Lenguajes de Programación 2023-1
Nota de clase 2: Herramientas Matemáticas
Introducción

Javier Enríquez Mendoza

16 de agosto de 2022

Para el estudio formal de los Lenguajes de Programación es necesario definirlos de tal forma que sea sencillo abstraer sus características y propiedades de con el fin de razonar sobre éstas. La forma de hacerlo es mediante metalenguajes de formalización matemática, estos metalenguajes se van a elegir dependiendo de las propiedades que se quieran estudiar sobre el lenguaje de programación.

En esta nota se estudiaran diferentes herramientas y técnicas que nos ayudarán a definir lenguajes de programación formalmente y a estudiar sus diferentes características así como probar propiedades de estos.

1 Objetos y juicios

Como primera instancia es necesario indicar que un elemento cumple cierta propiedad, para esto se utilizan **objetos** y **juicios** que se tratan de una notación muy simple similar a la pertenencia de conjuntos ($n \in \mathbb{N}$) para indicar que un individuo cumple una propiedad.

Definición 1.1 (Objeto)**.** Son entidades primitivas y previamente definidas. Algunos ejemplos pueden ser: números, tipos, caracteres, listas, valores, entre otros.

Definición 1.2 (Juicios)**.** Un juicio es una afirmación acerca de un objeto particular. Los juicios pueden expresar: la pertenencia a un conjunto, que un objeto cumple cierta propiedad, que dos o mas objetos pertenecen a una relación, que un valor tiene cierto tipo, entre otros.

En general se utiliza notación postfija para indicar que un objeto x cumple una propiedad P y se escribe como $x\ \mathsf{P}$. Si el juicio involucra una relación binaria, entonces se usa notación infija por ejemplo $x = y$ para indicar la relación de igualdad.

Ejemplos. Algunos ejemplos de juicios son:

$n\ \mathbb{N}$	n es un número natural
$n = m + k$	n es la suma de m con k
t asa	t es un árbol sintáctico
$3 + 4 * 5$ ExpAr	$3 + 4 * 5$ es una expresión aritmética válida
0.1234 float	0.1234 es un valor flotante
"aba" pal	"aba" es una cadena palíndroma
T type	T es un tipo
$e : T$	La expresión e tiene tipo T
$e \to e'$	la expresión e se reduce a e'
$e \Downarrow v$	la expresión e se evalua al valor v.

Se puede observar que los juicios son predicados en lógica escritos en otra notación.

Nuestro propósito es usar juicios para definir conjuntos, pero por sí solos pueden resultar aburridos y hasta cierto punto inútiles. La mayoría de los conjuntos que nos interesan modelar tienen un número infinito de elementos y enunciarlos todos mediante juicios resultaría imposible. Podemos definir sistemáticamente conjuntos y sus propiedades usando **reglas de inferencia**.

2 Reglas de Inferencia

Las **reglas de inferencia** son un esquema o forma lógica que consta de una serie de premisas (o hipótesis) y una conclusión. Tanto las premisas como la conclusión se modelan como **juicios** que se combinan para obtener nuevos juicios mas complejos. El esquema se ve de la siguiente forma:

$$\frac{J_1 \quad J_2 \quad \dots \quad J_n}{J}$$

En donde los juicios $J_1, J_2, \dots, J_n$ son las premisas y el juicio J es la conclusión. Si no hay premisas entonces la regla es un axioma. Esta regla se lee como: *Para que se cumpla J es necesario que se cumplan $J_1, J_2, \dots, J_n$.*

Ejemplo 2.1 (Números Naturales)**.** En este ejemplo se define el conjunto de los números naturales utilizando reglas de inferencia.

$$\frac{}{0\ \mathsf{Nat}}\ \text{zero} \qquad \frac{n\ \mathsf{Nat}}{s(n)\ \mathsf{Nat}}\ \text{suc}$$

Estas reglas indican que 0 es un número natural y esta regla es un axioma. Y que $s(n)$ es un número natural siempre y cuando n también lo sea.

Ejemplo 2.2 (Funciones de paridad)**.** Para este ejemplo vamos a definir las funciones que nos dicen si un número natural es par o impar mediante reglas de inferencia.

$$\frac{}{0\ \mathsf{par}}\ \text{parB} \qquad \frac{n\ \mathsf{impar}}{s(n)\ \mathsf{par}}\ \text{parR}$$

$$\frac{}{s(0)\ \mathsf{impar}}\ \text{imparB} \qquad \frac{n\ \mathsf{par}}{s(n)\ \mathsf{impar}}\ \text{imparR}$$

Los axiomas indican que 0 es par y $s(n)$ impar, mientras que las otras reglas definen la (im)paridad de un número respecto a (im)paridad de su predecesor.

Mediante el uso de reglas de inferencia se definió al conjunto de números naturales y las funciones de paridad sobre estos. La pregunta ahora sería como podemos usar estas definiciones para saber si un objeto es en realidad un número natural, o para saber si es un número par o impar.

Las reglas de inferencia funcionan en dos formas, podemos usarlas para definir una propiedad y también para probar que esta propiedad se cumple.

Para probar un juicio J tenemos que encontrar la regla que tenga a J como conclusión si esta regla es una axioma entonces ya terminamos con la prueba. Pero si la regla tiene premisas, entonces tenemos que construir una prueba para cada premisa aplicando la misma estrategia recursivamente. Este proceso se conoce como una derivación y el resultado se muestra en forma estructural como un árbol, al que llamamos **árbol de derivación**.

Ejemplo 2.3. Como ejemplo de derivación probaremos que el $s(s(0))$ es un número natural construyendo el árbol de derivación para el juicio $s(s(0))\ \mathsf{Nat}$. Buscamos en nuestras reglas la que tenga como conclusión el juicio que queremos probar. Se puede observar que la regla que corresponde es suc, por lo que ahora basta probar el juicio $s(0)\ \mathsf{Nat}$ y repetimos el proceso hasta llegar a un axioma, lo cual resulta en el siguiente árbol:

$$\dfrac{\dfrac{\dfrac{}{0\ \mathsf{Nat}}\ \text{zero}}{s(0)\ \mathsf{Nat}}\ \text{suc}}{s(s(0))\ \mathsf{Nat}}\ \text{suc}$$

En la derivación anterior se llego al axioma $0\ \mathsf{Nat}$, y con eso se cerraron todas las ramas, que en este caso solo se trata de una, por lo que podemos concluir que el juicio $s(s(0))\ \mathsf{Nat}$ es válido.

Observación. Las reglas de inferencia son puramente sintácticas, es decir, no es posible probar la validez del juicio $2\ \mathsf{Nat}$ a pesar de que sabemos que 2 es equivalente a $s(s(0))$ ya que ese conocimiento no se puede aplicar en la derivación pues no existe una regla que nos permita hacerlo. Las derivaciones simplemente nos permiten manipular términos mecánicamente de acuerdo a las reglas.

2.1 Reglas derivables y admisibles

No todas las reglas de inferencia mediante las cuales se define un conjunto o un lenguaje son necesarias, en algunos casos se pude prescindir de algunas reglas sin alterar los elementos definidos. Las reglas derivables y admisibles son reglas prescindibles en la definición de un conjunto.

Definición 2.1 (Regla derivable)**.** Una regla de inferencia $\mathcal{R}$ es derivable respecto a un conjunto de reglas $\Delta = \{\mathcal{R}_1, ..., \mathcal{R}_k\}$, si y sólo si puede obtenerse usando las reglas primitivas de Δ.

Consideremos el siguiente conjunto de reglas para definir cadenas de paréntesis balanceados.

Definición 2.2 (Lenguaje de Dyck)**.** Reglas de inferencia que definen el lenguaje Dyck en donde todas las palabras son cadenas de paréntesis bien balanceados:

$$\frac{}{\varepsilon\ \mathsf{Dyck}}\ \text{eps}$$

$$\frac{s_1\ \mathsf{Dyck} \qquad s_2\ \mathsf{Dyck}}{s_1 s_2\ \mathsf{Dyck}}\ \text{con}$$

$$\frac{s\ \mathsf{Dyck}}{(s)\ \mathsf{Dyck}}\ \text{par}$$

Con estas reglas se pueden construir las cadenas:

$$() \qquad (()) \qquad ()() \qquad (()()) \qquad ()(())$$

entre otras.

Ahora analicemos que es lo que sucede si agregamos al conjunto de reglas anterior la siguiente regla:

$$\frac{s\ \mathsf{Dyck}}{((s))\ \mathsf{Dyck}}\ \text{dob}$$

Al agregar una nueva regla, podría pensarse que aumentó la expresividad del lenguaje. Esto es, si existe una cadena s la cual se pueda derivar s Dyck usando forzosamente la regla dob.

Esto suena muy poco probable, pues lo único que está haciendo la nueva regla es agregar dos pares de paréntesis a una cadena que ya estaba balanceada, y en el sistema original se puede lograr el mismo efecto aplicando dos veces la regla par.

$$\frac{\dfrac{s\ \mathsf{Dyck}}{(s)\ \mathsf{Dyck}}}{((s))\ \mathsf{Dyck}} \equiv \frac{s\ \mathsf{Dyck}}{((s))\ \mathsf{Dyck}}$$

Esto quiere decir que la regla dob es derivable de las reglas originales.

Definición 2.3 (Regla Admisible)**.** Una regla $\mathcal{R}$ es admisible respecto a un conjunto de reglas $\Delta = \{\mathcal{R}_1, ..., \mathcal{R}_k\}$ si y sólo si el hecho de que sus premisas sean derivables respecto a Δ implica que su conclusión también es derivable en Δ.

Ahora pensemos en agregar al conjunto de reglas que definen el lenguaje de Dyck la siguiente regla:

$$\frac{()s\ \mathsf{Dyck}}{s\ \mathsf{Dyck}}\ \text{red}$$

Ésta regla es diferente al resto de las reglas que hemos visto, pues la cadena que se tiene en la conclusión es estructuralmente más simple que la de la premisa. Esta nueva regla no agrega nuevas cadenas a Dyck, es decir, si agregamos la regla red podemos derivar las mismas cadenas que si no estuviera.

Adicionalmente, la nueva regla no es derivable ya que no es posible obtenerla a partir de las reglas originales. Veamos como se derivaría la premisa de la nueva regla con el conjunto original.

$$\dfrac{\dfrac{\dfrac{}{\varepsilon\ \mathsf{Dyck}}}{()\ \mathsf{Dyck}} \qquad \dfrac{\vdots}{s\ \mathsf{Dyck}}}{()s\ \mathsf{Dyck}}$$

Podemos observar que para derivar $()s$ Dyck es necesario derivar s Dyck, es decir, para derivar la premisa de red se debe derivar su conclusión. Por lo tanto, la regla red es admisible.

Observación. Una regla derivable siempre es admisible, mas al revés no siempre es cierto.

Las reglas de inferencia pueden utilizarse en diferentes ámbitos, por ejemplo en lógica se usan para definir consecuencias lógicas. En este curso las emplearemos para modelar distintos aspectos de lenguajes de programación, en general estas características se definen de forma inductiva. En la siguiente sección se habla sobre definiciones inductivas así como el uso de las reglas de inferencia para este fin.

3 Definiciones Inductivas

Se dice que una regla de inferencia es inductiva si al menos una de sus premisas es un juicio de la misma forma que el juicio de su conclusión, es decir, si se refieren a la misma propiedad. Por ejemplo la regla suc de la definición de los números naturales

$$\frac{n\ \mathsf{Nat}}{s(n)\ \mathsf{Nat}}\ \text{suc}$$

Se trata de una regla inductiva pues el juicio n Nat es de la misma forma que la conclusión $s(n)$ Nat, ambos hablan de la propiedad de pertenencia al conjunto de naturales.

Definición 3.1 (Definición Inductiva)**.** Un conjunto finito de reglas de inferencia $\Delta = \{\mathcal{R}_1, ..., \mathcal{R}_k\}$ es una definición inductiva si y sólo si al menos una $\mathcal{R}_i \in \Delta$ es una regla inductiva.

Hasta ahora hemos visto ya un par de ejemplos de definiciones inductivas, los números naturales y el lenguaje de Dyck, veamos un par de ejemplos mas:

Ejemplo 3.1 (El lenguaje Scratches)**.** En este ejemplo se da una definición inductiva para el lenguaje Scratches también conocido como números de cavernícolas, mediante el siguiente conjunto de reglas de inferencia:

$$\frac{}{|\ \mathsf{Sc}}\ \text{one}$$

$$\frac{s\ \mathsf{Sc}}{s|\ \mathsf{Sc}}\ \text{next}$$

Se trata de una definición inductiva ya que la regla next es inductiva, pues la premisa y la conclusión se definen como juicios de la misma forma.

Con estas reglas se pueden construir las cadenas:

$$|\qquad ||\qquad |||\qquad ||||\qquad |||||\qquad \cdots$$

Ejemplo 3.2 (Listas)**.** Se define el conjunto de listas de elementos de A como una definición inductiva mediante el siguiente conjunto de reglas de inferencia:

$$\frac{}{nil\ \mathsf{List_A}}\ \text{nil}$$

$$\frac{x\ A \qquad xs\ \mathsf{List_A}}{cons(x, xs)\ \mathsf{List_A}}\ \text{cons}$$

Este conjunto de reglas es una definición inductiva pues la regla cons es inductiva.

Las reglas anterior representan inductivamente, la definición recursiva clásica de las listas de elementos de A, que dice:

- nil es una lista de elementos de A.
- Si x es un elemento de A y xs es una lista de elementos de A, entonces $cons(x, xs)$ es una lista de elementos de A.
- Son todas.

4 Inducción Estructural

La deducción natural por si sola no es un herramienta muy poderosa para probar propiedades ya que, como vimos en secciones anteriores, se trata de un mecanismo puramente sintáctico. Por esta razón es necesario contar con otra técnica que nos permita demostrar propiedades sobre los conjuntos o relaciones definidas con las reglas de inferencia, para esto usaremos **inducción**.

En las clases de álgebra superior se estudia la inducción matemática o inducción sobre naturales, este es un caso particular de un principio de inducción mas general, la **Inducción Estructural**.

Definición 4.1 (Principio de Inducción Estructural)**.** Sea A una propiedad o relación definida inductivamente por un conjunto de reglas de inferencia Δ, para probar que una propiedad $\mathcal{P}$ es válida para todos los elementos de A basta probar que para cada regla $\mathcal{R} \in \Delta$ de la forma

$$\frac{x_1\ A \quad \cdots \quad x_n\ A}{x\ A}$$

se cumple que:

$$\text{Si } \mathcal{P} \text{ es válida para } x_1, \cdots, x_n \text{ entonces } \mathcal{P} \text{ es válida para } x.$$

Otra forma de verlo es probar que cada regla:

$$\frac{x_1\ \mathcal{P} \quad \cdots \quad x_n\ \mathcal{P}}{x\ \mathcal{P}}$$

Obtenidas al sustituir A por $\mathcal{P}$ en las reglas de Δ, es una fórmula válida.

Y el anterior es el principio de inducción estructural para A.

Con cada definición inductiva se genera su propio principio de inducción estructural.

Ejemplo 4.1 (Principio de inducción estructural para el lenguaje de Dyck)**.** Retomando el ejemplo del lenguaje de Dyck, definido inductivamente como sigue:

$$\frac{}{\varepsilon\ \mathsf{Dyck}}\ \text{eps}$$

$$\frac{s_1\ \mathsf{Dyck} \quad s_2\ \mathsf{Dyck}}{s_1 s_2\ \mathsf{Dyck}}\ \text{con}$$

$$\frac{s\ \mathsf{Dyck}}{(s)\ \mathsf{Dyck}}\ \text{par}$$

Si queremos probar que una propiedad $\mathcal{P}$ se cumple para todas las cadenas en Dyck mediante inducción estructural, basta probar que $s\ \mathsf{Dyck}$ implica $s\ \mathcal{P}$, y como sabemos que existe una derivación para $s\ \mathsf{Dyck}$, entonces solo tenemos que probar:

- $\varepsilon\ \mathcal{P}$
- Si $s_1\ \mathcal{P}$ y $s_2\ \mathcal{P}$ entonces $s_1 s_2\ \mathcal{P}$

- Si $s\ \mathcal{P}$ entonces $(s)\ \mathcal{P}$

Es decir, se tiene que probar la validez del siguiente conjunto de reglas:

$$\frac{}{\varepsilon\ \mathcal{P}}$$

$$\frac{s_1\ \mathcal{P} \qquad s_2\ \mathcal{P}}{s_1 s_2\ \mathcal{P}}$$

$$\frac{s\ \mathcal{P}}{(s)\ \mathcal{P}}$$

Para ejemplificar el uso de la inducción estructural probaremos la siguiente proposición:

Proposición 4.2. Si s Dyck entonces s tiene el mismo número de paréntesis izquierdos y derechos. Es decir, $\texttt{izq}\ s = \texttt{der}\ s$ en donde `izq` y `der` son las funciones que calculan el número de paréntesis izquierdos y derechos respectivamente.

Demostración. Inducción estructural sobre el lenguaje de Dyck

Caso Base (eps)

$$\texttt{izq}\ \varepsilon = 0 = \texttt{der}\ \varepsilon$$

Por lo que ε tiene el mismo número de paréntesis izquierdos y derechos.

Paso Inductivo (con)

Por demostrar: Si $\texttt{izq}\ s_1 = \texttt{der}\ s_1$ y $\texttt{izq}\ s_2 = \texttt{der}\ s_2$ entonces $\texttt{izq}\ s_1 s_2 = \texttt{der}\ s_1 s_2$

- Hipótesis de Inducción 1: $\texttt{izq}\ s_1 = \texttt{der}\ s_1$
- Hipótesis de Inducción 2: $\texttt{izq}\ s_2 = \texttt{der}\ s_2$

$$\begin{aligned}
& \texttt{izq}\ s_1 s_2 = \texttt{der}\ s_1 s_2 \\
= & \quad \{\text{Propiedad de } izq\} \\
& \texttt{izq}\ s_1 + \texttt{izq}\ s_2 = \texttt{der}\ s_1 s_2 \\
= & \quad \{\text{Hipótesis de Inducción 1}\} \\
& \texttt{der}\ s_1 + \texttt{izq}\ s_2 = \texttt{der}\ s_1 s_2 \\
= & \quad \{\text{Hipótesis de Inducción 2}\} \\
& \texttt{der}\ s_1 + \texttt{der}\ s_2 = \texttt{der}\ s_1 s_2 \\
= & \quad \{\text{Propiedad de } der\} \\
& \texttt{der}\ s_1 s_2 = \texttt{der}\ s_1 s_2
\end{aligned}$$

Paso Inductivo (par)

Por demostrar: Si $\texttt{izq}\ s = \texttt{der}\ s$ entonces $\texttt{izq}\ (s) = \texttt{der}\ (s)$

- Hipótesis de Inducción: $\texttt{izq}\ s = \texttt{der}\ s$

$$
\begin{aligned}
& \texttt{izq}\ (s) = \texttt{der}\ (s) \\
= \quad & \{\text{Definición de } izq\} \\
& 1 + \texttt{izq}\ s) = \texttt{der}\ (s) \\
= \quad & \{\text{Propiedad de } izq\} \\
& 1 + \texttt{izq}\ s = \texttt{der}\ (s) \\
= \quad & \{\text{Hipótesis de Inducción}\} \\
& 1 + \texttt{der}\ s = \texttt{der}\ (s) \\
= \quad & \{\text{Definición de } der\} \\
& 1 + \texttt{der}\ s = 1 + \texttt{der}\ (s \\
= \quad & \{\text{Propiedad de } der\} \\
& 1 + \texttt{der}\ s = 1 + \texttt{der}\ s
\end{aligned}
$$

□

5 Sistemas de Transición

Un sistema de transición es un mecanismo que nos ayuda a modelar el comportamiento de la ejecución de un programa mediante un dispositivo de cómputo abstracto. Es decir, que se simula la ejecución de un programa mediante un modelo teórico de computo.

Definición 5.1 (Sistema de Transición)**.** Un sistema de transición es un modelo de computo abstracto con los siguientes componentes:

- Un conjunto Γ de estados.
- Una función de transición que depende principalmente del estado actual y la instrucción del programa que se está ejecutando. Esta función define el comportamiento del sistema.
- Un conjunto finito I de estados iniciales. Tal que $I \subseteq \Gamma$

Existen sistemas de transición con un mayor número de componentes, sin embargo estos son los mínimos necesarios.

Se puede utilizar reglas de inferencia para definir sistemas de transición de la siguiente forma:

- Se definen los estados mediante juicios s estado.

$$\frac{}{s \text{ estado}}$$

- Los estados iniciales se modelan con juicios s inicial

$$\frac{s \text{ estado}}{s \text{ inicial}}$$

  En donde la única premisa es que s también sea un estado.
- Para definir las transiciones se usan juicios $s_1 \to s_2$ que indica que del estado s_1 se transita al estado s_2

$$\frac{s_1 \text{ estado} \qquad s_2 \text{ estado}}{s_1 \to s_2}$$

Ejemplo 5.1 (Sistema de transición)**.** Se define un sistema de transición para el siguiente programa:

```
x := 0;
while (x < 2) do
    x := x + 1;
end;
x := 25
```

El sistema de transición es el siguiente:

- Estados: $\Gamma = \{S_0, S_1, S_2, S_F\}$
- Estados Iniciales: $I = \{S_0\}$
- Función de transición: la función está definida por los cambios en el valor de la variable `x`. Y se define con el siguiente conjunto de transiciones $\{S_0 \to S_1, S_1 \to S_2, S_2 \to S_F\}$

También se define el sistema como un conjunto de reglas de inferencia.

Estados:

$$\frac{}{S_0 \text{ estado}} \qquad \frac{}{S_1 \text{ estado}} \qquad \frac{}{S_2 \text{ estado}} \qquad \frac{}{S_F \text{ estado}}$$

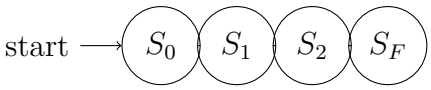
Estados iniciales:

$$\frac{S_0 \text{ estado}}{S_0 \text{ inicial}}$$

Función de transición:

$$\frac{S_0 \text{ estado} \quad S_1 \text{ estado}}{S_0 \to S_1} \qquad \frac{S_1 \text{ estado} \quad S_2 \text{ estado}}{S_1 \to S_2} \qquad \frac{S_2 \text{ estado} \quad S_F \text{ estado}}{S_2 \to S_F}$$

Un sistema de transición puede representar de forma gráfica, de manera similar a como se hace con los autómatas.



Todas las herramientas matemáticas estudiadas en esta nota de clase nos serán de utilidad a lo largo del curso, para poder formalizar la estructura y el comportamiento de los lenguajes de programación que estudiaremos así como los programas escritos en ellos.

Referencias

[1] Miranda Perea F., González Huesca L., Nota de Clase del curso de Lenguajes de Programación, Facultad de Ciencias UNAM, Semestre 2021-1.

[2] Ramírez Pulido K., Soto Romero M., Enríquez Mendoza J., Nota de Clase del curso de Lenguajes de Programación, Facultad de Ciencias UNAM, Semestre 2021-2

[3] Keller G., O'Connor-Davis L., Class Notes from the course Concepts of programming language

design, Department of Information and Computing Sciences, Utrecht University, The Netherlands, Fall 2020.

[4] Fiore M., Class Notes from the course Discrete Mathematics II, Department of Computer Science and Technology, University of Cambridge, England, Fall 2013.

[5] Harper R., Practical Foundations for Programming Languages. Working draft, 2010.

[6] Mitchell J., Foundations for Programming Languages. MIT Press 1996.

[7] Krishnamurthi S., Programming Languages Application and Interpretation; Version 26.04.2007.