer paso de la demostración consiste en utilizar la Prop. 3.2.5 para descomponer el segundo intersecando; se utilizan las Props. B.13 y B.14 para anteponer el dominio correcto a dicho subtérmino.

$$Dom(B + C;AC);((B + C;AC) \rightarrow Anc) =$$

$$\left(\textit{por Prop. 3.2.5} \right)$$

$$\begin{aligned} = \; & Dom(B);\left[(B \rightarrow Anc) \bullet \left(Dom(C;AC);(C;AC \rightarrow Anc) + \left(\overline{Dom(C;AC)} \bullet 1\right);\infty\right)\right] + \\ & Dom(C;AC);\left[(C;AC \rightarrow Anc) \bullet \left(Dom(B);(B \rightarrow Anc) + \left(\overline{Dom(B)} \bullet 1\right);\infty\right)\right] = \end{aligned}$$

$$\left(\begin{array}{l} \textit{por Prop. B.14 y} \\ \textit{distribuyendo ; sobre +} \end{array}\right)$$

$$\begin{aligned} = \; & [(Dom(B);(B \rightarrow Anc)) \bullet \\ & \left(Dom(B);Dom(C;AC);(C;AC \rightarrow Anc) + Dom(B);\left(\overline{Dom(C;AC)} \bullet 1\right);\infty\right)] + \\ & [(Dom(C;AC);(C;AC \rightarrow Anc)) \bullet \\ & \left(Dom(C;AC);Dom(B);(B \rightarrow Anc) + Dom(C;AC);\left(\overline{Dom(B)} \bullet 1\right);\infty\right)] = \end{aligned}$$

$$\left(\begin{array}{l} \textit{por Prop. 3.2.6 y} \\ \textit{por ser } Dom(C;AC) \preceq Dom(B) \end{array}\right)$$

$$\begin{aligned} = \; & \left[(B;Anc^{\smile}) \bullet \left(Dom(C;AC);(C;AC \rightarrow Anc) + Dom(B);\left(\overline{Dom(C;AC)} \bullet 1\right);\infty\right)\right] + \\ & [(Dom(C;AC);(C;AC \rightarrow Anc)) \bullet (Dom(C;AC);B;Anc^{\smile})] = \end{aligned}$$

$$\left(\begin{array}{l} \textit{distribuyendo } \bullet \textit{ sobre +} \end{array}\right)$$

$$\begin{aligned} = \; & (B;Anc^{\smile}) \bullet (Dom(C;AC);(C;AC \rightarrow Anc)) + \\ & (B;Anc^{\smile}) \bullet \left(Dom(B);\left(\overline{Dom(C;AC)} \bullet 1\right);\infty\right) + \\ & (Dom(C;AC);(C;AC \rightarrow Anc)) \bullet (Dom(C;AC);B;Anc^{\smile}) = \end{aligned}$$

$$\left(\begin{array}{l} \textit{por Prop. B.14 y} \\ \textit{eliminando sumandos iguales y} \\ \textit{distribuyendo } \bullet \textit{ sobre +} \end{array}\right)$$

$$= (B;Anc^{\smile}) \bullet \left[Dom(C;AC);(C;AC \rightarrow Anc) + Dom(B);\left(\overline{Dom(C;AC)} \bullet 1\right);\infty\right] =$$

$$\left(\begin{array}{l} \textit{aplicando unfold sobre C y} \\ \textit{por Prop. B.15-2} \end{array}\right)$$

$$\begin{aligned} = \; & (B;Anc^{\smile}) \bullet \\ & [(Dom(C_1;AC) + Dom(C_2;AC));((C_1;AC + C_2;AC) \rightarrow Anc) + \\ & Dom(B);\left(\overline{Dom(C;AC)} \bullet 1\right);\infty] = \end{aligned}$$

$$\left(\begin{array}{l} \textit{por Prop. 3.2.2} \end{array}\right)$$

$$\begin{aligned} = \; & (B;Anc^{\smile}) \bullet \\ & [(Dom(C_1;AC) + Dom(C_2;AC));((C_1;AC \rightarrow Anc) \bullet (C_2;AC \rightarrow Anc)) + \\ & Dom(B);\left(\overline{Dom(C;AC)} \bullet 1\right);\infty] = \end{aligned}$$

$$\left(\begin{array}{l} \textit{distribuyendo ; sobre + y} \\ \textit{por Prop. B.14} \end{array}\right)$$

$$\begin{aligned} = \; & (B;Anc^{\smile}) \bullet \\ & [(Dom(C_1;AC);(C_1;AC \rightarrow Anc) \bullet Dom(C_1;AC);(C_2;AC \rightarrow Anc)) + \\ & (Dom(C_2;AC);(C_1;AC \rightarrow Anc) \bullet Dom(C_2;AC);(C_2;AC \rightarrow Anc)) + \\ & Dom(B);\left(\overline{Dom(C;AC)} \bullet 1\right);\infty] = \end{aligned}$$

$$\left(\begin{array}{l} Dom(C_1;AC) \bullet Dom(C_2;AC) = 0 \textit{ y} \\ \textit{por Prop. 3.2.4 y propiedades de } \bullet \end{array}\right)$$

$$\begin{aligned} = \; & (B;Anc^{\smile}) \bullet \\ & [Dom(C_1;AC);(C_1;AC \rightarrow Anc) + Dom(C_2;AC);(C_2;AC \rightarrow Anc) + \\ & Dom(B);\left(\overline{Dom(C;AC)} \bullet 1\right);\infty] = \end{aligned}$$

$$\left(\begin{array}{l} Dom(C_i;AC) \preceq Dom(C_i) \ y \\ por\ Prop.\ 3.2.7 \end{array} \right)$$

$$= (B;Anc^{\smile}) \bullet$$
$$[Dom(C_1;AC);C_1;(AC \rightarrow Anc) + Dom(C_2;AC);C_2;(AC \rightarrow Anc) +$$
$$Dom(B);\left(\overline{Dom(C;AC)} \bullet 1\right);\infty]$$

Finalmente, distribuyendo $\bullet$ sobre $+$ se obtiene el enunciado. ■

Proposición 4.2.15 *Se cumple que*

1. $(\pi \nabla A) \preceq (A;Anc^{\smile})$
2. $(\pi \nabla B) \preceq (B;Anc^{\smile})$
3. $(\pi \nabla(C;AC)) \preceq (B;Anc^{\smile})$

Dem. Para demostrar el primero de los ítems, demostraremos la siguiente proposición, que es equivalente: $A^{\smile};(\pi \nabla A) \preceq (A^{\smile};A;Anc^{\smile})$

$A^{\smile};(\pi \nabla A) =$

$$\left(\textit{por Prop. B.11} \right)$$

$$= \left(\begin{array}{c} A^{\smile};\pi \\ \nabla \\ A^{\smile};A \end{array} \right) =$$

$$\left(\begin{array}{l} \textit{aplicando unfold sobre } A^{\smile} \ y \\ \textit{por Prop. B.7} \end{array} \right)$$

$$= \left(\begin{array}{c} \left(\begin{array}{c} raíz^{\smile};1_{T_1} \\ \nabla \\ 2 \\ \nabla \\ A^{\smile};A \end{array} \right);\pi \end{array} \right) =$$

$$\left(\textit{por Prop. B.5-1} \right)$$

$$= \left(\begin{array}{c} raíz^{\smile};1_{T_1} \\ \nabla \\ A^{\smile};A \end{array} \right) =$$

$$\left(\begin{array}{l} Dom(raíz^{\smile}) = Ran(A) \ y \\ A \ es\ funcional \end{array} \right)$$

$$= \left(\begin{array}{c} A^{\smile};A;raíz^{\smile};1_{T_1} \\ \nabla \\ A^{\smile};A \end{array} \right) =$$

$$\left(\textit{por Prop. B.11} \right)$$

$$= A^{\smile}; A; \begin{pmatrix} \text{raíz}^{\smile}; l_{T_1} \\ \nabla \\ l \end{pmatrix} \preceq$$

(*por conversa de la Ecu. 3.6*)

$$\preceq A^{\smile}; A; \text{Anc}^{\smile}$$

Hemos demostrado el primer ítem.

Para demostrar el segundo ítem, primero escribiremos $(B; \text{Anc}^{\smile})$ como $(B_1 + B_2)$, y luego demostraremos que $(\pi \nabla B) \preceq B_1$.

$(B; \text{Anc}^{\smile}) =$

(*aplicando unfold sobre Ecu. 3.6 y distribuyendo $^{\smile}$ sobre +*)

$$= B; \Bigg(\underbrace{\left(\begin{pmatrix} \pi; l_{T>1}; \text{raíz} \\ \nabla \\ (\text{en} \otimes l); \breve{2} \end{pmatrix}; \pi \right)^{\smile}}_{\text{Anc}_1} + \underbrace{\left(\left((l_{T_1}; \text{raíz}) \otimes l \right); \breve{2} + \begin{pmatrix} \text{der} + \text{izq} \\ \otimes \\ l \end{pmatrix}; \text{Anc} \right)^{\smile}}_{\text{Anc}_2} \Bigg) =$$

(*distribuyendo ; sobre +*)

$$= \underbrace{B; \text{Anc}_1^{\smile}}_{B_1} + \underbrace{B; \text{Anc}_2^{\smile}}_{B_2} =$$

(*aplicando fold sobre las abreviaturas definidas*)

$$= B_1 + B_2$$

A continuación, transformamos $(\pi \nabla B)$:

$(\pi \nabla B) =$

(*aplicando unfold sobre B*)

$$= \begin{pmatrix} \pi \\ \nabla \\ (l_{T>1} \otimes l); \phi; \pi; \text{raíz} \end{pmatrix} =$$

(*por Prop. B.12*)

$$= (l_{T>1} \otimes l); \phi; \begin{pmatrix} \pi \\ \nabla \\ \pi; \text{raíz} \end{pmatrix} =$$

(*π es funcional y por Prop. B.11*)

$$= (1_{T>1} \otimes 1); \phi; \pi; \begin{pmatrix} 1 \\ \nabla \\ \text{raíz} \end{pmatrix} =$$

$$\left(\textit{propagando } (1_{T>1} \otimes 1) \textit{ hasta el } \nabla \right)$$

$$= (1_{T>1} \otimes 1); \phi; \pi; \begin{pmatrix} 1_{T>1} \\ \nabla \\ \text{raíz} \end{pmatrix}$$

Por otro lado, B_1 se transforma en:

$B_1 =$

$$\left(\begin{array}{l} \textit{aplicando unfold sobre } B_1 \textit{ y } B \textit{ y} \\ \textit{distribuyendo } \breve{} \textit{ sobre ;} \end{array} \right)$$

$$= (1_{T>1} \otimes 1); \phi; \pi; \text{raíz}; \breve{\pi}; \begin{pmatrix} \pi; 1_{T>1}; \text{raíz} \\ \nabla \\ (\text{en} \otimes 1); \breve{2} \end{pmatrix}\breve{} =$$

$$\left(\begin{array}{l} \textit{aplicando unfold sobre Def. 2.2.12-1 y} \\ \textit{por Def. 2.2.9-3} \end{array} \right)$$

$$= (1_{T>1} \otimes 1); \phi; \pi; \text{raíz}; \left((\pi; 1_{T>1}; \text{raíz})\breve{} \bullet \infty; \left((\text{en} \otimes 1); \breve{2} \right)\breve{} \right) =$$

$$\left(\begin{array}{l} \textit{distribuyendo } \breve{} \textit{ sobre ; y } \breve{} \textit{ sobre } \otimes \textit{ y} \\ \text{raíz } \textit{es funcional y} \\ \textit{subdistributividad de ; sobre } \bullet \end{array} \right)$$

$$= (1_{T>1} \otimes 1); \phi; \pi; \left(\text{raíz}; \text{raíz}\breve{}; 1_{T>1}; \breve{\pi} \bullet \text{raíz}; \infty; 2; (\text{en}\breve{} \otimes 1) \right) \succeq$$

$$\left(\begin{array}{l} 1_{T>1} \preceq \text{raíz}; \text{raíz}\breve{} \textit{ y} \\ 1 \preceq \infty \textit{ y} \\ \textit{aplicando unfold sobre Def. 2.2.12-4 y} \\ \textit{por Prop. B.7} \end{array} \right)$$

$$\succeq (1_{T>1} \otimes 1); \phi; \pi; \left(1_{T>1}; \breve{\pi} \bullet \text{raíz}; (\text{en}\breve{} \nabla 1) \right) =$$

$$\left(\begin{array}{l} \textit{aplicando unfold sobre Def. 2.2.12-1 y} \\ \textit{por Prop. B.11} \end{array} \right)$$

$$= (1_{T>1} \otimes 1); \phi; \pi; \left(\begin{pmatrix} 1_{T>1} \\ \nabla \\ 1_{T>1}; \infty \end{pmatrix} \bullet \begin{pmatrix} \text{raíz}; \text{en}\breve{} \\ \nabla \\ \text{raíz} \end{pmatrix} \right) =$$

$$\left(\begin{array}{l} \textit{por Prop. B.8 y} \\ \text{raíz} \preceq 1_{T>1}; \infty \textit{ y} \\ 1_{T>1} \preceq \text{raíz}; \text{en}\breve{} \end{array} \right)$$

$$= (1_{T>1} \otimes 1); \phi; \pi; \begin{pmatrix} 1_{T>1} \\ \nabla \\ \text{raíz} \end{pmatrix}$$

El último término es igual a la forma obtenida para $(\pi \nabla B)$, y por lo tanto hemos demostrado el segundo ítem.

Para el tercer ítem, distribuímos ; y ∇ sobre + en la ecuación $(\pi \nabla(C;AC))$ obteniendo

$$\left(\left(\begin{array}{c}\pi\\ \nabla\\ C_1;AC\end{array}\right)+\left(\begin{array}{c}\pi\\ \nabla\\ C_2;AC\end{array}\right)\right)$$

Demostraremos, ahora, que $(\pi \nabla C_1;AC) \preceq (B;Anc^{\smile})$. Para el otro sumando la demostración es análoga.

$(\pi \nabla(C_1;AC)) =$

(*aplicando unfold sobre* C_1)

$$= \left(\begin{array}{c}\pi\\ \nabla\\ (1_{T>1} \otimes 1);\left(\begin{array}{c}\mathsf{der}\\ \otimes\\ 1\end{array}\right);AC\end{array}\right) =$$

(*por Prop. B.12*)

$$= (1_{T>1} \otimes 1);\left(\begin{array}{c}\pi\\ \nabla\\ \left(\begin{array}{c}\mathsf{der}\\ \otimes\\ 1\end{array}\right);AC\end{array}\right) \preceq$$

(*por Prop. B.13 y por Prop. 4.2.7*)

$$\preceq (1_{T>1} \otimes 1);\left(\begin{array}{c}\pi\\ \nabla\\ \left(\begin{array}{c}\mathsf{der}\\ \otimes\\ 1\end{array}\right);\phi;\pi;\mathsf{en}\end{array}\right) =$$

($(\mathsf{der} \otimes 1);\phi \preceq \phi$)

$$\preceq (1_{T>1} \otimes 1);\left(\begin{array}{c}\pi\\ \nabla\\ \phi;\pi;\mathsf{en}\end{array}\right) =$$

(*por Prop. B.12*)

$$= (1_{T>1} \otimes 1);\phi;\left(\begin{array}{c}\pi\\ \nabla\\ \pi;\mathsf{en}\end{array}\right) =$$

(*por Prop. B.11*)

$$= (1_{T>1} \otimes 1);\phi;\pi;\begin{pmatrix} 1 \\ \nabla \\ \mathsf{en} \end{pmatrix} =$$

$$\left(\textit{propagando } (1_{T>1} \otimes 1) \textit{ hasta el } \nabla \right)$$

$$= (1_{T>1} \otimes 1);\phi;\pi;\begin{pmatrix} 1_{T>1} \\ \nabla \\ \mathsf{en} \end{pmatrix}$$

Pero, en la demostración del segundo ítem, vimos que $(\mathsf{B};\mathsf{Anc}^{\smile}) \succeq$

$$\left(\textit{definition of } \preceq \right)$$

$$\succeq \mathsf{B}_1 =$$

$$\left(\textit{aplicando unfold sobre } \mathsf{B}_1 \right)$$

$$= (1_{T>1} \otimes 1);\phi;\pi;\left(\mathsf{raíz};\mathsf{raíz}^{\smile};1_{T>1};\overline{\pi} \;\bullet\; \mathsf{raíz};\infty;2;(\mathsf{en}^{\smile} \otimes 1)\right) \succeq$$

$$\left(\begin{array}{l} 1_{T>1} \preceq \mathsf{raíz};\mathsf{raíz}^{\smile} \textit{ y} \\ \mathsf{raíz}^{\smile};\infty \preceq \infty \end{array} \right)$$

$$\succeq (1_{T>1} \otimes 1);\phi;\pi;\left(1_{T>1};\overline{\pi} \;\bullet\; \mathsf{raíz};\mathsf{raíz}^{\smile};\infty;2;(\mathsf{en}^{\smile} \otimes 1)\right) \succeq$$

$$\left(\begin{array}{l} \textit{aplicando unfold sobre Def. 2.2.12-1 y por Prop. B.12 y} \\ 1_{T>1} \preceq \mathsf{raíz};\mathsf{raíz}^{\smile} \textit{ y} \\ \textit{aplicando unfold sobre Def. 2.2.12-4 y por Prop. B.7} \end{array} \right)$$

$$\succeq (1_{T>1} \otimes 1);\phi;\pi;((1_{T>1} \nabla \infty) \;\bullet\; 1_{T>1};\infty;(\mathsf{en}^{\smile} \nabla 1)) \succeq$$

$$\left(\mathsf{en} \preceq \infty \right)$$

$$\succeq (1_{T>1} \otimes 1);\phi;\pi;((1_{T>1} \nabla \mathsf{en}) \;\bullet\; 1_{T>1};\infty;(\mathsf{en}^{\smile} \nabla 1)) \succeq$$

$$\left(\begin{array}{l} \mathsf{X};(\mathsf{en}^{\smile} \nabla 1) \preceq \overline{(1 \nabla \mathsf{en})} \textit{ sii (por Schröder) } (1 \nabla \mathsf{en});((\mathsf{en}^{\smile} \nabla 1))^{\smile} \preceq \overline{\mathsf{X}} \textit{ sii} \\ \mathsf{X} \preceq \overline{(1 \nabla \mathsf{en});((\mathsf{en}^{\smile} \nabla 1))^{\smile}} \\ \textit{y tomando } \mathsf{X} = \overline{(1 \nabla \mathsf{en});((\mathsf{en}^{\smile} \nabla 1))^{\smile}} \textit{ y reemplázandolo por } \infty \end{array} \right)$$

$$= (1_{T>1} \otimes 1);\phi;\pi;\begin{pmatrix} 1_{T>1} \\ \nabla \\ \mathsf{en} \end{pmatrix}$$

y por lo tanto, $(\pi \nabla \mathsf{C}_1;\mathsf{AC}) \preceq (\mathsf{B};\mathsf{Anc}^{\smile})$.

Hemos demostrado los tres ítems, y por lo tanto, la proposición.

■

Proposición 4.2.16 *Se cumple que*

1. $(\pi \nabla \mathsf{B}) \bullet (Dom(\mathsf{C}_1;\mathsf{AC});\mathsf{C}_1;(\mathsf{AC} \rightarrow \mathsf{Anc})) = 0$
2. $(\pi \nabla \mathsf{B}) \bullet (Dom(\mathsf{C}_2;\mathsf{AC});\mathsf{C}_2;(\mathsf{AC} \rightarrow \mathsf{Anc})) = 0$
3. $(\pi \nabla(\mathsf{C};\mathsf{AC})) \bullet \left(Dom(\mathsf{B});\left(\overline{Dom(\mathsf{C};\mathsf{AC})} \bullet 1\right);\infty\right) = 0$

Dem. En primer lugar demostraremos los dos primeros ítems, cuya demostración es idéntica. Para ello llamamos D_i a $(Dom(C_i;AC);C_i;(AC \rightarrow Anc))$, con i=1,2, fijo durante toda la demostración, y l_i a der para i=1, izq para i=2, y procedemos como sigue:

$(\pi \nabla B) \bullet D_i =$

$$\left(\begin{array}{l} \textit{aplicando unfold sobre } B \textit{ y } D_i \textit{ y} \\ \textit{por Prop. B.14} \end{array} \right)$$

$= Dom(C_i;AC);(\pi;(1 \nabla raíz) \bullet C_i;(AC \rightarrow Anc)) \preceq$

$$\left(\begin{array}{l} \textit{por Prop. 3.2.3 y} \\ 1_X;R \preceq R \end{array} \right)$$

$\preceq (\pi;(1 \nabla raíz)) \bullet (C_i;AC;Anc^{\smile}) \preceq$

$$\left(\begin{array}{l} \textit{aplicando unfold sobre Ecu. 3.2 y} \\ \textit{eliminando uno de los intersecandos} \end{array} \right)$$

$\preceq (\pi;(1 \nabla raíz)) \bullet (C_i;(1 \otimes \pi);Anc;Anc^{\smile}) =$

$$\left(\begin{array}{l} \textit{por Prop. B.11 y} \\ \textit{por Def. 2.2.9-2} \end{array} \right)$$

$= (\pi;\breve{\pi}) \bullet (\pi;raíz;\breve{\rho}) \bullet (C_i;(1 \otimes \pi);Anc;Anc^{\smile}) \preceq$

$$\left(\begin{array}{l} \textit{por Prop. B.4-1 y} \\ \textit{por Prop. 4.2.7} \end{array} \right)$$

$\preceq (1 \otimes \infty) \bullet (\pi;raíz;\breve{\rho}) \bullet (C_i;(1 \otimes \pi);\pi;en;Anc^{\smile}) =$

$$\left(\begin{array}{l} \textit{aplicando unfold sobre definición de } C_i \textit{ y} \\ \textit{por Props. B.9 y B.6-1} \end{array} \right)$$

$= (1 \otimes \infty) \bullet (\pi;raíz;\breve{\rho}) \bullet (\pi;l_i;en;Anc^{\smile}) =$

$$\left(Dom(Anc);\rho \preceq Anc \right)$$

$= (1 \otimes \infty) \bullet (\pi;raíz;\breve{\rho}) \bullet (\pi;l_i;en;\breve{\rho};Dom(Anc)) =$

$$\left(\textit{por Prop. 4.2.8} \right)$$

$= 0$

De esta manera, quedan demostrados los dos primeros ítems.

Para demostrar el tercer ítem, vemos primero que

$$Dom((\pi \nabla(C;AC))) = Dom(C;AC)$$

y que

$$Dom\left(\left(Dom(B);\left(\overline{Dom(C;AC)} \bullet 1\right);\infty\right)\right) \preceq$$
$$Dom\left(\left(\overline{Dom(C;AC)} \bullet 1\right);\infty\right) =$$
$$\overline{Dom(C;AC)} \bullet 1$$

Por lo tanto, la intersección de los dominios es 0, y eso implica la validez del tercer ítem. ∎

Proposición 4.2.17 *Se cumple que*

$$(\pi \nabla \mathsf{B}) \bullet \left(Dom(\mathsf{B}); \left(\overline{Dom(\mathsf{C};\mathsf{AC})} \bullet 1\right); \infty\right) =$$
$$Dom\begin{pmatrix} \overline{Dom((\mathsf{der} \otimes 1);\mathsf{AC})} \bullet 1 \\ \nabla \\ \overline{Dom((\mathsf{izq} \otimes 1);\mathsf{AC})} \bullet 1 \end{pmatrix}; \phi; \pi; \begin{pmatrix} 1 \\ \nabla \\ \mathsf{raíz} \end{pmatrix}$$

Dem. La demostración se compondrá de tres partes: primero transformaremos cada uno de los dos intersecandos, y luego procederemos a combinarlos.

La transformación de $(\pi \nabla \mathsf{B})$ es sencilla, ya que sólo consiste en utilizar las Props. B.11 y B.12 para obtener:

$$(\pi \nabla \mathsf{B}) = \phi; \pi; (1 \nabla \mathsf{raíz})$$

Para el segundo subtérmino, hacemos:

$\left(Dom(\mathsf{B}); \left(\overline{Dom(\mathsf{C};\mathsf{AC})} \bullet 1\right); \infty\right) =$

$\left(\text{pues } Dom(\mathsf{B}) = \phi\right)$

$= \phi; \left(\overline{Dom(\mathsf{C};\mathsf{AC})} \bullet 1\right); \infty =$

$\left(\text{por Prop. B.17}\right)$

$= \left(\phi \bullet \overline{Dom(\mathsf{C};\mathsf{AC})} \bullet 1\right); \infty =$

$\left(\begin{array}{l}\text{por Prop. B.18 y} \\ \text{por ser } \phi \preceq 1\end{array}\right)$

$= \left(\phi \bullet \overline{((\mathsf{C};\mathsf{AC};\infty) \bullet 1)}\right); \infty =$

$\left(\begin{array}{l}\text{distribuyendo ; sobre + y • sobre + y} \\ \text{por Ley de DeMorgan}\end{array}\right)$

$= \left(\phi \bullet \overline{(\mathsf{C}_1;\mathsf{AC};\infty \bullet 1)} \bullet \overline{(\mathsf{C}_2;\mathsf{AC};\infty \bullet 1)}\right); \infty =$

$\left(\begin{array}{l}\text{asociando y conmutando • y} \\ \text{por Prop. B.19}\end{array}\right)$

$= \left(\left(\phi \bullet \overline{(Dom(\mathsf{C}_1;\mathsf{AC});\infty \bullet 1)}\right) \bullet \left(\phi \bullet \overline{(Dom(\mathsf{C}_2;\mathsf{AC});\infty \bullet 1)}\right)\right); \infty =$

$\left(\begin{array}{l}\text{por Ley de DeMorgan y} \\ \phi \preceq 1 \text{ luego de distribuir}\end{array}\right)$

$= \left(\left(\phi \bullet \overline{Dom(\mathsf{C}_1;\mathsf{AC});\infty}\right) \bullet \left(\phi \bullet \overline{Dom(\mathsf{C}_2;\mathsf{AC});\infty}\right)\right); \infty =$

$\left(\text{por Prop. B.20}\right)$

$= \left(\left(\phi \bullet \left(\left(\overline{Dom(\mathsf{C}_1;\mathsf{AC})} \bullet 1\right); \infty\right)\right) \bullet \left(\phi \bullet \left(\left(\overline{Dom(\mathsf{C}_2;\mathsf{AC})} \bullet 1\right); \infty\right)\right)\right); \infty =$

$\left(\text{por Prop. B.21}\right)$

$$= \left(\left(\begin{array}{c} \left(\overline{Dom\,(C_1;AC)} \bullet 1\right) \\ \nabla \\ \phi \end{array} \right); \rho \bullet \left(\begin{array}{c} \left(\overline{Dom\,(C_2;AC)} \bullet 1\right) \\ \nabla \\ \phi \end{array} \right); \rho \right); \infty =$$

$$\left(\begin{array}{l} \textit{por Prop. B.5-2 y} \\ \textit{subdistributividad de ; sobre } \bullet \end{array} \right)$$

$$= \left(Dom\left(\overline{Dom\,(C_1;AC)} \bullet 1\right) \bullet Dom\left(\overline{Dom\,(C_2;AC)} \bullet 1\right)\right); \phi; \infty =$$

$$\left(\textit{por Prop. B.16} \right)$$

$$= \underbrace{Dom \left(\begin{array}{c} \overline{Dom\,(C_1;AC)} \bullet 1 \\ \nabla \\ \overline{Dom\,(C_2;AC)} \bullet 1 \end{array} \right); \phi; \infty}_{\varphi}$$

Para combinar los dos intersecandos utilizamos la Prop. B.22.

Primero vemos que $\varphi; \phi; \pi; (1 \nabla \text{raíz}) \preceq (\phi; \pi; (1 \nabla \text{raíz})) \bullet (\varphi; \phi; \infty)$, lo cual se cumple pues $\varphi; \phi \preceq \phi$ y $\pi; (1 \nabla \text{raíz}) \preceq \infty$; además, el dominio del primer término es igual al del segundo, pues ambos son iguales a $\varphi; \phi$. Como el segundo término es funcional, entonces, por la proposición mencionada, ambos son iguales, concluyendo la demostración. ■

Lema 4.2.18 *Se cumple que*

$$((\pi \nabla AC); \rho) \bullet (((AC \rightarrow Anc) \bullet (1 \otimes \infty)); \rho) = ((\pi \nabla AC) \bullet ((AC \rightarrow Anc) \bullet (1 \otimes \infty))); \rho$$

Dem. Para demostrar la igualdad, demostraremos las dos desigualdades ($\preceq$ y $\succeq$).

La desigualdad $\succeq$ es consecuencia de la monotonía de la composición relacional (;).

Para demostrar la desigualdad $\preceq$, primero utilizamos la Prop. B.1,

$((\pi \nabla AC); \rho) \bullet (((AC \rightarrow Anc) \bullet (1 \otimes \infty)); \rho) \preceq$

$$\left(\textit{por Prop. B.1} \right)$$

$$\preceq \left((\pi \nabla AC) \bullet \left(((AC \rightarrow Anc) \bullet (1 \otimes \infty)); \rho; \breve{\rho}\right)\right); (\rho \bullet ((\pi \nabla AC)^{\smile}; ((AC \rightarrow Anc) \bullet (1 \otimes \infty)); \rho)) \preceq$$

$$\left(\textit{propiedades de } \bullet \right)$$

$$\preceq \left((\pi \nabla AC) \bullet \left(((AC \rightarrow Anc) \bullet (1 \otimes \infty)); \rho; \breve{\rho}\right)\right); \rho =$$

$$\left(\textit{por Prop. B.4 -2} \right)$$

$$= ((\pi \nabla AC) \bullet (((AC \rightarrow Anc) \bullet (1 \otimes \infty)); (\infty \otimes 1))); \rho =$$

$$\left(\begin{array}{l} \textit{por Def. 2.2.9-2 y} \\ \textit{por Prop. B.4-1} \end{array} \right)$$

$$= \left((1 \nabla \infty) \bullet \left(AC; \breve{\rho}\right) \bullet (((AC \rightarrow Anc) \bullet (1 \otimes \infty)); (\infty \otimes 1))\right); \rho =$$

$$\left(\textit{propiedades de } \bullet \right)$$

$= \big((1 \nabla \infty) \bullet (1 \nabla \infty) \bullet \big(\mathsf{AC}; \breve{\rho}\big) \bullet (\mathsf{AC} \rightarrow \mathsf{Anc})\big); \rho =$

$\left(\begin{array}{l} \textit{por Prop. B.4-1 y} \\ \textit{por Def. 2.2.9-2} \end{array} \right)$

$= ((\pi \nabla \mathsf{AC}) \bullet (\mathsf{AC} \rightarrow \mathsf{Anc}) \bullet (1 \nabla \infty)); \rho$

■

Lema 4.2.19 *Se cumple que* $\mathsf{MAC} = \mathsf{MAC}^{\circ}; \rho = \mathsf{AC} \bullet (((\mathsf{AC} \rightarrow \mathsf{Anc}) \bullet (1 \otimes \infty)); \rho)$.

Dem. Como primer paso demostraremos que

$$(\pi \nabla \mathsf{AC}) = (\pi \nabla \mathsf{AC}) \bullet (1 \otimes \infty)$$

lo cual implica que $\mathsf{MAC}^{\circ} = (\pi \nabla \mathsf{AC}) \bullet ((\mathsf{AC} \rightarrow \mathsf{Anc}) \bullet (1 \otimes \infty))$.

$(\pi \nabla \mathsf{AC}) =$

$\left(\textit{por Def. 2.2.9-2} \right)$

$= (\pi ; \pi^{\smile}) \bullet (\mathsf{AC}; \rho^{\smile}) =$

$\left(\textit{idempotencia de} \bullet \right)$

$= (\pi ; \pi^{\smile}) \bullet (\pi ; \pi^{\smile}) \bullet (\mathsf{AC}; \rho^{\smile}) =$

$\left(\textit{por Def. 2.2.9-2} \right)$

$= (\pi ; \pi^{\smile}) \bullet (\pi \nabla \mathsf{AC}) =$

$\left(\textit{por Prop. B.4 -1} \right)$

$= (1 \otimes \infty) \bullet (\pi \nabla \mathsf{AC})$

Por otra parte, por Lema 4.2.18, vale que

$$((\pi \nabla \mathsf{AC}) \bullet ((\mathsf{AC} \rightarrow \mathsf{Anc}) \bullet (1 \otimes \infty))); \rho = ((\pi \nabla \mathsf{AC}); \rho) \bullet (((\mathsf{AC} \rightarrow \mathsf{Anc}) \bullet (1 \otimes \infty)); \rho)$$

Finalmente, por Prop. B.5-1,

$$\mathsf{MAC} = \mathsf{MAC}^{\circ}; \rho = \mathsf{AC} \bullet (((\mathsf{AC} \rightarrow \mathsf{Anc}) \bullet (1 \otimes \infty)); \rho)$$

■

Proposición 4.2.20 *Se cumple que*

1. $(\pi \nabla (\mathsf{C}; \mathsf{AC})) \bullet (\mathit{Dom}(\mathsf{C}_1; \mathsf{AC}); \mathsf{C}_1; (\mathsf{AC} \rightarrow \mathsf{Anc})) =$
   $= \mathit{Dom}((\mathsf{der} \otimes 1); \mathsf{AC}); (\pi \nabla ((\mathsf{der} \otimes 1); \mathsf{MAC}))$
2. $(\pi \nabla (\mathsf{C}; \mathsf{AC})) \bullet (\mathit{Dom}(\mathsf{C}_2; \mathsf{AC}); \mathsf{C}_2; (\mathsf{AC} \rightarrow \mathsf{Anc})) =$
   $= \mathit{Dom}((\mathsf{izq} \otimes 1); \mathsf{AC}); (\pi \nabla ((\mathsf{izq} \otimes 1); \mathsf{MAC}))$

Dem. Utilizaremos el nombre D_i para $(Dom(C_i;AC);C_i;(AC \rightarrow Anc))$ (el cual fue introducido en la Prop. 4.2.16), con i=1, para la demostración del primer ítem; para el segundo ítem es análogo, con i=2.

$(\pi \nabla(C;AC)) \bullet D_i =$

$$\left(\textit{por Prop. B.13-1} \right)$$

$$= \left(Dom(C;AC); \begin{pmatrix} \pi \\ \nabla \\ C;AC \end{pmatrix} \right) \bullet D_1 =$$

$$\left(\textit{aplicando unfold sobre definición de } C;AC \right)$$

$$= \left(Dom((C_1+C_2);AC); \begin{pmatrix} \pi \\ \nabla \\ (C_1+C_2);AC \end{pmatrix} \right) \bullet D_1 =$$

$$\left(\begin{array}{l} \textit{distribuyendo } + \textit{ sobre } ; \textit{ y } + \textit{ sobre } \nabla \textit{ y} \\ \textit{propiedades de los dominios} \end{array} \right)$$

$$= \left[Dom(C_1;AC); \begin{pmatrix} \pi \\ \nabla \\ C_1;AC \end{pmatrix} + \underbrace{Dom(C_2;AC); \begin{pmatrix} \pi \\ \nabla \\ C_1;AC \end{pmatrix}}_{=0} + \right.$$

$$\left. \underbrace{Dom(C_1;AC); \begin{pmatrix} \pi \\ \nabla \\ C_2;AC \end{pmatrix}}_{=0} + Dom(C_2;AC); \begin{pmatrix} \pi \\ \nabla \\ C_2;AC \end{pmatrix} \right] \bullet D_1 =$$

$$\left(\begin{array}{l} \textit{eliminando 0 's obtenidos por} \\ Dom(C_1;AC) \bullet Dom(C_2;AC) = 0 \textit{ y} \\ \textit{por Prop. B.16} \end{array} \right)$$

$$= \left[Dom(C_1;AC); \begin{pmatrix} \pi \\ \nabla \\ C_1;AC \end{pmatrix} + Dom(C_2;AC); \begin{pmatrix} \pi \\ \nabla \\ C_2;AC \end{pmatrix} \right] \bullet D_1 =$$

$$\left(\textit{distribuyendo } \bullet \textit{ sobre } + \right)$$

$$= \underbrace{\left(\left(Dom(C_1;AC); \begin{pmatrix} \pi \\ \nabla \\ C_1;AC \end{pmatrix} \right) \bullet D_1 \right) + \left(\left(Dom(C_2;AC); \begin{pmatrix} \pi \\ \nabla \\ C_2;AC \end{pmatrix} \right) \bullet D_1 \right)}_{=0} =$$

$$\left(\begin{array}{l} \textit{eliminando 0 's obtenidos por} \\ Dom(C_1;AC) \bullet Dom(C_2;AC) = 0 \end{array} \right)$$

$$= \left(Dom(C_1;AC); \begin{pmatrix} \pi \\ \nabla \\ C_1;AC \end{pmatrix} \right) \bullet D_1 =$$

(*aplicando unfold sobre* D_1)

$$= \left(Dom(C_1;AC); \begin{pmatrix} \pi \\ \nabla \\ C_1;AC \end{pmatrix} \right) \bullet (Dom(C_1;AC);C_1;(AC \rightarrow Anc)) =$$

($Dom(C_1;AC)$ *es funcional*)

$$= Dom(C_1;AC); \left(\begin{pmatrix} \pi \\ \nabla \\ C_1;AC \end{pmatrix} \bullet (C_1;(AC \rightarrow Anc)) \right) =$$

(*por Def. 2.2.9-2*)

$$= Dom(C_1;AC); \left(\left(\pi ; \breve{\pi} \right) \bullet \left(C_1;AC; \breve{\rho} \right) \bullet (C_1;(AC \rightarrow Anc)) \right) =$$

(*por Prop. B.4-1 y subdistributividad de ; sobre* $\bullet$)

$$= Dom(C_1;AC); \left((I \otimes \infty) \bullet \left(C_1; \left(AC; \breve{\rho} \bullet (AC \rightarrow Anc) \right) \right) \right) =$$

(*propiedades de* $\bullet$)

$$= Dom(C_1;AC); \left((I \otimes \infty) \bullet \left(C_1; \left(\left(AC; \breve{\rho} \right) \bullet (((AC \rightarrow Anc) \bullet (I \otimes \infty)); (\infty \otimes I)) \right) \right) \right) =$$

(*por Prop. B.4 y subdistributividad de ; sobre* $\bullet$)

$$= Dom(C_1;AC); \left(\left(\pi ; \breve{\pi} \right) \bullet \left(C_1; (AC \bullet (((AC \rightarrow Anc) \bullet (I \otimes \infty)); \rho)); \breve{\rho} \right) \right) =$$

(*por Def. 2.2.9-2*)

$$= Dom(C_1;AC); \begin{pmatrix} \pi \\ \nabla \\ C_1;(AC \bullet)(((AC \rightarrow Anc) \bullet (I \otimes \infty)); \rho) \end{pmatrix} =$$

(*por Lema 4.2.19*)

$$= Dom(C_1;AC); \begin{pmatrix} \pi \\ \nabla \\ C_1;MAC \end{pmatrix}$$

■

Capítulo 5

Conclusiones

"Pero videmus nunc per speculum et in aenigmate y la verdad, antes de manifestarse a cara descubierta, se muestra en fragmentos (¡ay, cuán ilegibles!), mezclada con el error de este mundo, de modo que debemos deletrear sus fieles signáculos incluso allí donde nos parecen oscuros y casi forjados por una voluntad totalmente orientada hacia el mal."

Umberto Eco, El Nombre de la Rosa.

La programación transformacional tiene sus raíces en las matemáticas, más particularmente en la aritmética, al punto de que se ha propuesto llamar *algorítmica* a esta técnica; por lo tanto, el uso de un álgebra de programas como base para desarrollarla es una elección que resulta natural. Pero no cualquier álgebra es igualmente adecuada para esta tarea; si bien las álgebras relacionales parecen serlo, su limitado poder expresivo las invalida como candidato. Las Fork Álgebras surgieron como intento de aumentar el poder expresivo de las álgebras relacionales, y han demostrado ser una elección adecuada.

A pesar de que las bases están fundadas, el estado actual del cálculo de programas basado en fork álgebras es sustancialmente primitivo, existiendo varias áreas donde se pueden realizar aportes de importancia para mejorarlo. Esta tesis se enmarca en la tarea de desarrollar, basándose en el marco algebraico de las fork álgebras, una metodología de programación transformacional.

Una de estas áreas es la que consiste en recopilar propiedades de las fork álgebras que permitan armar una "base de datos" de reglas de transformación. El principal aporte de esta tesis es haber enunciado y demostrado un conjunto de reglas que permiten transformar una especificación que involucra cuantificadores universales (en la forma de implicaciones relacionales) en un término algebraico que puede considerarse como un algoritmo. La caracterización de las fórmulas cuantificadas universalmente y las reglas de transformación de éstas en expresiones algebraicas son de gran importancia pues una gran cantidad de problemas se expresan naturalmente en términos de este tipo de fórmulas, y la definición de la implicación relacional no sugiere un algoritmo eficiente

para computarla. El método utilizado para obtener el conjunto de reglas consistió en el estudio de varios casos de problemas conocidos: mínimo de una lista, ocho reinas, clausura transitiva de una relación y mínimo ancestro común, principalmente. Estos problemas fueron especificados utilizando la implicación relacional, y luego se realizaron derivaciones hasta obtener formas algorítmicas de los mismos, extrayendo en el proceso las propiedades de la implicación que permitieron su computación. Con el fin de mostrar el uso de las reglas, fue presentado el ejemplo del mínimo ancestro común, por ser el más completo de ellos.

Otra de estas áreas es la construcción de herramientas que permitan automatizar la tarea de construcción de términos y la definición, demostración y utilización de reglas de transformación. Dichas herramientas se conocen con el nombre de *editores*, y hay varios trabajos destinados a capitalizar el trabajo realizado en otros temas (como la teoría de tipos de Martin-Löf, [NPS89, MN94]) para la construcción de un editor orientado a las fork álgebras.

Otra más de estas áreas es la de expresar tipos de datos relacionalmente. A pesar de que existen varios trabajos que la enfocan, son pocos los que se basan en el marco de las fork álgebras (escencialmente homogéno, o sea sin tipos), y ninguno hasta ahora apunta a la utilización de los tipos por un programador. Es nuestra intención continuar la investigación en este área, proponiendo una sintaxis estándar para tipos de datos, junto con su semántica, y probar su utilidad en ejemplos concretos.

La observación detallada de las demostraciones nos permite observar el uso de heurísticas, o sea patrones de derivación, tales como la técnica de fold-unfold ([Par90]) y el análisis de casos ([HV91]), entre las ya conocidos y difundidas entre los profesionales de la informática, y otras propias de las fork álgebras, como ser la utilización de las propiedades de los tipos de datos expresadas como ecuaciones e inecuaciones, la caracterización de expresiones condicionales en función de ∇, y la transformación de términos con $\bullet$ en términos con ∇, ya que estos últimos sugieren mejor el comportamiento de entrada-salida de un programa, lo que facilita la aplicación de la intuición al razonamiento.

La principal conclusión obtenida es que la *algorítmica* es posible, si bien aún falta mucho camino por recorrer, tanto en la obtención de reglas más poderosas, como en programas que permitan automatizar parte del proceso de construcción.

APÉNDICES

Apéndice A

Definición de las Fork Álgebras

En este apéndice se presentan las definiciones de las álgebras, tal cual fueron presentadas en la Secc. 2.2.

Definición A.1 *[Def. 2.2.1] Un* Campo de Conjuntos *sobre un conjunto no vacío U es una estructura*

$$\mathsf{SF} = < \mathcal{R}, \cup, \cap, ', \emptyset, U >$$

tal que:

- $U \subseteq \mathcal{P}(U)$, *cerrado para todas las operaciones,*
- $\cup$ *es la unión de conjuntos,*
- $\cap$ *es la intersección de conjuntos,*
- $'$ *es el complemento respecto de* U,
- $\emptyset$ *es el conjunto vacío, y*
- U *es llamado conjunto universal.*

Se utilizará el símbolo clásico ($\subseteq$) *para la inclusión de conjuntos.* ■

Definición A.2 *[Def. 2.2.2] Un* Álgebra de Boole *es una estructura*

$$\mathsf{BA} = < A, \vee, \wedge, ', 0, 1 >$$

donde

- $\vee$ *y* $\wedge$ *son operaciones binarias sobre* A *(llamadas* supremo *e* ínfimo, *respectivamente),*
- $'$ *una operación unaria sobre* A *(llamada* complemento*),*
- 0 *y* 1 *dos elementos de* A *(llamados* primer *y* último elemento, *respectivamente),*

y tal que para toda terna a, b, c, de elementos de A:

1. $a \vee a = a \quad y \quad a \wedge a = a$ *(idempotencia)*
2. $a \vee b = b \vee a \quad y \quad a \wedge b = b \wedge a$ *(conmutatividad)*
3. $a \vee (b \vee c) = (a \vee b) \vee c \quad y \quad a \wedge (b \wedge c) = (a \wedge b) \wedge c$ *(asociatividad)*
4. $a \vee (a \wedge b) = a \quad y \quad a \wedge (a \vee b) = a$ *(absorción)*
5. $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \quad y \quad a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ *(distributividad)*
6. $0 \neq 1$
7. $a \vee 0 = a \quad y \quad a \wedge 1 = a$
8. $a \vee a' = 1 \quad y \quad a \wedge a' = 0.$

El símbolo $\leq$ *se utilizará para el orden subyacente, y queda definido por* $a \leq b \leftrightarrow a \vee b = b$ *(o, equivalentemente,* $a \wedge b = a$*).* ■

Definición A.3 *[Def. 2.2.3] Un* Álgebra Relacional Propia *sobre un conjunto* U *es una estructura*

$$\mathsf{PRA} = < \mathcal{R}, \cup, \cap, |, ', \sim, \emptyset, V, \Delta_{Dom(V)} >$$

tal que:

- $\mathcal{R} \subseteq \mathcal{P}(U \times U)$,
- $< \mathcal{R}, \cup, \cap, ', \emptyset, V >$ *es un campo de conjuntos sobre* $\mathcal{P}(U \times U)$,
- $|$ *denota el* producto relacional,
- $\sim$ *denota la operación* conversa *y*
- $\Delta_{Dom(V)}$ *denota la relación* identidad *sobre el dominio (o equivalentemente el rango) de la relación* V.

El símbolo $\subseteq$ *se utiliza para la inclusión usual de relaciones.* ■

Definición A.4 *[Def. 2.2.4] Un* Álgebra Relacional (Abstracta) *es una estructura*

$$\mathsf{ARA} = < A, +, \bullet, ^{-}, ;, ^{\smile}, 0, \infty, 1 >$$

tal que:

1. $< A, +, \bullet, ^{-}, 0, \infty >$ *es un álgebra de Boole, completa y atómica, donde*
   - $+$ *denota la operación supremo,*
   - $\bullet$ *denota la operación ínfimo,*

- $^-$ *denota la operación complemento,*
- 0 *denota el elemento* cero,
- ∞ *denota el elemento* unidad, *y*
- $\subset$ *(llamada* inclusión*) denota el orden subyacente.*

2. $< A, ;, 1 >$ *es un semigrupo, es decir,*
   - $(R;S);T = R;(S;T)$, *y*
   - $R;1 = 1;R = R$.
3. $R;S \subset T \Leftrightarrow R^{\smile};\overline{T} \subset \overline{S} \Leftrightarrow \overline{T};S^{\smile} \subset \overline{R}$, *(regla de Schröder).*
4. $R \neq 0 \Rightarrow \infty;R;\infty = \infty$, *(regla de Tarski)* ■

Definición A.5 *[Def. 2.2.7] Una* $\star$ Fork Álgebra Propia *sobre un conjunto* U *es una estructura*

$$\star \text{PFA} = < \mathcal{R}, U, \cup, \cap, |, \underline{\nabla}, ', ^{\smile}, \emptyset, V, \Delta_{Dom(V)}, \star >$$

tal que:

1. $< \mathcal{R}, \cup, \cap, |, ', ^{\smile}, \emptyset, V, \Delta_{Dom(V)} >$ *es un álgebra relacional propia.*
2. $\star$ *es una operación binaria inyectiva de* V *en* U.
3. $\mathcal{R}$ *es cerrada para la operación* $\underline{\nabla}$ *definida por*

$$R \underline{\nabla} S = \{< x, (y \star z) > / < x, y > \in R,\ < x, z > \in S\ y < y, z > \in V\}$$

■

Definición A.6 *[Def. 2.2.8] Una* Fork Álgebra Propia *sobre un conjunto* U *es el reducto*

$$\text{PFA} = < \mathcal{R}, \cup, \cap, |, \underline{\nabla}, ', ^{\smile}, \emptyset, V, \Delta_{Dom(V)} >$$

de una $\star$ *fork álgebra propia.* ■

Definición A.7 *[Def. 2.2.9] Una* Fork Álgebra (Abstracta) *es una estructura*

$$\text{AFA} = < A, +, \bullet, ;, \nabla, ^-, ^{\smile}, 0, \infty, 1 >$$

(donde la operación ∇ *es llamada* fork*) tal que:*

1. $< A, +, \bullet, ;, ^-, ^{\smile}, 0, \infty, 1 >$ *es un álgebra relacional, con* 0 *su elemento* cero, ∞ *su elemento* unidad *y* $\preceq$ *(llamada* inclusión relacional*) el orden subyacente.*
2. $R \nabla S = (R;(1 \nabla \infty)) \bullet (S;(\infty \nabla 1))$
3. $(R \nabla S);(T \nabla Q)^{\smile} = (R;T^{\smile}) \bullet (S;Q^{\smile})$
4. $(1 \nabla \infty)^{\smile} \nabla (\infty \nabla 1)^{\smile} \preceq 1$ ■

Apéndice B

Propiedades de las Fork Álgebras

En este apéndice se presentan, sin demostración, algunas propiedades válidas en fork álgebras, las cuales sirven como reglas de derivación a la hora de calcular programas formalmente (ver Cap. 2).

Proposición B.1 $((Q;R) \bullet S) = (Q \bullet (S;R^{\smile}));(R \bullet (Q^{\smile};S))$ ■

Proposición B.2 *Se cumple que*

1. $(\pi \nabla \rho;\pi) = (1 \otimes \pi)$
2. $(\pi \nabla \rho;\rho) = (1 \otimes \rho)$ ■

Proposición B.3 $\pi \bullet \rho = \widetilde{2}$ ■

Proposición B.4 *Se cumple que*

1. $\pi;\pi^{\smile} = (1 \otimes \infty)$
2. $\rho;\rho^{\smile} = (\infty \otimes 1)$ ■

Proposición B.5 *Se cumple que*

1. $(R \nabla S);\pi = Dom(S);R$
2. $(R \nabla S);\rho = Dom(R);S$ ■

Proposición B.6 *Se cumple que*

1. $(R \otimes S);\pi = Dom(\rho;S);\pi;R$
2. $(R \otimes S);\rho = Dom(\pi;R);\rho;S$ ■

Proposición B.7 $\begin{pmatrix} R \\ \nabla \\ S \end{pmatrix};\begin{pmatrix} P \\ \otimes \\ Q \end{pmatrix} = \begin{pmatrix} R;P \\ \nabla \\ S;Q \end{pmatrix}$ ■

Proposición B.8 $\begin{pmatrix} R \\ \nabla \\ S \end{pmatrix} \bullet \begin{pmatrix} P \\ \nabla \\ Q \end{pmatrix} = \begin{pmatrix} R \bullet P \\ \nabla \\ S \bullet Q \end{pmatrix}$ ■

Proposición B.9 $\begin{pmatrix} R \\ \otimes \\ S \end{pmatrix} ; \begin{pmatrix} P \\ \otimes \\ Q \end{pmatrix} = \begin{pmatrix} R ; P \\ \otimes \\ S ; Q \end{pmatrix}$ ■

Proposición B.10 $\begin{pmatrix} R \\ \otimes \\ S \end{pmatrix} \bullet \begin{pmatrix} P \\ \otimes \\ Q \end{pmatrix} = \begin{pmatrix} R \bullet P \\ \otimes \\ S \bullet Q \end{pmatrix}$ ■

Proposición B.11 *Sea* A *una relación funcional. Entonces* $A ; (P \nabla Q) = ((A ; P) \nabla (A ; Q))$ ■

Proposición B.12 $((1_X ; P) \nabla (1_Y ; Q)) = (1_X \bullet 1_Y) ; (P \nabla Q)$ ■

Proposición B.13 *Se cumple que:*

1. $Dom(R) ; R = R$
2. $Dom(1_X) = 1_X$ ■

Proposición B.14 $(1_X ; R) \bullet (1_Y ; R) = (1_X \bullet 1_Y) ; R$ ■

Proposición B.15 *Se cumple que:*

1. $Dom(R \bullet S) = Dom(R) \bullet Dom(S)$
2. $Dom(R + S) = Dom(R) + Dom(S)$ ■

Proposición B.16 $Dom(R \nabla S) = Dom(R) \bullet Dom(S)$ ■

Proposición B.17 $1_X ; 1_Y = 1_X \bullet 1_Y$ ■

Proposición B.18 $Dom(R) = (R ; \infty) \bullet 1$ ■

Proposición B.19 $R ; \infty = Dom(R) ; \infty$ ■

Proposición B.20 *Si* $P + Q = 1$ *y* $P \bullet Q = 0$, *entonces* $\overline{P ; \infty} = Q ; \infty$ ■

Proposición B.21 $(R ; \infty) \bullet S = (R \nabla S) ; \rho$ ■

Proposición B.22 *Si* $A \preceq B$, $Dom(A) = Dom(B)$, *y* B *es funcional, entonces* $A = B$ ■

Apéndice C

Propiedades de la Implicación Relacional

En este ápendice se presentan las propiedades de la implicación relacional, que son la principal contribución de esta tesis.

Proposición C.1 *[Prop. 3.2.2] Sean* P, Q *y* R *relaciones cualesquiera. Se cumple que*

1. $((P+Q)\rightarrow R) = (P\rightarrow R) \bullet (Q\rightarrow R)$
2. $(P\rightarrow(Q\bullet R)) = (P\rightarrow Q) \bullet (P\rightarrow R)$

Dem. Ver Prop. 4.1.1. ■

Proposición C.2 *[Prop. 3.2.4] Siendo* P *y* Q *relaciones cualesquiera, se cumple que*

$$\left(\overline{Dom(P)}\bullet 1\right);(P\rightarrow Q) = \left(\overline{Dom(P)}\bullet 1\right);\infty$$

Dem. Ver Prop. 4.1.2. ■

Proposición C.3 *[Prop. 3.2.3] Se cumple que* $Dom(R);(R\rightarrow S) \preceq R;S^{\smile}$.

Dem. Ver Prop. 4.1.3. ■

Proposición C.4 *[Prop. 3.2.5] Sean* P, Q *y* R *relaciones cualesquiera. Se cumple que*

$$\begin{aligned}
&Dom(P+Q);((P+Q)\rightarrow R) = \\
&Dom(P);\left((P\rightarrow R)\bullet\left(Dom(Q);(Q\rightarrow R) + \left(\overline{Dom(Q)}\bullet 1\right);\infty\right)\right) \\
&+ \\
&Dom(Q);\left((Q\rightarrow R)\bullet\left(Dom(P);(P\rightarrow R) + \left(\overline{Dom(P)}\bullet 1\right);\infty\right)\right)
\end{aligned}$$

Dem. Ver Prop. 4.1.4. ■

Proposición C.5 *[Prop. 3.2.6] Sea* A *una relación funcional. Entonces* $Dom(A);(A\rightarrow P) = A;P^{\smile}$.

Dem. Ver Prop. 4.1.5. ■

Proposición C.6 *[Prop. 3.2.7] Sea* B *una relación funcional. Entonces se cumple que*

$$Dom(\mathsf{B});(\mathsf{B};\mathsf{P}\rightarrow\mathsf{Q}) = \mathsf{B};(\mathsf{P}\rightarrow\mathsf{Q})$$

Dem. Ver Prop. 4.1.6. ■

Proposición C.7 *[Prop. 3.2.8] Sean* A *y* B *relaciones funcionales y sean* $\mathsf{P}=\mathsf{A}+(\mathsf{B};\mathsf{P})$ *y* $\mathsf{T}=\mathsf{P}\rightarrow\mathsf{Q}$, *cumpliendo que*

- $Dom(\mathsf{P}) = Dom(\mathsf{A})+Dom(\mathsf{B})$, *y*
- $Dom(\mathsf{A})\bullet Dom(\mathsf{B}) = 0$.

Entonces $Dom(\mathsf{P});\mathsf{T} = \mathsf{A};\mathsf{Q}^{\smile} + \mathsf{B};\mathsf{T}$.

Dem. Ver Prop. 4.1.7. ■

Proposición C.8 *[Prop. 3.2.9] Sean* A, B *y* C *relaciones funcionales, y sean* $\mathsf{T} = (\mathsf{P}\rightarrow\mathsf{Q})$ *y* $\mathsf{P} = \mathsf{A}+(\mathsf{B};\mathsf{P})+(\mathsf{C};\mathsf{P})$ *cumpliendo que*

- $Dom(\mathsf{P}) = Dom(\mathsf{A})+Dom(\mathsf{B})+Dom(\mathsf{C})$,
- $Dom(\mathsf{A})\bullet Dom(\mathsf{B}) = 0$,
- $Dom(\mathsf{A})\bullet Dom(\mathsf{C}) = 0$, *y*
- $Dom(\mathsf{B})\bullet Dom(\mathsf{C}) = 0$.

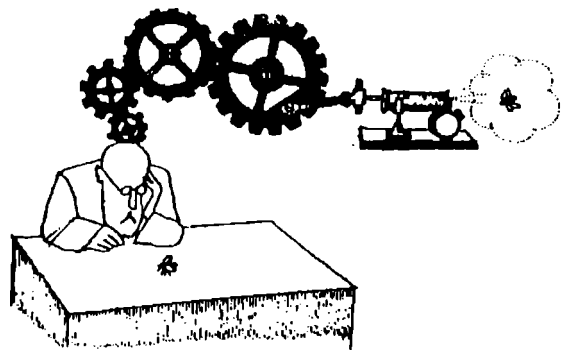
Entonces $Dom(\mathsf{P});\mathsf{T} = \mathsf{A};\mathsf{Q}^{\smile} + \mathsf{B};\mathsf{T} + \mathsf{C};\mathsf{T}$.

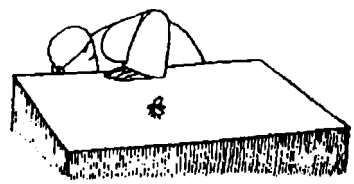
Dem. Ver Prop. 4.1.8. ■

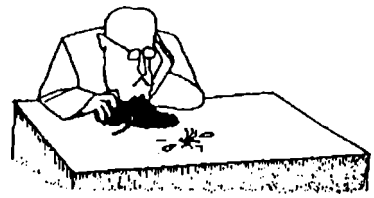
Apéndice D

Preface or...

"The right way to kill flies"







Quino, Hombres de bolsillo

"Look at the picture on the previous page.

Every member of the computer science community knows what specification and formal verification tools (the flies-killing machine) are and how they are used.

Assume that a fly is a sequential program. Usually, even though a system designer knows about those tools, (s)he just thinks a bit about the problem which has to be solved and starts to write the program code from the first line to the last one. It is a fact that the first time the program will not run correctly. Applying intuition and debugging, (s)he obtains a good approximation to the correct operation. Finally, the writing of some goto's in proper places optimizes the execution time of the program. If it is a lucky day, the programmer writes some comments into the program text and makes the minimal indispensable documentation. This process is equivalent to the blow with the shoe.

However, the fly could have flown, and hence the same process should start again.

Now, look at the maintenance. Using the machine, the scientist waits until the fly dies and falls, and so (s)he collects it and throws it away. Using the shoe, the scientist has a harder job: (s)he will need some water, a brush and time to clean the remaining stain.

Sit in the scientist's chair, but consider a lot of flies in front of you. Try to kill them using your shoe. It is possible for you to kill one of them, but the other ones, although in a primitive manner, can communicate among themselves, can realize that something has happened and run away. Obviously, with the shoe in your hand, you will pursue the remaining flies. Finally, you will get a crushed-flies-stained room and the doubt whether some flies have escaped.

The last case is exactly the analogous situation we have to regard in concurrent programming. Moreover, that is why we have to use formal tools as we are going to study in the present work."

Pedro D'Argenio, Tesis de grado ([D'A]).

BIBLIOGRAFÍA

Bibliografía

[BHSV93] Rudolph Berghammer, Armando M. Haeberer, Gunther Schmidt, and Paulo A. S. Veloso. Comparing two different approaches to products in abstract relation algebras. In *Proceedings of the Third International Conference on Algebraic Methodology and Software Technology, AMAST '93*, pages 167–176. Springer-Verlag, 1993.

[BM77] John Bell and Moshé Machover. *A Course in Mathematical Logic*. Elsevier Science Publishers, 1977. Ch.4,Ex.2.2 and Ch.5,§5.

[BW88] Richard Bird and Philip Wadler. *Introduction to Functional Programming*. C. A. R. HOARE. Prentice Hall, 1988.

[CT51] L. H. Chin and Alfred Tarski. Distributive and modular laws in the arithmetic of relation algebras. In *University of California Publications in Mathematics*, pages 341–384. University of California, 1951.

[D'A] Pedro Rubén D'Argenio. A comparative analysis of concurrency theories. Graduation Thesis, UNLP, 1994.

[dBdR72] J. W. de Bakker and W. P. de Roever. A calculus for recursive program schemes. In *Proc. IRIA Symp. on Automata, Formal Languajes and Programming*, pages 167–196, Amsterdam, North Holland, 1972.

[DF88] Edsger W. Dijkstra and W.H.J. Feijen. *A Method of Programming*. Addison-Wesley, 1988.

[FA94] Marcelo Fabián Frias and N. G. Aguayo. Natural specifications vs. abstract specifications. A relational approach. In *Proceedings of SOFSEM '94*, Milovy, Czech Republic, November 1994.

[FAN93] Marcelo Fabián Frias, N. G. Aguayo, and B. Novak. Development of graph algorithms with fork algebras. In *Proceedings of the XIX Latinamerican Conference on Informatics*, pages 529–554, 1993.

[FBHV93] Marcelo Fabián Frias, Gabriel A. Baum, Armando M. Haeberer, and Paulo A. S. Veloso. A representation theorem for fork-algebras. Res. Rept. MCC 29, PUC-Rio, 1993.

[FBHV95] Marcelo Fabián Frias, Gabriel A. Baum, Armando M. Haeberer, and P. A. S. Veloso. Fork algebras are representable. In *Bulletin of the Section of Logic, Vol.24, N.2*, pages 64–75, University of Lódź, June 1995.

[FHV95] Marcelo Fabián Frias, Armando M. Haeberer, and P. A. S. Veloso. On the metalogical properties of fork algebras. In *Proceedings of the Winter Meeting of the ASL*, San Francisco, California, January 1995.

[FHVB94] Marcelo Fabián Frias, Armando M. Haeberer, P. A. S. Veloso, and Gabriel A. Baum. Representability of fork algebras. In *Proceedings of the Logic Colloquium '94*, page 51, San Francisco, California, July 1994.

[Gri81] David Gries. *The Science of Programming*. Springer-Verlag, 1981.

[Hal62] P. R. Halmos. *Algebraic Logic*. New York: Chelsea, 1962.

[HBS93] Armando M. Haeberer, Gabriel A. Baum, and Gunther Schmidt. On the smooth calculation of relational recursive expressions out of first-order non-constructive specifications involving quantifiers. In *Proceedings of the Intl. Conference on Formal Methods in Programming and Their Applications, LNCS 735*, pages 281–298, 1993.

[HH86a] C. A. R. Hoare and J. He. The weakest prespecification. Part I. In *Fundamenta Informatica, vol.4, no.9*, pages 51–54, 1986.

[HH86b] C. A. R. Hoare and Jifeng He. The weakest prespecification. Part II. In *Fundamenta Informatica, vol.4, no.9*, pages 217–252, 1986.

[HMT71] L. Henkin, J. D. Monk, and Alfred Tarski. *Cylindric Algebras, part I*, volume 64. Studies in Logic and the Foundations of Mathematics, North Holland, 1971.

[HMT85] L. Henkin, J. D. Monk, and Alfred Tarski. *Cylindric Algebras, part II*, volume 115. Studies in Logic and the Foundations of Mathematics, North Holland, 1985.

[HV91] Armando M. Haeberer and P. A. S. Veloso. Partial relations for program derivation: Adequacy, inevitability and expressiveness. In *Constructing Programs from Specifications - Proceedings of the IFIP TC2 Working Conference on Constructing Programs from Specifications*, pages 319–371, North Holland, 1991. IFIP WG. 2.1.

[JT51] Bjarni Jónsson and Alfred Tarski. Boolean algebras with operators. Part I. In *American Journal of Mathematics, vol. 73*, pages 891–939, 1951.

[JT52] Bjarni Jónsson and Alfred Tarski. Boolean algebras with operators. Part II. In *American Journal of Mathematics, vol. 74*, pages 127–162, 1952.

[Lyn50] Roger Lyndon. The representation of relation algebras. In *Annals of Mathematics (series 2), vol. 51*, pages 707–729, 1950.

[Mac90] Bruce J. Mac Lennan. *Functional Programming: Practice and Theory*. Addison-Wesley Publishing Company, 1990.

[MB95a] Pablo E. Martínez López and Gabriel A. Baum. Listas: Una formalización relacional. In *Anales del 1er. Congreso Argentino de Ciencias de la Computación*. Universidad Nacional del Sur, Bahía Blanca, Octubre 1995. URL: "ftp://www-lifia.info.unlp.edu.ar/pub/papers/fork-algebras/".

[MB95b] Pablo E. Martínez López and Gabriel A. Baum. Los números naturales como tipo de datos relacional. In *Anales de las 2das Jornadas Universitarias de Informática San Juan 95*. Universidad Nacional de San Juan, Noviembre 1995. URL: "ftp://www-lifia.info.unlp.edu.ar/pub/papers/fork-algebras/".

[MN94] Lena Magnusson and Bengt Nordström. The ALF proof editor and its proof engine. In *The Formal Proceeding of the 1993 Workshop on Types for Proofs and Programs*, Nijmegen, 1994.

[MSS92] S. Mikulás, I. Sain, and A. Simon. Complexity of equational theory of relational algebras with projection elements. In *Bulletin of the Section of Logic, vol.21, n.3*, pages 103–111. University of Lódź, October 1992.

[MT76] Roger Maddux and Alfred Tarski. A sufficient condition for the representability of relation algebras. In *Notices Amer. Math. Soc., vol. 23*, pages A–477, 1976.

[NPS89] Bengt Nordström, Kent Petersson, and Jan M. Smith. *Programming in Martin-Löf's Type Theory: An Introduction*. Programming Methodology Group. Department of Computer Science, University of Göteborg, Chalmers, Sweden, Midsummer 1989.

[OZ94] Lía Oubiña and Rubén Zucchello. *Estructuras Algebraicas*. Editorial Exacta, March 1994.

[Par90] Helmut A. Partsch. *Specification an Transformation of Programs: A Formal Approach to Software Development*. Texts and Monographs in Computer Science. Springer–Verlag, 1990.

[Smi83] D. R. Smith. A problem reduction approach to program synthesis. In *Proc. Int. Joint Conf. on Artificial Intelligence*, pages 32–36, 1983.

[SN94] I. Sain and I. Németi. Fork algebras in usual as well as in non–well–founded set theories. In *preprint of the Mathematical Institue of the Hungarian Academy of Sciences*, 1994. Also appeared in Relational Methods in Computer Science, Dagstuhl–Seminar–Report 80, Schloss Dagstuhl, C. Brink and G. Schmidt eds., January 1994.

[SS93] Gunther Schmidt and T. Ströhlein. *Relations and Graphs*. EATCS Monographs on Theoretical Computer Science. Springer–Verlag, 1993.

[Sto36] M. H. Stone. The theory of representations for Boolean algebras. In *Transactions of the American Mathematical Society, vol.40*, pages 37–111, 1936.

[Tar41] Alfred Tarski. On the calculus of relations. In *Journal of Symbolic Logic, vol. 6*, pages 73–89, 1941.

[TG87] Alfred Tarski and Steve Givant. *A Formalization of Set Theory without Variables*, volume 41. A.M.S. Coll. Pub. 1987.

[VH91] Paulo A. S. Veloso and Armando M. Haeberer. A finitary relational algebra for classical first-order logic. In *Bull. Section of Logic, vol. 20, no. 2*, pages 52-62. Polish Acad. Sciences, 1991.

[VHB92] Paulo A. S. Veloso, Armando M. Haeberer, and Gabriel A. Baum. Formal program construction within an extended calculus of binary relations. Res. Rept. MCC 19, PUC-Rio, 1992.

[VHF94] P. A. S. Veloso, Armando M. Haeberer, and Marcelo Fabián Frias. Fork algebras as algebras of logic. In *Proceedings of the Logic Colloquium '94*, page 127, July 1994. To appear also in: Journal of Symbolic Logic.

[Wad85] P. Wadler. How to replace failure by a list of successes: a method for exception handling, backtracking, and pattern matching in lazy functional languages. In J.P. Jouannaud, editor, *Functional Programming Languages and Computer Architecture, LNCS 201*, pages pp. 113–128. Springer, 1985.