Tema 1: Algoritmos y funciones computables

Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Ciencias de la Computación
(4º curso, Grado en Matemáticas)
Curso 2021–22

Contenido

Problemas decidibles e indecidibles
- Primera definición informal
- Segunda definición informal

El lenguaje GOTO
- Sintaxis de GOTO
- Semántica intuitiva de los programas GOTO

Macros
- Uso de macros

Semántica
- Estados y descripciones instantáneas
- Computación de un programa

Definición formal de las macros

Apéndice: Conjuntos y funciones

Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal

El lenguaje `GOTO`
Sintaxis de `GOTO`
Semántica intuitiva de los programas `GOTO`

Macros
Uso de macros

Semántica
Estados y descripciones instantáneas
Computación de un programa

Definición formal de las macros

Apéndice: Conjuntos y funciones

Funciones computables (informalmente)

- A continuación desarrollamos de manera informal una teoría básica de funciones computables apoyándonos en una noción informal de **algoritmo**.
- Un hecho clave será la existencia de una **enumeración efectiva** de los algoritmos (que puede justificarse informalmente imponiendo algunas restricciones razonables a la definición intuitiva de algoritmo)
- En primer lugar justificaremos la necesidad de incluir **funciones parciales** en nuestro tratamiento.

“Definición” de algoritmo (primer intento)

Definición.

“Un algoritmo, A, es una sucesión finita de instrucciones, $I_1, \dots, I_n$, que para cada dato de entrada, x, nos proporciona al cabo de un número finito de pasos, un dato de salida que denominamos el resultado de aplicar el algoritmo A al dato x y que denotaremos por $A(x)$.”

Algunas precisiones

1. Un algoritmo debe proporcionar, a partir del dato de entrada, un resultado mediante un procedimiento totalmente rutinario, susceptible de ser llevado a cabo por una dispositivo mecánico.
   - Dicho dispositivo funcionará de manera discreta realizando sus operaciones básicas secuencialmente.
   - La operación realizada en cada momento está determinada por la configuración del dispositivo y sólo existen una cantidad finita de configuraciones y operaciones básicas.
2. Es posible describir cada algoritmo mediante una cadena de caracteres sobre un alfabeto finito fijado de antemano, de tal modo que sea "fácil" (una tarea mecánica) reconocer si una cadena dada describe un algoritmo o no.
3. Cada instrucción especifica una operación *efectiva*, que podría ser realizada por una persona en un tiempo finito, con una cantidad finita de recursos (memoria).

Enumeración de los algoritmos

- Es posible asignar un número a cada algoritmo de modo que los algoritmos queden enumerados en una lista infinita $A_0, A_1, \ldots, A_n, \ldots$
  - Para ello, puesto que cada algoritmo puede ser descrito por una cadena de caracteres en un alfabeto finito fijo, basta ordenar las cadenas de este alfabeto usando un algoritmo de ordenación lexicográfica en el que las de menor longitud aparezcan antes en dicho orden.
- Además, debido al carácter efectivo y uniforme que las condiciones (1)–(3) de nuestra "definición" imponen a la descripción de los algoritmos mediante cadenas de caracteres, existen procedimientos efectivos que permiten:
  1. Dado un algoritmo obtener su número, y
  2. Dado un número obtener el algoritmo que le corresponde.
- Más aún, existe un procedimiento efectivo, U, que, dados n y x, obtiene el resultado de aplicar A_n a x. El procedimiento U se denomina *algoritmo universal*.

Diagonalización

- Ahora podemos definir el siguiente algoritmo D:

  Entrada: $n \in \mathbb{N}$
  Salida: $D(n) = A_n(n) + 1$

  1. Hallar A_n.
  2. Calcular $A_n(n)$ (utilizando el algoritmo universal U).
  3. Devolver $A_n(n) + 1$ (y parar).

- Puesto que D es un algoritmo, debe aparecer en la enumeración de los algoritmos que dimos anteriormente. Es decir, existe un número $e \in \mathbb{N}$ tal que $D = A_e$.
- Pero entonces

$$\left.\begin{array}{l} D(e) = A_e(e) + 1 \\ D(e) = A_e(e) \end{array}\right\} \Longrightarrow A_e(e) = A_e(e){+}1 \Longrightarrow 0 = 1 \text{ !!!}$$

  Una contradicción.

Contenido

Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal

El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO

Macros
Uso de macros

Semántica
Estados y descripciones instantáneas
Computación de un programa

Definición formal de las macros

Apéndice: Conjuntos y funciones

Diagonalización (II)

- La contradicción que acabamos de obtener nos muestra que la noción intuitiva de algoritmo que hemos presentado es insostenible.
- Un análisis detenido de nuestros razonamientos permite establecer la causa de la contradicción anterior: **no es posible exigir que todo algoritmo pare siempre (sobre cualquier dato de entrada) y, a la vez, que exista una descripción uniforme y efectiva de los algoritmos**.
- Por tanto, modificaremos nuestra definición de algoritmo para que incluya la posibilidad de que un algoritmo **no proporcione ningún resultado sobre ciertos datos de entrada**.

“Definición” de algoritmo (segundo intento)

Definición.

“Un algoritmo A es una sucesión finita de instrucciones $I_1, \ldots, I_n$ que al recibir un dato de entrada x, especifican:

1. *Qué operación (efectiva) ha de realizarse.*
2. *Cuál es la siguiente instrucción que debe llevarse a cabo. Si tal instrucción no existe, decimos que el algoritmo para y debe entonces proporcionar un dato de salida o resultado que denotamos por $A(x)$.”*

- Notación: Dado x escribiremos $A(x)\downarrow$ para indicar que el algoritmo A para sobre el dato x y escribiremos $A(x)\uparrow$ para expresar que no para.

Funciones computables

- Las precisiones (1), (2) y (3) que añadimos a la primera "definición" siguen teniendo validez, con la única salvedad de que para cada algoritmo pueden existir datos de entrada para los que dicho algoritmo no proporciona ningún resultado. De este modo, nuestros algoritmos no siempre definirán funciones totales.
- Escribiremos $f : \mathbb{N}^k - \to \mathbb{N}$ y diremos que f es una función parcial de $\mathbb{N}^k$ en $\mathbb{N}$ si existe $D \subseteq \mathbb{N}^k$ tal que $f : D \to \mathbb{N}$.
  - D es el **dominio** de f, $\mathrm{dom}(f) = D$.
- Una función parcial f es **computable** si existe un algoritmo A tal que para todo x,

$$x \in \mathrm{dom}(f) \iff A(x)\downarrow$$

  y para cada $x \in \mathrm{dom}(f)$, $f(x) = A(x)$.
- Es usual escribir $f(x)\downarrow$ para indicar $x \in \mathrm{dom}(f)$.

Enumeración efectiva

- Utilizando un método efectivo para enumerar los algoritmos tal y como hicimos anteriormente podemos probar ahora la existencia de **funciones universales**.

Teorema.
Existe un función computable universal, es decir, existe $U : \mathbb{N}^2 - \rightarrow \mathbb{N}$ *tal que para cada función parcial computable* $f : \mathbb{N} - \rightarrow \mathbb{N}$ *existe* $e \in \mathbb{N}$ *tal que:*

$$\forall x \in \mathbb{N},\ U(x, e)\downarrow \iff f(x)\downarrow$$

y además, $\forall x \in dom(f),\ U(x, e) = f(x)$.

- Como consecuencia de este resultado podemos enumerar las funciones computables en una sucesión:

$$f_0, f_1, \ldots, f_n, \ldots$$

siendo, para todo $x, n \in \mathbb{N},\ f_n(x) = U(x, n)$.

Algoritmos y funciones computables

Contenido

Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal

El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO

Macros
Uso de macros

Semántica
Estados y descripciones instantáneas
Computación de un programa

Definición formal de las macros

Apéndice: Conjuntos y funciones

Diagonalización

- Ahora el método diagonal empleado anteriormente no produce una contradicción. Por el contrario, dicho argumento prueba la existencia de **funciones no computables**.
- En efecto, la función $g : \mathbb{N} \to \mathbb{N}$ definida por

$$g(x) = \begin{cases} U(x,x) + 1 & \text{si } U(x,x)\downarrow \\ 0 & \text{si } U(x,x)\uparrow \end{cases}$$

NO es computable.

Problemas decidibles

- Dos de los problemas básicos que podemos resolver mediante un algoritmo son:
  (P1) Calcular valores de una función.
  (P2) Determinar si un elemento pertenece a un conjunto.
- Identificando cada conjunto $B \subseteq \mathbb{N}$ con su función característica,

$$\mathcal{C}_B(x) = \begin{cases} 1 & \text{si } x \in B \\ 0 & \text{si } x \notin B \end{cases}$$

  el problema (P2) pueden considerarse un caso particular de (P1).

Contenido
Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal
El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO
Macros
Uso de macros
Semántica
Estados y descripciones instantáneas
Computación de un programa
Definición formal de las macros
Apéndice: Conjuntos y funciones

Problemas decidibles (II)

- Si $\mathcal{C}_B$ es computable entonces tenemos un algoritmo que "decide" la pertenencia a B, es decir, dado x devuelve como valor SI cuando $x \in B$ y devuelve NO en caso contrario. Esto motiva la siguiente definición:

Definición.

- *Un conjunto $C \subseteq \mathbb{N}^k$ es decidible si existe un algoritmo A tal que:*

$$\begin{array}{rcl} \vec{x} \in C & \iff & A(\vec{x})\downarrow \ \text{ y } A(\vec{x}) = 1 \\ \vec{x} \notin C & \iff & A(\vec{x})\downarrow \ \text{ y } A(\vec{x}) = 0 \end{array}$$

- *Diremos que $C \subseteq \mathbb{N}^k$ es semidecidible si existe un algoritmo A tal que:*

$$\vec{x} \in C \iff A(\vec{x})\downarrow$$

Problemas decidibles (III)

- Utilizando la función universal que hemos presentado podemos establecer informalmente algunas relaciones básicas entre conjuntos decidibles y semidecidibles de $\mathbb{N}$.
  1. Un conjunto $A \subseteq \mathbb{N}$ es decidible si y sólo si A y $\mathbb{N} - A$ son semidecidibles.
  2. Dado $B \subseteq \mathbb{N}$, $B \neq \emptyset$, son equivalentes:
     (a) B es semidecidible.
     (b) Existe $f : \mathbb{N} \to \mathbb{N}$ computable tal que $B = \text{rang}(f)$
- Aplicando del método diagonal podemos probar la existencia de problemas **indecidibles**:
  1. $K_0 = \{(x, y) : \; U(x, y)\downarrow\}$ es semidecidible, pero no decidible.
  2. El conjunto $Tot = \{n : \; A_n \text{ es total}\}$ no es semidecidible.

Contenido
Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal
El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO
Macros
Uso de macros
Semántica
Estados y descripciones instantáneas
Computación de un programa
Definición formal de las macros
Apéndice: Conjuntos y funciones

Sintaxis del lenguaje GOTO

Es un modelo secuencial, determinista con conjunto de datos $\mathbb{N}$.

1. **Variables (tipo natural):**
   - **Variables de entrada**: $\mathtt{X}_1 (= \mathtt{X}), \dots \mathtt{X}_n, \dots$
   - **Variable de salida**: $\mathtt{Y}$.
   - **Variables de trabajo** (auxiliares): $\mathtt{Z}_1 (= \mathtt{Z}), \dots, \mathtt{Z}_n, \dots$.
2. **Etiquetas:** $A_1, B_1, C_1, D_1, E_1, A_2, B_2, C_2, D_2, E_2, \dots$
   - Notación: $A = A_1$, $B = B_1$, $C = C_1$, $D = D_1$, $E = E_1$.
3. **Instrucciones:** Para cada variable $\mathtt{V}$ y cada etiqueta $\mathtt{L}$:
   - **Incremento:** $\mathtt{V} \leftarrow \mathtt{V} + 1$
     - Interpretación: Incrementa en 1 el valor de $\mathtt{V}$.
   - **Decremento:** $\mathtt{V} \leftarrow \mathtt{V} - 1$
     - Interpretación: Si el valor de $\mathtt{V}$ es 0, no tiene efecto. En otro caso, su valor decrece en 1.
   - **Condicional:** $\mathtt{IF}\ \mathtt{V} \neq 0\ \mathtt{GOTO}\ \mathtt{L}$
     - Interpretación: Si el valor de $\mathtt{V}$ es 0, no tiene efecto. Si no es 0, salta el control a la instrucción de etiqueta $\mathtt{L}$. Si no existe, sin efecto.
   - **Skip:** SKIP, $\mathtt{V} \leftarrow \mathtt{V}$
     - Interpretación: No tiene efecto.

Programa GOTO

- Estas instrucciones pueden ser **etiquetadas** con cualquier etiqueta K. Por ejemplo,

$$[K]\ \mathtt{V} \leftarrow \mathtt{V} + \mathtt{1}$$
$$[K]\ \mathtt{IF}\ \mathtt{V} \neq 0\ \mathtt{GOTO}\ \mathtt{L}$$

- Un **programa** GOTO (o G–**programa**) es una sucesión finita de instrucciones (posiblemente etiquetadas a la izquierda) del lenguaje GOTO,

$$\mathtt{I}_1, \ldots, \mathtt{I}_n$$

  donde la última instrucción **no** es $\mathtt{Y} \leftarrow \mathtt{Y}$.
- Si $n = 0$ lo denominaremos *programa vacío*.
- Al conjunto de los G–programas lo denotaremos por $\mathtt{GOTO}_\mathtt{P}$

Contenido
Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal
El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO
Macros
Uso de macros
Semántica
Estados y descripciones instantáneas
Computación de un programa
Definición formal de las macros
Apéndice: Conjuntos y funciones

Ejecución en GOTO

La ejecución de p sobre la entrada $X_1 = a_1, X_2 = a_2, \ldots$

- Consiste en realizar la tarea descrita por la primera instrucción sobre esos valores, ejecutar la siguiente sobre el resultado obtenido, y así sucesivamente,
  - Si en algún momento **no** existe siguiente instrucción, entonces **para** y devuelve el valor almacenado en Y.
  - Si la ejecución es infinita (siempre existe siguiente instrucción a ejecutar), no devuelve ningún resultado, **no para**.
- **Convenio:** Las variables auxiliares y la variable de salida, Y, toman valor inicial 0.
- **NOTA:** Al ejecutar IF $V \neq 0$ GOTO L cuando V tiene valor no nulo, existen dos casos especiales:
  - No existe instrucción etiquetada con [L]. El programa para. Por comodidad usaremos la etiqueta [E].
  - Existen varias instrucciones etiquetadas con [L]. La siguiente instrucción a ejecutar es la **primera** con dicha etiqueta.

Función calculada por un programa

- **Definición:** Dado $\mathtt{p} \in \mathtt{GOTO_P}$ y $n \in \mathbb{N}$, **la función** *n*–**aria que calcula** $\mathtt{p}$

$$[\![\mathtt{p}]\!]^{(n)} : \mathbb{N}^n - \to \mathbb{N}$$

se define como

$$[\![\mathtt{p}]\!]^{(n)}(a_1, \dots, a_n) = \begin{cases} b & \textit{si al ejecutar } \mathtt{p} \\ & \textit{sobre } \mathtt{X}_1 = a_1, \dots, \mathtt{X}_n = a_n, \mathtt{X}_{n+1} = 0, \dots \\ & \textit{para y finalmente } \mathtt{Y} = b. \\ & \\ \uparrow & \textit{en otro caso} \end{cases}$$

(Si no hay confusión, omitiremos el superíndice)

- **Definición** $f : \mathbb{N}^k - \to \mathbb{N}$ es $\mathtt{GOTO}$–computable (o $\mathtt{G}$–computable) si existe $\mathtt{p} \in \mathtt{GOTO_P}$ tal que $f = [\![\mathtt{p}]\!]^{(k)}$
- $\mathtt{GCOMP}^{(k)}$ es el conjunto de las funciones k–arias y $\mathtt{G}$–computables

Algoritmos y funciones computables

Contenido

Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal

El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO

Macros
Uso de macros

Semántica
Estados y descripciones instantáneas
Computación de un programa

Definición formal de las macros

Apéndice: Conjuntos y funciones

Ejemplos

- Sea p el programa

$$\begin{array}{ll} [\texttt{A}] & \texttt{X} \leftarrow \texttt{X} - 1 \\ & \texttt{Y} \leftarrow \texttt{Y} + 1 \\ & \texttt{IF X} \neq 0 \texttt{ GOTO A} \end{array}$$

- Entonces

$$[\![\mathrm{p}]\!]^{(1)}(a) = \begin{cases} 1 & \text{si a=0} \\ a & \text{e.o.c.} \end{cases}$$

- $[\![\mathrm{p}]\!]^{(2)}(a, b) = [\![\mathrm{p}]\!]^{(1)}(a)$. Es decir, este programa *copia* valor de X en Y, si el valor no es 0.
- ¿Qué funciones calcula el programa vacío?

Algoritmos y funciones computables

Contenido

Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal

El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO

Macros
Uso de macros

Semántica
Estados y descripciones instantáneas
Computación de un programa

Definición formal de las macros

Apéndice: Conjuntos y funciones

Ejemplos (II): identidad y salto incondicional

- ¿Programa que calcule la función identidad $Id_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$?
-
$$\begin{array}{ll} & \texttt{IF X} \neq 0 \texttt{ GOTO B} \\ & \texttt{Z} \leftarrow \texttt{Z} + 1 \\ & \texttt{IF Z} \neq 0 \texttt{ GOTO E} \\ [\texttt{B}] & \texttt{X} \leftarrow \texttt{X} - 1 \\ & \texttt{Y} \leftarrow \texttt{Y} + 1 \\ & \texttt{IF X} \neq 0 \texttt{ GOTO B} \end{array}$$
- El *bloque* $\left\{ \begin{array}{l} \texttt{Z} \leftarrow \texttt{Z} + 1 \\ \texttt{IF Z} \neq 0 \texttt{ GOTO L} \end{array} \right.$
  es "equivalente" a *ir a la instrucción de etiqueta* `L`.
  Simplificación:
$$\texttt{GOTO L}$$
  es lo que llamaremos una **macro**

Macros

Existen bloques de instrucciones que podemos considerar como "nuevas instrucciones" ya que podemos utilizarlos en cualquier programa para realizar una cierta subtarea. Veamos algunos ejemplos de esto:

- El bloque de instrucciones:

$$\begin{array}{l} \mathtt{Z} \leftarrow \mathtt{Z} + 1 \\ \mathtt{IF}\ \mathtt{Z} \neq 0\ \mathtt{GOTO}\ \mathtt{L} \end{array}$$

  nos permite realizar un <u>salto incondicional</u> a la instrucción etiquetada por L (o terminar la ejecución del programa).

- Podemos poner a cero una variable V mediante el siguiente bloque de instrucciones:

$$\begin{array}{ll} [\mathtt{L}] & \mathtt{V} \leftarrow \mathtt{V} - 1 \\ & \mathtt{IF}\ \mathtt{V} \neq 0\ \mathtt{GOTO}\ \mathtt{L} \end{array}$$

Contenido
Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal
El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO
Macros
Uso de macros
Semántica
Estados y descripciones instantáneas
Computación de un programa
Definición formal de las macros
Apéndice: Conjuntos y funciones

Macros (II)

- Es útil introducir abreviaturas para poder usar cómodamente los bloques anteriores en la descripción de un programa.
- Dichas abreviaturas se denominan **macros** y para poder usarlas debemos especificar con precisión la forma en que dichas abreviaturas se reemplazan por verdadero código.

$$\underbrace{\texttt{GOTO L}}_{\text{MACRO}} \implies \underbrace{\left.\begin{array}{l} \texttt{Z}_k \leftarrow \texttt{Z}_k + 1 \\ \texttt{IF } \texttt{Z}_k \neq 0 \texttt{ GOTO L} \end{array}\right\}}_{\text{EXPANSIÓN}}$$

$$\underbrace{\texttt{V} \leftarrow 0}_{\text{MACRO}} \implies \underbrace{\left\{\begin{array}{ll} [\texttt{K}] & \texttt{V} \leftarrow \texttt{V} - 1 \\ & \texttt{IF V} \neq 0 \texttt{ GOTO K} \end{array}\right.}_{\text{EXPANSIÓN}}$$

- En el primer caso $\texttt{Z}_k$ debe ser una variable que no aparece en el programa en que realizamos la expansión. En el segundo caso K es una etiqueta que no aparece en dicho programa.

Contenido
Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal
El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO
Macros
Uso de macros
Semántica
Estados y descripciones instantáneas
Computación de un programa
Definición formal de las macros
Apéndice: Conjuntos y funciones

Ejemplos (III): Anular una variable

- Cómo asignar el valor 0 a una variable; es decir,

$$\mathtt{V} \leftarrow \mathtt{0}$$

- Una macroexpansión posible es

$$\begin{array}{ll} [\mathtt{A}] & \mathtt{V} \leftarrow \mathtt{V} - \mathtt{1} \\ & \mathtt{IF}\ \mathtt{V} \neq \mathtt{0}\ \mathtt{GOTO}\ \mathtt{A} \end{array}$$

con [A] una nueva etiqueta.

Ejemplos (IV): Macro para $\mathtt{V} \leftarrow \mathtt{V}'$

- El programa para $Id_{\mathbb{N}}$ no vale: *destruye* el valor original de $\mathtt{X}$
- Debemos combinar los ejemplos anteriores:

$$\begin{array}{ll} [\mathtt{A}] & \mathtt{IF\ X \neq 0\ GOTO\ B} \\ & \mathtt{GOTO\ C} \\ [\mathtt{B}] & \mathtt{X} \leftarrow \mathtt{X} - 1 \\ & \mathtt{Y} \leftarrow \mathtt{Y} + 1 \\ & \mathtt{Z} \leftarrow \mathtt{Z} + 1 \\ & \mathtt{GOTO\ A} \end{array} \left.\vphantom{\begin{array}{l}a\\a\\a\\a\\a\\a\end{array}}\right\} \text{Copia el valor de } \mathtt{X} \text{ en } \mathtt{Y}, \mathtt{Z}$$

$$\begin{array}{ll} [\mathtt{C}] & \mathtt{IF\ Z \neq 0\ GOTO\ D} \\ & \mathtt{GOTO\ E} \\ [\mathtt{D}] & \mathtt{Z} \leftarrow \mathtt{Z} - 1 \\ & \mathtt{X} \leftarrow \mathtt{X} + 1 \\ & \mathtt{GOTO\ C} \end{array} \left.\vphantom{\begin{array}{l}a\\a\\a\\a\\a\end{array}}\right\} \text{Restaura el valor de } \mathtt{X}$$

  transforma el estado $\mathtt{X}_1 = a$ en el estado $\mathtt{X}_1 = a, \mathtt{Y} = a, \mathtt{Z} = 0$.
- **Macroexpansión** de $V \leftarrow V'$: sustituir $\mathtt{X}/\mathtt{V}'$ y $Y/\mathtt{V}$, con $\mathtt{Z}$ **nueva y** $\mathtt{V}$ **inicializada a 0 antes de ejecutar la macro**.

Contenido
Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal
El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO
Macros
Uso de macros
Semántica
Estados y descripciones instantáneas
Computación de un programa
Definición formal de las macros
Apéndice: Conjuntos y funciones

Macroexpansión para $\mathtt{V} \leftarrow \mathtt{V}'$

$$
\begin{array}{ll}
 & \mathtt{V} \leftarrow 0 \\
[\mathtt{A}] & \mathtt{IF}\ \mathtt{V}' \neq 0\ \mathtt{GOTO}\ \mathtt{B} \\
 & \mathtt{GOTO}\ \mathtt{C} \\
[\mathtt{B}] & \mathtt{V}' \leftarrow \mathtt{V}' - 1 \\
 & \mathtt{V} \leftarrow \mathtt{V} + 1 \\
 & \mathtt{Z} \leftarrow \mathtt{Z} + 1 \\
 & \mathtt{GOTO}\ \mathtt{A} \\
[\mathtt{C}] & \mathtt{IF}\ \mathtt{Z} \neq 0\ \mathtt{GOTO}\ \mathtt{D} \\
 & \mathtt{GOTO}\ \mathtt{E} \\
[\mathtt{D}] & \mathtt{Z} \leftarrow \mathtt{Z} - 1 \\
 & \mathtt{V}' \leftarrow \mathtt{V}' + 1 \\
 & \mathtt{GOTO}\ \mathtt{C}
\end{array}
$$

Copia el valor de V' en V, Z

Restaura el valor de V'

La macroexpansión se puede usar en un programa p si:

- Las etiquetas utilizadas no aparecen en p .
- La etiqueta E de salida de la macroexpansión se debe cambiar por nueva etiqueta **que dirige a la siguiente instrucción a ejecutar en** p **.**
- Como Z no ocurre en p, no es necesario añadir $\mathtt{Z} \leftarrow 0$.

La suma es $\mathtt{G}$–computable

- $+ = f$, donde $f = [\![\mathrm{p}_+]\!]^{(2)}$, donde p_+ es el $\mathtt{G}$–programa

$$\begin{array}{ll} & \mathtt{Y} \leftarrow \mathtt{X}_1 \\ & \mathtt{Z} \leftarrow \mathtt{X}_2 \\ [\mathtt{B}] & \mathtt{IF\ Z} \neq \mathtt{0\ GOTO\ A} \\ & \mathtt{GOTO\ E} \\ [\mathtt{A}] & \mathtt{Z} \leftarrow \mathtt{Z} - \mathtt{1} \\ & \mathtt{Y} \leftarrow \mathtt{Y} + \mathtt{1} \\ & \mathtt{GOTO\ B} \end{array}$$

- Por tanto, usaremos como macro $\mathtt{V} \leftarrow \mathtt{V}_1 + \mathtt{V}_2$
- El programa es correcto:

$$(\forall x, y \in \mathbb{N})([\![\mathrm{p}_+]\!]^{(2)}(x, y) = x + y)$$

(por inducción en b).

La función producto

- $f(x, y) = x \cdot y$, con $f = [\![\mathrm{p}_\bullet]\!]^{(2)}$, donde $p_\bullet$ es

p	con macroexp. de +	
$\mathtt{Z_2 \leftarrow X_2}$ [B] $\mathtt{IF\ Z_2 \neq 0\ GOTO\ A}$ $\mathtt{GOTO\ E}$ [A] $\mathtt{Z_2 \leftarrow Z_2 - 1}$ $\mathtt{Z_1 \leftarrow X_1} + Y$ $\mathtt{Y \leftarrow Z_1}$ $\mathtt{GOTO\ B}$	$\mathtt{Z_2 \leftarrow X_2}$ [B] $\mathtt{IF\ Z \neq 0\ GOTO\ A}$ $\mathtt{GOTO\ E}$ [A] $\mathtt{Z_2 \leftarrow Z_2 - 1}$ $\mathtt{Z_1 \leftarrow X_1}$ $\mathtt{Z_3 \leftarrow Y}$ [$\mathtt{B_2}$] $\mathtt{IF\ Z_3 \neq 0\ GOTO\ A_2}$ $\mathtt{GOTO\ E_2}$ [$\mathtt{A_2}$] $\mathtt{Z_3 \leftarrow Z_3 - 1}$ $\mathtt{Z_1 \leftarrow Z_1 + 1}$ $\mathtt{GOTO\ B_2}$ [$\mathtt{E_2}$] $\mathtt{Y \leftarrow Z_1}$ $\mathtt{GOTO\ B}$	Macroexp. de $\mathtt{Z_1 \leftarrow X_1 + Y}$

- Se demuestra (por inducción en y) que

$$(\forall x, y \in \mathbb{N})([\![\mathrm{p}_\bullet]\!]^{(2)}(x, y) = x \cdot y)$$

Sustracción parcial

$$f : \mathbb{N}^2 - \to \mathbb{N}$$

$$f(x, y) = z \iff x = y + z$$

- Es $\mathtt{G}$–computable, pues $f = [\![\mathtt{p}]\!]^{(2)}$, donde $\mathtt{p}$ es

```
     Y ← X1
     Z ← X2
[C]  IF Z ≠ 0 GOTO A
     GOTO E
[A]  IF Y ≠ 0 GOTO B
     GOTO A
[B]  Y ← Y − 1
     Z ← Z − 1
     GOTO C
```

(Es una función $\mathtt{G}$–computable parcial)

La función vacía es $\mathtt{G}$-computable

- $f_\emptyset : \mathbb{N} - \to \mathbb{N}$ tal que $f(n)\uparrow$ para todo $n \in \mathbb{N}$
- Computable por el programa $\mathtt{p}_\emptyset$:

$$\begin{array}{ll} [\mathtt{A}] & \mathtt{Z} \leftarrow \mathtt{Z} + 1 \\ & \mathtt{IF}\ \mathtt{Z} \neq 0\ \mathtt{GOTO}\ \mathtt{A} \end{array}$$

- Si en un programa $\mathtt{p}$ no aparece la variable $\mathtt{Y}$, entonces $[\![p]\!]$ es la función constante e igual a 0.

Estados

- Notación:
  - $\mathit{var}(\mathtt{p})$ es el conjunto de las variables que ocurren en $\mathtt{p}$.
  - $|\mathtt{p}|$ es la longitud del programa.
- **Definición:** Un **estado** σ de un programa $\mathtt{p} \in \mathtt{GOTO}_{\mathtt{P}}$, es un conjunto de ecuaciones del tipo

$$\mathtt{V} = m$$

donde

  - $\mathtt{V}$ es una variable y $m \in \mathbb{N}$.
  - Ninguna variable aparece en dos ecuaciones de σ.
  - Para cada variable de $\mathtt{p}$ existe una ecuación en σ.

Denotaremos por $\mathit{var}(\sigma)$ al conjunto de variables que ocurren en σ.

Si σ es un estado de $\mathtt{p}$, entonces $\mathit{var}(\mathtt{p}) \subseteq \mathit{var}(\sigma)$.

Ejemplo

Consideremos el programa $\mathtt{p}$

$$
\begin{array}{ll}
[\mathtt{A}] & \mathtt{IF\ X} \neq 0 \ \mathtt{GOTO\ B} \\
 & \mathtt{Z} \leftarrow \mathtt{Z} + 1 \\
 & \mathtt{IF\ Z} \neq 0 \ \mathtt{GOTO\ E} \\
[\mathtt{B}] & \mathtt{X} \leftarrow \mathtt{X} - 1 \\
 & Y \leftarrow Y + 1 \\
 & \mathtt{IF\ X} \neq 0 \ \mathtt{GOTO\ A}
\end{array}
$$

- $\sigma = \{\mathtt{Z} = 6, \mathtt{X} = 3, \mathtt{Y} = 10, \mathtt{X}_2 = 1\}$ es un estado de $\mathtt{p}$.
- $\sigma' = \{\mathtt{Z} = 5, \mathtt{X} = 2\}$ no lo es, pues $var(\mathtt{p}) \not\subseteq var(\sigma)$.
- $\sigma'' = \{\mathtt{Z} = 12, \mathtt{X} = 1, \mathtt{Y} = 3, \mathtt{Z} = 1\}$ no es un estado de $\mathtt{p}$.

Descripción instantánea

Una **descripción instantánea** (d.i.) de un programa $\mathtt{p}$ es un par $s = (i, \sigma)$ donde

- $1 \leq i \leq |\mathtt{p}| + 1$ y σ es un estado de $\mathtt{p}$.

Una d.i. $s = (i, \sigma)$ para $\mathtt{p}$ se dice

1. **Inicial** si $i = 1$
2. **Terminal** si $i = |\mathtt{p}| + 1$.

Definición: Sea $\mathtt{p}$ un $\mathtt{G}$-programa.

- Sea σ un estado para $\mathtt{p}$, y sea $\mathtt{V} \in var(\sigma)$. El valor de $\mathtt{V}$ en σ es el único $m \in \mathbb{N}$ tal que la ecuación $\mathtt{V} = m$ ocurre en σ, que lo notaremos por $\sigma(\mathtt{V})$.
  - Si $\mathtt{V} \notin var(\sigma)$ entonces, definimos el valor de $\mathtt{V}$ en σ como $\sigma(\mathtt{V}) = 0$.
- De manera análoga se define el valor en una d.i. $s = (i, \sigma)$, sin más que tomar $s(\mathtt{V}) = \sigma(\mathtt{V})$.

D.i. sucesora

$\mathtt{p} \equiv \mathtt{I}_1, \ldots \mathtt{I}_n$ y $s = (j, \sigma)$ no terminal. La **d.i. sucesora** o siguiente de s en $\mathtt{p}$ es $s' = (j', \tau)$, donde

CASO 1: $\mathtt{I}_j \equiv \mathtt{V} \leftarrow \mathtt{V} + 1$. Si $s(\mathtt{V}) = m$, entonces

- $j' = j + 1$.
- τ se obtiene de σ cambiando $\mathtt{V} = m$ por $\mathtt{V} = m + 1$.

CASO 2: $\mathtt{I}_j \equiv \mathtt{V} \leftarrow \mathtt{V} - 1$. Si $s(\mathtt{V}) = m$, entonces

- $j' = j + 1$.
- τ se obtiene de σ sustituyendo $\mathtt{V} = m$ por $\mathtt{V} = m - 1$ si $m > 0$, o $\sigma' = \sigma$ en otro caso.

CASO 3: $\mathtt{I}_i \equiv \mathtt{IF}\ \mathtt{V} \neq 0\ \mathtt{GOTO}\ \mathtt{L}$. En este caso $\tau = \sigma$, y

3.A: $s(\mathtt{V}) = 0$. En este caso, $j' = j + 1$

3.B: $s(\mathtt{V}) \neq 0$. En este caso,

- j es el menor índice k tal que $\mathtt{I}_k$ está etiquetado con $\mathtt{L}$, si existe tal k. Si no existiese, $j' = |\mathtt{p}| + 1$.

CASO 4: $\mathtt{I}_i \equiv \mathtt{SKIP}$. Entonces $j' = j + 1$ y $\sigma' = \sigma$.

Escribiremos $s \vdash_{\mathtt{p}} s'$ si s' es la d.i. siguiente a s en $\mathtt{p}$.

Ejemplo

Consideremos el programa $\mathtt{p}$

$$\begin{array}{ll}
[\mathtt{A}] & \mathtt{IF\ X} \neq 0\ \mathtt{GOTO\ B} \\
 & \mathtt{Z} \leftarrow \mathtt{Z} + 1 \\
 & \mathtt{IF\ Z} \neq 0\ \mathtt{GOTO\ E} \\
[\mathtt{B}] & \mathtt{X} \leftarrow \mathtt{X} - 1 \\
 & Y \leftarrow Y + 1 \\
 & \mathtt{IF\ X} \neq 0\ \mathtt{GOTO\ A}
\end{array}$$

y el estado $\sigma = \{\mathtt{X} = 40, \mathtt{Z} = 7, \mathtt{Y} = 6\}$

- $(1, \sigma) \vdash_{\mathtt{p}} (4, \sigma)$
- $(5, \sigma) \vdash_{\mathtt{p}} (6, \{\mathtt{X} = 40, \mathtt{Z} = 7, \mathtt{Y} = 7\})$
- $(3, \sigma) \vdash_{\mathtt{p}} (7, \sigma)$, que es terminal.

Computación de un programa

Sea $\mathrm{p} \in \mathtt{GOTO}_{\mathrm{P}}$. Una **computación** de p a partir de una d.i. s es una sucesión de d.i.

$$s = s_1, \ldots, s_n, \ldots$$

finita o no, tal que:

1. $s_1 = (1, \sigma)$ donde σ es un estado de p .
2. Para todo i, $s_i \vdash_{\mathrm{p}} s_{i+1}$.
3. Si la sucesión es finita, entonces la última d.i. es terminal.

- Si la sucesión es finita, diremos que la computación de p a partir de s para, y lo notaremos $\mathrm{p}(s) \downarrow$.
- En caso contrario, diremos que diverge (o no para), y lo notaremos $\mathrm{p}(s) \uparrow$.

Propiedades fundamentales

Lema: La semántica definida en esta sección para el lenguaje `GOTO` verifica las siguientes propiedades, para todo $\mathtt{p} \in \mathtt{GOTO}_{\mathtt{P}}$:

1. **Ausencia de bloqueo:** Para toda d.i. s no terminal para $\mathtt{p}$ existe s' d.i. tal que $s \vdash_p s'$.
2. **Determinismo:** Para toda d.i. s para $\mathtt{p}$ existe una única computación de $\mathtt{p}$ partiendo se s.

D.i. inicial

Sean $n \geq 1$ y $\vec{r} = r_1, \ldots r_n \in \mathbb{N}$, y σ el siguiente estado:

- $\sigma(X_i) = r_i$ $(1 \leq i \leq n)$, y las restantes variables de entrada que ocurren en $\mathtt{p}$ se le asigna el valor 0.
- $\sigma(Z_i) = 0$ para todo i, y $\sigma(Y) = 0$.

Se denomina **descripción instantánea inicial** para $r_1, \ldots r_n$ a la d.i $s_1 = (1, \sigma)$.

Notación para d.i. iniciales: $(1, (a_1, \ldots, a_n))$ para hacer referencia la d.i. cuyo estado σ es

$$\begin{cases} \sigma(\mathtt{X}_j) = a_j & (1 \leq j \leq n) \\ \sigma(\mathtt{V}) = 0 & \mathtt{V} \in var(\sigma) \setminus \{\mathtt{X}_1, \ldots, \mathtt{X}_n\} \end{cases}$$

Función calculada por un programa

La **función de aridad n calculada por** $\mathtt{p}$ es la función

$$[\![\mathtt{p}]\!]^{(n)}(\vec{r}) = \begin{cases} s_k(\mathtt{Y}) & \text{si existe una computación finita } s_1, \ldots s_k \\ & \text{ de } \mathtt{p} \text{ a partir de } (1, (a_1, \ldots, a_n)) \\ \uparrow & \text{en otro caso} \end{cases}$$

Ejemplos: Si $\mathtt{p} \equiv \mathtt{p}_+$ es el programa que calcula la suma, entonces

- $[\![\mathtt{p}]\!]^{(2)}(x, y) = x + y$, y
- $[\![\mathtt{p}]\!]^{(1)}(x) = x$.

Algoritmos y funciones computables
Contenido
Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal
El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO
Macros
Uso de macros
Semántica
Estados y descripciones instantáneas
Computación de un programa
Definición formal de las macros
Apéndice: Conjuntos y funciones

Ejemplo

$[A]$	$IF\ X \neq 0\ GOTO\ B$
	$Z \longleftarrow Z + 1$
	$IF\ Z \neq 0\ GOTO\ E$
$[B]$	$X \longleftarrow X - 1$
	$Y \longleftarrow Y + 1$
	$IF\ X \neq 0\ GOTO\ A$

$$
\begin{aligned}
s_1 &= (1, \{X = 3,\ \ Y = 0,\ \ Z = 0\})\\
s_2 &= (4, \{X = 3,\ \ Y = 0,\ \ Z = 0\})\\
s_3 &= (5, \{X = 2,\ \ Y = 0,\ \ Z = 0\})\\
s_4 &= (6, \{X = 2,\ \ Y = 1,\ \ Z = 0\})\\
s_5 &= (1, \{X = 2,\ \ Y = 1,\ \ Z = 0\})\\
s_6 &= (4, \{X = 2,\ \ Y = 1,\ \ Z = 0\})\\
s_7 &= (5, \{X = 1,\ \ Y = 1,\ \ Z = 0\})\\
s_8 &= (6, \{X = 1,\ \ Y = 2,\ \ Z = 0\})\\
s_9 &= (1, \{X = 1,\ \ Y = 2,\ \ Z = 0\})\\
s_{10} &= (4, \{X = 1,\ \ Y = 2,\ \ Z = 0\})\\
s_{11} &= (5, \{X = 0,\ \ Y = 2,\ \ Z = 0\})\\
s_{12} &= (6, \{X = 0,\ \ Y = 3,\ \ Z = 0\})\\
s_{13} &= (7, \{X = 0,\ \ Y = 3,\ \ Z = 0\})
\end{aligned}
$$

Algoritmos y funciones computables

Contenido

Problemas decidibles e indecidibles

Primera definición informal

Segunda definición informal

El lenguaje GOTO

Sintaxis de GOTO

Semántica intuitiva de los programas GOTO

Macros

Uso de macros

Semántica

Estados y descripciones instantáneas

Computación de un programa

Definición formal de las macros

Apéndice: Conjuntos y funciones

Observaciones

1. Cada programa $\texttt{p} \in \texttt{GOTO}_\texttt{P}$ determina una función $\texttt{G}$-computable para cada aridad.
2. Si las variables de entrada de $\texttt{p}$ están contenidas en $\{\texttt{X}_1, \ldots, \texttt{X}_m\}$, entonces para todo $k > m$ se verifica que

$$[\![\texttt{p}]\!]^{(k)}(r_1, \ldots r_k) = [\![\texttt{p}]\!]^{(m)}(r_1, \ldots, r_m)$$

   (La igualdad es entre funciones parciales)
3. La función $f_\emptyset : \mathbb{N}^m - \rightarrow \mathbb{N}$ *siempre indefinido* es computable por

$$\begin{array}{ll} [\texttt{A}] & \texttt{Z} \leftarrow \texttt{Z} + 1 \\ & \texttt{IF Z} \neq 0 \texttt{ GOTO A} \end{array}$$

4. Dada una función computable n–aria, existe un programa $\texttt{GOTO}$ tal que $[\![\texttt{p}]\!]^{(n)}$ y con sólo variables de entrada $\texttt{X}_1, \ldots, \texttt{X}_n$.

Algoritmos y funciones computables

Contenido

Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal

El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO

Macros
Uso de macros

Semántica
Estados y descripciones instantáneas
Computación de un programa

Definición formal de las macros

Apéndice: Conjuntos y funciones

Apéndice: Definición formal de las macros

Sean $f \in \mathtt{GCOMP}^{(n)}$ y $\mathtt{p} \in \mathtt{GOTO}_{\mathtt{P}}$ tales que:

- $[\![\mathtt{p}]\!]^{(n)} = f$.
- $var(\mathtt{p}) \subseteq \{\mathtt{Y}, \mathtt{X}_1, \ldots, \mathtt{X}_n, \mathtt{Z}_1, \ldots, \mathtt{Z}_k\}$
- Las etiquetas que ocurren en $\mathtt{p}$ están contenidas en $\{\mathtt{E}, \mathtt{A}_1, \ldots, \mathtt{A}_l\}$.
- La única instrucción del tipo '$\mathtt{IF\ V} \neq 0\ \mathtt{GOTO\ L}$' que es de salida es con $\mathtt{L} = \mathtt{E}$.

Notación: $\mathtt{p} \equiv \mathtt{p}(\mathtt{Y}, , \mathtt{X}_1, \ldots, \mathtt{X}_n, \mathtt{Z}_1, \ldots, \mathtt{Z}_k; \mathtt{E}, \mathtt{A}_1, \ldots \mathtt{A}_l)$
Dado $m \in \mathbb{N}$,

$$\mathtt{q}_m \equiv \mathtt{p}(\mathtt{Z}_m, , \mathtt{Z}_{m+1}, \ldots, \mathtt{Z}_{m+n}, \mathtt{Z}_{m+n+1}, \ldots, \mathtt{Z}_{m+n+k}; \mathtt{E}_m, \mathtt{A}_{m+1}, \ldots \mathtt{A}_{m+l})$$

Definición formal de las macros

Definimos '$\mathtt{W} \leftarrow f(\mathtt{V}_1, \ldots, \mathtt{V}_n)$' a través de la macroexpansión

$$
\begin{array}{ll}
 & \mathtt{Z}_m \leftarrow 0 \\
 & \mathtt{Z}_{m+1} \leftarrow \mathtt{V}_1 \\
 & \vdots \\
 & \mathtt{Z}_{m+n} \leftarrow \mathtt{V}_n \\
 & \mathtt{Z}_{m+n+1} \leftarrow 0 \\
 & \vdots \\
 & \mathtt{Z}_{m+n+k} \leftarrow 0 \\
 & \mathtt{q}_m \\
{[\mathtt{E}_m]} & \mathtt{W} \leftarrow \mathtt{Z}_m
\end{array}
$$

de tal manera que si dicha macro se utiliza en un programa $\mathtt{p}_0$, se tomará m suficientemente grande para que $var(\mathtt{q}_m) \cap var(\mathtt{p}_0) = \emptyset$.

Observaciones

1. El bloque de asignaciones a 0 es necesario, pues puede ocurrir que esta macro forme parte de un bucle.
2. Si $f(a_1, \dots a_n) \uparrow$ entonces al ejecutar la macro

$$\mathtt{W} \leftarrow f(\mathtt{V}_1, \dots, \mathtt{V}_n)$$

sobre un estado con esos valores, la macro no para.
3. Si tenemos un programa que calcula una función f, entonces f se puede usar como macro
4. **Condicionales con macros (predicados)**

$$\mathtt{IF}\ P(\mathtt{V}_1, \dots \mathtt{V}_n)\ \mathtt{GOTO\ L}$$

con $P(\vec{x})$ predicado computable (función booleana sobre $\mathbb{N}$) por algún G–programa. La macroexpansión es

$$\begin{array}{l} \mathtt{Z} \leftarrow P(\mathtt{V}_1, \dots, \mathtt{V}_n) \\ \mathtt{IF\ Z} \neq 0\ \mathtt{GOTO\ L} \end{array}$$

Ejemplo y propiedad

Ejemplo:

$$\texttt{IF V} = 0 \ \texttt{GOTO L}$$

se puede utilizar, pues el predicado

$$P(v) = \begin{cases} 1 & \text{si } v = 0 \\ 0 & \text{si } v \neq 0 \end{cases}$$

es computable por el programa

$$\begin{array}{l} \texttt{IF X} \neq 0 \ \texttt{GOTO E} \\ \texttt{Y} \leftarrow \texttt{Y} + 1 \end{array}$$

Proposición: Toda función computable por programas usando macros para funciones G–computables es G–computable (sin macros).

Apéndice: Conjuntos y funciones

- Escribimos $x \in C$ para expresar que x es un elemento de C.
- Además, $x \notin C$ expresa que x NO es un elemento de C.
- Dada una propiedad Θ, denotamos por $\{t : \Theta(t)\}$ al conjunto formado por aquellos elementos que verifican la propiedad Θ.
- El conjunto vacío de denota por $\emptyset$.

Igualdad entre conjuntos

- Dos conjuntos A y B son iguales si tienen los mismos elementos, es decir, si

$$\text{Para todo } x,\ x \in A \Longleftrightarrow x \in B$$

- Por tanto, dada una propiedad, Θ, si $A = \{x : \ \Theta(x)\}$ entonces, para todo x,

$$x \in A \Longleftrightarrow \Theta(x)$$

Producto cartesiano

- **Producto cartesiano**:
  $A \times B = \{(u, v) : \; u \in A \text{ y } v \in B\}$,
  donde (u, v) denota al **par ordenado** formado por u y v.
- La propiedad característica de los pares ordenados es

$$(a, b) = (c, d) \iff a = c \text{ y } b = d$$

- En general, para cada $n \geq 3$, podemos considerar **n-tuplas**, $(a_1, a_2, ..., a_n)$, cuya propiedad característica es:

$$(a_1, \dots, a_n) = (b_1, \dots, b_n) \iff a_1 = b_1, a_2 = b_2, \dots, a_n = b_n$$

- $A_1 \times ... \times A_n = \{(x_1, ..., x_n) : x_1 \in A_1 \wedge ... \wedge x_n \in A_n\}$.
- Si $A_1 = ... = A_n = A$, escribiremos $A^n = A \times A \times \overset{(n}{\dots} \times A$. Además, la n–tupla $(x_1, \dots, x_n)$ se denotará por $\vec{x}$.

Funciones

- Una **función** f es un conjunto de **pares ordenados**, tal que
$$(a, b) \in f \text{ y } (a, c) \in f \Longrightarrow b = c$$
- El **dominio** de una función f es el conjunto:
$$dom(f) = \{x : \exists y\, ((x, y) \in f)\}$$
- El **rango** de f es el conjunto:
$$rang(f) = \{y : \ \exists x\, ((x, y) \in f)\}$$
- Si f es una función, para cada $a \in dom(f)$ existe un único $b \in rang(f)$ tal que $(a, b) \in f$. Por ello usaremos la notación habitual, $f(a) = b$ para expresar que $(a, b) \in f$.
- Ejemplo: Sea A un conjunto. La función **identidad** de A es la función $Id_A : A \to A$, dada por $Id_A(x) = x$.

Contenido
Problemas decidibles e indecidibles
Primera definición informal
Segunda definición informal
El lenguaje GOTO
Sintaxis de GOTO
Semántica intuitiva de los programas GOTO
Macros
Uso de macros
Semántica
Estados y descripciones instantáneas
Computación de un programa
Definición formal de las macros
Apéndice: Conjuntos y funciones

Igualdad entre funciones

- Notación: Escribiremos $f(x)\downarrow$ para expresar que $x \in dom(f)$ y utilizaremos la notación $f(x)\uparrow$ para expresar que $x \notin dom(f)$.
(En este último caso se dice que f no está definida en x).
- Si f y g son funciones, entonces $f = g$ si y sólo si tienen los mismos elementos (pares ordenados). Por tanto,

$$f = g \iff \begin{cases} dom(f) = dom(g) \\ y \\ \text{para todo } x \in dom(f),\ f(x) = g(x) \end{cases}$$

$$\iff \text{Para todo } x, \begin{cases} f(x)\downarrow \iff g(x)\downarrow \\ y \\ f(x) = g(x) \end{cases}$$

Tipos de funciones

Sean A y B dos conjuntos y f una función.

- f es una **aplicación** de A en B (o **función total** de A en B) y escribiremos $f : A \longrightarrow B$, si

$$dom(f) = A \quad \text{y} \quad rang(f) \subseteq B$$

- f es una **función parcial** de A en B, $f : A - \to B$, si

$$dom(f) \subseteq A \quad \text{y} \quad rang(f) \subseteq B$$

- Una función $f : A - \to B$ es:
  - **inyectiva** si para todo $x_1, x_2 \in dom(f)$,

$$f(x_1) = f(x_2) \Longrightarrow x_1 = x_2$$

  - **sobreyectiva** (o suprayectiva o exhaustiva) si $rang(f) = B$, es decir,

$$\forall y \in B \exists x \in A (f(x) = y)$$

  - **biyectiva** si es total, inyectiva y sobreyectiva.