1.1. Inducción

Principio de Inducción como propiedad de los naturales y técnica para demostraciones matemáticas. Muy usado en computación además como técnica de construcción de estructuras.

1.1.1. Principios de Inducción

Existen distintas formulaciones para el principio de inducción, veremos las más usadas.

Teorema 1.1.1: Principio de Buen Orden (PBO). Todo subconjunto no vacío de los naturales tiene un menor elemento

$$\text{si } A \neq \emptyset \text{ y } A \subseteq \mathbb{N} \text{ entonces existe un } x \in A \text{ tal que } x \leq y \text{ para todo } y \in A.$$

Este principio no lo cumplen por ejemplo los números racionales ¿Cuál es el menor elemento del conjunto $A = \{q \in \mathbb{Q} \mid q > 0\}$? No existe un menor elemento, de hecho, supongamos que existiera tal $q_0 \in A$ el menor elemento, claramente $\frac{q_0}{2} \in A$ y cumple que $0 < \frac{q_0}{2} < q_0$ lo que contradice la hipótesis de que q_0 es el menor. Los reales tampoco cumplen este principio (buen orden), de hecho es definitorio para $\mathbb{N}$.

El anterior principio se asume para formular uno más útil:

Teorema 1.1.2: Principio de Inducción Simple (PIS). Sea A un subconjunto de $\mathbb{N}$. Si A cumple con:

1. $0 \in A$
2. si $n \in A$ entonces $n + 1 \in A$

entonces $A = \mathbb{N}$.

Demostración: Asumimos el PBO. Supongamos que tenemos un conjunto $A \subseteq \mathbb{N}$ que cumple las anteriores características y tal que $A \neq \mathbb{N}$. Entonces el conjunto $B = \mathbb{N} - A$ cumple con $B \subseteq \mathbb{N}$ y con $B \neq \emptyset$. Por el PBO, B debe tener un menor elemento, digamos $b \in B$. Es claro que $b \neq 0$ (ya que $0 \in A$), luego $b - 1$ pertenece a $\mathbb{N}$ y no a B, por lo que se cumple $b - 1 \in A$. Dado que estamos suponiendo que A cumple las características del PIS entonces $b \in A$ lo que contradice el hecho de que b sea el menor elemento de B. La contradicción ocurre por el hecho de suponer que $A \neq \mathbb{N}$, luego necesariamente se cumple que $A = \mathbb{N}$. □

El anterior principio nos dice que cada vez que nos encontremos con un subconjunto de los naturales que contenga al 0 y que para cada uno de sus elementos, el sucesor de él también está contenido, entonces el conjunto es exactamente el conjunto de todos los naturales. Habitualmente a la propiedad $0 \in A$ se le llama *base de inducción* (BI), a la suposición de que $n \in A$ se le llama *hipótesis de inducción* (HI), y a la demostración de que $n + 1 \in A$ se le llama *tésis de inducción* (TI).

¿De qué nos sirve este principio? Principalmente para demostrar que algunas propiedades son cumplidas por todos los números naturales.

Ejemplo: Demostraremos que el 0 es el menor número natural usando el PIS. Para esto definimos el siguiente conjunto:

$$A = \{x \in \mathbb{N} \mid x \geq 0\}$$

Si demostramos que $A = \mathbb{N}$ estamos demostrando que para todo elemento $x \in \mathbb{N}$, x es mayor o igual a 0 y que por lo tanto 0 es el menor natural.

Demostración: La demostración es bastante simple:

B.I. Claramente $0 \in A$ ya que $0 \geq 0$.

H.I. Supongamos que un natural n fijo cumple con $n \geq 0$.

T.I. Dado que $n \geq 0$ se cumple que $n + 1 \geq 1$ y por lo tanto $n + 1 \geq 0$

Por PIS se sigue que $A = \mathbb{N}$. □

Generalmente el PIS se formula de una manera alternativa que hace más fácil plantear ciertos teoremas:

Teorema 1.1.3: Principio de Inducción Simple (segunda formulación). Sea P una propiedad cualquiera sobre elementos de $\mathbb{N}$. Si se tiene que:

1. $P(0)$ es verdadero (0 cumple la propiedad P)
2. $P(n) \Rightarrow P(n+1)$ (cada vez que n cumple la propiedad $n+1$ también la cumple)

Entonces todos los elementos de $\mathbb{N}$ cumplen la propiedad P.

Demostración: La demostración es inmediata a partir de la primera formulación del PIS, sólo tome $A = \{n \in \mathbb{N} \mid P(n) \text{ es verdadera } \}$. □

Al igual que en la formulación anterior, a $P(0)$ se le llama BI, la suposición de $P(n)$ es HI, y la demostración de $P(n+1)$ a partir de $P(n)$ es la TI.

Ejemplo: Demostraremos que la siguiente propiedad se cumple para todo n:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Para ocupar el PIS debemos definir nuestra propiedad P, en este caso:

$$P(n) \ : \ \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Demostración:

B.I. Si $n = 0$, $\sum_{i=0}^{n} = \sum_{i=0}^{0} = 0$ que es igual a $\frac{n(n+1)}{2} = 0$, por lo que $P(0)$ es verdadera

H.I. Supongamos que $P(n)$ se cumple, o sea que $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

T.I. Queremos demostrar ahora que $P(n+1)$ se sigue cumpliendo, o sea, queremos demostrar que

$$P(n+1) \ : \ \sum_{i=0}^{n+1} i = \frac{(n+1)((n+1)+1)}{2}$$

es verdadero.

Ahora, es claro que

$$\sum_{i=0}^{n+1} i = (\sum_{i=0}^{n} i) + (n+1)$$

Por HI se cumple que

$$\sum_{i=0}^{n+1} i = (\frac{n(n+1)}{2}) + (n+1)$$

de lo que resulta

$$\sum_{i=0}^{n+1} i = \frac{(n^2+n+2n+2)}{2} = \frac{n^2+3n+2}{2} = \frac{(n+1)(n+2)}{2}$$

finalmente

$$\sum_{i=0}^{n+1} i = \frac{(n+1)((n+1)+1)}{2}$$

por lo que $P(n+1)$ es también verdadero.

Por PIS se sigue que P se cumple para todos los naturales. □

A veces necesitamos demostrar propiedades que se cumplen para todos los naturales exceptuando una cantidad finita de ellos. Generalmente son propiedades de los naturales que empiezan a cumplirse desde un punto en adelante. El PIS puede ser modificado para que la BI pueda iniciarse en cualquier número natural distinto de 0, lo que nos importa en estos casos es que cierta propiedad se cumple para todos los naturales mayores o iguales que cierto natural fijo. Debemos cambiar entonces la demostración de nuestra base a ese natural fijo. El siguiente ejemplo nos muestra una aplicación de esta variación del PIS.

Ejemplo: Para todo natural $n \geq 4$ se cumple que

$$n! > 2^n$$

Nuestra propiedad en este caso es $P(n) \ : \ n! > 2^n$.

Demostración:

B.I. En este caso la base debiera iniciarse en $n = 4$, entonces nos preguntamos si $P(4)$ es o no verdadero. Ahora, $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24 > 16 = 2^4$, por lo que la propiedad se cumple para 4.

H.I. Supongamos que efectivamente $n! > 2^n$

T.I. Queremos demostrar que $(n+1)! > 2^{n+1}$. Ahora,

$$(n+1)! = (n+1)n!$$

dado que estamos suponiendo que n cumple la propiedad (HI) tenemos que

$$(n+1)! = (n+1)n! > (n+1)2^n$$

Ahora, dado que la propiedad que queremos demostrar se inicia en $n = 4$ sabemos que $n+1$ es necesariamente mayor que 4 por lo que obtenemos

$$(n+1)! > (n+1)2^n > 4 \cdot 2^n > 2 \cdot 2^n = 2^{n+1}$$

de donde obtenemos que

$$(n+1)! > 2^{n+1}$$

por lo que la propiedad se cumple también para $n+1$.

□

¿Cómo podemos justificar este nuevo uso del PIS a partir de la formulación con base en 0? En vez de considerar la propiedad P de arriba, pordiamos considerar la propiedad:

$$n < 4 \text{ o } n! > 2^n.$$

note que esta propiedad es verdadera (se puede demostrar usando PIS) para todos los elementos de $\mathbb{N}$.

Existen casos donde la inducción sirve para problemas que parecen estar muy apartados de propiedades numéricas como en el siguiente ejemplo:

Ejemplo: Queremos demostrar que cualquier tablero cuadriculado de dimensiones $2^n \times 2^n$ con $n \geq 1$ al que le falta exactamente un casillero, puede ser cubierto completamente con *trominós*. Un *trominó* es una ficha como la que se muestra en la figura 1.1.

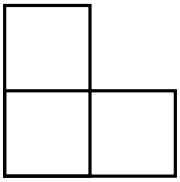

Figura 1.1: Un *trominó* recto

La propiedad en este caso tiene que ver con la potencia que nos da las dimensiones del tablero, o sea P sería:

$P(n)$: un tablero de $2^n \times 2^n$ con un casillero menos, puede ser completamente cubierto con *trominós*.

Lo segundo que debemos observar es que en este caso la inducción comienza en $n = 1$.

Demostración: (Demostración en clases) □

Al alumno de computación el anterior ejemplo debiera de inmediato parecerle, además de una demostración, un método de construcción. Usando exactamente la misma idea de la anterior demostración se podría programar un algoritmo recursivo que, dado un tablero de dimensiones $2^n \times 2^n$ al que le falta un casillero, pueda encontrar (efectivamente) una forma de cubrirlo usando *trominós*.

Existe una tercera formulación del Principio de Inducción que usa una suposición más fuerte y resulta de gran utilidad para demostrar propiedades cuando la información de que n cumple la propiedad no basta para concluir que $n + 1$ también la cumple.

Teorema 1.1.4: Principio de Inducción por Curso de Valores (PICV). Sea A un subconjunto de $\mathbb{N}$. Si para todo $n \in \mathbb{N}$ se cumple que

$$si \{x \in \mathbb{N} \mid x < n\} \subseteq A \text{ entonces } n \in A,$$

entonces $A = \mathbb{N}$.

En este caso la parte $\{0, 1, \ldots, n-1\} \subseteq A$ es la HI, y la demostración de $n \in A$ es la TI. Un punto interesante es que *pareciera* no haber una base de inducción... (¿la hay? piense en el caso $\emptyset \subseteq A$).

Al igual que con el PIS, existe una segunda formulación que hace las demostraciones más naturales.

Teorema 1.1.5: Principio de Inducción por Curso de Valores (segunda formulación). Sea P una propiedad cualquiera sobre elementos de $\mathbb{N}$ y suponga que se cumple que:

si para todo $k \in \mathbb{N}$ menor que n $P(k)$ es verdadero, entonces $P(n)$ es verdadero.

Entonces se tiene que P es verdadero para todos los elementos de $\mathbb{N}$.

El teorema anterior nos dice que si, al suponer que cierta propiedad se cumple para todos los números naturales menores que cierto n podemos concluir que n también la cumple, entonces se concluye que todos los números naturales cumplen la propiedad.

Ejemplo: La sucesión de Fibonacci, es una serie de números naturales $F_0, F_1, F_2, \ldots, F_n, F_{n+1} \ldots$ que cumplen la siguiente relación de recurrencia:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \quad \forall n \geq 2 \end{aligned}$$

Demostraremos que $F_n < 2^n$ para todo $n \in \mathbb{N}$.

Demostración: La demostración la haremos usando el PICV. Un punto interesante es que usaremos dos casos base, $n = 0$ y $n = 1$, la razón debiera quedar clara cuando finalizemos la demostración.

B.I. Para $n = 0$ se tiene $0 < 1 = 2^0$, para $n = 1$ se tiene que $1 < 2 = 2^1$ por lo que los casos base funcionan.

H.I. Supongamos que para todo $k < n$ se cumple que $F_k < 2^k$.

T.I. Queremos demostrar que usando HI podemos concluir que $F_n < 2^n$. Usaremos el hecho de que $F_n = F_{n-1} + F_{n-2}$, y la HI:

$$\begin{aligned} F_n &= F_{n-1} + F_{n-2} \\ &\overset{\text{(HI)}}{<} 2^{n-1} + 2^{n-2} = \tfrac{3}{4} \cdot 2^n \\ &< 2^n \end{aligned}$$

Por el PICV se sigue que $F_n < 2^n$ para todo $n \in \mathbb{N}$. □

Un error muy frecuente entre los alumnos es hacer la inducción "al revés". Inicialmente suponen que la tesis de inducción es correcta y haciendo movimientos algebráicos obtiene la hipótesis de inducción con lo que dan por terminada su demostración. Este tipo de desarrollo se considerará siempre incorrecto ya que **no se puede partir una demostración desde lo que se quiere concluir**. Siempre se debe tener muy en claro que lo que debemos suponer es la hipótesis de inducción y a partir de ella concluir la tesis de inducción.

Más adelante en el curso veremos aplicaciones directas de los principios de inducción en el área de algoritmos computacionales, principalmente en el cálculo de la eficiencia de un algoritmo y en el establecimiento de su corrección (que el algoritmo efectivamente hace lo que dice que hace).

1.1.2. Inducción Estructural

Hemos visto distintos principios de inducción (y formulaciones de estos), todos aplicados al conjunto de los números naturales. ¿Qué tiene de especial $\mathbb{N}$? Principalmente su característica de ser un conjunto que se puede construir a partir de un elemento base y un operador. El elemento base es el 0 y el operador es "el sucesor". Intuitivamente todo natural se puede obtener a partir de sumarle 1 a otro natural, o sea de aplicarle el operador sucesor a otro natural (excepto el 0 que es la base). Así una forma de definir al conjunto de los números naturales es la siguiente:

1. El 0 es un número natural.
2. Si n es un número natural entonces $n + 1$ también es un número natural.
3. Todos los números naturales y sólo ellos se obtienen a partir de la aplicación de reglas 1 y 2.

Mirando la definición debiera quedar clara la naturaleza constructiva de los números naturales y como se relaciona con el principio de inducción. Lo que hacemos entonces para demostrar que una propiedad se cumple para todo el conjunto de los números naturales es demostrar que se cumple para su elemento base y que si suponemos que se cumple para un elemento cualquiera, el operador de construcción (sucesor en este caso) mantiene la propiedad.

¿Pueden otros conjuntos definirse de manera similar? La respuesta es afirmativa y a estas definiciones les llamaremos *definiciones inductivas*. La implicancia más importante es que podremos usar inducción para demostrar propiedades que cumplen otros conjuntos, no sólo los naturales. Una implicancia adicional es que podremos definir nuevos objetos (funciones, operaciones, etc) usando la definición inductiva del conjunto.

En general para definir un conjunto inductivamente necesitaremos:

1. Un conjunto (no necesariamente finito) de elementos base que se supondrá que inicialmente pertenecen al conjunto que queremos definir.
2. Un conjunto finito de reglas de construcción de nuevos elementos del conjunto a partir de elementos que ya pertenecen.

(Omitiremos la afirmación "Todos los elementos del conjunto y sólo ellos se obtienen a partir de la aplicación de las anteriores reglas" pero supondremos que está implícita en la definición.)

Ejemplo: Un ejemplo muy sencillo es la definición de los números pares.

1. El 0 es un número par.
2. Si n es un número par entonces $n + 2$ es un número par.

nada muy extraño...

Un punto muy importante es que ningún elemento del conjunto que queremos definir se debe escapar de nuestra definición, por ejemplo la siguiente **no** es una definición válida de los números pares:

1. El 0 es un número par.
2. Si n es un número par entonces $2n$ es un número par.

¿Cuál es el problema?...

Ejemplo: Muchas veces cuando estudiamos computación nos encontramos con estructuras de datos. Generalmente las usamos y no nos interesa demasiado formalizar ni su construcción ni las operaciones sobre ella. En este ejemplo veremos como podemos formalizar un concepto similar al de "lista enlazada" muy usada en cursos de computación. Para simplificar la definición, supondremos que nuestras listas sólo pueden contener números naturales.

Un ejemplo de lista enlazada (de las que usaremos nosotros) es:

$$\to 5 \to 7 \to 1 \to 0 \to 3 \to 1 \to 4$$

Diremos en este caso que la lista contiene a los valores $5, 7, 1, 0, 3, 1, 4$ en ese orden (note la repetición del elemento 1). Una lista muy especial es la lista vacía que representaremos por

$$\emptyset$$

que es una lista que no contiene elemento alguno (para los que son más orientados a la programación esta lista representaría a un puntero nulo, `NULL` en C). En este caso la lista

$$\emptyset \to 10 \to 6$$

y la lista

$$\to 10 \to 6$$

son exactamente iguales, ambas contienen exactamente a los elementos 10 y 6 en ese orden.

Con estas consideraciones no es difícil imaginar que cualquier lista que se nos ocurra se formará a partir de una lista más pequeña a la que se le ha agregado un elemento "al final". La única lista que no puede ser creada entonces de esta manera es la lista vacía, que debiera ser nuestro caso base. Así la siguiente es una definición para el conjunto $\mathcal{L}_{\mathbb{N}}$ de todas las listas formadas con elementos en $\mathbb{N}$.

1. $\emptyset$ es una lista y representa a la lista vacía ($\emptyset \in \mathcal{L}_{\mathbb{N}}$).
2. Si L es una lista y k es un natural, entonces $L \to k$ es una lista ($L \in \mathcal{L}_{\mathbb{N}} \Rightarrow L \to k \ \in \mathcal{L}_{\mathbb{N}}, \ \forall k \in \mathbb{N}$).

En este caso la "operación" que estamos usando para crear listas es tomar una lista, y agregar una flecha ($\to$) seguida de un natural. La "operación flecha seguida de natural" correspondería a sumar uno en el caso de la inducción sobre los naturales. La anterior definición nos dice que $\emptyset$ es una lista, que $\to 4$ es una lista ya que se forma a partir de la lista vacía $\emptyset$ agregándole el natural 4. De la misma forma $\to 4 \to 7$ es una lista ya que se forma a partir de $\to 4$ que sabemos que es una lista, al agregar el natural 7.

Esta anterior definición además nos entrega una noción de igualdad de listas (concepto muy importante en los ejemplos posteriores):

$$L_1 \to k = L_2 \to j \quad \text{si y sólo si} \quad L_1 = L_2 \text{ y } k = j$$

esto quiere decir que dos listas son iguales cuando ambas han sido creadas a partir de la misma lista agregándole el mismo elmento al final.

Podemos plantear algunas propiedades que debieran cumplir todas las listas y demostrarlas por inducción estructural, es decir usando inducción en el dominio constructible de las listas. Demostraremos a modo de ejemplo que toda lista tiene exactamente la misma cantidad de elementos que de flechas ($\to$).

Demostración: En este caso la propiedad P es sobre el conjunto $\mathcal{L}_{\mathbb{N}}$ de todas las listas con elementos naturales y se define por:

$$P(L) \ : \ L \text{ tiene el mismo número de flechas que de elementos.}$$

B.I. El caso base es la lista vacía, ella tiene ningún elemento y ninguna flecha por lo que cumple con la propiedad, o sea $P(\emptyset)$ es verdadero.

H.I. Supongamos que una lista cualquiera L tiene exactamente tantos elementos como flechas, o sea que $P(L)$ es verdadero.

T.I. Queremos demostrar que $P(L \to k)$ es verdadero, o sea que la lista $L \to k$ con $k \in \mathbb{N}$, también cumple la propiedad. Es claro que la lista $L \to k$ tiene exactamente una flecha más y exactamente un elemento más que L. Dado que estamos suponiendo que $P(L)$ se cumple (HI), concluimos que $L \to k$ tiene exactamente el mismo número de flechas que de elementos, por lo que $P(L \to k)$ también se cumple.

Por inducción estructural se sigue que todas las listas en $\mathcal{L}_{\mathbb{N}}$ tienen la misma cantidad de flechas que de elementos. $\square$

Algo muy interesante de las definiciones inductivas de conjuntos, es la posibilidad de aprovechar el carácter de constructivo para definir operadores o funciones sobre los elementos. Cuando estas definiciones se hacen en los naturales generalmente se les llama "definiciones recursivas", por ejemplo, la definición del operador factorial (!) sobre $\mathbb{N}$ se hace de la siguiente manera:

1. $0! = 1$.
2. $(n+1)! = (n+1) \cdot n!$.

Aquí se está aprovechando la forma de construcción de los naturales para definir de manera elegante un operador sobre todos los naturales. Se define el caso base (0) y se explicita como operar el siguiente elemento

que ha sido creado por inducción (sucesor) suponiendo que el operador ya está definido sobre los demás elementos del conjunto.

De manera similar podemos definir funciones y operadores sobre otros conjuntos creados por inducción estructural, el siguiente ejemplo muestra definiciones para el dominio de las listas.

Ejemplo: La función $|\cdot| : \mathcal{L}_{\mathbb{N}} \to \mathbb{N}$ (que a la lista L se aplica como $|L|$) toma una lista como argumento y entrega el entero correspondiente a la cantidad de elementos de la lista, es decir el largo de la lista. Queremos definir la función $|\cdot|$ inductivamente sobre el dominio constructible de las listas $\mathcal{L}_{\mathbb{N}}$. Primero debemos definir el caso base, o sea el resultado de $|\emptyset|$. Naturalmente el resultado debiera ser 0, luego la primera parte de nuestra definición debiera ser:

$$|\emptyset| = 0$$

Queremos definir ahora que pasa con el largo de una lista que ha sido creada a partir de otra anterior. La única forma que conocemos de crear una nueva lista es agregarle un elemento al final de la primera, es claro que el largo de la nueva lista será el largo de la primera más 1, luego la segunda parte de nuestra definición debiera ser:

$$|L \to k| = |L| + 1$$

Finalmente la definición completa de la función $|\cdot|$ que toma una lista y entrega su largo resulta:

1. $|\emptyset| = 0$
2. $|L \to k| = |L| + 1$

con L lista y $k \in \mathbb{N}$.

La noción de largo de una lista ya la habíamos usamos, de manera intuitiva, cuando demostramos que toda lista tiene exactamente la misma cantidad de flechas que de elementos. De la misma forma como definimos $|\cdot|$ podríamos definir la función $|\overrightarrow{\cdot}|$: $\mathcal{L}_{\mathbb{N}} \to \mathbb{N}$ que toma una lista cualquiera y entrega como resultado la cantidad de flechas de la lista. Así la demostración de que toda lista tiene exactamente la misma cantidad de flechas que de elementos puede formularse como:

$$\forall L \in \mathcal{L}_{\mathbb{N}} \quad |\overrightarrow{L}| = |L|$$

Más adelante veremos propiedades de las listas que tienen que ver con funciones y operadores definidos para ellas y que pueden demostrarse por inducción.

Ejemplo: La función $\text{sum} : \mathcal{L}_{\mathbb{N}} \to \mathbb{N}$ toma una lista como argumento y entrega el entero correspondiente a la suma de todos los elementos de la lista. Por ejemplo

$$\text{sum}(\to 5 \to 7 \to 1 \to 0 \to 3 \to 1 \to 4) = 21$$

Definiremos la función sum de forma inductiva sobre el dominio constructible de las listas. Primero debemos definir el caso base, o sea el resultado de aplicar sum a la lista vacía. Naturalmente el resultado debiera ser 0, ya que la lista no contiene elemento alguno. Ahora tenemos que arreglárnosla para definir sum para una lista cualquiera construida a partir de una lista anterior. Es claro que sumar todos los elementos de una lista es equivalente a sumar todos los elementos del tramo inicial de la lista y al resultado sumarle el último elemento. Finalmente la definición completa de la función sum que toma una lista y entrega la suma de sus valores resulta:

1. $\text{sum}(\emptyset) = 0$
2. $\text{sum}(L \to k) = \text{sum}(L) + k$

con L lista y $k \in \mathbb{N}$.

Usando esta definición podemos calcular la suma de la lista $\to 2 \to 3 \to 5$ de la siguiente manera:

$$\begin{aligned} \text{sum}(\to 2 \to 3 \to 5) &= \text{sum}(\to 2 \to 3) + 5 \\ &= \text{sum}(\to 2) + 3 + 5 \\ &= \text{sum}(\emptyset) + 2 + 3 + 5 \\ &= 0 + 2 + 3 + 5 \\ &= 10 \end{aligned}$$

Ejemplo: Definiremos la función máx : $\mathcal{L}_{\mathbb{N}} \to \mathbb{N} \cup \{-1\}$ de una lista, que entrega el valor del elemento más grande en la lista. Por convención, supondremos que el elemento máximo de la lista vacía es -1 (¿Por qué tiene sentido esta suposición?). Nuestra definición entonces resulta:

1. $\text{máx}(\emptyset) = -1$
2. $\text{máx}(L \to k) = \begin{cases} \text{máx}(L) & \text{si máx}(L) \geq k \\ k & \text{si } k > \text{máx}(L) \end{cases}$

Como ejercicio se puede hacer algo similar al ejemplo anterior para calcular máx$(\to 4 \to 1 \to 7 \to 3)$.

Ejemplo: En este ejemplo veremos la definición de la función $Head : \mathcal{L}_{\mathbb{N}} \to \mathbb{N}$ que dada una lista entrega el primer elemento contenido en ella (la "cabeza" de la lista). Una cosa interesante de esta función es que no está definida para todas las listas de naturales, de hecho la lista vacía no tiene elemento alguno, por lo tanto no tiene un primer elemento. La función estará entonces, parcialmente definida por inducción.

1. $Head(\to k) = k$
2. Si L es una lista no vacía ($L \neq \emptyset$), $Head(L \to k) = Head(L)$.

Se debe notar que para esta definición existe una infinidad de casos base (tantos como elementos de $\mathbb{N}$).

Todos los anteriores ejemplos tienen que ver con funciones sobre listas que entregan un elemento natural, en la siguiente definición veremos un operador sobre listas, es decir, una función que toma una lista y entrega como resultado otra lista.

Ejemplo: Queremos definir el operador $Tail : \mathcal{L}_{\mathbb{N}} \to \mathcal{L}_{\mathbb{N}}$ que toma una lista y entrega la lista que resulta de ella al sacar el primer elemento. La definición entonces resulta:

1. $Tail(\to k) = \emptyset$
2. Si L es una lista no vacía, $Tail(L \to k) = Tail(L) \to k$.

Note que en este caso el operador tampoco está definido para la lista vacía.

Ahora con las varias funciones definidas podemos plantear muchas propiedades acerca de listas y demostrarlas usando inducción estructural.

Teorema 1.1.6: Las siguientes son propiedades de las listas:

1. $\forall L \in \mathcal{L}_{\mathbb{N}}$ se cumple $\text{sum}(L) \geq 0$.
2. $\forall L \in \mathcal{L}_{\mathbb{N}}$ se cumple $\text{máx}(L) \leq \text{sum}(L)$.

3. $\forall L \in \mathcal{L}_{\mathbb{N}}$, $\text{sum}(L) = \mathit{Head}(L) + \text{sum}(\mathit{Tail}(L))$.
4. $\forall L_1, L_2 \in \mathcal{L}_{\mathbb{N}}$, $L_1, L_2 \neq \emptyset$, se cumple $L_1 = L_2$ si y sólo si $\mathit{Tail}(L_1) = \mathit{Tail}(L_2)$ y $\text{sum}(L_1) = \text{sum}(L_2)$.
5. Muchas otras propiedades que se pueden plantear...

Demostración: A modo de ejemplo demostraremos sólo las propiedades 2 y 4, las demás se proponen como ejercicios.

2. Por inducción estructural en $\mathcal{L}_{\mathbb{N}}$:

   B.I. $\text{máx}(\emptyset) = -1 \leq 0 = \text{sum}(\emptyset)$, por lo que $\emptyset$ cumple la propiedad.

   H.I. Supongamos que para toda lista L se cumple que $\text{máx}(L) \leq \text{sum}(L)$.

   T.I. Queremos demostrar que $L \to k$ con $k \in \mathbb{N}$ también cumple, o sea, $\text{máx}(L \to k) \leq \text{sum}(L \to k)$.

   La definición de la función máx nos habla de dos casos

$$\text{máx}(L \to k) = \begin{cases} \text{máx}(L) & \text{si } \text{máx}(L) \geq k \\ k & \text{si } k > \text{máx}(L) \end{cases}$$

   seguiremos la demostración para cada uno de estos casos:
   Si $\text{máx}(L \to k) = \text{máx}(L)$ tenemos que

$$\begin{array}{rcl} \text{máx}(L \to k) & = & \text{máx}(L) \\ & \leq & \text{máx}(L) + k \quad (\text{ya que } k \in \mathbb{N}) \\ & \overset{\text{HI}}{\leq} & \text{sum}(L) + k \;=\; \text{sum}(L \to k) \end{array}$$

   Si $\text{máx}(L \to k) = k$ tenemos que

$$\text{máx}(L \to k) \;=\; k \overset{(1.)}{\leq} \text{sum}(L) + k \;=\; \text{sum}(L \to k)$$

   En cualquier caso se cumple que $\text{máx}(L \to k) \leq \text{sum}(L \to k)$, luego por inducción estructural se sigue que la propiedad se cumple para todas las listas.

4. Primero, es claro que si $L_1 = L_2$ entonces se cumple que $\mathit{Tail}(L_1) = \mathit{Tail}(L_2)$ y que $\text{sum}(L_1) = \text{sum}(L_2)$ ya que ambas son funciones y la igualdad está bien definida. El punto complicado es demostrar la implicación inversa: si $\mathit{Tail}(L_1) = \mathit{Tail}(L_2)$ y $\text{sum}(L_1) = \text{sum}(L_2)$ entonces $L_1 = L_2$. Demostraremos esto último por inducción estructural en $\mathcal{L}_{\mathbb{N}}$. Lo haremos haciendo inducción sobre la construcción de L_1 y la propiedad la demostraremos para todo L_2.

   B.I. En este caso no podemos tomar $\emptyset$ como base ya que *Tail* no está definido para $\emptyset$. Tomaremos como base entonces listas con un elemento. Sea $L_1 = \to k$. Dado que $\mathit{Tail}(L_1) = \emptyset$ y $\mathit{Tail}(L_1) = \mathit{Tail}(L_2)$ obtenemos que $\mathit{Tail}(L_2) = \emptyset$. Ahora, dado que $\text{sum}(L_1) = k$ y $\text{sum}(L_1) = \text{sum}(L_2)$, tenemos que $\text{sum}(L_2) = k$. Tenemos que $L_2 \neq \emptyset$, $\mathit{Tail}(L_2) = \emptyset$ y $\text{sum}(L_2) = k$, concluimos que $L_2 = \to k$, y por lo tanto $L_1 = L_2$.

   H.I. Supongamos que si $\mathit{Tail}(L_1) = \mathit{Tail}(L_2)$ y $\text{sum}(L_1) = \text{sum}(L_2)$ entonces $L_1 = L_2$.

   T.I. Considere ahora la lista $L_1 \to k$ y supongamos que

$$\begin{array}{l} \mathit{Tail}(L_1 \to k) = \mathit{Tail}(L_2) \quad \text{y} \\ \text{sum}(L_1 \to k) = \text{sum}(L_2). \end{array}$$

   Primero, dado que $\mathit{Tail}(L_1 \to k) = \mathit{Tail}(L_2)$ sabemos que L_2 no es vacía, y por lo tanto $L_2 = L_2' \to j$ para algún j. Ahora, por la definición de *Tail* y sum obtenemos

$$\begin{array}{l} \mathit{Tail}(L_1) \to k = \mathit{Tail}(L_2') \to j \\ \text{sum}(L_1) + k = \text{sum}(L_2') + j \end{array}$$

de la primera de estas ecuaciones y usando la definición de igualdad de listas obtenemos que $Tail(L_1) = Tail(L_2')$ y que $k = j$. Usando este último resultado en la segunda ecuación obtenemos que $\text{sum}(L_1) = \text{sum}(L_2')$. Tenemos entonces que $Tail(L_1) = Tail(L_2')$, $\text{sum}(L_1) = \text{sum}(L_2')$, y por la HI resulta que $L_1 = L_2'$ y dado que $k = j$ obtenemos que $L_1 \to k = L_2' \to j = L_2$.

□

Los ejemplos anteriores tienen que ver con la construcción de listas. Un punto que se debe notar es que cada lista se construye a partir de una única lista anterior, al igual que en el PIS en donde el paso inductivo tiene que ver exclusivamente con el antecesor de un natural. De manera similar al PICV podemos definir conjuntos inductivamente, usando para la construcción de un elemento particular uno o más de los elementos anteriores (anteriormente construidos). Luego para definir propiedades y demostrar teoremas sobre el nuevo conjunto definido tendremos que usar una estrategia más similar al PICV que al PIS. En el siguiente ejemplo veremos como se aplican estas ideas.

Ejemplo: Queremos definir el conjunto $\mathcal{E}_{\mathbb{N}}$ de todas las expresiones aritméticas que se pueden formar con números naturales, el símbolo `+`, el símbolo `*` y los símbolos de paréntesis `(` y `)`. Por ejemplo, los siguientes son elementos de $\mathcal{E}_{\mathbb{N}}$ (son expresiones aritméticas)

```
( 4 + 5 * 7 ) * 9
12 + 2 + 3 + 2 * 11
( 143 + 9 )
3
```

Note que no nos interesa el *valor* de la expresión, sólo nos interesa la forma en que esta "se ve". El conjunto $\mathcal{E}_{\mathbb{N}}$ puede definirse inductivamente usando una definición inductiva por curso de valores, es decir, definiendo un elemento posiblemente a partir de varios de los elementos anteriores. Una expresión vacía no tiene sentido, así que nuestro caso base (la expresión más pequeña) sería un natural cualquiera, luego:

Si k es un natural, entonces k es una expresión ($k \in \mathbb{N} \Rightarrow k \in \mathcal{E}_{\mathbb{N}}$).

No es difícil notar que otra manera de crear una expresión es "sumando" dos expresiones, o más formalmente, poniéndole un símbolo `+` entre las expresiones, así uno de los pasos inductivos será:

Si E_1 y E_2 son expresiones, entonces E_1 `+` E_2 es una expresión ($E_1, E_2 \in \mathcal{E}_{\mathbb{N}} \Rightarrow E_1$ `+` $E_2 \in \mathcal{E}_{\mathbb{N}}$).

Necesitamos completar la definición inductiva de las expresiones aritméticas, especificando como crear expresiones usando `*` y `()`. Finalmente nuestra definición resulta:

1. Si k es un natural, entonces k es una expresión ($k \in \mathbb{N} \Rightarrow k \in \mathcal{E}_{\mathbb{N}}$).
2. Si E_1 y E_2 son expresiones, entonces E_1 `+` E_2 es una expresión ($E_1, E_2 \in \mathcal{E}_{\mathbb{N}} \Rightarrow E_1$ `+` $E_2 \in \mathcal{E}_{\mathbb{N}}$).
3. Si E_1 y E_2 son expresiones, entonces E_1 `*` E_2 es una expresión ($E_1, E_2 \in \mathcal{E}_{\mathbb{N}} \Rightarrow E_1$ `*` $E_2 \in \mathcal{E}_{\mathbb{N}}$).
4. Si E es una expresión, entonces `(`E`)` es una expresión ($E \in \mathcal{E}_{\mathbb{N}} \Rightarrow$ `(`E`)` $\in \mathcal{E}_{\mathbb{N}}$).

En este caso los "operadores" usados para crear las expresiones son unir dos expresiones mediante un `+` o mediante un `*` y cerrar una expresión entre `()`.

Ejemplo: La siguiente es una definición inductiva sobre $\mathcal{E}_{\mathbb{N}}$ del operador $\#_L : \mathcal{E}_{\mathbb{N}} \to \mathbb{N}$ que dada una expresión, entrega la cantidad de paréntesis izquierdos de ella.

1. $\#_L(k) = 0$ para todo $k \in \mathbb{N}$.
2. $\#_L(E_1 \texttt{ + } E_2) = \#_L(E_1) + \#_L(E_2)$ para todas $E_1, E_2 \in \mathcal{E}_\mathbb{N}$.
3. $\#_L(E_1 \texttt{ * } E_2) = \#_L(E_1) + \#_L(E_2)$ para todas $E_1, E_2 \in \mathcal{E}_\mathbb{N}$.
4. $\#_L(\texttt{(}E\texttt{)}) = 1 + \#_L(E)$ para toda $E \in \mathcal{E}_\mathbb{N}$.

En esta definición se debe tener muchísimo cuidado en comprender la diferencia entre `+` y $+$. El primero es el símbolo utilizado para la creación de las operaciones aritméticas (es sólo un símbolo, no debiera significar nada...), el segundo representa a la suma de números naturales y tiene el sentido habitual.

De manera similar se puede definir el operador $\#_R : \mathcal{E}_\mathbb{N} \to \mathbb{N}$ que cuenta la cantidad de paréntesis derechos de una expresión aritmética:

1. $\#_R(k) = 0$ para todo $k \in \mathbb{N}$.
2. $\#_R(E_1 \texttt{ + } E_2) = \#_R(E_1) + \#_R(E_2)$ para todas $E_1, E_2 \in \mathcal{E}_\mathbb{N}$.
3. $\#_R(E_1 \texttt{ * } E_2) = \#_R(E_1) + \#_R(E_2)$ para todas $E_1, E_2 \in \mathcal{E}_\mathbb{N}$.
4. $\#_R(\texttt{(}E\texttt{)}) = 1 + \#_R(E)$ para toda $E \in \mathcal{E}_\mathbb{N}$.

El siguiente resulta ser un teorema muy simple acerca de las expresiones aritméticas.

Teorema 1.1.7: Toda expresión aritmética tiene exactamente la misma cantidad de paréntesis derechos que izquierdos, o sea:

$$\forall E \in \mathcal{E}_\mathbb{N} \quad \#_L(E) = \#_R(E).$$

Demostración: La demostración resulta inmediata a partir de las definiciones inductivas de ambos operadores, sus resultados aplicados a las mismas expresiones son exactamente los mismos. De todas maneras y sólo por completitud se presenta la demostración por inducción estructural en la construcción de $\mathcal{E}_\mathbb{N}$:

B.I. Si la expresión es un natural $k \in \mathbb{N}$ por definición tenemos que $\#_L(k) = 0 = \#_R(k)$.

H.I. Supongamos que E_1 y E_2 son expresiones que cumplen con $\#_L(E_1) = \#_R(E_1)$ y $\#_L(E_2) = \#_R(E_2)$

T.I. Tenemos tres casos para nuestra tesis que aparecen de la construcción inductiva de $\mathcal{E}_\mathbb{N}$:

- $E_1 \texttt{ + } E_2$: Tenemos que $\#_L(E_1 \texttt{ + } E_2) = \#_L(E_1) + \#_L(E_2) \overset{HI}{=} \#_R(E_1) + \#_R(E_2) = \#_R(E_1 \texttt{ + } E_2)$.
- $E_1 \texttt{ * } E_2$: Igual al caso anterior.
- $\texttt{(}E_1\texttt{)}$: Tenemos que $\#_L(\texttt{(}E_1\texttt{)}) = 1 + \#_L(E_1) \overset{HI}{=} 1 + \#_R(E_1) = \#_R(\texttt{(}E_1\texttt{)})$.

En cada caso la propiedad se cumple para los pasos inductivos de la construcción de $\mathcal{E}_\mathbb{N}$. □

A pesar de que los únicos ejemplos que usamos para inducción estructural fueron las listas enlazadas y las expresiones aritméticas, existen muchos otros dominios constructibles que pueden definirse de manera similar. En lo que sigue del curso, varias veces nos encontraremos con definiciones inductivas de objetos (conjuntos) y con definiciones de funciones y operadores sobre ellos. El alumno debe practicar planteándose dominios aptos para ser construidos en forma inductiva, plantear funciones sobre sus elementos y demostrar algunos teoremas que puedan surgir en el dominio.