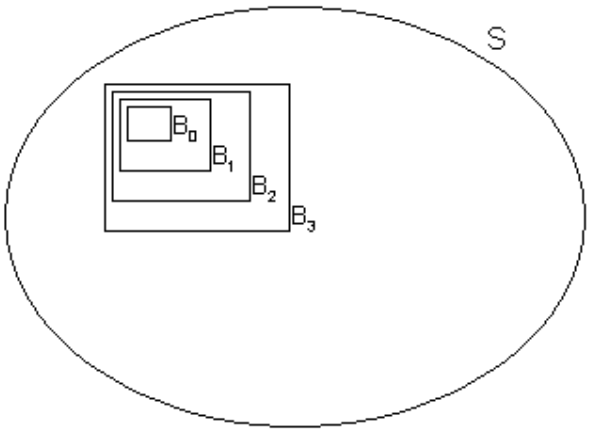
Conjuntos inductivos, recursión y principio de inducción

Esquema de inducción

Veremos la inducción como mecanismo primitivo para definir conjuntos. La idea es tomar un conjunto conocido S, partir de algún subconjunto $B_0 \subseteq S$, definir formas de agrandar sucesivamente B_0 y considerar el conjunto que se obtiene al límite de este proceso.



El conjunto S debe ser un conjunto conocido y podrá ser por ejemplo el conjunto N de los números naturales o el conjunto de las tiras de caracteres sobre determinado alfabeto.

A modo de ejemplo, veamos una definición inductiva del conjunto de los números pares:

Tomemos:

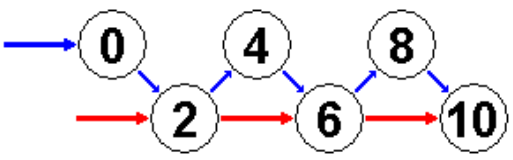
- Como S a los naturales
- Como base al conjunto $\{0\}$
- Como única regla: *"si* n *es par, entonces* $n+2$ *es par"*

Una observación interesante en este punto es que podríamos haber realizado otra definición inductiva para el mismo conjunto de los números pares:

Tomemos:

- Como S a los naturales
- Como base al conjunto $\{0,2\}$
- Como única regla: *"si* n *es par, entonces* $n+4$ *es par"*

Los conjuntos son iguales, pero la forma de recorrer los elementos es distinta. Así, demostrar que 10 es par podría implicar seguir distintos caminos (e incurrir en distintos costos), dependiendo de la definición inductiva que se considere.



Para continuar, deberemos presentar algunas definiciones.

Un alfabeto es un conjunto finito (o numerable) de símbolos o letras, por ejemplo:

$D = \{0,1,2,3,4,5,6,7,8,9\}$

```
digit ::= 0|1|2|3|4|5|6|7|8|9
```

Una palabra sobre un alfabeto es una secuencia (o cadena) de cero o más símbolos del alfabeto, concatenados.

$S^* = \{w \mid w$ es una palabra con letras del alfabeto $S\}$
$S^+ = \{w \mid w$ es una palabra no vacía con letras del alfabeto $S\}$

A la palabra vacía la notaremos ε.

A un conjunto de palabras se lo denomina lenguaje. S^* y S^+ son ejemplos de lenguajes para cualquier alfabeto S, pero nos concentraremos en lenguajes definidos de forma recursiva (aunque S^* y S^+ también podrían definirse en forma recursiva).

Por ejemplo:

Sea $S = \{a,b,c\}$, y $L_1 \subseteq S^*$ definido inductivamente por:

i. $a \in L_1$ (cláusula base: no depende de otros objetos de L_1)

ii. Si $w \in L_1$, entonces $bwb \in L_1$ (cláusula recursiva: depende de otros objetos de L_1)

Sea $S = \{a,b,c\}$, y $L_2 \subseteq S^*$ definido inductivamente por:

i. $b \in L_2$

ii. Si $w \in L_2$, entonces $awc \in L_2$

Con una definición inductiva no sólo definimos un conjunto sino que decimos la forma de construir objetos en él, e implícitamente decimos que es la única forma de construir objetos en el conjunto.

Para probar que un objeto pertenece a un conjunto inductivo basta con mostrar cómo lo formamos (su secuencia de formación está dada por las cláusulas utilizadas).

Por ejemplo:

$bbabb \in L_1$ ya que:

i. $a \in L_1$ (aplicamos i)

ii. Luego, $bab \in L_1$ (aplicamos ii)

iii. Finalmente, $bbabb \in L_1$ (aplicamos ii)

En el ejemplo anterior, se hace evidente que todas las palabras del lenguaje L_1 tendrán un número par de símbolos b, pero ¿qué es "tener un número par de símbolos b"?

Esquema de recursión primitiva

Sea A un conjunto definido inductivamente.

Para definir una función $f : A \to B$, alcanza con definir a f mediante ecuaciones que determinen:

- El valor de f para los objetos de A obtenidos de aplicar cláusulas base
- El valor de f para los objetos de A obtenidos de aplicar cláusulas inductivas, utilizando el valor de f en el objeto inmediatamente anterior (llamada recursiva).

Dado un conjunto inductivo, sabemos exactamente cómo se construyen sus elementos. Esta información nos sirve para definir funciones sobre sus elementos.

Por ejemplo, para nuestro lenguaje L_1:

> Sea B un conjunto arbitrario.
> Un conjunto de ecuaciones como el siguiente basta para definir una única función $f : L_1 \to B$.
>
> i. $f(a) := \ldots$
> ii. $f(bwb) := \ldots f(w) \ldots$

Un ejemplo más concreto y muy intuitivo será el de la función largo:

> Consideremos la función $\text{largo} : L_1 \to N$, definida por:
>
> i. $\text{largo}(a) := 1$
> ii. $\text{largo}(bwb) := \text{largo}(w) + 2$

El esquema de recursión primitiva nos permite definir funciones, siguiendo los pasos de una definición inductiva.

Principio de inducción primitiva

Sea A un conjunto definido inductivamente.

Para probar que una propiedad se cumple para todos los objetos de A alcanza con:

- Probar que la propiedad se cumple para los objetos de A obtenidos de aplicar cláusulas base
- Probar que la propiedad se cumple para los objetos de A obtenidos de aplicar cláusulas inductivas, suponiendo que la misma se cumple para objetos anteriores (hipótesis inductiva).

Por ejemplo, podríamos demostrar que *"todos los objetos de L_1 tienen una cantidad impar de símbolos"*. Una prueba que aprovecha el conocimiento de cómo se generan los objetos de L_1 sería de la forma:

> i. a tiene una cantidad impar de símbolos
> ii. Si w tiene una cantidad impar de símbolos, entonces bwb también

En este punto vemos que podría ser útil para la demostración servirse de funciones definidas recursivamente sobre el conjunto inductivo. En lugar de probar que los objetos de L_1 *tienen "una cantidad impar de símbolos"* podríamos considerar probar que tienen *"largo impar"*, utilizando la función largo que vimos previamente.

En el caso general, si queremos probar una propiedad P para las palabras de L_1, aplicamos el principio de inducción para L_1.

> Sea P una propiedad sobre los objetos de L_1 tal que:
>
> i. $P(a)$ se cumple
>
> ii. Si $P(w)$ se cumple, entonces $P(bwb)$ se cumple
>
> Entonces, P se cumple para todos los objetos de L_1.

Debemos notar que el principio de inducción es dependiente del conjunto inductivo, y más aún, de las cláusulas base e inductivas de su definición.

El principio de inducción, aplicado a un conjunto inductivo, nos permite probar proposiciones sobre todos los elementos del conjunto inductivo:

- Para todo w de L_1, w tiene una cantidad impar de símbolos
- Para todo w de L_1, $\text{largo}(w)$ es impar
- Para todo w de L_1, w es palíndromo

Ejemplo: las expresiones aritméticas

Consideremos el alfabeto $A = \{+, *, (,), 0, 1, 2, ... \}$.

El lenguaje de las expresiones aritméticas, $Exp \subseteq A^*$ puede ser definido inductivamente por:

i. Si $n \in \{0, 1, 2, ... \}$, entonces $n \in Exp$

ii. Si $\{x1, x2\} \subseteq Exp$, entonces $(x1 + x2) \in Exp$

iii. Si $\{x1, x2\} \subseteq Exp$, entonces $(x1 * x2) \in Exp$

Construimos entonces nuestro conjunto de expresiones aritméticas mediante el esquema de inducción. Ahora podemos definir funciones sobre este conjunto inductivo, aplicando el esquema de recursión primitiva.

Sería útil, por ejemplo, definir una función de evaluación:

Sea la función $eval : Exp \to N$, definida por:

i. $eval(n) := n$

ii. $eval((x1 + x2)) := eval(x1) + eval(x2)$

iii. $eval((x1 * x2)) := eval(x1) * eval(x2)$

Es importante darse cuenta que Exp es un lenguaje (un subconjunto de A^*), y por lo tanto sus elementos son palabras (cadenas de símbolos sobre un alfabeto). En este contexto, los elementos $+$, $*$ y cualquier n son simplemente símbolos. Sin embargo, del lado derecho de la definición de función, una aparición de n significa un número natural, y una aparición de $+$ o $*$ implica la operación binaria cerrada en los naturales de suma o producto.

Al esquema de recursión, y al principio de inducción que vimos los llamamos primitivos porque se basan en el objeto inmediatamente anterior del conjunto inductivo.