2.2. Recursión e inducción

En la sección anterior comenzamos a discutir técnicas fundamentales de demostración inspiradas en la deducción natural. En esta sección nos enfocaremos en un tipo de demostración que se aplica a objetos construidos de manera recursiva. Dichos objetos aparecen de manera frecuente en matemáticas discretas y en ciencias de la computación.

Partiremos formalizando la noción de estructura recursiva, luego veremos cómo definir funciones sobre esos objetos siguiendo la misma estructura. Finalmente estudiaremos la técnica de inducción estructural, una técnica de demostración diseñada precisamente para objetos recursivos.

2.2.1. Definiciones recursivas

Una definición recursiva tiene dos ingredientes: los objetos base y reglas que explican cómo combinar objetos ya construidos para obtener otros nuevos. Precisaremos esta idea en la siguiente definición.

Definición (Producción)**.** Sea $\mathcal{U}$ un dominio de objetos. Supongamos que tenemos:

- $B \subseteq \mathcal{U}$ *objetos base*,
- $\mathcal{F}$ *reglas de producción* donde cada $f \in \mathcal{F}$ es una función $f : \mathcal{U}^k \to \mathcal{U}$ de alguna aridad $k > 0$.

Diremos que $u \in \mathcal{U}$ se *construye* a partir de B y $\mathcal{F}$ si existe una secuencia (finita) $u_1, \ldots, u_m = u$ tal que, para todo $i \in \{1, \ldots, m\}$,

- $u_i \in B$, (caso base)
- o bien $u_i = f(u_{j_1}, \ldots, u_{j_k})$, para alguna regla $f \in \mathcal{F}$ de aridad k e índices $j_1, \ldots, j_k < i$. (caso recursivo)

A la secuencia $u_1, \ldots, u_m$ se le llama *producción* de u. El conjunto de todos los objetos que se construyen a partir de B y $\mathcal{F}$ se llama el *conjunto generado* por B y $\mathcal{F}$.

Ejemplo. Antes de considerar estructuras más complejas, revisemos varias definiciones recursivas conocidas. En cada caso, el punto importante no es solamente reconocer el conjunto, sino identificar sus objetos base y sus reglas de producción.

Números naturales. El conjunto $\mathbb{N}$ se genera tomando 0 como objeto base y la función $n \mapsto n+1$ como regla de producción. Así, por ejemplo, $0, 1, 2, 3$ es una producción de 3.

Potencias de 2. El conjunto $\texttt{Pot} = \{2^n : n \in \mathbb{N}\}$ se genera a partir del objeto base 1 y de la regla $n \mapsto 2n$. La secuencia $1, 2, 4, 8$ es una producción de 8.

Strings binarios y strings generales. El conjunto $\{0, 1\}^*$ de strings binarios tiene como objeto base el string vacío ϵ y usa las reglas $w \mapsto 0w$ y $w \mapsto 1w$. Por ejemplo, una producción de 101 es $\epsilon, 1, 01, 101$. Más generalmente, si Σ es un alfabeto finito, el conjunto Σ^* de strings sobre Σ se genera a partir de ϵ mediante una regla $w \mapsto \sigma w$ por cada símbolo $\sigma \in \Sigma$.

Paréntesis bien balanceados. Un paréntesis bien balanceado es un string de paréntesis que cumple las siguientes condiciones:

- el número de paréntesis que abren es igual al número de paréntesis que cierran, y
- en cada prefijo del string, el número de paréntesis que abren es mayor o igual al número de paréntesis que cierran.

El conjunto `BB` de strings de paréntesis bien balanceados se genera a partir de ϵ con las reglas $w \mapsto (w)$ y $(w, z) \mapsto wz$. Por ejemplo, una producción de ()() es $\epsilon, (), ()()$. Notemos que un objeto puede tener varias producciones distintas; por ejemplo, dos producciones de ()(())((())) son $\epsilon, (), (()), ()(()), ((())), ()(())((()))$ y $\epsilon, (), (()), ((())), (())((())), ()(())((()))$.

Strings binarios palíndromos. Un string es un *palíndromo* si se lee igual de izquierda a derecha y de derecha a izquierda. Describamos el conjunto de strings binarios palíndromos mediante una definición recursiva. Los objetos base son ϵ, 0 y 1; las reglas $w \mapsto 0w0$ y $w \mapsto 1w1$ agregan el mismo símbolo en ambos extremos.

Fórmulas bien formadas. Las fórmulas bien formadas, que aparecieron en la sección anterior, se generaron recursivamente a partir de las variables y las constantes $\bot$ y $\top$, y los conectivos $\wedge$, $\vee$, $\neg$ y $\to$.

Usualmente B y $\mathcal{F}$ estarán definidos de manera implicita. En lugar de nombrar B y $\mathcal{F}$, escribiremos oraciones como "$0 \in \mathbb{N}$" y "si $n \in \mathbb{N}$, entonces $n + 1 \in \mathbb{N}$". Cada oración sigue identificando uno de los dos ingredientes de la definición: un objeto base o una forma permitida de combinar objetos anteriores.

Veamos dos ejemplos de definiciones recursivas un poco más sofisticadas: fórmulas de Horn y árboles 1–2.

Definición (Fórmulas de Horn)**.**

- $\bot$ y $\top$ son fórmulas de Horn.
- Si p es una variable, p es fórmula de Horn.
- Si H_1 y H_2 son fórmulas de Horn, $(H_1 \wedge H_2)$ también es una fórmula de Horn.
- Si H es una fórmula de Horn y p una variable, $(p \to H)$ es fórmula de Horn.

Ejemplo. Por ejemplo, la fórmula $((p \to q) \wedge (r \to (s \to t)))$ es de Horn, como podemos ver en la siguiente producción:

$$q, (p \to q), t, (s \to t), (r \to (s \to t)), ((p \to q) \wedge (r \to (s \to t))).$$

Definición (Árboles 1–2)**.** Sea $\mathcal{T}_{1-2}$ el conjunto de *árboles 1–2* definido por las siguientes reglas:

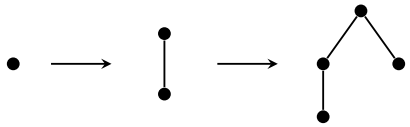
- $\bullet \in \mathcal{T}_{1-2}$.
- Si $T_1, T_2 \in \mathcal{T}_{1-2}$, entonces $\bullet(T_1) \in \mathcal{T}_{1-2}$ y $\bullet(T_1, T_2) \in \mathcal{T}_{1-2}$.

Aquí $\bullet$ representa una vértice, $\bullet(T_1)$ un árbol cuya raíz tiene un hijo y $\bullet(T_1, T_2)$ un árbol cuya raíz tiene dos hijos. Ningún otro objeto pertenece a $\mathcal{T}_{1-2}$.

Ejemplo. Por ejemplo, el árbol $\bullet(\bullet(\bullet), \bullet)$ tiene como producción

$$\bullet, \bullet(\bullet), \bullet(\bullet(\bullet), \bullet).$$

Gráficamente, se representa como



2.2.2. Funciones en objetos recursivos

Como los objetos recursivos se construyen a partir de objetos base y reglas de producción, podemos definir funciones sobre ellos siguiendo la misma estructura. Para esto, bastará con definir la función en los objetos base y explicar cómo calcularla a través de cada regla de producción.

Definición (Definición recursiva de una función)**.** Sea S un conjunto de objetos construidos a partir de objetos básicos B y reglas de producción $\mathcal{F}$.
Una función $g\colon S \to X$ se *define recursivamente* especificando $g(b)$ para $b \in B$ y para cada regla de producción cómo obtener $g(f(s_1, \ldots, s_k))$ a partir de $g(s_1), \ldots, g(s_k)$.

Notemos que si un objeto admite varias producciones, para que la función quede bien definida debemos asegurarnos de que el valor de la función no dependa de la producción utilizada.

Ejemplo. Veamos algunos ejemplos de funciones definidas recursivamente sobre los objetos que ya hemos considerado.

Paridad. La función $\texttt{par}\colon \mathbb{N} \to \{\text{par}, \text{impar}\}$ se define mediante $\texttt{par}(0) = \text{par}$ y

$$\texttt{par}(n+1) = \begin{cases} \text{impar}, & \text{si } \texttt{par}(n) = \text{par}, \\ \text{par}, & \text{si } \texttt{par}(n) = \text{impar}. \end{cases}$$

El caso recursivo cambia la paridad obtenida para el natural anterior.

Largo de un string. Para $w \in \Sigma^*$, la función $|w|$ que denota el *largo* de w se define por $|\epsilon| = 0$ y $|\sigma w| = |w| + 1$ para $\sigma \in \Sigma$.

Número de unos. Para $w \in \{0,1\}^*$, definimos $|\epsilon|_1 = 0$, $|0w|_1 = |w|_1$ y $|1w|_1 = |w|_1 + 1$. De manera análoga se define $|w|_0$, el número de ceros de w.

Valor numérico de un string binario. Interpretaremos el símbolo de más a la izquierda como el bit más significativo. La función $\texttt{val}\colon \{0,1\}^* \to \mathbb{N}$ que entrega el valor numérico de un string binario se define por

$$\texttt{val}(\epsilon) = 0, \qquad \texttt{val}(0w) = \texttt{val}(w), \qquad \texttt{val}(1w) = 2^{|w|} + \texttt{val}(w).$$

Por ejemplo, esta definición calcula $\texttt{val}(101) = 2^2 + \texttt{val}(01) = 2^2 + \texttt{val}(1) = 2^2 + 2^1 + \texttt{val}(\epsilon) = 4 + 2 + 0 = 6$.

Valuación de una fórmula de Horn. Sea v una valuación de las variables proposicionales. Su extensión $\widehat{v}$ a una valuación de fórmula de Horn se define por

$$\widehat{v}(\bot) = 0, \quad \widehat{v}(\top) = 1, \quad \widehat{v}(p) = v(p), \quad \widehat{v}(H_1 \wedge H_2) = \min\{\widehat{v}(H_1), \widehat{v}(H_2)\}$$

y $\widehat{v}(p \to H) = \max\{1 - v(p), \widehat{v}(H)\}$.

Altura de un árbol 1–2. Una *hoja* en un árbol 1-2 es un vértice sin hijos. La *altura* de un árbol es el número máximo de pasos hacia un hijo que se pueden dar comenzando en la raíz y terminando en un $\bullet$ que es una hoja. Entonces

$$h(\bullet) = 0, \qquad h(\bullet(T)) = 1 + h(T), \qquad h(\bullet(T_1, T_2)) = 1 + \max\{h(T_1), h(T_2)\}.$$

Número de hojas de un árbol 1–2. La función ℓ se define mediante

$$\ell(\bullet) = 1, \qquad \ell(\bullet(T)) = \ell(T), \qquad \ell(\bullet(T_1, T_2)) = \ell(T_1) + \ell(T_2).$$

Una raíz unaria no crea hojas nuevas, mientras que una raíz binaria reúne las hojas de sus dos subárboles.

2.2.3. Inducción estructural

La inducción estructural es una técnica de demostración que nos permitirá mostrar que una propiedad se cumple para todos los objetos de un conjunto recursivo.

Intuitivamente, la técnica consiste en probar que la propiedad se cumple para los objetos base y que se *propaga* a través de las reglas de producción. Esto nos permitirá concluir la propiedad para un objeto arbitrario de la familia mirando su producción y *propagando* la propiedad desde los objetos base hasta el objeto final.

Principio (Inducción estructural)**.** Sea S un conjunto de objetos construidos a partir de objetos básicos B y reglas de producción $\mathcal{F}$. Sea $P(s)$ una propiedad que depende de un objeto $s \in S$ tal que se cumple lo siguiente:

- $P(b)$ es verdadera para todo $b \in B$. (caso base)
- Para toda $f \in \mathcal{F}$ de aridad k y objetos $s_1, \ldots, s_k \in S$, se cumple

  $P(s_1) \wedge \cdots \wedge P(s_k) \to P(f(s_1, \ldots, s_k))$. (paso inductivo)

Entonces, $P(s)$ es verdadera para todo $s \in S$.

Notemos que una demostración por inducción estructural debe reflejar todas las oraciones de la definición recursiva: debemos verificar $P(b)$ para *cada objeto base* y, que *para cada regla de producción* f, al suponer las propiedades de sus entradas y podemos demostrar la propiedad en la salida.

Veamos algunos ejemplos de demostraciones por inducción estructural.

Ejemplo. La primera aplicación relaciona tres funciones recursivas sobre bitstrings: el largo, el número de ceros y el número de unos.

> **Proposición.** Para todo bitstring $w \in \{0,1\}^*$, tenemos que $|w|_0 + |w|_1 = |w|$.

Demostración. Procedemos por inducción estructural.
Caso base: Si $w = \epsilon$,

$$|\epsilon|_0 + |\epsilon|_1 = 0 + 0 = 0 = |\epsilon|.$$

Paso inductivo: Recordemos que hay dos reglas de producción para los bitstrings: $w \mapsto 0w$ y $w \mapsto 1w$.

- **Regla 1:** Supongamos que w es un bitstring y que la igualdad se cumple para w. Si anteponemos un cero, entonces

$$|0w|_0 + |0w|_1 = (|w|_0 + 1) + |w|_1 = |w| + 1 = |0w|.$$

- **Regla 2:** Supongamos que w es un bitstring y que la igualdad se cumple para w. Si anteponemos un uno, obtenemos

$$|1w|_0 + |1w|_1 = |w|_0 + (|w|_1 + 1) = |w| + 1 = |1w|.$$

Por inducción estructural, la igualdad vale para todo bitstring. □

Ejemplo. Continuamos ahora mostrando nuevamente algo natural: el valor numérico de un string binario es menor que $2^{|w|}$.

> **Proposición.** Para todo bitstring $w \in \{0,1\}^*$, tenemos que $0 \leq \texttt{val}(w) < 2^{|w|}$.

Demostración. Usaremos inducción estructural sobre w.
Caso base: Para $w = \epsilon$, se cumple

$$0 \leq \texttt{val}(\epsilon) = 0 < 1 = 2^{|\epsilon|}.$$

Paso inductivo: Recordemos que hay dos reglas de producción para los bitstrings: $w \mapsto 0w$ y $w \mapsto 1w$.

- **Regla 1:** Supongamos que w es un bitstring y que $0 \leq \texttt{val}(w) < 2^{|w|}$. Si anteponemos un cero,

$$0 \leq \texttt{val}(0w) = \texttt{val}(w) < 2^{|w|} < 2^{|0w|}.$$

- **Regla 2:** Supongamos que w es un bitstring y que $0 \leq \texttt{val}(w) < 2^{|w|}$. Si anteponemos un uno,
$$2^{|w|} \leq \texttt{val}(1w) = 2^{|w|} + \texttt{val}(w) < 2^{|w|} + 2^{|w|} = 2^{|1w|}.$$

Por inducción estructural, la desigualdad vale para todo bitstring. □

Ejemplo. Pasamos a una demostración que requerirá un poco más de argumentos. En particular, usaremos la identidad $|xy| = |x| + |y|$ (que se puede demostrar por inducción estructural sobre x) y algunos calculos para ver que la función `val` se comporta bien con la concatenación de strings. Algo interesante también de esta demostración es que buscamos probar algo sobre *pares* de strings, pero la inducción estructural se hace sobre uno de ellos, mientras que el otro es arbitrario y se mantiene fijo.

> **Proposición.** Para todo par de bitstrings $x, y \in \{0,1\}^*$, tenemos que $\texttt{val}(xy) = 2^{|y|}\texttt{val}(x) + \texttt{val}(y)$.

Demostración. Sea y un bitstring arbitrario y procedamos por inducción estructural sobre x.
Caso base: Si $x = \epsilon$, entonces

$$\texttt{val}(\epsilon y) = \texttt{val}(y) = 2^{|y|}\texttt{val}(\epsilon) + \texttt{val}(y).$$

Paso inductivo: Recordemos que hay dos reglas de producción para los bitstrings: $x \mapsto 0x$ y $x \mapsto 1x$.

- **Regla 1:** Supongamos la identidad para un bitstring x. Para la regla que antepone un cero,
$$\texttt{val}((0x)y) = \texttt{val}(0(xy)) = \texttt{val}(xy) = 2^{|y|}\texttt{val}(0x) + \texttt{val}(y).$$

- **Regla 2:** Supongamos la identidad para un bitstring x. Para la regla que antepone un uno, usando $|xy| = |x| + |y|$ y la hipótesis inductiva,
$$\begin{aligned}
\texttt{val}((1x)y) &= \texttt{val}(1(xy)) \\
&= 2^{|xy|} + \texttt{val}(xy) \\
&= 2^{|x|+|y|} + 2^{|y|}\texttt{val}(x) + \texttt{val}(y) \\
&= 2^{|y|}\big(2^{|x|} + \texttt{val}(x)\big) + \texttt{val}(y) \\
&= 2^{|y|}\texttt{val}(1x) + \texttt{val}(y).
\end{aligned}$$

Por inducción estructural, la identidad vale para todo bitstring x. Como y era un bitstring arbitrario, concluimos para todo par de bitstrings x, y. □

Ejemplo. Veamos ahora un ejemplo sobre una estructura recursiva más compleja: los árboles 1–2.

> **Proposición.** Para todo árbol 1–2 T, se tiene que $\ell(T) \leq 2^{h(T)}$.

Demostración. Procedemos por inducción estructural sobre T.
Caso base: Para la hoja $\bullet$,

$$\ell(\bullet) = 1 = 2^0 = 2^{h(\bullet)}.$$

Paso inductivo: Recordemos que hay dos reglas de producción recursivas para los árboles 1–2: agregar un hijo o agregar dos hijos a la raíz.

- **Regla 1:** Supongamos la propiedad para un árbol T. Para una raíz con un hijo,

$$\ell(\bullet(T)) = \ell(T) \leq 2^{h(T)} \leq 2^{1+h(T)} = 2^{h(\bullet(T))}.$$

- **Regla 2:** Supongamos la propiedad para dos árboles T_1 y T_2. Para una raíz con dos hijos, las dos hipótesis inductivas entregan

$$\begin{aligned} \ell(\bullet(T_1, T_2)) &= \ell(T_1) + \ell(T_2) \\ &\leq 2^{h(T_1)} + 2^{h(T_2)} \\ &\leq 2 \cdot 2^{\max\{h(T_1), h(T_2)\}} \\ &= 2^{h(\bullet(T_1,T_2))}. \end{aligned}$$

Por inducción estructural, la desigualdad vale para todo árbol 1–2. □

Ejemplo. Terminamos con una aplicación de inducción estructural a la lógica proposicional, que nos permitirá demostrar una propiedad clave de las fórmulas de Horn.

Para un conjunto de variables X, denotemos por v_X la valuación que hace verdadera exactamente cada variable de X.

> **Proposición.** Sean A, B conjuntos de variables y ϕ una fórmula de Horn. Si ϕ es verdadera con v_A y con v_B, entonces también es verdadera con $v_{A\cap B}$.

Demostración. Demostraremos por inducción estructural sobre ϕ que, si $\widehat{v}_A(\phi) = \widehat{v}_B(\phi) = 1$, entonces $\widehat{v}_{A\cap B}(\phi) = 1$.
Caso base: Para $\phi = \top$ la conclusión es inmediata. El caso $\phi = \bot$ también queda cubierto, por vacuidad. Si $\phi = p$ es una variable y es verdadera con v_A y v_B, entonces $p \in A$ y $p \in B$; por lo tanto, $p \in A \cap B$ y $v_{A\cap B}(p) = 1$.
Paso inductivo: Recordemos que hay dos reglas de producción recursivas para las fórmulas de Horn: agregar una conjunción o agregar una implicancia.

- **Regla 1:** Supongamos que la propiedad se cumple para H_1 y H_2. Si $\phi = H_1 \wedge H_2$ es verdadera con ambas valuaciones, cada H_i es verdadera tanto con v_A como con v_B. Las dos hipótesis inductivas muestran que H_1 y H_2 son verdaderas con $v_{A\cap B}$, de modo que su conjunción también lo es.

- **Regla 2:** Supongamos que la propiedad se cumple para H y que $\phi = p \to H$ es verdadera con v_A y v_B. Si $p \notin A \cap B$, entonces p es falsa con $v_{A\cap B}$ y por lo tanto $p \to H$ es verdadera con $v_{A\cap B}$. Si $p \in A \cap B$, entonces p es verdadero con v_A y v_B; como ambas valuaciones satisfacen $p \to H$, también satisfacen H. La hipótesis inductiva implica que H es verdadera con $v_{A\cap B}$, y por tanto $p \to H$ también lo es.

Por inducción estructural, la afirmación vale para toda fórmula de Horn ϕ. □

Ejercicios

1. *a*) Sea Σ un alfabeto. El conjunto Σ^* se genera a partir de ϵ mediante una regla $x \mapsto \sigma x$ por cada $\sigma \in \Sigma$. La concatenación y el largo de strings se definen mediante

$$\epsilon y = y, \qquad (\sigma x)y = \sigma(xy), \qquad |\epsilon| = 0, \qquad |\sigma x| = |x| + 1.$$

   Demuestre que $|xy| = |x| + |y|$ para todos $x, y \in \Sigma^*$.

   b) Sea $\texttt{BB}$ el conjunto de strings de paréntesis generado por las siguientes reglas: $\epsilon \in \texttt{BB}$; si $w \in \texttt{BB}$, entonces $(w) \in \texttt{BB}$; y si $w, z \in \texttt{BB}$, entonces $wz \in \texttt{BB}$. En este inciso puede usar, sin demostrarla, la identidad $|wz| = |w| + |z|$. Demuestre que todo string $w \in \texttt{BB}$ tiene largo par.

   c) En un árbol 1–2, llamaremos *hoja* a una aparición de $\bullet$ sin hijos. Demuestre que todo árbol 1–2 tiene al menos una hoja.

   d) El conjunto de palíndromos binarios está generado a partir de ϵ, 0 y 1 mediante las reglas $w \mapsto 0w0$ y $w \mapsto 1w1$. Para un string w, sean $|w|_0$ y $|w|_1$ sus números de ceros y unos, respectivamente. Demuestre que un palíndromo binario no puede tener simultáneamente $|w|_0$ y $|w|_1$ impares.

2. Una fórmula proposicional es una *y-o fórmula* si sólo usa los conectivos $\wedge$ y $\vee$.

   a) Defina recursivamente el conjunto de y-o fórmulas.

   b) Defina recursivamente las funciones $c(\phi)$ y $v(\phi)$ que cuente el número de conectivos y variables en una y-o fórmula, respectivamente.

   c) Demuestre por inducción estructural que, para toda y-o fórmula ϕ, tenemos que $v(\phi) \leq c(\phi) + 1$.

   d) Defina recursivamente la semántica de las y-o fórmulas.

   e) Muestre que las y-o fórmulas son *monótonas*, es decir, si ϕ es una y-o fórmula y $A \subseteq B$ son conjuntos de variables, entonces $\widehat{v}_A(\phi) \leq \widehat{v}_B(\phi)$.

3. Sea $\texttt{BB}$ el conjunto de strings de paréntesis balanceados definido en el ejercicio 1(b). Para un string w, defina su *balance* por $b(w) = |w|_(- |w|_)$. Diremos que u es un *prefijo* de w si $w = uz$ para algún string z.

   a) Demuestre por inducción estructural que, para todo $w \in \texttt{BB}$,

$$b(w) = 0 \text{ y } b(u) \geq 0 \text{ para todo prefijo } u \text{ de } w.$$

   b) Demuestre que todo string no vacío $w \in \texttt{BB}$ admite una descomposición $w = (u)v$ con $u, v \in \texttt{BB}$ y $u \neq \epsilon$.

   c) Muestre que la descomposición anterior es única.

   d) Los *árboles binarios* son árboles 1–2 en los que cada vértice tiene exactamente cero o dos hijos. Defina recursivamente el conjunto $\mathcal{T}_2$ de árboles binarios.

e) Defina recursivamente una función $f\colon \texttt{BB} \to \mathcal{T}_2$ que asocie a cada string de paréntesis balanceados un árbol binario. Muestre que f es biyectiva.

Indicación: recuerde que f es biyectiva si para todo $T \in \mathcal{T}_2$ existe un único $w \in \texttt{BB}$ tal que $f(w) = T$.

f) ¿Qué puede concluir sobre los árboles binarios y los strings de paréntesis balanceados?