Curso:
Teoría de la Computación.
Unidad 1, Sesión 1: Introducción

Instituto de Computación, Facultad de Ingeniería
Universidad de la República, Montevideo, Uruguay

dictado semestre 2 - 2010

Contenido:

1. Modalidad del curso.
2. Equipo docente.
3. Bibliografía.
4. Introducción.

Modalidad del curso:

- Curso en modalidad semi-presencial.
- Material organizado en 2 Unidades divididas en Sesiones (siguiendo el esquema general del contenido del curso).
- Material teórico para auto-estudio, disponible en web (`www.fing.edu.uy/inco/cursos/teocomp`), siguiendo el avance programado del curso (una sesión por semana) para auto-estudio (en web).
- Material complementario: lecturas de libros recomendados.
- Cada sesión implica entre 3 y 4 hs de lectura (incluyendo lectura de libros recomendados), estudio y elaboración de ejercicios.

- Una clase semanal (presencial) de 1 hora de teórico y 1hs. de práctico y consulta presencial.
- Se espera entre 4 y 8 hs adicionales (a lo largo del curso) de interacción por news (`fing.cursos.teocomp`).
- Prueba escrita.
- Cronograma propuesto: ver la página web del curso.

Equipo docente 2010:

- Héctor Cancela, Dina Wonsever.

Bibliografía

Material de base:

- A. J. Kfoury, Robert N. Moll, Michael A. Arbib, A Programming Approach to Computability. Springer-Verlag, 1982.
- Lewis, H., Papadimitriou, C., Elements of the theory of computation. Prentice-Hall 1981.
- El publicado en las páginas web (basado en apuntes elaborados por J.J. Prada y D. Wonsever, y la bibliografía del curso).

Otros libros relevantes.

- Garey, M., Johnson, D. Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, 1979.

- Hermes, H. Enumerability, Decidability, Computability. Second Revised Edition, Springer Verlag, 1969.

Introducción

(esta transparencia y las siguientes son en buena parte una traducción libre de la introducción del libro *Elements of the theory of computation*, Lewis, H., Papadimitriou, C.,. Prentice-Hall 1981.)

La Informática es una realidad omnipresente en nuestra sociedad actual, y ha afectado profundamente todos los aspectos de nuestra vida, desde la producción de bienes y servicios, hasta la educación y la manera en que nos comunicamos con nuestros amigos y familiares.

El enorme desarrollo de la Computación ha sido posible a través de las investigaciones que desde el siglo pasado han permitido descubrir métodos sofisticados para gestionar los recursos computacionales, permitir las comunicaciones, traducir programas, diseñar desde circuitos hasta sistemas de información, crear computadoras y programas que son cada vez más rápidos, potentes, fáciles de usar y seguros.

Los éxitos en la práctica de la Computación han estado basados en fundamentos teóricos elegantes y sólidos. Esto es usual en todas las

grandes disciplinas, cada una de las cuales tiene sus propias preguntas fundamentales. Por ejemplo, en la Física y en la Biología algunas preguntas esenciales son sobre la naturaleza de la materia, y sobre las bases y orígenes de la vida. Para la Computación, las preguntas fundamentales equivalentes son estas: ¿Qué es un algoritmo? ¿Que puede y no puede ser computado? ¿Cuando un algoritmo puede ser considerado útil en la práctica?

Estas preguntas han sido consideradas por investigadores antes aún de la construcción de las primeras computadoras electrónicas programables, a mediados del siglo XX. Una parte esencial de esta teoría (que aún sigue siendo un campo fértil de investigación, y posee muchas preguntas abiertas) ya ha sido completamente desarrollada, y subyace toda la práctica de la Computación, influenciando de manera esencial su desarrollo.

El objetivo de este curso es dar una introducción a algunas de estas ideas fundamentales, que son los paradigmas básicos de nuestro campo de estudio. No sólo estas ideas son la base esencial de la informática, sino que también son ejemplos de modelos potentes, elegantes y productivos, y cuyos resultados son de gran alcance.

El curso incluirá algunos elementos de los siguientes aspectos:

- Un repaso de conceptos fundamentales ligados a la definición de relaciones y funciones, así como de algunas técnicas básicas de demostración en matemática discreta.

- La descripción de un lenguaje simple de programación, que al mismo tiempo puede verse como un modelo de cómputo. Siendo un lenguaje simple, su potencia es suficiente para describir un algoritmo arbitrario (y es equivalente a la de otros formalismos de cómputo)

- La noción de indecidibilidad, correspondiente a constatar que hay funciones bien definidas que sin embargo no pueden ser nunca computadas con un algoritmo.

- La teoría de complejidad algorítmica, que corresponde a, dada una tarea que puede ser computada, estudiar si existen o no algoritmos razonablemente rápidos, que en la práctica puedan realizar el cómputo.

Citando a Lewis y Papadimitriou: "Computation is essential, powerful, beautiful, challenging, ever-expanding – and so is its theory. This book only tells the beginning of an exciting story. It is a modest introduction to a few basic and carefully selected topics from the treasure chest of the theory of computartion. We hope that it will motivate the readers to seek out more;...".

Objetivos del curso

- Objetivos:
  - Que el estudiante conozca diversos modelos matemáticos de computabilidad.
  - Que estudie la equivalencia de algunos de estos modelos y la tesis de formalización de la noción de procedimiento mecánico sustentada en dicha equivalencia.
  - Que conozca la existencia de clases de problemas no resolubles por métodos mecánicos y métodos de demostración asociados.
  - Que estudie las clases de complejidad de problemas, casos de problemas y su clasificación y métodos de demostración asociados.

- Para cubrir estos objetivos, la unidad 1 presenta un modelo básico de computabilidad, sus propiedades, y sus limitaciones, incluyendo la existencia de problemas indecidibles. Asimismo, se discute en dicha unidad la equivalencia de los diversos modelos matemáticos de

computabilidad (aunque no se ven en detalle los mismos). La unidad 2 presenta las nociones básicas de la teoría de la complejidad.

- Dentro de este enfoque, hay algunas preguntas que resultan centrales:
  - ¿Cuáles son los modelos que permiten formalizar las nociones intuitivas de cómputación?
  - Dada una función, ¿existe un procedimiento automatizable que dado un valor de los parámetros de la función, puede computar el valor correspondiente de la función? ¿La respuesta a esta pregunta depende del modelo elegido en respuesta a la pregunta anterior?
  - Dada una función computable, ¿existen procedimientos eficientes para efectuar el cómputo?

Repaso Técnicas de demostración

En estas transparencias revisaremos dos técnicas o esquemas de demostración que aparecen en pruebas de muchas propiedades, especialmente en el contexto de la matemática discreta. Se trata del *principio de inducción completa* y del *principio del palomar*.

Proposición: *Principio de inducción completa .*
Enunciado. *Sea A un conjunto de números naturales tales que:*

- $0 \in A$,
- *para cada número natural n, si $\{0, 1, \ldots, n\} \subseteq A$, entonces $n+1 \in A$.*

Entonces $A = N$ $\diamondsuit$
En términos intuitivos, el principio de inducción completa indica que cualquier conjunto de números naturales que contenga el 0, y con la propiedad de que si contiene todos los números entre 0 y n entonces contiene también $n + 1$, es en realidad el conjunto de todos los números naturales.

$\clubsuit$

Ejercicio. (opcional) Demuestre que para cualquier conjunto finito A, el conjunto 2^A (es decir, el conjunto de los subconjuntos de A) tiene cardinal $2^{|A|}$. $\diamondsuit$

Proposición: $Principio\ del\ palomar$.
Enunciado. $Si\ A\ y\ B\ son\ dos\ conjuntos\ finitos,\ y\ |A| > |B|,\ entonces\ no\ existe\ ninguna\ función\ inyectiva\ de\ A\ en\ B\ \diamondsuit$
Es posible demostrar esta propiedad utilizando el principio de inducción completa.

Paso baso: supongamos $|B| = 0$, es decir, $B = \emptyset$. Entonces no existe ninguna función $f : A \to B$, por lo tanto trivialmente no existe ninguna función inyectiva.

Hipótesis de inducción: supongamos que para todo A y para todo B tal que $|A| > |B|$, y $|B| \leq n$ se cumple la propiedad (es decir, no existe función inyectiva de A en B.

Tesis de Inducción: supongamos que $|A| > |B| = n + 1$ y que tomamos una función cualquiera $f : A \to B$. La tesis de inducción es que se cumple la propiedad, por lo tanto f no puede ser inyectiva.

Demostración: Elegimos un $a \in A$ (posible al no ser A vacío). Si hubiera

otro elemento $a' \in A$ distinto de a tal que $f(a) = f(a')$, entonces la función no es inyectiva.

Entonces a es el único elemento de A que es mapeado en $f(a)$. Consideramos los conjuntos $A - \{a\}$ y $B - \{f(a)\}$, y la funcion g de $A - \{a\}$ en $B - \{f(a)\}$ que coincide con f en todos los elementos de $A - \{a\}$. Por hipótesis de inducción, g no puede ser inyectiva, por lo tanto hay dos elementos de $A - \{a\}$ que se mapean al mismo elemento de $B - \{f(a)\}$ y por lo tanto f no es inyectiva. ♣

Repaso Relaciones y Funciones

Consideremos un par de conjuntos arbitrarios, y ciertos vínculos que se establecen entre sus elementos. Para fijar ideas, consideremos una agenda telefónica.

Ejemplo: **Agenda telefónica.** Disponemos de una agenda telefónica que vincula personas con números telefónicos. Cada persona tiene asociada una lista de teléfonos donde puede ser ubicada. A la vez, cada número telefónico sirve para contactar a un grupo de personas. ♣

Estos vínculos se modelan por medio de relaciones. Nuestra agenda telefónica es una relación entre las personas y los números telefónicos.

Definición de **Relación binaria.**

Dados: Conjuntos A y B

Escribimos: R es una relación binaria entre A y B

Escribimos: $A \times B$ es el dominio de definición de R

Condición: $R \subseteq A \times B$

Definición de **Relación n-aria.**

Dados: Conjuntos A_1, A_2, . . . A_n

Escribimos: R es una relación n-aria entre A_1, A_2, . . . A_n

Condición: $R \subseteq A_1 \times A_2 \times \ldots \times A_n$

♣

Definición de **Tupla, n-upla .**

Dados: Conjuntos A_1, A_2, . . . A_n, y relación n-aria R

Escribimos: a es una tupla o n-upla de R

Notación: $\langle a_1, a_2, \ldots a_n \rangle := a$

Condición: $\langle a_1, a_2, \ldots a_n \rangle \in R$

♣

Definición de **Agregar a una tupla.**

Dados: Conjunto A

Dados: $a \in A$

Dados: $\langle a_1, \ldots, a_n \rangle$ tupla de largo n

Escribimos: $a : \langle a_1, \ldots, a_n \rangle$ es una tupla de largo $n + 1$

Condición: $a : \langle a_1, \ldots, a_n \rangle = \langle a, a_1, \ldots, a_n \rangle$

Casos simples de **Tuplas.**

Tuplas de naturales. Las tuplas de naturales constituyen una construcción muy usada en las presentaciones sobre computabilidad. Esto se debe a que todo tipo de dato inductivo puede codificarse usando solamente naturales; luego, si fuera posible codificar con naturales tanto los programas como los datos, las presentaciones se simplifican. Y como sabemos de la numerabilidad de objetos con alfabetos numerables, esta codificación es posible.

Definición de **Estar relacionado.**

Dados: Relación R entre A y B. Elementos $a \in A, b \in B$.

Escribimos: a está relacionado con b en R

Notación: $R.a.b$

Condición: $\langle a, b \rangle \in R$

Casos simples de **relaciones entre A y B.**

Relación vacía $\emptyset$

Relación total $A \times B$

Ejemplo: **Agenda telefónica.** La agenda telefónica es una relación entre el conjunto de todas las personas y el conjunto de todos los números telefónicos. Resulta razonable averiguar precisamente de qué personas disponemos el número telefónico; no estamos interesados en todas las personas, sino solamente en aquellas de las que conocemos su teléfono. ♣

Las relaciones afirman la existencia de vínculos entre individuos de los conjuntos A y B. El recorrido de la relación es el conjunto de aquellos individuos vinculados por la relación.

Definición de **Recorrido de la primera (segunda) coordenada.**

Dados: Relación R entre A y B.

Escribimos: S es el recorrido de la primera (segunda) coordenada de R

Notación: $\pi_1.R$ $(\pi_2.R)$

Condición: $S = \{a \in A : (\exists b \in B :: \langle a, b\rangle \in R)\}$
$(\{b \in B : (\exists a \in A :: \langle a, b\rangle \in R)\})$

Ejemplo: **Agenda telefónica.** Usamos una agenda para obtener los teléfonos de alguna persona. ♣

La imagen de un elemento en la relación es el conjunto de aquellos otros elementos con los que está relacionado.

Definición de **Imagen de la primera (segunda) coordenada.**

Dados: Relación R entre A y B, y un elemento $a \in A$ ($b \in B$).

Escribimos: S es la imagen de a (b) por la relación R en la primera (segunda) coordenada

Escribimos: S es la imagen de a por la relación R (S es la preimagen de b por la relación R)

Notación: $R.a._-$ ($R._-.b$)

Condición: $S = \{b \in B : \langle a, b\rangle \in R\}$ ($\{a \in A : \langle a, b\rangle \in R\}$)

♣

Ejemplo: **Agenda telefónica.** Supongamos que todas las personas tienen algún teléfono. Podemos pensar en la existencia de una guía telefónica universal donde aparecen listados todos los teléfonos en los cuáles podemos ubicar a cualquier persona. ♣

Definición de **Relación total en la primera coordenada.**

Dados: Relación R entre A y B.

Escribimos: R es una relación total en la primera coordenada

Condición: $\pi_1.R = A$

♣

Definición de **Relación total en la segunda coordenada.**

Dados: Relación R entre A y B.

Escribimos: R es total en la segunda coordenada

Condición: $\pi_2.R = B$

♣

Ejemplo: **Agenda telefónica.** Supongamos que todas las personas tienen, a lo más, un teléfono. Es decir, cada persona tiene asociado un único teléfono, o ninguno. La agenda establece una relación funcional entre las personas y sus números telefónicos. ♣

Definición de **Relación funcional en la primera coordenada.**

Dados: Relación R entre A y B

Escribimos: R es una relación funcional en la primera coordenada entre A y B

Condición: $(\forall a \in A, \{b, b'\} \subseteq B : \langle a, b\rangle \in R \wedge \langle a, b'\rangle \in R : b = b')$

♣

Definición de **Relación funcional en la segunda coordenada.**

Dados: Relación R entre A y B

Escribimos: R es una relación funcional en la segunda coordenada entre A y B

Condición: $(\forall \{a, a'\} \subseteq A, b \in B : \langle a, b \rangle \in R \wedge \langle a', b \rangle \in R : a = a')$

♣

Ejemplo: **Suma y resta.** La suma en los naturales puede modelarse como una relación entre parejas de naturales, los sumandos, y naturales, el resultado de la suma. Para indicar que esta relación se cumple entre la pareja de sumandos 2 y 3, y el resultado 5, podemos escribir:

$$\langle\langle 2, 3\rangle, 5\rangle \in +$$

Sin embargo, usualmente preferimos notaciones que reflejen la situación de que la relación suma es una relación funcional. El uso de la igualdad explicita esta situación

$$2 + 3 = 5$$

En ocasiones nos encontramos con relaciones en las que aparecen elementos no relacionados. Esto nos sucede con la resta en los naturales, dónde la pareja $\langle 2, 3\rangle$ no tiene correspondiente. El uso de *indef* explicita esta situación

$$2 - 3 = \textit{indef}$$

Observemos que *indef* no es un valor, sino que representa un comportamiento de la expresión de la izquierda: "no hay ningún objeto

que cumpla con la igualdad". Este uso de *indef* tiene como consecuencia desagradable que la expresión

$$indef = indef$$

no tenga sentido, así como tampoco lo tiene

$$f.indef.$$

Sin embargo, habitualmente no se tiene tanta preocupación con el uso de esta expresión, y se emplea libremente como abreviatura de otras expresiones sin declarar. Sugerimos al lector atarse a la interpretación estricta que dimos antes hasta que alcance una fluidez adecuada con el concepto. ♣

Nota: **Igualdad.** El uso de la igualdad es, en muchas ocasiones, ambiguo. Por ejemplo, el texto

$$2 + 3 = 5$$

puede interpretarse, al menos, de las formas siguientes:

- El resultado de evaluar lo que hay a la izquierda coincide con el resultado de evaluar lo que hay a la derecha;

- El resultado de evaluar lo que hay a la izquierda coincide con lo que hay a la derecha.

La primera lectura está asociada a la idea de una relación simétrica; la segunda a una noción de cómputo. Para entender mejor la diferencia entre ellas hagamos el ejercicio de pensar el siguiente problema escolar:

Resuelva la siguiente ecuación: $2 + 3 = x$.

Supongo que el lector propondrá como solución el juicio $x = 5$; pero de acuerdo a la primera lectura, tanto $x = 555 - 550$ como $x = 2 + 3$ hubieran sido respuestas satisfactorias. Nuestro ejemplo se lee de acuerdo a la segunda interpretación, donde lo que hay que colocar a la derecha no es una expresión a computar, sino algo que podemos llamar, para no innovar, forma normal. ♣

Las funciones son relaciones funcionales. El adjetivo de *total* o *parcial* habla del recorrido de la primera coordenada.

Definición de **Función parcial.**

Dados: Conjuntos A y B

Escribimos: f es una función parcial de A en B

Escribimos: $f : A \to_\perp B$

Condición: f es una relación funcional en la primera coordenada

Definición de **Función total.**

Dados: Conjuntos A y B

Escribimos: f es una función total de A en B

Escribimos: $f : A \to B$

Condición: f es una relación funcional y total en la primera coordenada

Las funciones parciales son, por lo tanto, una clase especial de relaciones. Usamos nombres especiales para referirnos a los recorridos de estas relaciones.

Definición de **Dominio y codominio.**

Dados: Una función parcial $f : A \rightarrow_{\perp} B$

Escribimos: A es el dominio de f, y B es el codominio de f

Casos simples de **Dominio y codominio.**

Naturales En este curso trabajaremos casi siempre con funciones parciales cuyos dominio y codominio serán tuplas de naturales. Es decir, la forma de las funciones será $\theta : N^k \rightarrow_{\perp} N^l$. En muchas ocasiones, l será 1.

Estas funciones serán computadas por programas con un único tipo de dato primitivo: `natural` [1].

♣

Definición de **Dominio de definición.**

Dados: Una función parcial $f : A \rightarrow_{\perp} B$.

Escribimos: $DOM.f$ es el dominio de definición de f

Condición: $DOM.f = \pi_1.f$

♣

Definición de **Rango.**

Dados: Una función parcial $f : A \rightarrow_{\perp} B$.

[1] El tipo de dato que usamos en nuestro programa puede representar cualquier natural, sin limitación alguna de magnitud.

Escribimos: *RAN.*f es el rango de f

Condición: $\mathit{RAN}.f = \pi_2.f$

Casos simples de **funciones parciales.**

Función vacía $\emptyset : A \rightarrow_\perp B$. En este curso la llamamos $\perp$. Está definida con la regla siguiente:

$$(\forall a \in A :: \perp.a = \mathit{indef})$$

Ejercicio. Calcule $DOM.\perp$ y $RAN.\perp$. $\diamondsuit$

Ejercicio. Muestre cómo considerar que la función vacía es una función parcial y total. $\diamondsuit$

Casos simples de **funciones totales.**

Dados: $A \subseteq B$

Inclusión $I : A \to B$. Está definida con la regla siguiente:

$$(\forall a \in A :: I.a = a)$$

Cumple $DOM.I = A$ y $RAN.I = A$.

Identidad $id : A \to A$. Está definida con la regla siguiente:

$$(\forall a \in A :: id.a = a)$$

Cumple $DOM.id = A$ y $RAN.id = A$.

Ejemplo: **Programas Pascal y funciones parciales.** Los programas Pascal[2] implementan funciones parciales. Por ejemplo, considere el siguiente programa:

```
FUNCTION ejemplo1 (n : natural) : natural;
BEGIN
   WHILE n MOD 2 = 0 DO
      n := n + 2;
   ejemplo1 := n
END
```

La función parcial implementada por este código es

$$\begin{array}{ll} \theta : N \to_{\perp} N & \\ \theta.n = n, & \text{si } n \text{ es impar} \\ \theta.n = \mathit{indef}, & \text{si } n \text{ es par} \end{array}$$

[2]Hablamos de programas Pascal para referirnos a los módulos `FUNCTION` de Pascal. A nuestros fines, esto es suficiente.

La función parcial $\perp : N \to_\perp N$ puede implementarse con el siguiente código Pascal.

```
FUNCTION vacia (n : natural) : natural;
BEGIN
   WHILE n = n DO;
   vacia := n
END
```

♣

Proposición: *Funciones totales y parciales.*
Enunciado. *Las nociones de función parcial y total pueden intercambiarse.* $\diamondsuit$
Toda función total $f : A \to B$ es una función parcial. Por otra parte, toda función parcial $g : A \to_\perp B$ es una función total cuando restringimos su dominio al dominio de definición de g. Es decir, podemos considerar en su lugar la función total $g' : \mathit{DOM}.g \to B$ definida con la regla de

correspondencia siguiente

$$(\forall a \in \mathit{DOM}.g :: g'.a = g.a)$$

Nota: **Funciones parciales y totales.** ¿Por qué distinguir entre funciones totales y parciales? Si nuestra visión de las funciones es extensional, o sea, nos interesamos en conocer cuáles son las parejas relacionadas, la distinción entre estos puntos de vista resulta poco interesante. La única dificultad reside en identificar adecuadamente el dominio de definición de las funciones parciales.

La situación es diferente si estudiamos las funciones desde el punto de vista intensional, o sea, interesándonos en las reglas que me permiten descubrir la imagen de un elemento dado. Si nuestro único conocimiento acerca de las funciones consiste en el mecanismo de cálculo asociado a la función (o sea, su algoritmo, o programa), nos encontramos ante tres posibilidades:

- aplicamos el programa a un elemento, y obtenemos la imagen correspondiente

- aplicamos el programa a un elemento, y en algún momento del cómputo el programa realiza alguna operación ilegal

- aplicamos el programa a un elemento, y el cómputo del programa no termina

A nuestros fines, el valor *indef* será asignado solamente al tercer caso. El segundo caso suele modelarse agregando al codominio un valor especial de error.

Por ejemplo, considere la siguiente implementación[3] PASCAL (defectuosa) que recibe un natural mayor que 10, y calcula la cantidad de los pares entre 10 y el argumento.

```
FUNCTION sumapar (n:natural) : natural;
VAR res, i : natural;
BEGIN
   i := 10; res := 0;
   IF (n <= 10) THEN writeln('error')
   ELSE
      WHILE (n <> i) do
```

[3] Se asume una implementación que admite representaciones numéricas de tamaño arbitrario.

```
      BEGIN
         res := res + 1; i := i + 2
      END;
   sumapar := res
END;
```

El programa `sumapar` implementa la siguiente función:

$$
\begin{array}{l}
f : \texttt{natural} \rightarrow_{\perp} \texttt{natural} \cup \{\texttt{error}\} \\
f.n = \frac{n}{2} - 5, \qquad \text{si } n = \dot{2} \text{ y } n > 10 \\
f.n = \textit{indef}, \quad \text{si } n = \dot{2} + 1 \text{ y } n > 10 \\
f.n = \texttt{error}, \qquad \text{si } n \leq 10
\end{array}
$$

El dominio de la función es `natural`; su codominio es $\texttt{natural} \cup \{\texttt{error}\}$; su dominio de definición es $\{n \in \texttt{natural} : n \leq 10 \vee n = \dot{2}\}$; y su rango, es $\{n \in \texttt{natural} : n \geq 0 \wedge n = \dot{2}\}$. ♣

Definición de **Función inyectiva.**

Dados: f es una función[4]

Escribimos: f es inyectiva

Condición: f es una relación funcional en la segunda coordenada

Definición de **Función inyectiva.**

Dados: $f : A \rightarrow_{\perp} B$

Escribimos: f es inyectiva

Condición: $(\forall \{a, a'\} \subseteq \mathit{DOM}.f : f.a = f.a' : a = a')$

---

[4] No aclaro para funciones totales, por haber una inclusión conceptual inmediata.

Definición de **Función sobreyectiva.**

Dados: f es una función parcial

Escribimos: f es sobreyectiva

Condición: f es una relación total en la segunda coordenada

Definición de **Función sobreyectiva.**

Dados: $f : A \rightarrow_{\perp} B$

Escribimos: f es sobreyectiva

Condición: $(\forall b \in B :: (\exists a \in A :: b = f.a))$

Definición de **Función biyectiva.**

Dados: $f : A \to_\perp B$

Escribimos: f es biyectiva

Condición: f es inyectiva y f es sobreyectiva

♣

Nota: **Operaciones sobre funciones.** Conocemos distintos mecanismos que nos permiten construír nuevas funciones a partir de funciones ya conocidas. Como en la computabilidad estudiamos los programas como funciones, estos mecanismos se reflejan en algún sentido en la construcción de programas más complejos. ♣

Definición de **Composición de funciones.**

Dados: $\theta : A \to_\perp B, \zeta : B \to_\perp C$

Escribimos: $\zeta \circ \theta : A \to_{\perp} C$ es la función compuesta, o composición, de las funciones θ y ζ.

Condición:

$(\forall a \in A :: \zeta \circ \theta.a = (\texttt{IF}\ \ \theta.a = \textit{indef}\ \texttt{THEN}\ \ \textit{indef}\ \texttt{ELSE}\ \ \zeta.(\theta.a)))$

Ejercicio. Considere $\theta : A \to_\perp B, \zeta : B \to_\perp C$. Demuestre que

1. $\mathit{DOM}.\zeta \circ \theta \subseteq \mathit{DOM}.\theta$

2. $\mathit{RAN}.\zeta \circ \theta \subseteq \mathit{RAN}.\zeta$

3. $\mathit{DOM}.\zeta \circ \theta = \{x \in \mathit{DOM}.\theta : \theta.x \in \mathit{DOM}.\zeta\}$

4. $\mathit{RAN}.\zeta \circ \theta = \{\zeta.x : x \in \mathit{RAN}.\theta\}$

5. La función vacía juega el rol de absorsor: $(\forall \theta :: \theta \circ \perp = \perp \wedge \perp \circ \theta = \perp)$

6. La función identidad juega el rol de neutro a izquierda :
   $(\forall \theta :: \theta \circ \mathit{id} = \theta)$

Pruebe que no es cierto que

1. $DOM.\zeta \circ \theta = DOM.\theta$

2. $RAN.\zeta \circ \theta = RAN.\zeta$

3. La función identidad juega el rol de neutro a derecha : $(\forall \theta :: id \circ \theta = \theta)$

Podemos precisar mejor la relación entre funciones parciales y totales. Demuestre que

1. $(\exists \pi_\theta : A \rightarrow_\perp DOM.\theta, \theta_t : DOM.\theta \rightarrow B :: \theta = \theta_t \circ \pi_\theta)$

$\diamondsuit$

Ejemplo: **Composición de funciones.** Consideremos la función `por2` que a cada natural le asigna su doble, y la función `suma1` que a cada natural le suma uno.

```
FUNCTION por2 (n:natural) : natural;
VAR i : natural;
BEGIN
   i := 2 * n;
   por2 := i
END;
FUNCTION suma1 (n:natural) : natural;
VAR i : natural;
BEGIN
   i := 1 + n;
   suma1 := i
END;
```

Naturalmente, podemos evaluar en un programa arbitrario las siguientes expresiones:

```
por2 (suma1 (k))
```

y

```
suma1 (por2 (k))
```

En el primer caso, esperamos que la expresión evalúe al natural $2 \times k + 2$. Observemos que esta invocación podría haber sido codificada en una nueva función Pascal, colocando primero un texto y luego (casi) el otro.

```
FUNCTION por2suma1 (n : natural) : natural;
VAR i, j : natural;
BEGIN
   i := 1 + n;
   j := 2 * i;
   por2suma1 := j
END
```

Esta nueva función implementa la *composición* de `por2` y `suma1`. Constate que el orden de la composición es relevante. ♣

Nota: **Composición de funciones.** En ocasiones se define extensionalmente

$$(\forall x :: \zeta \circ \theta.x = \zeta.(\theta.x))$$

Esta definición no es completamente satisfactoria. Observemos que la expresión $\zeta.y$ tiene sentido cuando $y \in \mathit{DOM}.\zeta$. Si al objeto x no se encuentra en $\mathit{DOM}.\theta$, deberemos evaluar $\zeta.\mathit{indef}$, lo que no tiene sentido. ♣

Ejemplo: **Función inversa.** Consideremos una función f que transforma un cierto objeto. La función inversa de f nos permite recuperar el objeto inicial a partir del transformado. En ocasiones, la operación de encontrar la inversa de una función es un trabajo difícil; pero en otras ocasiones, esta tarea es directamente imposible. ♣

Definición de **Función inversa.**

Dados: $\theta : A \rightarrow_{\perp} B$

Escribimos: $\theta^{-1} : B \rightarrow_{\perp} A$ es la función inversa de la función θ.

Condición: $(\forall a \in \mathit{DOM}.\theta :: \theta^{-1} \circ \theta.a = a)$

Nota: **Existencia de funciones inversas.** La definición anterior no garantiza la existencia del objeto definido. Por lo tanto, la definición de función inversa debe asociarse estrechamente con la condición de existencia de ese objeto. ♣

Condición necesaria de la existencia de funciones inversas.

Hipótesis: $\theta : A \rightarrow_{\perp} B$

Hipótesis: $\theta^{-1} : B \rightarrow_{\perp} A$

Hipótesis: $(\forall a \in A :: \theta^{-1} \circ \theta.a = a)$

Tesis: θ es inyectiva.

Demostración. Consideremos $\{a, a'\} \subseteq \mathit{DOM}.\theta$.

$$
\begin{aligned}
&\theta.a = \theta.a' \\
&\Rightarrow \\
&\theta^{-1}.\theta.a = \theta^{-1}.\theta.a' \\
&\Rightarrow \\
&\theta^{-1} \circ \theta.a = \theta^{-1} \circ \theta.a' \\
&\Rightarrow \\
&a = a'.
\end{aligned}
$$

Nota: **CNS y existencia.** La existencia de un objeto se puede asegurar mediante alguna condición suficiente. Las condiciones necesarias pueden servir de inspiración en esa búsqueda; pero no realizan ningún aporte desde el punto de vista deductivo. ♣

Condición suficiente para la existencia de funciones inversas.

Hipótesis: $\theta : A \rightarrow_{\perp} B$

Hipótesis: θ es inyectiva.

Tesis: $(\exists \zeta : (\forall a \in A :: \zeta \circ \theta.a = a))$

Demostración. Definimos ζ de la forma siguiente:

$$\begin{array}{ll} \zeta : B \rightarrow_{\perp} A & \\ \zeta.b = a, & \text{si } b \in \mathit{RAN}.\theta \text{ y } \theta.a = b \\ \zeta.b = \mathit{indef}, & \text{en cualquier otro caso} \end{array}$$

Le dejamos al lector demostrar que la función así definida cumple con la propiedad deseada.

Proposición: $Inversa\ sobreyectiva.$
La función inversa de una función inyectiva total, es sobreyectiva. ♣

Definición de **Extensión de funciones.**

Dados: $\psi : A \rightarrow_{\perp} B$

Dados: $\tau : A \rightarrow_{\perp} B$

Escribimos: ψ es una extensión de τ

Notación: $\tau \leq \psi$

Condición: $\tau \subseteq \psi$

♣

Ejercicio. Demuestre que a toda función sobreyectiva $\theta : B \rightarrow_{\perp} A$ se le puede asociar una función inyectiva $\zeta : A \rightarrow B$ tal que $\zeta^{-1} \leq \theta$. ♢

Ejercicio. **1.1.1.** Construya una secuencia infinita de funciones parciales distintas $\psi_i : N \to_\perp N$ tales que

$$\psi_1 \leq \psi_2 \leq \ldots \leq \psi_n \leq \ldots$$

$\diamondsuit$

Nota: **Tamaño de los conjuntos.** Los conjuntos están determinados por sus elementos. La inclusión es una relación de orden que podemos establecer entre los conjuntos de acuerdo a los elementos que los conforman. Aquí estamos interesados en ordenar los conjuntos de acuerdo a la cantidad de elementos, y no de acuerdo a su calidad.

Por ejemplo, consideremos los conjuntos $A = \{\star, \bullet, \circ\}$, $B = \{\star, \bullet\}$, y $C = \{\odot, \bullet\}$. La inclusión no es un orden total, y no nos permite relacionar ni los conjuntos A, C, ni los conjuntos B, C. La cantidad de elementos sí nos permitiría establecer la relación "tener menos elementos que", y podríamos escribir $B \leq A$, $C \leq A$, y $B = C$[5]

[5]Esta última igualdad no es la igualdad extensional de conjuntos, sino la abreviatura de "$B \leq C$ y $C \leq B$". Más adelante introducimos la notación de cardinal que permite desambiguar esta situación.

Esta propuesta de contar los elementos resulta simple y natural cuando pensamos en conjuntos finitos. La observación que nos permite generalizar esta noción a conjuntos infinitos es la siguiente: hay inyecciones de B en A, pero no puede haberlas de A en B. ♣

Definición de **Menor o igual cardinal.**

Dados: Conjuntos A, B

Escribimos: A tiene menor o igual cardinal que B

Escribimos: $|A| \leq |B|$

Condición: Hay una inyección de A en B

Definición de **Orden en cardinales.**

Dados: Conjuntos A, B, C

Escribimos: A tiene menor cardinal que B

Escribimos: A tiene igual cardinal que C

Escribimos: $|A| < |B|$, $|A| = |C|$

Condición: $|A| \leq |B|$ y no pasa $|B| \leq |A|$

Condición: $|A| \leq |C|$ y $|C| \leq |A|$

Nota: **Orden total y cardinales.** La relación de orden entre cardinales es total (alternativamente notado orden lineal). La prueba de ello exige el uso del axioma de elección; no la proporcionamos. ♣

Nota: **Numeración .** Consideremos un conjunto cualquiera A; en muchas ocasiones estamos más interesados en ordenar los elementos de A, antes que en conocer precisamente la naturaleza de cada elemento. Por ejemplo, muchas panaderías ordenan a sus clientes por orden de llegada, numerándolos con un entero; al llamar al siguiente cliente, el vendedor no está interesado en su cliente como persona[6] sino como portador de un número determinado. El orden de los naturales tiene dos propiedades que son muy relevantes a los intereses del cliente:

conjunto infinito hay suficientes naturales para todos los seres humanos del planeta; todo cliente sabe que, en la medida en que alcance el pan y el vendedor no quede afónico, será llamado.

[6] Aunque eventualmente lo esté unos momentos más tarde.

orden discreto no existe la posibilidad de que se cuele alguien; si se ha llamado al cliente/número n, y el cliente/número k sabe precisamente cuándo será llamado.

Numerar un conjunto consiste en indicar precisamente la posición de cada elemento en una lista; quién es el primero, quién el segundo, y así sucesivamente. ♣

Definición de **Numeración.**

Dados: Conjunto A

Dados: $h : N \to A$

Escribimos: h es una numeración de A

Escribimos: A está numerado por h

Condición: h es sobreyectiva

Definición de **Conjunto numerable.**

Dados: Conjunto A

Escribimos: A es numerable

Condición: $A = \emptyset$ o A esta numerado[7]

[7] Por alguna numeración.

Ejercicio. Pruebe que

$$A \text{ es numerable } \Leftrightarrow |A| \leq |N|$$

♢

Casos simples de **Conjuntos numerables.**

El conjunto vacío

El conjunto de los naturales

Cualquier conjunto finito

♣

Nota: **Numerabilidad de lenguajes con alfabeto finito.** Considere un alfabeto finito $\{a_1, \ldots, a_n\}$ con n marcas diferentes, y un lenguaje no vacío C que usa ese alfabeto. Una numeración posible sigue los siguientes pasos

1. Agregue una marca nueva, a la que llamaremos a_0

2. Asigne a cada marca su índice: $\#a_i = i$.

3. Considere la siguiente numeración (cambio de base $n+1$), donde hemos fijado una palabra arbitraria $\star \in C$.

$$h : N \to C$$
$$h.(\#a_{i_k}(n+1)^k + \ldots + \#a_{i_0}(n+1)^0) := a_{i_k} \ldots a_{i_0} \quad \text{, si } a_{i_k} \ldots a_{i_1} \in C$$
$$h.(\#a_{i_k}(n+1)^k + \ldots + \#a_{i_0}(n+1)^0) := \star \quad \text{, en cualquier otro caso}$$

Nota: **Otros conjuntos numerables.**

- Los enteros Z también son un conjunto numerable; nos basta con numerar cada uno de ellos. Una numeración posible consiste en asignarle números pares crecientes a los positivos e impares crecientes a los negativos.

$$h_Z : N \to Z$$
$$h_Z.(2k) := k$$
$$h_Z.(2k+1) := -k$$

- Las parejas de naturales también son numerables. Una numeración posible consiste en desagregar el factor primo 2 de cada natural.

$$h_2 : N \to N \times N$$
$$h_2.(2^k \times l) := \langle k, l \rangle \quad , l \neq \dot{2}$$

- El conjunto de las fórmulas del cálculo proposicional también es numerable[8]. Para numerarlo considere las siguientes dos magnitudes: el

[8] Ver práctico del curso de Lógica.

índice de las letras proposicionales y la cantidad de conectivos usados; observe que el conjunto de las fórmulas que usan la letra proposicional con mayor índice y que tienen una cantidad fija de conectivos es finito.

- Cualquier lenguaje inductivo con una cantidad numerable de casos bases y una cantidad numerable de casos inductivos es numerable. Para numerarlo usamos la misma estrategia que en el caso particular del cálculo proposicional.

- El conjunto de secuencias finitas N^* de naturales también es numerable. Se infiere fácilmente de lo anterior por estar definido inductivamente con las siguientes reglas:

  Un caso base: $\langle\rangle \in N^*$
  Numerables casos inductivos: $(\forall n : s \in N^* : n : s \in N^*)$

- El conjunto de todas las palabras en español; y el conjunto de todos los libros que se puedan escribir en cualquier idioma con un alfabeto dado [9].

[9] Y si los alfabetos son numerables, con cualquier alfabeto.

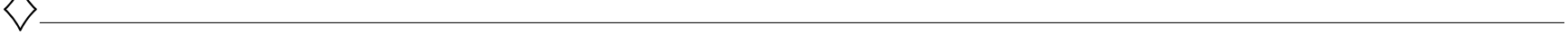
- El conjunto de todos los programas Pascal.
- El conjunto de todos los programas escritos en cualquier lenguaje de programación.
- El conjunto de todos los programas escritos en todos los lenguajes de programación.

Ejercicio. Muestre que los siguientes conjuntos son numerables, proporcionándoles una numeración.

1. enteros
2. racionales
3. algebraicos

4. programas PASCAL

5. funciones parciales de N en N con dominio de definición finito



Nota: **Numerabilidad y tamaño.** Los conjuntos numerables son finitos o del tamaño de los naturales. Uno de los resultados más interesantes de la teoría de conjuntos propuesta por Georg Cantor consistió en poder expresar la existencia de conjuntos infinitos de distinto tamaño. El conjunto infinito de menor tamaño es el de los naturales; los conjuntos de mayor tamaño son tales que nunca podremos numerarlos. El teorema de Cantor establece que hay conjuntos de mayor tamaño que los naturales. ♣

Teorema de Cantor.

Tesis: $|N| < |R|$

Demostración. La estrategia de prueba es el contrarrecíproco.

Suponemos $|R| \leq |N|$, y consideramos una numeración h de todos los reales. Es decir, podemos colocar todos los números reales en una lista infinita $h.0, h.1, \ldots$. Construiremos un número real que no aparece en dicha lista, contradiciendo el supuesto de que h era una numeración de los reales.

La visualización de este teorema le da nombre al método usado, llamado diagonalización. Para fijar ideas, consideremos que la enumeración h está dada por la lista siguiente, donde la primera fila se corresponde con la imagen de 0 en la numeración, la segunda con la imagen de uno, y así sucesivamente. Además, destinamos la primera columna para la parte entera del real, la segunda columna para el dígito correspondiente a la décima, la tercera para el correspondiente a la centésima, y así. Es decir, el número real asociado al uno es el 12,45678. . . . Usamos la notación de arreglos habitual para referenciarnos a cada columna; por tanto, $h.0[0]$ será 22, $h.2[4]$ será 1, etc.

$$
\begin{array}{cccccc}
\mathbf{22} & 3 & 1 & 5 & 4 & 3 & \ldots \\
12 & \mathbf{4} & 5 & 6 & 7 & 8 & \ldots \\
1 & 3 & \mathbf{1} & 3 & 1 & 9 & \ldots \\
0 & 9 & 8 & \mathbf{7} & 6 & 5 & \ldots \\
9 & 9 & 9 & 9 & \mathbf{9} & 9 & \ldots \\
2 & 2 & 2 & 2 & 2 & \mathbf{2} & \ldots \\
\vdots & & & & & &
\end{array}
$$

Observemos la diagonal en negritas, formada por la parte entera $h.0[0]$ y los dígitos de la parte fraccionaria $h.i[i]$ para $i > 0$. Consideremos el real que se obtiene a partir de esa diagonal (infinita) tomando, en lugar del dígito en negrita, su sucesor módulo 10. Es decir, fijemos la atención en el número 3,52803. . . .

Este número no está en el primer lugar de la lista, ya que la parte entera no puede concidir con su sucesor módulo 10. Pero tampoco en el segundo lugar de la lista, porque su primer dígito decimal tampoco puede coincidir con él mismo módulo 10. De igual forma con las restantes posiciones de la lista.

La técnica aplicada se puede usar para cualquier enumeración. En general, si se supone una numeración h, y llamamos f a la función

$$f.k := (k + 1) \ \texttt{MOD} \ 10,$$

el número r construído está formado por los siguientes dígitos

concatenados:

$$f.(h.0[0]), f.(h.1[1]), f.(h.2[2]), f.(h.3[3]), f.(h.4[4]), \ldots$$

Si r apareciera en la posición n de la lista, debería cumplirse la igualdad

$$h.n[n] = f.(h.n[n]).$$

Pero elegimos la función f de forma que no tuviera puntos fijos, mostrando la imposibilidad de esta construcción.

Nota: **Diagonalización.** El método de diagonalización aparece en varias construcciones de la computabilidad. Permite demostrar la imposibilidad de numerar distintos conjuntos. ♣

Funciones totales no numerables.

Tesis: El conjunto de funciones totales de N en N no es numerable.

Demostración. Considere una enumeración h de estas funciones, de forma que $h.i$ es una función total. Consideremos ahora la función total definida de la forma siguiente:

$$f.n := (h.n).n + 1$$

Esta función no está en la enumeración dada por h.

♣

Inclusión y numerabilidad.

Hipótesis: Conjuntos A, B

Hipótesis: A es numerable

Hipótesis: $B \subseteq A$

Tesis: B es numerable

Demostración. Consideramos dos casos, de acuerdo a B

1. $B = \emptyset$. Luego, B es numerable.
2. $\star \in B$. Sea h una numeración de A; definimos h' como

$$\begin{array}{ll} h' : N \to B & \\ h'.m := h.m, & \text{si } h.m \in B \\ h'.m := \star, & \text{si } h.m \notin B \end{array}$$

Complete la prueba.

Nota: **Probando la no numerabilidad.** Tenemos al momento dos formas de probar que un conjunto no es numerable, aparte del uso directo de la definición:

diagonalización: suponer una numeración, y construír un elemento del conjunto a partir de la modificación de la diagonal que no puede estar en esa lista.

inclusión: encontrar un subconjunto no numerable.

Ejercicio. Demuestre que los siguientes conjuntos no son numerables.

1. reales entre 0 y 1
2. funciones parciales de N en N
3. funciones totales de N en $\{0, 1\}$
4. conjunto potencia de N
5. funciones parciales de N en N con rango finito

Bibliografía

References

[KMA82] A. J. Kfoury, Robert N. Moll, Michael A. Arbib, A Programming Approach to Computability. Springer-Verlag, 1982.

[PL98] Lewis, H., Papadimitriou, C., Elements of the theory of computation. Prentice-Hall 1998 (2da edición).