Universidad de Los Andes

http://cesimo.ing.ula.ve

Unidad 2: Introducción a la matemática discreta

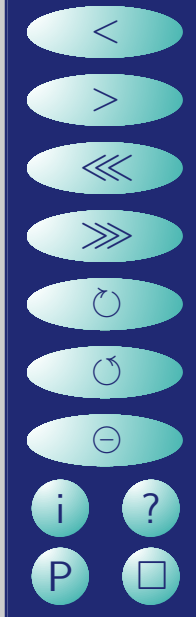
Jacinto Dávila

mailto:jacinto@ula.ve

Centro de Simulación y Modelos (CESIMO)

A diferencia de las otras unidades, en esta tocamos mas de un tema separable:

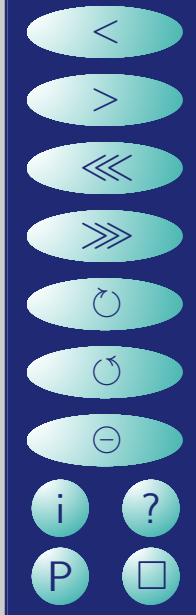
1. Inducción matemática: Deducción en matemáticas.
2. Nociones elementales de latises y algebras y su relación con las estructuras de "significados" en lógica.
3. Elementos de la teoría de grafos en lógica.



Inducción matemática

El método de **inducción matemática** se aplica cuando:

1. sabemos la respuesta al principio.
2. sabemos cómo determinar la respuesta en una etapa a partir de la respuesta en la etapa anterior, y
3. tenemos una expectativa (una idea) de la respuesta general.



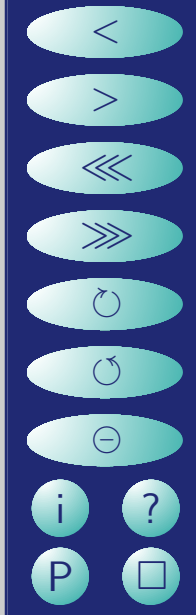
Versión elemental del principio de inducción matemática

Consideremos una lista p(1), p(2), ... de proposiciones con índices en $\mathbb{P}$. Todas estas proposiciones son ciertas si:

- B) $p(1)$.
- I) $(\forall N \in \mathbb{P})(p(N+1) \leftarrow p(N))$

Observe que el principio de inducción NO es una prueba de que p(N) sea cierta para toda N.

El principio dice que si B) e I) son verdaderas, entonces las p(N) lo son. Precisamente la deducción que todas las p(N) son ciertas es lo que uno obtiene al aplicar el principio.



El principio de inducción matemática es *deductivo*!!

$p(1)$

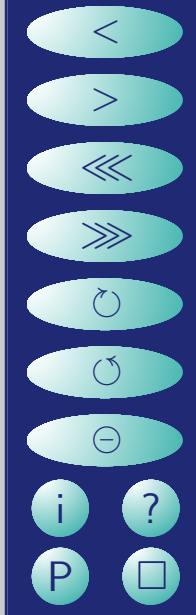
$\underline{\forall N(p(N+1) \leftarrow p(N))}$

$\underbrace{\forall N(p(N))}$

Esta conclusión es deducida, no inducida!!.

Así termina una demostración por el principio de inducción matemática:

... entonces p(N+1) es verdadera siempre que p(N) lo sea. Así que, por el principio de inducción matemática podemos concluir p(N) es verdadera para toda N.



Primer forma del principio de inducción matemática

(B) $p(m)$

(I) $\underline{(\forall N \geq m)(p(N+1) \leftarrow p(N))}$

$\vdash (\forall N \geq m)(p(N))$

Segunda forma del principio de inducción matemática

(B) $p(m), ..., p(m+l)$

(I) $\underline{(\forall N \geq m)(p(N) \leftarrow p(m) \wedge ... \wedge p(N-1))}$

$\vdash (\forall N \geq m)(p(N))$

Principio del buen orden

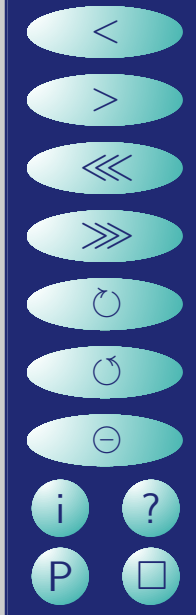
Principio del buen orden: todo subconjunto no vacío de los naturales $\mathbb{N}$ tiene elemento mínimo.

Esta propiedad no es una consecuencia de las reglas aritméticas de $\mathbb{N}$.

El principio del buen orden implica que:

Dado $m \in \mathbb{Z}$, todo subconjunto no vacío de $\{n \in \mathbb{Z} : n \geq m\}$ tiene elemento mínimo.

El principio del buen orden es usado para probar la validez de los principios de inducción matemática.



Un ejemplo de inducción matemática

Teorema LM2-1: Todo número entero mayor o igual que 2 ($N \geq 2$) puede escribirse como el producto de otros numeros primos.

Prueba: Sea p(N) la proposición: "N es el producto de primos".

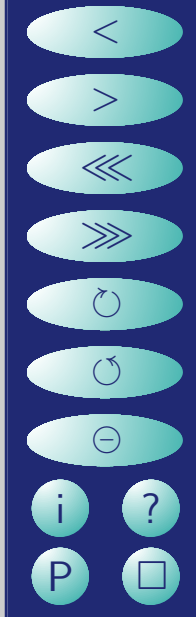
B) Dado que el 2 es primo tenemos p(2) (puesto que $2 = 2 * 1$).

I) Veamos que ocurre con $N > 2$. Supongamos que p(K) es cierto para todo K mayor o igual que 2 y menor que N. Con esta suposición tratemos de probar que p(N) es verdadero.

Si N es primo, p(N) es directamente cierta de la misma forma que B).

Si N no es primo, se le puede escribir como el producto de dos números, J y W, mayores que 1. Pero esto implica que J y W son ambos mayores o iguales que 2 y estrictamente menores que N. Es decir que, en virtud de la suposición que hicimos en esta parte, ambos son productos de primos.

Pero entonces, N que es igual a J*W, es también producto de primos (Fin de la prueba).

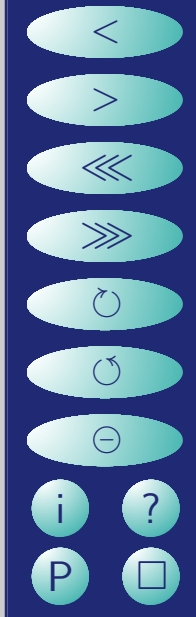


Conjuntos parcialmente ordenados

En muchos conjuntos que surgen naturalmente, sabemos cómo comparar algunos elementos con otros, pero también hay parejas que no son comparables.

Podemos comparar dos subconjuntos de un conjunto S (es decir, 2 elementos de $\mathcal{P}(\mathrm{S})$) si uno está contenido en el otro. Si S tiene más de un elemento entonces tiene subconjuntos incomparables.

Si $s_1 \subset S$ y $s_2 \subset S$, entonces los conjuntos $\{s_1\}$ y $\{s_2\}$ no son comparables (si $\subset$ es la inclusión estricta).



La definición de los conjuntos parcialmente ordenados.

Aquellos en los cuales las relaciones de comparación permiten pares de elementos que no son comparables.

Sea $\preceq$ una relación sobre el conjunto S tal que $< x, y > \in \preceq$ si $< x, y > \in S \otimes S$ y x está relacionada con y (que se escribe precisamente $x \preceq y$).

La relación $\preceq$ es un **orden parcial** si satisface lo siguiente:

- Ley reflexiva: $\forall s(s \in S \to s \preceq s)$
- Ley antisimétrica: $\forall s, t(s \preceq t \wedge t \preceq s \to s = t)$
- Ley transitiva: $\forall s, t, u(s \preceq t \wedge t \preceq u \to s \preceq u)$

Decimos que $\preceq$ es un orden parcial en S y $(S, \preceq)$ es un conjunto parcialmente ordenado o **copo** (*poset*).

Ejercicio: Construya conjuntos parcialmente ordenados, si es posible, con las relaciones "subconjunto de", "múltiplo de", "amigo de", "es un", "es parte de" y "es causa de".

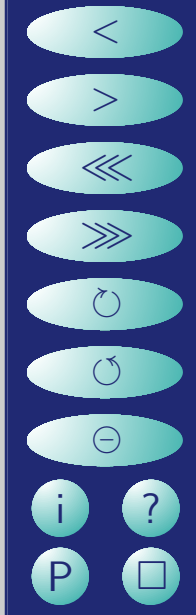
Orden parcial estricto

Sea $\prec$ definida como:

$$x \prec y \leftrightarrow x \preceq y \wedge x \neq y$$

La relación $\prec$ satisface la:

- Ley antirreflexiva: $\forall s \in S(s \prec s \rightarrow falso)$ (atención con esta fórmula ¿Qué dice?)
- Ley transitiva: $\forall s, t, u(s \prec t \wedge t \prec u \rightarrow s \prec u)$



Los diagramas de Hasse

La relación "cubre a" y los diagramas de Hasse.

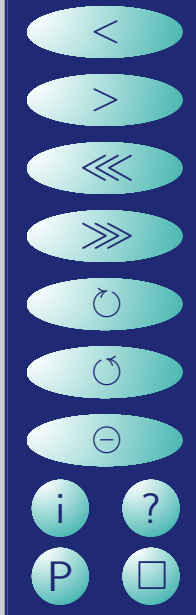
t **cubre** a s en un orden parcial $\preceq$ (en S) si

$$s \prec t \wedge \neg \exists u (s \prec u \prec t)$$

Un **diagrama de Hasse** del copo $(S, \preceq)$

es un grafo cuyos vértices son los elementos de S y hay una arista de s a t siempre que t cubre a s.

Los diagramas generalmente se dibujan de abajo hacia arriba.

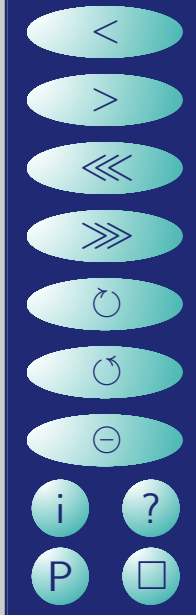


Cúspide y base en un diagrama de Hasse.

Si $(P, \preceq)$ es un copo, $x \in P$ es maximal si no existe $y \in P$, tal que $x \prec y$.

Si $(P, \preceq)$ es un copo, $x \in P$ es minimal si no existe $y \in P$, tal que $y \prec x$.

Un subcopo de P es un subconjunto del copo P que hereda el orden parcial de P y el mismo es un copo.



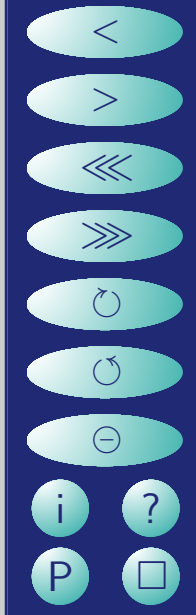
Elementos mayores y menores.

Si S es un subcopo de $(P, \preceq)$ y existe M tal que $\forall s \in S(s \preceq M)$ entonces $M = \max(S)$ y M es el mayor elemento de S.

y

si existe un $m \in S$ tal que $\forall s \in S(m \preceq s)$ entonces m es el mínimo de S y $m = \min(S)$.

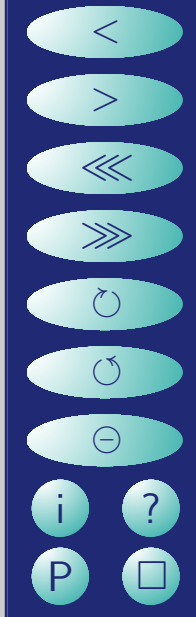
A lo sumo hay un M y un m, ¿Por qué?.



Cotas superiores (Upper bounds)

A pesar de que no exista un máximo para el subcopo S de $(P, \preceq)$ puede existir un $x \in P$ tal que $\forall s \in S(s \preceq x)$. x es la cota superior (*upper bound*) para S en P.

Si, además, $x \preceq y$ para toda y que es cota superior de S en P, entonces x es una **mínima cota superior** (*least upper bound*) ó **mcs** de S en P.

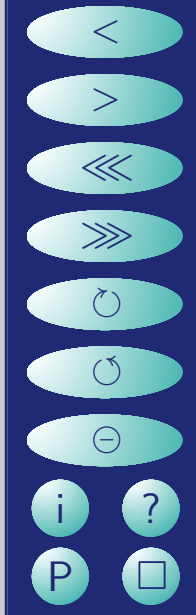


Cotas inferiores (lower bounds)

En forma similar a la lámina anterior, un $z \in P$ tal que $z \preceq s$, para toda $s \in S$, es una cota inferior (*lower bound*) para S en P.

Y, una cota inferior z tal que $w \preceq z$, para toda w que sea cota inferior para S en P es una **máxima cota inferior** (*greatest lower bound*) ó **mci** de S en P.

Por antisimetría, un subconjunto de P no puede tener 2 mínimas cotas superiores o 2 máximas cotas superiores.



Latises ó reticulados (*lattices*)

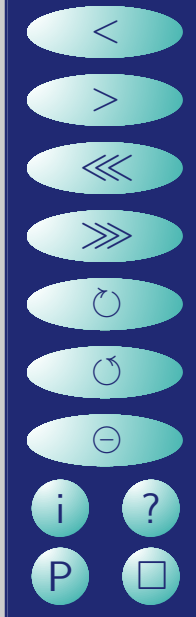
Un copo $(P, \preceq)$ es una latís si $mcs\{x, y\}$ y $mci\{x, y\}$ existen para todo par $x \in P$, $y \in P$.

Si $(P, \preceq)$ es una latís entonces para todos $x \in P$, $y \in P$

$x \vee y = mcs\{x, y\}$ y $x \wedge y = mci\{x, y\}$

$\vee$ y $\wedge$ se interpretan como la unión e intersección respectivamente en esta lámina.

Actividad: Demuestre por inducción que todo subconjunto finito de una latís tiene mcs y mci.



¿Latises para qué?

Latises o reticulados (*lattices*) y las interpretaciones de programas.

Considere el programa lógico siguiente:

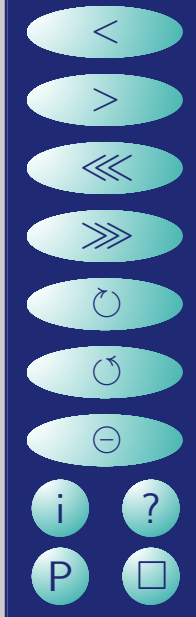
```
masculino(adan).

femenino(eva).
```

y el reticulado (latis) en la lámina siguiente:

Actividad: Complete el diagrama de Hasse (observe que está orientado de izquierda a derecha).

Este latís contiene TODAS las interpretaciones posibles del programa anterior. Actividad: Indique cuáles son MODELOS.



$$
\begin{pmatrix}
 & & m(a) & \\
 & m(a), f(e) & & \\
 & & f(e) & \\
m(a), f(e), m(e) & & & \\
 & m(a), m(e) & & \\
m(a), f(e), f(a) & & & \\
 & m(a), f(a) & & \\
m(a), f(e), m(e), f(a) & & & \emptyset \\
 & f(e), f(a) & & \\
f(e), m(e), f(a) & & & \\
 & f(e), m(e) & & \\
 & & f(a) & \\
m(a), m(e), f(a) & & & \\
 & m(e), f(a) & & \\
 & & m(e) &
\end{pmatrix}
$$

Ordenes especiales

Un **árbol** puede considerarse un diagrama de Hasse en el que $mcs\{x, y\}$ existe para cada par $< x, y >$, pero $mci\{x, y\}$ existe sólo si $x \preceq y$ ó $y \preceq x$.

Una **cadena** es un copo en el cual siempre se puede comparar a los elementos por pares.

Un orden parcial es un **orden total o lineal** si para cada elección de s y t en S o bien $s \preceq t$ o $t \preceq s$.

Una cadena es un conjunto totalmente ordenado.

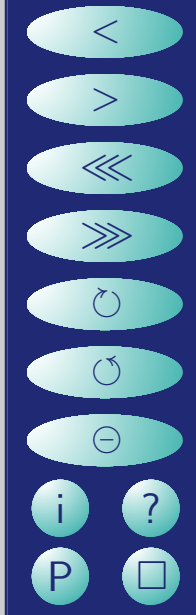
Todo subcopo de una cadena es, a su vez, una cadena.

Relaciones de equivalencia

La relación $\sim$ (ó $\equiv$) se dice relación de equivalencia si satisface:

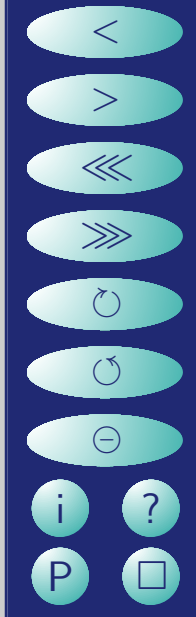
- (R) $\forall s \in S(s \sim s)$.
- (S) $\forall s, t \in S(s \sim t \to t \sim s)$
- (T) $\forall s, t, u \in S(s \sim t \wedge t \sim u \to s \sim u)$

las reglas anteriores son condiciones que debe satisfacer una relación para que se considerada de equivalente. Sin embargo, esa no es la definición de la relación. Para construir, por ejemplo, la definición de equivalencia lógica uno tendría que escribir fórmulas como las siguientes:



1. $(A \wedge cierto) \equiv A$
2. $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$
3. $(H \leftarrow (A \vee B) \wedge C) \equiv (H \leftarrow A \wedge C) \wedge (H \leftarrow B \wedge C)$

Actividad: Agrega por lo menos dos fórmulas más a esta lista y construye un programa PROLOG que compute la relación de equivalencia



Clases de equivalencias

Consideremos una relación de equivalencia $\sim$ en un conjunto S:

$$\forall s \in S([s] = \{t \in S : s \sim t\})$$

Sea S un conjunto de camisas (no todos llamamos canicas a las canicas). Decimos que dos camisas s y t son equivalentes ($s \sim t$) si son del mismo color (Verifique que esta es una relación de equivalencia). La clase de equivalencia $[s]$ de una camisa s, es el conjunto de todas las camisas que tiene el mismo color que s.

$[S]$ es el conjunto de todas las clases de equivalencias de S:

$$[S] = \{[s] : s \in S\}$$

Cerradura (Clausura) de relaciones *(Closure)*

Teorema lm2-2: Si R es una relación en un conjunto S y si $E = \{< x, x >: x \in S\}$ y $R^{-1} = \{< y, x > \in S \otimes S :< x, y > \in R\}$, entonces:

r $r(R) = R \cup E$;

s $s(R) = R \cup R^{-1}$;

t $t(R) = \bigcup_{k=1}^{\infty} R^k$

Grafos

Actividad: Investigemos el vocabulario de los grafos

grafo o gráfica:

aristas y vértices:

aristas múltiples o paralelas: aristas entre los mismos vértices.

lazo: arista que conecta a vértice consigo mismo.

camino: sucesión de aristas.

longitud de un camino: número de aristas en el camino.

camino cerrado: sucesión circular.

ciclo: camino cerrado en que ningún vértice se repite.

grafo acíclico

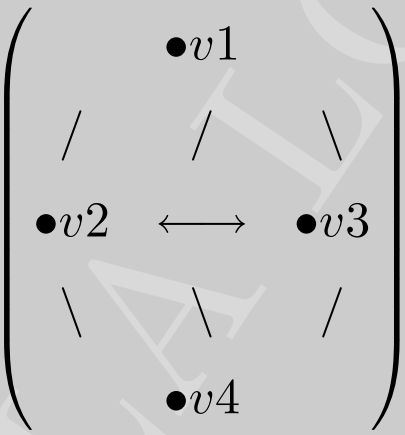
Circuito de Euler o Euleriano

Un camino cerrado que contiene exactamente cada una de las aristas.

La *valencia* de un vértice es el número de aristas que confluyen en él.

[Euler]: Si todos los vértices de una gráfica *conexa* tienen valencia par, entonces la gráfica tiene un circuito euleriano.

El grafo de Königsberg.



Una representación de un grafo en lógica

$$G = [arista(a,[w,x]),\ arista(b,[w,x]),\ arista(c,[w,u]),\ arista(d,[u,y]),\ arista(e,[w,y]),\ arista(f,[y,x]),\ arista(g,[x,z]),\ arista(h,[y,z]).]$$

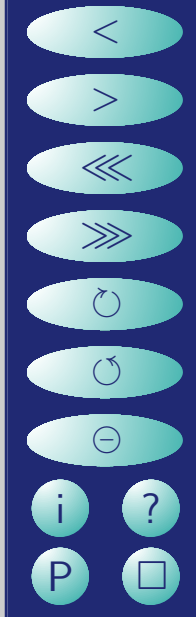
Este es un programa para recolectar los vértices de esos grafos:

$vértices([],[]).$

$vértices([arista(\_,Vertices) \mid RestoG],Todos)$

$\leftarrow vértices(RestoG,RestoVertices)$

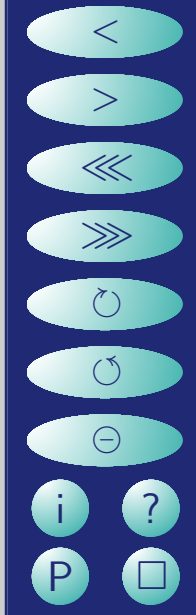
$\wedge\ une(Vértices,RestoVertices,Todos).$



$$une([], Lista, Lista).$$

$$une([\mathrm{Pr}\, imero|Resto], Lista, [\mathrm{Pr}\, imero|NResto]) \leftarrow \mathrm{Pr}\, imero \notin Lista$$
$$\wedge une(Resto, Lista, NResto).$$

$$une([\mathrm{Pr}\, imero|Resto], Lista, NResto) \leftarrow \mathrm{Pr}\, imero \in Lista$$
$$\wedge une(Resto, Lista, NResto).$$



Valencias y vértices

% la valencia de un vértice V en un grafo vacío en nula.

$valencia(V, [], 0).$

$valencia(V, [arista(\_, Vertices)|Resto], N + 1) \leftarrow V \in Vertices$

$\wedge valencia(V, Resto, N).$

$valencia(V, [arista(\_, Vertices)|Resto], N) \leftarrow V \notin Vertices$

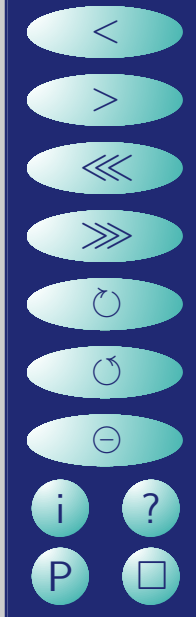
$\wedge valencia(V, Resto, N).$

$valencias([], G, []).$

$valencias([V|R], G, [(V, C)|NR]) \leftarrow valencia(\ V, G, C)$

$\wedge valencias(\ R, G, NR).$

¿Cómo se leen en español las cláusulas de esta lámina?



Algoritmo de Fleury

Paso 1. Empiece en cualquier vértice V de valencia impar si es que lo haya. Si no lo haya empiece en cualquier vértice V. Sea Vs = [V] y sea Es=[].

Paso 2. Si no hay ninguna arista que pase por V, pare.

Paso 3. Si hay exactamente una arista que pase por V, digamos arista(E, [V,W]), entonces sustraiga a arista E del grafo A(G) y a V sus vertices V(G) y siga con el paso 5.

Paso 4. Si hay más de una arista en V, elija una de ellas, digamos E, tal que arista(E,[V,W]), de tal modo que su eliminación no desconecte la gráfica; quite entonces a E de A(G).

Paso 5. Añada W al final de Vs, añada E al final de Es, reemplace V por W y regrese al paso 2.

Reconstrucción lógica del algoritmo de Fleury

(*Olvidándonos de la eficiencia y tomándolo con mucha calma*).

$$eureliano(G) \leftarrow vertices(\ G, Vertices)$$
$$\wedge valencias(Vertices, G, Valencias)$$
$$\wedge vertice\_apropiado(Valencias, V)$$
$$\wedge sucesion(G, V, Sucesion)$$
$$\wedge cerrada(V, Sucesion).$$
$$cerrada(V, Sucesion) \leftarrow Sucesion = [V|Resto]$$
$$\wedge ciclo\_desde\_vertice(V, Sucesion).$$
$$ciclo\_desde\_vertice(\ V, [V]).$$
$$ciclo\_desde\_vertice(V, [\_|R]) \leftarrow ciclo\_desde\_vertice(V, R).$$

¿Cómo se leen en español?

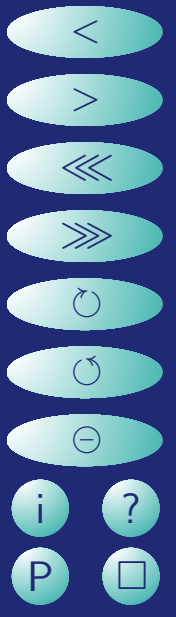
Más de la reconstrucción lógica de Fleury

$$sucesion(G, V, []) \leftarrow \neg\exists A((arista(A, VerticesdeA) \in G \wedge V \in VerticesdeA).$$

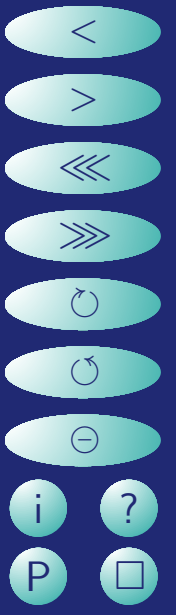
$$sucesion(G, V, [V|RestoSucV]) \leftarrow$$

$$selecciona\_arista(G, V, G', V')$$

$$\wedge sucesion(G', V', RestoSucV).$$



$$selecciona\_arista([arista(A, Vs)], V, [], W)$$
$$\leftarrow (Vs = [V, W] \vee Vs = [W, V]).$$
$$selecciona\_arista(G, V, NuevaG, W) \leftarrow arista(A, Vs) \in G$$
$$\wedge \neg desconecta(arista(A, Vs), G)$$
$$\wedge (Vs = [V, W] \vee Vs = [W, V])$$
$$\wedge sustrae(arista(A, Vs), G, NuevaG)$$



Actividad: No olvide leer estas cláusulas en español y procurar entender lo que dicen.

$$desconecta(A, G, G') \leftarrow A = arista(N, [X, Y])$$
$$\wedge sustrae(A, G, G')$$
$$\wedge desconectados(X, Y, G').$$

$$desconectados(X, Y, G) \leftarrow \neg ca\min o(X, Y, G).$$

$$sustrae(A, [A|Resto], Resto).$$

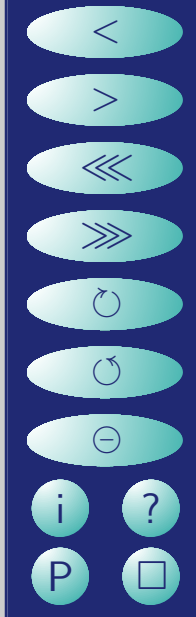
$$sustrae(A, [B|Resto], [B|Final]) \leftarrow A \neq B \wedge sustrae(A, Resto, Final).$$

¿Cúal es la suposición esencial en este programa de sustracción?

Circuito Hamiltoniano

Es un camino cerrado que utiliza cada vértice del grafo exactamente una vez, excepto el último que es igual al primero.

$$\begin{aligned} hamiltoniano(G,H) \leftarrow\ & vertices(G,Vertices) \\ & \wedge conecta(G,Vertices,H) \\ & \wedge cerrado(H). \end{aligned}$$



$$conecta(G, [V], [arista(A, [V, Z])])$$

$$\leftarrow arista(A, [V, Z]) \in G \vee arista(A, [Z, V]) \in G.$$

$$conecta(G, [V|RestoVs], [arista(A, [V, W])|RestoH])$$

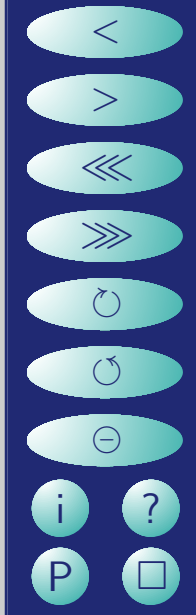
$$\leftarrow RestoVs \neq []$$

$$\wedge (arista(A, [V, W]) \in G \vee arista(A, [W, V]) \in G)$$

$$\wedge agrega(Frente, [W|Cola], RestoVs)$$

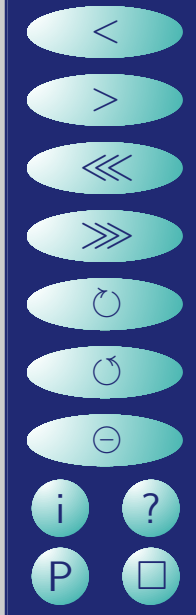
$$\wedge agrega(Frente, Cola, NuevoRestoV)$$

$$\wedge conecta(G, [W|NuevoRestoV], RestoH).$$



¿Qué pasa con la eficiencia si en lugar de $cerrado(H)$ usamos?:

$$cerrado^*(Vertices, H) \leftarrow Vertices = [V|\_] \wedge agrega(\_, [arista(\_, [\_, V])], H)?$$



El problema del agente viajero (*the traveller salesman's problem*)

Primera aproximación.

Para este problema tenemos que modificar nuestra representación de arista, para incluir un PESO en cada arista:

$$arista(Nombre, Peso, Vertices).$$

$$ruta\_del\_viajero(G, R) \leftarrow hamiltoniano(G, R) \wedge peso\_minimo(G, R).$$

$$peso\_minimo(G, R) \leftarrow peso(R, P)$$

$$\wedge(\neg\exists H(ruta\_del\_viajero(G, H) \wedge H \neq R \wedge peso(H, P^*) \wedge P^* < P)).$$

La solución que se vislumbra es muy ineficiente. **Actividad: Piense por un momento (sólo un momento) cómo podría ser más eficiente.**

Arboles

Son grafos acíclicos conexos. Volveremos a ellos más adelante en el curso. Aquí mostramos una forma de representarlos en lógica:

$$arbol(Nodo, Hijos)$$

Actividad: Dibuje el árbol: $arbol(a, [arbol(b, [d, e]), arbol(c, [arbol(f, [q])])])$.

Volveremos con los árboles en la unidad 5 de este curso.

En esta unidad 2 hemos revisado conceptos claves en matemáticas discretas que usaremos más adelante en el curso.

Es particularmente importante que entendamos qué es la latís de interpretaciones. Esa es la estructura matemática que emplearemos para precisar el "significado" de nuestros programas lógicos.

Los grafos, por su parte, son las estructuras matemáticas más populares en computación. Es muy importante que sepan que es un hamiltoniano y en que consiste en problema del agente viajero y que piensen en las posibles aplicaciones de estas nociones.

Todo el material sobre grafos en esta unidad está basado en el texto "Matemáticas Discretas", de K. Ross y C. Wright (Prentice Hall, 1990). Las reconstrucciones lógicas son nuestras.