Tema 10: Programación lógica y aprendizaje automático

José A. Alonso Jiménez

Jose-Antonio.Alonso@cs.us.es
http://www.cs.us.es/∼jalonso

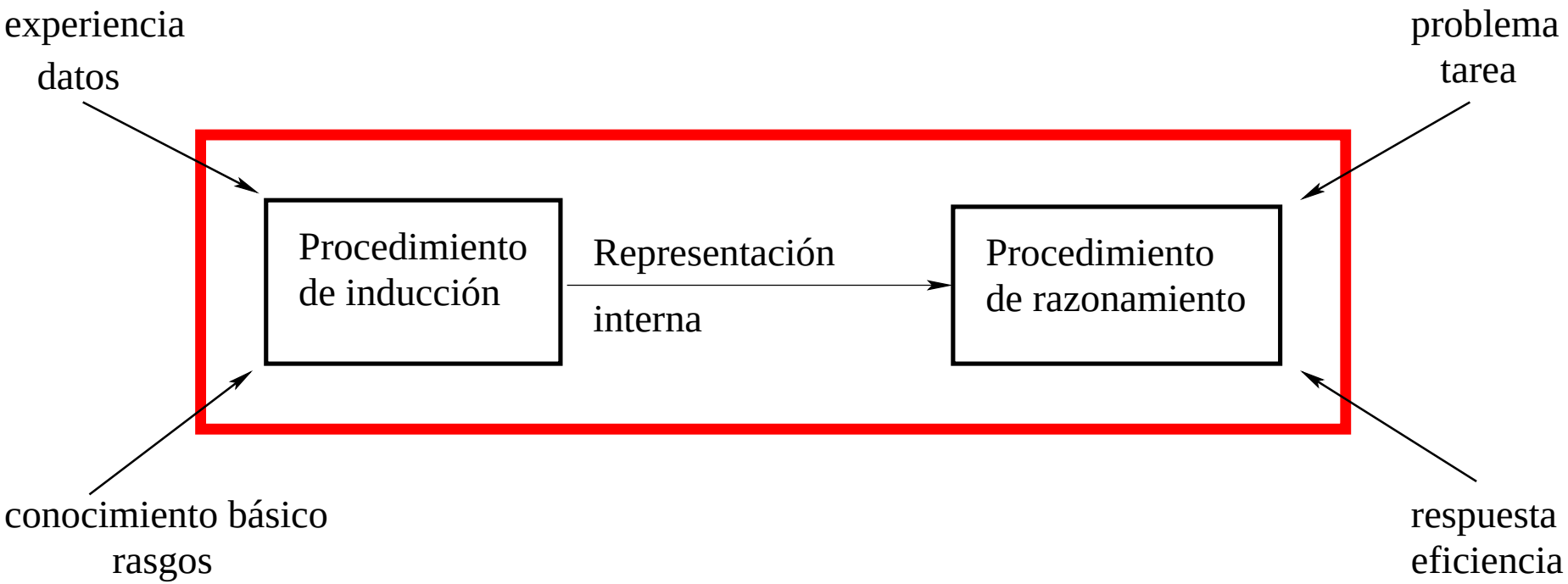
Dpto. de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Aprendizaje automático

- **El aprendizaje automático estudia cómo contruir programas que mejoren automáticamente con la experiencia.**
- **Aspectos a mejorar:**
  - **La amplitud: hacer más tareas.**
  - **La calidad: hacer mejor.**
  - **La eficiencia: hacer más rápido.**

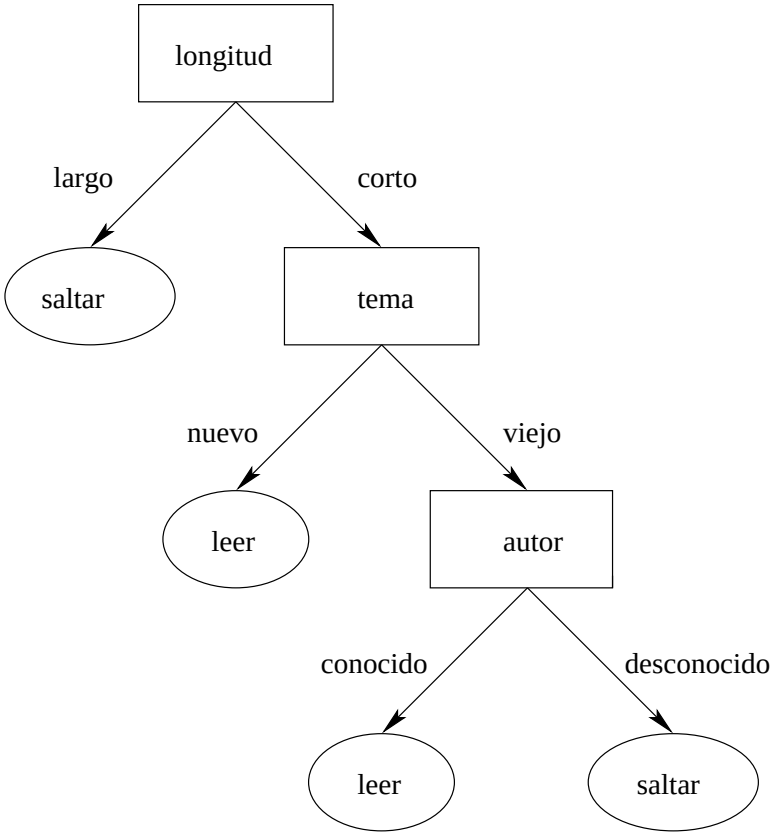
Arquitectura de aprendizaje automático



Ejemplo de datos

Ejemplo	Acción	Autor	Tema	Longitud	Sitio
e1	saltar	conocido	nuevo	largo	casa
e2	leer	desconocido	nuevo	corto	trabajo
e3	saltar	desconocido	viejo	largo	trabajo
e4	saltar	conocido	viejo	largo	casa
e5	leer	conocido	nuevo	corto	casa
e6	saltar	conocido	viejo	largo	trabajo
e7	saltar	desconocido	viejo	corto	trabajo
e8	leer	desconocido	nuevo	corto	trabajo
e9	saltar	conocido	viejo	largo	casa
e10	saltar	conocido	nuevo	largo	trabajo
e11	saltar	desconocido	viejo	corto	casa
e12	saltar	conocido	nuevo	largo	trabajo
e13	leer	conocido	viejo	corto	casa
e14	leer	conocido	nuevo	corto	trabajo
e15	leer	conocido	nuevo	corto	casa
e16	leer	conocido	viejo	corto	trabajo
e17	leer	conocido	nuevo	corto	casa
e18	leer	desconocido	nuevo	corto	trabajo

Árbol de decisión



Descripción de árboles de decisión

- **Elementos del árbol de decisión:**
  - **Nodos distintos de las hojas: atributos.**
  - **Arcos: posibles valores del atributo.**
  - **Hojas: clasificaciones.**
- **Reglas correspondientes a un árbol de decisión:**

```
longitud(E)=largo -> acción(E)=saltar
longitud(E)=corto y trama(E)=nuevo -> acción(E)=leer
longitud(E)=corto y trama(E)=viejo y autor(E)=conocido -> acción(E)=leer
longitud(E)=corto y trama(E)=viejo y autor(E)=desconocido -> acción(E)=saltar
```

- **Fórmula correspondiente a un árbol de decisión.**

Cuestiones sobre árboles de decisión

- ¿Cuál es el árbol de decisión correcto?:
  - Navaja de Occam.
  - El mundo es inherentemente simple.
  - El árbol de decisión más pequeño consistente con la muestra es el que tiene más probabilidades de identificar objetos desconocidos de manera correcta.
  - Menor profundidad.
  - Menor número de nodos.
- ¿Cómo se puede construir el árbol de decisión más pequeño?
  - Búsqueda en escalada

Búsqueda del árbol de decisión

- **¿Están todos los ejemplos en la misma clase?**

```
Ejemplos positivos = {E: acción(E)=leer} =
                       = {e2, e5, e8, e13, e14, e15, e16, e17, e18}
Ejemplos negativos = {E: acción(E)=saltar} =
                       = {e1, e3, e4, e6,  e7,  e9,  e10, e11, e12}
P = número de ejemplos positivos = 9
N = número de ejemplos negativos = 9
T = número total de ejemplos = P + N = 18
```

- **Información**
  - **Fórmula:** $\boldsymbol{I(P,N) = \begin{cases} 0, & \text{si } N*P=0; \\ -\frac{P}{T}\log_2\frac{P}{T} - \frac{N}{T}\log_2\frac{N}{T}, & \text{si } N*P\neq 0. \end{cases}}$
  - **Ejemplo:** $\boldsymbol{I(9,9) = -\frac{9}{18}\log_2\frac{9}{18} - \frac{9}{18}\log_2\frac{9}{18} = 1}$

Búsqueda del árbol de decisión

- **Información tras la división por un `Atributo`:**

  $$I = \frac{N1*I1+N2*I2}{N1+N2},$$

  donde

  - **N1 = número de ejemplos en la clase 1.**
  - **N2 = número de ejemplos en la clase 2.**
  - **I1 = cantidad de información en los ejemplos de la clase 1.**
  - **I2 = cantidad de información en los ejemplos de la clase 2.**

Búsqueda del árbol de decisión

- **Ganancia de información al dividir por autor.**
  - **Distribución:**

	leer	saltar
conocido	05,13,14,15,16,17	01,04,06,09,10 12
desconocido	02,08,18	03,07,11

  - **Información de** `autor(E)=conocido`: $\boldsymbol{I(6,6) = -\frac{6}{12}\log_2\frac{6}{12} - \frac{6}{12}\log_2\frac{6}{12} = 1}$
  - **Información de** `autor(E)=desconocido`: $\boldsymbol{I(3,3) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1}$
  - **Información de la división por** `autor`: $\boldsymbol{\frac{12*1+6*1}{12+6} = 1}$

Búsqueda del árbol de decisión

- **Ganancia de información al dividir por tema.**
  - **Distribución:**

	leer	saltar
nuevo	02,05,08,14,15,17,18	01,10,12
viejo	13,16	03,04,06,07,09,11

  - **Información de `tema(E)=nuevo`:** $\boldsymbol{I(7,3) = -\frac{7}{10}\log_2\frac{7}{10} - \frac{3}{10}\log_2\frac{3}{10} = 0.881}$
  - **Información de `tema(E)=viejo`:** $\boldsymbol{I(2,6) = -\frac{2}{8}\log_2\frac{2}{8} - \frac{6}{8}\log_2\frac{6}{8} = 0.811}$
  - **Información de la división por `tema`:** $\boldsymbol{\frac{10*0.881+8*0.811}{18} = 0.850}$

Búsqueda del árbol de decisión

- **Ganancia de información al dividir por longitud.**
  - **Distribución:**

	leer	saltar
largo		01,03,04,06,09,10,12
corto	02,05,08,13,14,15,16,17,18	07,11

  - **Información de** `longitud(E)=largo`: $I(0,7) = 0$
  - **Información de** `longitud(E)=corto`: $I(9,2) = -\frac{9}{11}\log_2\frac{9}{11} - \frac{2}{11}\log_2\frac{2}{11} = 0.684$
  - **Información de la división por** `longitud`: $\frac{7*0+11*0.684}{18} = 0.418$

Búsqueda del árbol de decisión

- **Ganancia de información al dividir por sitio de lectura.**
  - **Distribución:**

	leer	saltar
casa	05,13,15,17	01,04,09,11
trabajo	02,08,14,16,18	03,06,07,10,12

  - **Información de `sitio(E)=casa`:** $\boldsymbol{I(4,4) = -\frac{4}{8}\log_2\frac{4}{8} - \frac{4}{8}\log_2\frac{4}{8} = 1}$
  - **Información de `sitio(E)=trabajo`:** $\boldsymbol{I(5,5) = -\frac{5}{11}\log_2\frac{5}{11} - \frac{5}{11}\log_2\frac{1}{11} = 1}$
  - **Información de la división por `sitio`:** $\boldsymbol{\frac{8*1+10*1}{18} = 1}$

Búsqueda del árbol de decisión

- **Conclusiones:**
  - **Mejor atributo para dividir: longitud**
  - `Clase_1 = {E: longitud(E)=largo} = {} U {1,3,4,6,9,10,12}`
  - `Clase_2 = {E: longitud(E)=corto} = {2,5,8,13,14,15,16,17,18} U {7,11}`
- **Clasificación de `Clase_1`:**
  - **Están todos los ejemplos en la misma clase: `acción(E)=saltar`.**
- **Clasificación de `Clase_2`:**
  - **No están todos los ejemplos en la misma clase.**
  - **Repetir sobre `Clase_2` con los restantes atributos (autor, tema y sitio).**
  - **Información de `longitud(E)=corto`: $I(9,2) = -\frac{9}{11}\log_2\frac{9}{11} - \frac{2}{11}\log_2\frac{2}{11} = 0.684$**

Búsqueda del árbol de decisión

- **Ganancia de información al dividir `Clase_2` por autor.**
  - **Distribución:**

	leer	saltar
conocido	05,13,14,15,16,17	
desconocido	02,08,18	07,11

  - **Información de `autor(E)=conocido`:** $I(6,0) = 0$
  - **Información de `autor(E)=desconocido`:** $I(3,2) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$
  - **Información de la división por `autor`:** $\frac{6*0+5*0.971}{11} = 0.441$

Búsqueda del árbol de decisión

- **Ganancia de información al dividir `Clase_2` por tema.**
  - **Distribución:**

	`leer`	`saltar`
`nuevo`	`02,05,08,14,15,17,18`	
`viejo`	`13,16`	`07,11`

  - **Información de `tema(E)=nuevo`:** $\boldsymbol{I(7,0) = 0}$
  - **Información de `tema(E)=viejo`:** $\boldsymbol{I(2,2) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1}$
  - **Información de la división por `tema`:** $\boldsymbol{\frac{7*0+4*1}{11} = 0.364}$

Búsqueda del árbol de decisión

- **Ganancia de información al dividir `Clase_2` por sitio de lectura.**
  - **Distribución:**

	leer	saltar
casa	05,13,15,17	11
trabajo	02,08,14,16,18	07

  - **Información de `sitio(E)=casa`:** $\boldsymbol{I(4,1) = -\frac{4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5} = 0.722}$
  - **Información de `sitio(E)=trabajo`:** $\boldsymbol{I(5,1) = -\frac{5}{6}\log_2\frac{5}{6} - \frac{1}{6}\log_2\frac{1}{6} = 0.650}$
  - **Información de la división por `sitio`:** $\boldsymbol{\frac{5*0.722+6*0.650}{11} = 0.683}$

Búsqueda del árbol de decisión

- **Conclusiones:**
  - **Mejor atributo para dividir** `Clase_2`**: tema.**
  - `Clase_3 = {E: longitud(E)=corto, tema(E)=nuevo} = {2,5,8,14,15,17,18} U {}`
  - `Clase_4 = {E: longitud(E)=corto, tema(E)=viejo} = {13,16} U {7,11}`
- **Clasificación de** `Clase_3`**:**
  - **Están todos los ejemplos en la misma clase:** `acción(E)=leer`**.**
- **Clasificación de** `Clase_4`**:**
  - **No están todos los ejemplos en la misma clase.**
  - **Repetir sobre** `Clase_4` **con los restantes atributos (autor y sitio).**
  - **Información de** `longitud(E)=corto, tema(E)=viejo`**:** $\boldsymbol{I(2,2) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1}$

Búsqueda del árbol de decisión

- **Ganancia de información al dividir `Clase_4` por autor.**
  - **Distribución:**

```
             | leer   | saltar
-------------+--------+-------
conocido     | 13,16 |
desconocido |        | 07,11
```

  - **Información de `autor(E)=conocido`:** $I(2,0) = 0$
  - **Información de `autor(E)=desconocido`:** $I(0,2) = 0$
  - **Información de la división por `autor`:** $\frac{2*0+2*0}{4} = 0$

Búsqueda del árbol de decisión

- **Ganancia de información al dividir `Clase_4` por sitio de lectura.**
  - **Distribución:**

	leer	saltar
casa	13	11
trabajo	16	07

  - **Información de `sitio(E)=casa`:** $I(1,1) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$
  - **Información de `sitio(E)=trabajo`:** $I(1,1) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$
  - **Información de la división por `sitio`:** $\frac{2*1+2*1}{4} = 1$

Búsqueda del árbol de decisión

- **Conclusiones:**
  - **Mejor atributo para dividir** `Clase_4`**: autor.**
  - `Clase_5 = {E: longitud(E)=corto, tema(E)=nuevo, autor(E)= conocido} = {13,16} U {}`.
  - `Clase_6 = {E: longitud(E)=corto, tema(E)=viejo, autor(E)= desconocido} = {} U {7,11}`.
- **Clasificación de** `Clase_5`**:**
  - **Están todos los ejemplos en la misma clase:** `acción(E)=leer`.
- **Clasificación de** `Clase_6`**:**
  - **Están todos los ejemplos en la misma clase:** `acción(E)=saltar`.

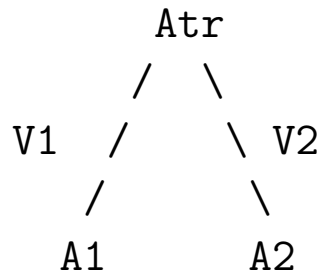
Algoritmo ID3

- **Sesión.**

```
?- ['aprende_ad.pl','aprende_ad_e1.pl'].
Yes

?- aprende_ad(acción,
                  [e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,e11,e12,e13,e14,e15,e16,e17,e18],
                  [autor,tema,longitud,sitio],
                  AD).
AD = si(longitud=largo, saltar,
            si(tema=nuevo, leer,
               si(autor=desconocido, saltar,
                  leer)))
Yes
```

Algoritmo ID3

1. **Si todos los ejemplos coinciden (es decir, si todos tienen el mismo valor en el atributo objetivo, entonces el árbol de decisión se reduce a una hoja con dicho valor.**
2. **En caso contrario,**
   (a) **elegir el atributo (`Atr`) con mayor ganancia de información,**
   (b) **sea `C1` el conjunto de ejemplos donde el atributo seleccionado toma un valor (`V1`) y `C2` el de los ejemplos donde toma el otro valor (`V2`),**
   (c) **sea `A1` el árbol construido aplicando el procedimiento a los ejemplos de `C1` y con el mismo atributo objetivo y `A2` el árbol construido aplicando el procedimiento a los ejemplos de `C2`,**
   (d) **el árbol de decisión es**



Algoritmo ID3

- **Representación del problema** `aprende_ad_e1.pl`**.**
  - `val(Objeto,Atributo,Valor)` **se verifica si el valor del** `Atributo` **del** `Objeto` **es** `Valor`**.**

```
val(e1,acción,saltar).
val(e1,autor,conocido).
val(e1,tema,nuevo).
val(e1,longitud,largo).
val(e1,sitio,casa ).

.........................

val(e18,acción,leer).
val(e18,autor,desconocido).
val(e18,tema,nuevo).
val(e18,longitud,corto).
val(e18,sitio,trabajo).
```

Algoritmo ID3

- **Algoritmo de aprendizaje de árboles de decisión.**
  - `aprende_ad(+Objetivo,+Ejemplos,+Atributos,-AD)` **se verifica si** `AD` **es el árbol de decisión inducido para el** `Objetivo` **a partir de la lista de** `Ejemplos` **y** `Atributos`**.**

```
aprende_ad(Objetivo, Ejemplos, _ , Val) :-
   coinciden_todos_ejemplos(Objetivo, Ejemplos, Val).
aprende_ad(Objetivo, Ejemplos, Atributos, si(Atr=ValPos,APos,ANeg)) :-
   not(coinciden_todos_ejemplos(Objetivo, Ejemplos, _)),
   selecciona_division(Objetivo, Ejemplos, Atributos, Atr, Resto_Atr),
   divide(Ejemplos, Atr, ValPos, Positivos, Negativos),
   aprende_ad(Objetivo, Positivos, Resto_Atr, APos),
   aprende_ad(Objetivo, Negativos, Resto_Atr, ANeg).
```

Algoritmo ID3

- **¿Están todos los ejemplos en la misma clase?**
  - `coindiden_todos_ejemplos(+Objetivo,+Ejemplos,-Valor)` **se verifica si el valor del atributo** `Objetivo` **de todos los** `Ejemplos` **es** `Valor`**.**

```
coinciden_todos_ejemplos(_,[],_).
coinciden_todos_ejemplos(Atributo,[Obj|Resto],Val) :-
   val(Obj,Atributo,Val),
   coinciden_todos_ejemplos(Atributo,Resto,Val).
```

Algoritmo ID3

- **Selección del mejor atributo para dividir.**
  - `selecciona_division(+Objetivo,+Ejemplos,+Atributos, -Atributo,-Restantes_atributos)` **se verifica si** `Atributo` **es el mejor elemento de la lista de** `Atributos` **para determinar el** `Objetivo` **a partir de los** `Ejemplos` **(es decir, la información resultante del** `Objetivo` **en los** `Ejemplos` **usando como división el** `Atributo` **es mínima), y** `Restantes_atributos` **es la lista de los restantes** `Atributos`**. Falla si para ningún atributo se gana en información.**

```
selecciona_division(Objetivo, Ejemplos, [A|R], Atributo, Resto_Atr) :-
    informacion_division(Objetivo,Ejemplos,A,I),
    selecciona_max_ganancia_informacion(Objetivo,Ejemplos,R,A,I,
                                        Atributo,[],Resto_Atr).
```

Algoritmo ID3

- `informacion_division(+Objetivo,+Ejemplos,+Atributo,-I)` **se verifica si** `I` **es la información resultante del** `Objetivo` **en los** `Ejemplos` **usando como división el** `Atributo`**; es decir,** `I = (N1*I1 + N2*I2) / (N1+N2)`.

```
informacion_division(Objetivo,Ejemplos,Atributo,Inf) :-
   divide(Ejemplos,Atributo,_,Clase_1,Clase_2),
   informacion(Objetivo,Clase_1,I1),
   informacion(Objetivo,Clase_2,I2),
   length(Clase_1,N1),
   length(Clase_2,N2),
   Inf is (N1*I1 + N2*I2)/(N1+N2).
```

Algoritmo ID3

- `informacion(+Objetivo,+Ejemplos,-I)` **se verifica si** `I` **es la cantidad de información en los** `Ejemplos` **respecto del** `Objetivo`**; es decir,**

$$\boldsymbol{I} = \begin{cases} \mathbf{0}, & \textbf{si } \boldsymbol{N} * \boldsymbol{P} = \mathbf{0}; \\ -\frac{\boldsymbol{P}}{\boldsymbol{T}} \log_2 \frac{\boldsymbol{P}}{\boldsymbol{T}} - \frac{\boldsymbol{N}}{\boldsymbol{T}} \log_2 \frac{\boldsymbol{N}}{\boldsymbol{T}}, & \textbf{si } \boldsymbol{N} * \boldsymbol{P} \neq \mathbf{0}. \end{cases}$$

```
informacion(Objetivo,Ejemplos,I) :-
   cuenta(Objetivo,_,Ejemplos,NP,NN),
   ( (NP=0 ; NN=0) ->
       I=0
   ; true ->
       NT is NP + NN,
       I is - NP/NT * log2(NP/NT) - NN/NT * log2(NN/NT)).
```

Algoritmo ID3

- `cuenta(+Atributo,?VP,+Ejemplos,-NP,-NN)` **se verifica si** `NP` **es el número de ejemplos positivos (es decir, elementos de** `Ejemplos` **tales que el valor de su** `Atributo` **es** `VP` **(valor positivo)) y** `NN` **es el número de ejemplos negativos.**

```
cuenta(_,_,[],0,0).
cuenta(Atributo,VP,[E|R],NP,NN) :-
   val(E,Atributo,VP),
   cuenta(Atributo,VP,R,NP1,NN),
   NP is NP1+1.
cuenta(Atributo,VP,[E|R],NP,NN) :-
   val(E,Atributo,VNeg),
   not(VP=VNeg),
   cuenta(Atributo,VP,R,NP,NN1),
   NN is NN1+1.
```

Algoritmo ID3

- `selecciona_max_ganancia_informacion(+Objetivo, +Ejemplos, +Atributos, +Mejor_atributo_actual, +Mejor_info_actual, -Atributo, +Atributos_analizados, -Resto_atributos)` **se verifica si** `Atributo` **es el elemento de** `Atributos` **tal la información resultante del** `Objetivo` **en los** `Ejemplos` **usando como división el** `Atributo` **es mínima.**

```
selecciona_max_ganancia_informacion(_,_,[],MejorA,_,
                                    MejorA, A_analizados, A_analizados).
selecciona_max_ganancia_informacion(Objetivo, Ejs, [A|R], MejorA, MejorI,
                                    Atributo, A_analizados, Resto_Atr) :-
   informacion_division(Objetivo,Ejs,A,Informacion),
   ( Informacion > MejorI ->
       selecciona_max_ganancia_informacion(
         Objetivo,Ejs,R,MejorA,MejorI,Atributo,[A|A_analizados],Resto_Atr)
   ;  true ->
       selecciona_max_ganancia_informacion(
         Objetivo,Ejs,R,A,Informacion,Atributo,[MejorA|A_analizados],Resto_Atr)
   ).
```

Algoritmo ID3

- **División de los ejemplos**
  - `divide(+Ejemplos,+Atributo,?Valor,-Positivos,-Negativos)` **se verifica si** `Positivos` **es la lista de elementos de** `Ejemplos` **tales que el valor de** `Atributo` **es** `Valor` **y** `Negativos` **es la lista de los restantes** `Ejemplos`**.**

```
divide([],_,_,[],[]).
divide([Ej|Rest],Atributo,ValPos,[Ej|Positivos],Negativos) :-
   val(Ej,Atributo,ValPos),
   divide(Rest,Atributo,ValPos,Positivos,Negativos).
divide([Ej|Rest],Atributo,ValPos,Positivos,[Ej|Negativos]) :-
   val(Ej,Atributo,ValNeg),
   not(ValNeg = ValPos),
   divide(Rest,Atributo,ValPos,Positivos,Negativos).
```

Sistemas TDIDP

- **Los sistemas basados en árboles de decisión forman una familia llamada TDIDT (*Top–Down Induction of Decision Tree*).**
- **Representantes de TDIDT:**
  - **ID3 (Interactive Dichotomizer) [Quinlan, 1986].**
  - **C4.5 [Quinlan, 93] es una variante de ID3 que permite clasificar ejemplos con atributos que toman valores continuos.**
- **Quinlan, J. R. *C4.5: Programs for Machine Learning* (Morgan Kaufmann, 1993).**

Limitaciones de árboles de decisión

- **Una representación formal limitada (lenguaje de pares atributo–valor equivalente al de la lógica proposicional).**
- **Tienden a ser demasiado grandes en aplicaciones reales.**
- **Dificultad del manejo del conocimiento base.**

Programación Lógica Inductiva

- **Datos:**
  - **Ejemplos positivos: $E^{\oplus}$.**
  - **Ejemplos negativos: $E^{\ominus}$.**
  - **Conocimiento base: T.**
  - **Lenguaje de hipótesis: L.**
- **Condiciones:**
  - ***Necesidad a priori:*** $(\exists e^{\oplus} \in E^{\oplus})[T \not\vdash e^{\oplus}]$.
  - ***Consistencia a priori:*** $(\forall e^{\ominus} \in E^{\ominus})[T \not\vdash e^{\ominus}]$.
- **Objetivo:**
  - **Encontrar un conjunto finito $H \subset L$ tal que se cumplan**
    - ***Suficiencia a posteriori:*** $(\forall e^{\oplus} \in E^{\oplus})[T \bigcup H \vdash e^{\oplus}]$.
    - ***Consistencia a posteriori:*** $(\forall e^{\ominus} \in E^{\ominus})[T \bigcup H \not\vdash e^{\ominus}]$.

Ejemplo con FOIL

- **Descripción del problema `familia.pl`.**
  - **Ejemplos positivos:**

```
padre(carlos,juan).        padre(carlos,eva).
```

  - **Ejemplos negativos:**

```
no(padre(carlos,aurora)).  no(padre(aurora,juan)).
```

  - **Conocimiento básico:**

```
hombre(carlos).            hombre(juan).
mujer(eva).                mujer(aurora).
progenitor(carlos,juan).   progenitor(carlos,eva).
progenitor(aurora,juan).   progenitor(aurora,eva).
```

  - **Parámetros:**

```
foil_predicates([padre/2,hombre/1,mujer/1,progenitor/2]).
foil_cwa(false).           % No usa la hipótesis del mundo cerrado
foil_use_negations(false). % No usa información negativa en el cuerpo
foil_det_lit_bound(0).     % No añade literales determinados
```

Ejemplo con FOIL

- **Sesión:**

```
?- [foil, familia].
Yes

?- foil(padre/2).

Uncovered positives: [padre(carlos, juan), padre(carlos, eva)]

Adding a clause ...
Specializing current clause: padre(A, B).

Covered negatives: [padre(carlos, aurora), padre(aurora, juan)]

Covered positives: [padre(carlos, juan), padre(carlos, eva)]
```

Ejemplo con FOIL

```
Ganancia:  0.830 Cláusula: padre(A, B):-hombre(A)
Ganancia:  0.000 Cláusula: padre(A, B):-hombre(B)
Ganancia:  0.000 Cláusula: padre(A, B):-mujer(A)
Ganancia:  0.000 Cláusula: padre(A, B):-mujer(B)
Ganancia:  0.000 Cláusula: padre(A, B):-progenitor(C, A)
Ganancia:  0.000 Cláusula: padre(A, B):-progenitor(A, C)
Ganancia:  0.830 Cláusula: padre(A, B):-progenitor(C, B)
Ganancia:  0.000 Cláusula: padre(A, B):-progenitor(B, C)
Ganancia:  0.000 Cláusula: padre(A, B):-progenitor(A, A)
Ganancia:  0.000 Cláusula: padre(A, B):-progenitor(B, A)
Ganancia:  0.830 Cláusula: padre(A, B):-progenitor(A, B)
Ganancia:  0.000 Cláusula: padre(A, B):-progenitor(B, B)

Specializing current clause: padre(A, B) :- hombre(A).

Covered negatives: [padre(carlos, aurora)]

Covered positives: [padre(carlos, juan), padre(carlos, eva)]
```

Ejemplo con FOIL

```
Ganancia:  0.000 Cláusula: padre(A, B):-hombre(A), hombre(A)
Ganancia:  0.585 Cláusula: padre(A, B):-hombre(A), hombre(B)
Ganancia:  0.000 Cláusula: padre(A, B):-hombre(A), mujer(A)
Ganancia: -0.415 Cláusula: padre(A, B):-hombre(A), mujer(B)
Ganancia:  0.000 Cláusula: padre(A, B):-hombre(A), progenitor(C, A)
Ganancia:  0.000 Cláusula: padre(A, B):-hombre(A), progenitor(A, C)
Ganancia:  1.170 Cláusula: padre(A, B):-hombre(A), progenitor(C, B)
Ganancia:  0.000 Cláusula: padre(A, B):-hombre(A), progenitor(B, C)
Ganancia:  0.000 Cláusula: padre(A, B):-hombre(A), progenitor(A, A)
Ganancia:  0.000 Cláusula: padre(A, B):-hombre(A), progenitor(B, A)
Ganancia:  1.170 Cláusula: padre(A, B):-hombre(A), progenitor(A, B)
Ganancia:  0.000 Cláusula: padre(A, B):-hombre(A), progenitor(B, B)

Clause found: padre(A, B) :- hombre(A), progenitor(A, B).

Found definition: padre(A, B) :- hombre(A), progenitor(A, B).
```

Ganancia de información en FOIL

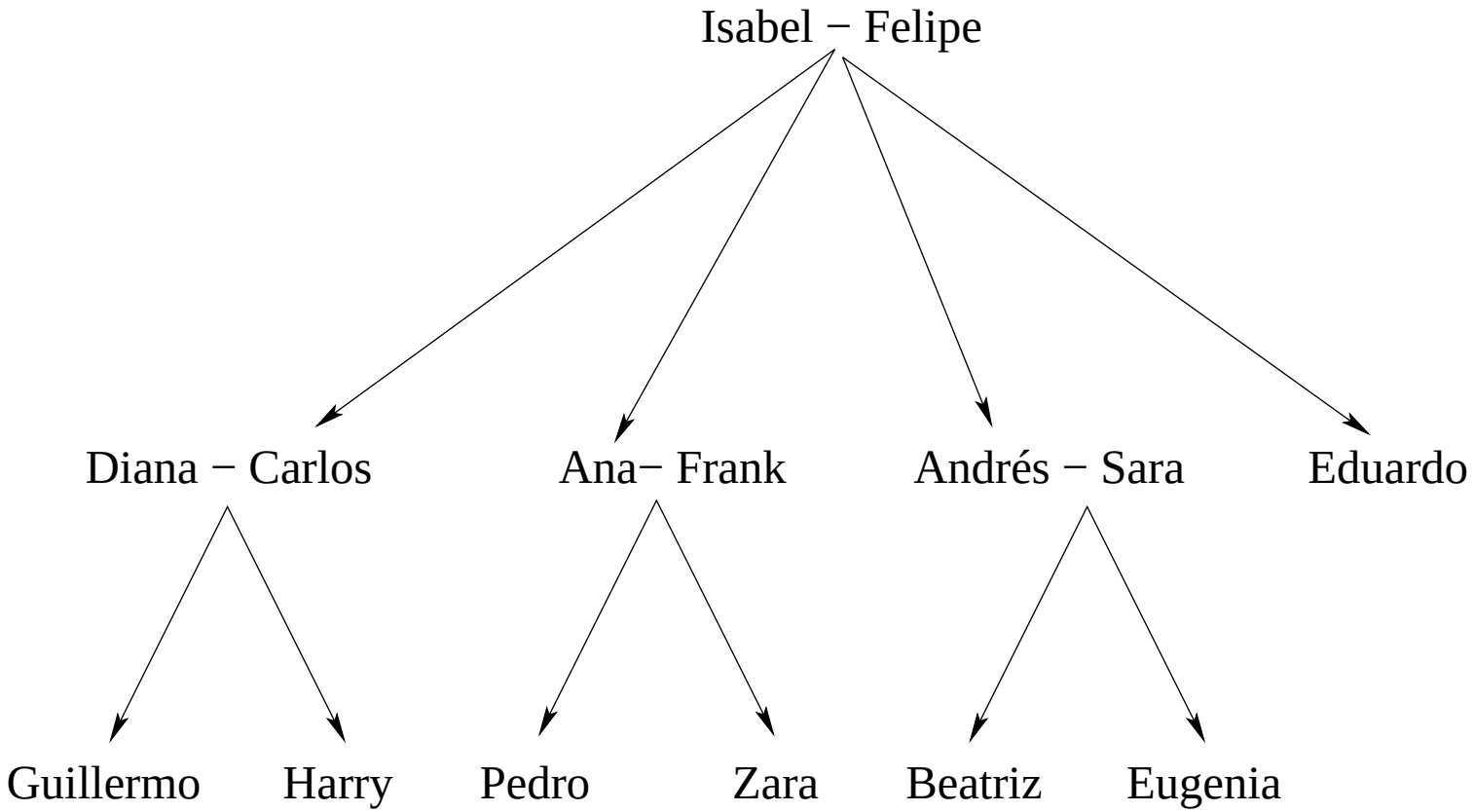
- **Información correspondiente a una cláusula:**
  Sean P el número de ejemplos positivos cubiertos por la cláusula y N el número de ejemplos negativos cubiertos por la cláusula.

$$I(C) = \begin{cases} 0, & \text{si } P = 0; \\ -\log_2 \frac{P}{P+N}, & \text{si } P \neq 0. \end{cases}$$

- **Ganancia de información de la cláusula C_1 a la cláusula C_2:**
  Sean P el número de ejemplos positivos cubiertos por C_2 y N el número de ejemplos negativos cubiertos C_2.

$$G(C_1, C_2) = \begin{cases} 0, & \text{si } P = 0; \\ P * (I(C_1) - I(C_2)), & \text{si } P \neq 0. \end{cases}$$

Ejemplo de FOIL con CWA



Ejemplo de FOIL con CWA

- **Representación** `familia_1.pl`**:**
  - **Ejemplos positivos**

```
padre(felipe,carlos).       padre(felipe,ana).      padre(felipe,andres).   ...
madre(isabel,carlos).       madre(isabel,ana).      madre(isabel,andres).   ...
abuelo(felipe,guillermo).   abuelo(felipe,harry).   abuelo(felipe,pedro).   ...
```

  - **Parámetros**

```
foil_predicates([padre/2, madre/2, abuelo/2]).
foil_cwa(true).                 % Usa la hipótesis del mundo cerrado
foil_use_negations(false).      % No usa información negativa en el cuerpo
foil_det_lit_bound(0).          % No añade literales determinados
```

Ejemplo de FOIL con CWA

- **Sesión**

```
?- [foil, familia_1].
Yes
?- foil(abuelo/2).

Uncovered positives: [(felipe,guillermo),(felipe,harry),(felipe,pedro),
                      (felipe,zara),(felipe,beatriz),(felipe,eugenia)]

Adding a clause ...
Specializing current clause: abuelo(A,B).

Covered negatives: [(ana,ana),(ana,andres),...]
Covered positives: [(felipe,guillermo),(felipe,harry),(felipe,pedro),
                    (felipe,zara),(felipe,beatriz),(felipe,eugenia)]
```

Ejemplo de FOIL con CWA

```
Ganancia:  0.000 Cláusula: abuelo(A,B):-padre(C,A)
Ganancia: 15.510 Cláusula: abuelo(A,B):-padre(A,C)
Ganancia:  3.510 Cláusula: abuelo(A,B):-padre(C,B)
Ganancia:  0.000 Cláusula: abuelo(A,B):-padre(B,C)
Ganancia:  0.000 Cláusula: abuelo(A,B):-padre(A,A)
Ganancia:  0.000 Cláusula: abuelo(A,B):-padre(B,A)
Ganancia:  0.000 Cláusula: abuelo(A,B):-padre(A,B)
Ganancia:  0.000 Cláusula: abuelo(A,B):-padre(B,B)
Ganancia:  0.000 Cláusula: abuelo(A,B):-madre(C,A)
Ganancia:  0.000 Cláusula: abuelo(A,B):-madre(A,C)
Ganancia:  3.510 Cláusula: abuelo(A,B):-madre(C,B)
Ganancia:  0.000 Cláusula: abuelo(A,B):-madre(B,C)
Ganancia:  0.000 Cláusula: abuelo(A,B):-madre(A,A)
Ganancia:  0.000 Cláusula: abuelo(A,B):-madre(B,A)
Ganancia:  0.000 Cláusula: abuelo(A,B):-madre(A,B)
Ganancia:  0.000 Cláusula: abuelo(A,B):-madre(B,B)

Specializing current clause: abuelo(A,B) :- padre(A,C).
```

Ejemplo de FOIL con CWA

```
Covered negatives: [(andres,ana),(andres,andres),...]
Covered positives: [(felipe,guillermo),(felipe,harry),(felipe,pedro),
                    (felipe,zara),(felipe,beatriz),(felipe,eugenia)]

Ganancia:  3.087 Cláusula: abuelo(A,B):-padre(A,C),padre(A,D)
Ganancia:  3.510 Cláusula: abuelo(A,B):-padre(A,C),padre(D,B)
Ganancia:  7.932 Cláusula: abuelo(A,B):-padre(A,C),padre(C,D)
Ganancia: 10.575 Cláusula: abuelo(A,B):-padre(A,C),padre(C,B)
Ganancia:  3.510 Cláusula: abuelo(A,B):-padre(A,C),madre(D,B)
Ganancia:  7.932 Cláusula: abuelo(A,B):-padre(A,C),madre(C,D)
Ganancia:  5.288 Cláusula: abuelo(A,B):-padre(A,C),madre(C,B)
....

Clause found: abuelo(A,B) :- padre(A,C),padre(C,B).
```

Ejemplo de FOIL con CWA

```
Uncovered positives: [(felipe,pedro),(felipe,zara)]

Adding a clause ...
Specializing current clause: abuelo(A,B).

Covered negatives: [(ana,ana),(ana,andres),...]
Covered positives: [(felipe,pedro),(felipe,zara)]

Ganancia:  5.444 Cláusula: abuelo(A,B):-padre(A,C)
Ganancia:  1.196 Cláusula: abuelo(A,B):-padre(C,B)
Ganancia:  1.196 Cláusula: abuelo(A,B):-madre(C,B)
...

Specializing current clause: abuelo(A,B) :- padre(A,C).

Covered negatives: [(andres,ana),(andres,andres),...]
Covered positives: [(felipe,pedro),(felipe,zara)]
```

Ejemplo de FOIL con CWA

```
Ganancia:  1.181 Cláusula: abuelo(A,B):-padre(A,C),padre(A,D)
Ganancia:  1.348 Cláusula: abuelo(A,B):-padre(A,C),padre(D,B)
Ganancia:  3.213 Cláusula: abuelo(A,B):-padre(A,C),padre(C,D)
Ganancia:  1.348 Cláusula: abuelo(A,B):-padre(A,C),madre(D,B)
Ganancia:  3.213 Cláusula: abuelo(A,B):-padre(A,C),madre(C,D)
Ganancia:  8.132 Cláusula: abuelo(A,B):-padre(A,C),madre(C,B)
...

Clause found: abuelo(A,B) :- padre(A,C),madre(C,B).

Found definition:
abuelo(A,B) :- padre(A,C),madre(C,B).
abuelo(A,B) :- padre(A,C),padre(C,B).
```

Ejemplo de FOIL con CWA y conocimiento básico

- **Representación** `familia_2.pl`**:**
  - **Ejemplos positivos:**

```
padre(felipe,carlos).       padre(felipe,ana).      padre(felipe,andres).   ...
madre(isabel,carlos).       madre(isabel,ana).      madre(isabel,andres).   ...
abuelo(felipe,guillermo).   abuelo(felipe,harry).   abuelo(felipe,pedro).   ...
```

  - **Conocimiento básico:**

```
progenitor(X,Y) :- padre(X,Y).
progenitor(X,Y) :- madre(X,Y).
```

  - **Parámetros:**

```
foil_predicates([padre/2, madre/2, abuelo/2, progenitor/2]).
foil_cwa(true).              % Usa la hipótesis del mundo cerrado
foil_use_negations(false).   % No usa información negativa en el cuerpo
foil_det_lit_bound(0).       % No añade literales determinados
```

Ejemplo de FOIL con CWA y conocimiento básico

- **Sesión:**

```
?- [foil, familia_2].
Yes

?- foil(abuelo/2).

Uncovered positives: [(felipe, guillermo), (felipe, harry), (felipe, pedro),
                      (felipe, zara), (felipe, beatriz), (felipe, eugenia)]

Adding a clause ...
Specializing current clause: abuelo(A, B).

Covered negatives: [(ana, ana), (ana, andres), ...]
Covered positives: [(felipe, guillermo), (felipe, harry), (felipe, pedro),
                    (felipe, zara), (felipe, beatriz), (felipe, eugenia)]
```

Ejemplo de FOIL con CWA y conocimiento básico

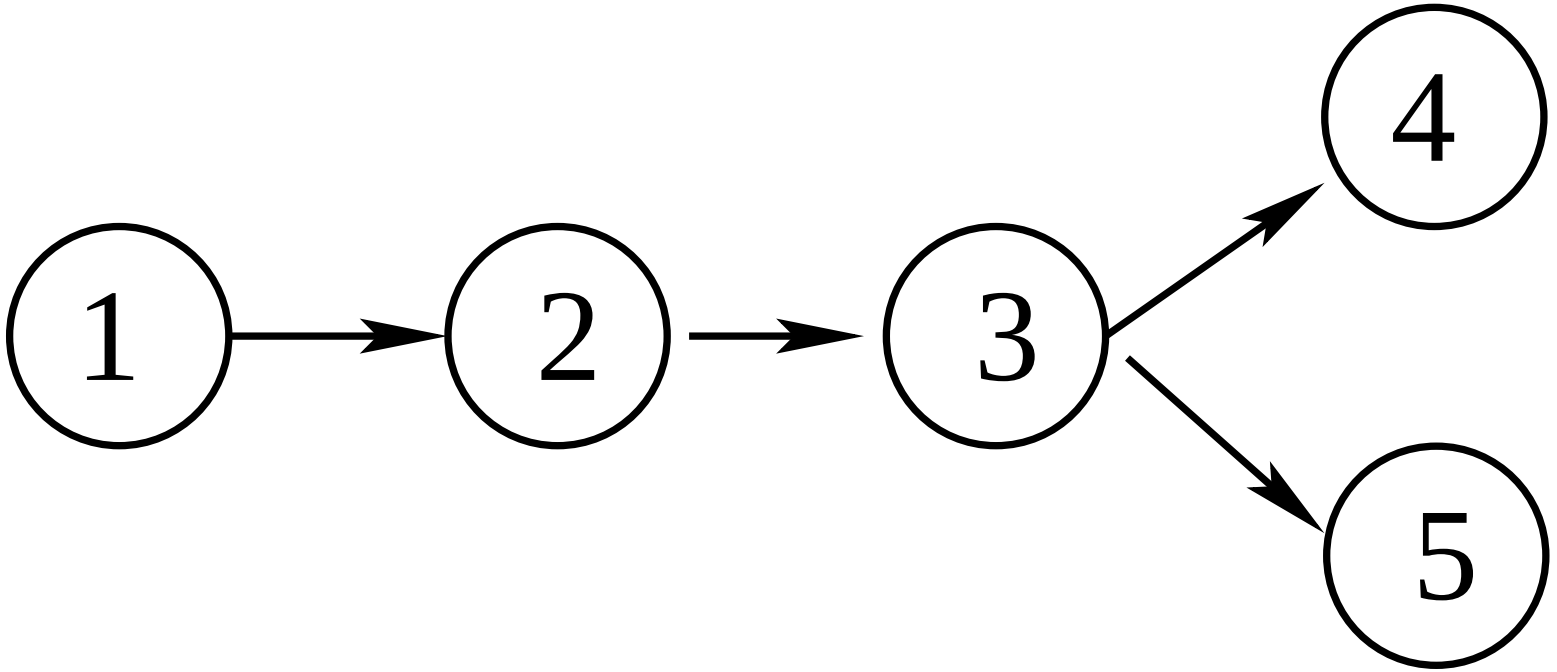
```
Ganancia: 15.510 Cláusula: abuelo(A, B):-padre(A, C)
Ganancia:  3.510 Cláusula: abuelo(A, B):-padre(C, B)
Ganancia:  3.510 Cláusula: abuelo(A, B):-madre(C, B)
Ganancia:  9.510 Cláusula: abuelo(A, B):-progenitor(A, C)
Ganancia:  3.510 Cláusula: abuelo(A, B):-progenitor(C, B)

Specializing current clause: abuelo(A, B) :- padre(A, C).

Covered negatives: [(andres, ana), (andres, andres), ...]
Covered positives: [(felipe, guillermo), (felipe, harry), (felipe, pedro),
                    (felipe, zara), (felipe, beatriz), (felipe, eugenia)]
```

Ejemplo de FOIL con CWA y conocimiento básico

```
Ganancia:  3.087 Cláusula: abuelo(A, B):-padre(A, C), padre(A, D)
Ganancia:  3.510 Cláusula: abuelo(A, B):-padre(A, C), padre(D, B)
Ganancia:  7.932 Cláusula: abuelo(A, B):-padre(A, C), padre(C, D)
Ganancia: 10.575 Cláusula: abuelo(A, B):-padre(A, C), padre(C, B)
Ganancia:  3.510 Cláusula: abuelo(A, B):-padre(A, C), madre(D, B)
Ganancia:  7.932 Cláusula: abuelo(A, B):-padre(A, C), madre(C, D)
Ganancia:  5.288 Cláusula: abuelo(A, B):-padre(A, C), madre(C, B)
Ganancia:  3.087 Cláusula: abuelo(A, B):-padre(A, C), progenitor(A, D)
Ganancia:  3.510 Cláusula: abuelo(A, B):-padre(A, C), progenitor(D, B)
Ganancia:  7.932 Cláusula: abuelo(A, B):-padre(A, C), progenitor(C, D)
Ganancia: 15.863 Cláusula: abuelo(A, B):-padre(A, C), progenitor(C, B)

Clause found: abuelo(A, B) :- padre(A, C), progenitor(C, B).

Found definition: abuelo(A, B) :- padre(A, C), progenitor(C, B).
```

Ejemplo con FOIL de relación recursiva

- Grafo



Ejemplo con FOIL de relación recursiva

- **Representación** `camino.pl`**:**
  - **Ejemplos:**

```
enlace(1,2).  enlace(2,3).  enlace(3,4).  enlace(3,5).
camino(1,2).  camino(1,3).  camino(1,4).  camino(1,5).
camino(2,3).  camino(2,4).  camino(2,5).
camino(3,4).  camino(3,5).
```

  - **Parámetros:**

```
foil_predicates([camino/2, enlace/2]).
foil_cwa(true).                % Usa la hipótesis del mundo cerrado
foil_use_negations(false).     % No usa información negativa
foil_det_lit_bound(0).         % No añade literales determinados
```

Ejemplo con FOIL de relación recursiva

- **Sesión**

```
?- [foil,camino].
?- foil(camino/2).
Uncovered positives: [(1,2),(1,3),(1,4),(1,5),(2,3),(2,4),(2,5),(3,4),(3,5)]

Adding a clause ...
Specializing current clause: camino(A,B).

Covered negatives: [(1,1),(2,1),(2,2),(3,1),(3,2),(3,3),(4,1),(4,2),(4,3),
                    (4,4),(4,5),(5,1),(5,2),(5,3),(5,4),(5,5)]
Covered positives: [(1,2),(1,3),(1,4),(1,5),(2,3),(2,4),(2,5),(3,4),(3,5)]

Ganancia: -2.630 Cláusula: camino(A,B):-enlace(C,A)
Ganancia:  5.503 Cláusula: camino(A,B):-enlace(A,C)
Ganancia:  2.897 Cláusula: camino(A,B):-enlace(C,B)
Ganancia: -1.578 Cláusula: camino(A,B):-enlace(B,C)
Ganancia:  0.000 Cláusula: camino(A,B):-enlace(A,A)
Ganancia:  0.000 Cláusula: camino(A,B):-enlace(B,A)
Ganancia:  5.896 Cláusula: camino(A,B):-enlace(A,B)
Ganancia:  0.000 Cláusula: camino(A,B):-enlace(B,B)
```

Ejemplo con FOIL de relación recursiva

```
Clause found: camino(A,B) :- enlace(A,B).

Uncovered positives: [(1,3),(1,4),(1,5),(2,4),(2,5)]

Adding a clause ...
Specializing current clause: camino(A,B).

Covered negatives: [(1,1),(2,1),(2,2),(3,1),(3,2),(3,3),(4,1),(4,2),(4,3),
                    (4,4),(4,5),(5,1),(5,2),(5,3),(5,4),(5,5)]
Covered positives: [(1,3),(1,4),(1,5),(2,4),(2,5)]

Ganancia: -2.034 Cláusula: camino(A,B):-enlace(C,A)
Ganancia:  2.925 Cláusula: camino(A,B):-enlace(A,C)
Ganancia:  1.962 Cláusula: camino(A,B):-enlace(C,B)
Ganancia: -1.017 Cláusula: camino(A,B):-enlace(B,C)
Ganancia:  0.000 Cláusula: camino(A,B):-enlace(A,A)
Ganancia:  0.000 Cláusula: camino(A,B):-enlace(B,A)
Ganancia:  0.000 Cláusula: camino(A,B):-enlace(A,B)
Ganancia:  0.000 Cláusula: camino(A,B):-enlace(B,B)
```

Ejemplo con FOIL de relación recursiva

```
Specializing current clause: camino(A,B) :- enlace(A,C).

Covered negatives: [(1,1),(2,1),(2,2),(3,1),(3,2),(3,3)]
Covered positives: [(1,3),(1,4),(1,5),(2,4),(2,5)]

Ganancia:  7.427 Cláusula: camino(A,B):-enlace(A,C),camino(C,B)
Ganancia: -1.673 Cláusula: camino(A,B):-enlace(A,C),enlace(D,A)
Ganancia: -2.573 Cláusula: camino(A,B):-enlace(A,C),enlace(A,D)
Ganancia:  2.427 Cláusula: camino(A,B):-enlace(A,C),enlace(D,B)
Ganancia: -1.215 Cláusula: camino(A,B):-enlace(A,C),enlace(B,D)
Ganancia:  3.539 Cláusula: camino(A,B):-enlace(A,C),enlace(C,D)
Ganancia:  4.456 Cláusula: camino(A,B):-enlace(A,C),enlace(C,B)

Clause found: camino(A,B) :- enlace(A,C), camino(C,B).

Found definition:
camino(A,B) :- enlace(A,C), camino(C,B).
camino(A,B) :- enlace(A,B).
```

Algoritmo de FOIL

```
% foil(+Positivos, +Objetivo, +Negativos, -Clausulas)
foil(Positivos, Objetivo, Negativos, Clausulas) :-
   foil(Positivos, Objetivo, Negativos, [], Clausulas).

foil(Positivos, Objetivo, Negativos, Acumulador, Clausulas) :-
   ( Positivos = [] -> Clausulas = Acumulador
   ; extiende_clausula(Negativos, Positivos, (Objetivo :- true), Clausula),
     ejemplos_no_cubiertos(Clausula, Positivos, Positivos1),
     foil(Positivos1, Objetivo, Negativos, [Clausula|Acumulador], Clausulas)).

% extiende_clausula(+Negativos, +Positivos, +Actual, -Clausula)
extiende_clausula(Neg0, Pos0, Clausula0, Clausula) :-
   ( Neg0 = [] -> Clausula = Clausula0
   ; genera_posibles_extensiones(Clausula0, L),
     informacion(Clausula0, Pos0, Neg0, Info),
     mejor_extension(L, Neg0, Pos0, Clausula0, Info, 0, Clausula0, Clausula1),
     Clausula0 \== Clausula1,
     ejemplos_cubiertos(Clausula1, Pos0, Pos1),
     ejemplos_cubiertos(Clausula1, Neg0, Neg1),
     extiende_clausula(Neg1, Pos1, Clausula1, Clausula))).
```

Ejemplo con Progol en espacios infinitos

- **Ejemplos positivos:**

```
p([]).          p([0]).         p([0,0]).       p([1,1]).       p([0,0,0]).
p([0,1,1]).     p([1,0,1]).     p([1,1,0]).     p([0,0,0,0]).   p([0,0,1,1]).
p([0,1,0,1]).   p([1,0,0,1]).   p([0,1,1,0]).   p([1,0,1,0]).   p([1,1,0,0]).
p([1,1,1,1]).
```

- **Ejemplos negativos:**

```
:- p([1]).          :- p([0,1]).        :- p([1,0]).        :- p([0,0,1]).
:- p([0,1,0]).      :- p([1,0,0]).      :- p([1,1,1]).      :- p([0,0,0,1]).
:- p([0,0,1,0]).    :- p([0,1,0,0]).    :- p([1,0,0,0]).    :- p([0,1,1,1]).
:- p([1,0,1,1]).    :- p([1,1,0,1]).    :- p([1,1,1,0]).
```

Ejemplo con Progol en espacios infinitos

- **Modos:**

```
:- modeh(1,p(+lista_binaria))?
:- modeb(1,+constant = #constant)?
:- modeb(1,+lista_binaria = [-binario|-lista_binaria])?
:- modeb(1,p(+lista_binaria))?
:- modeb(1,not(p(+lista_binaria)))?
```

- **Tipos:**

```
lista_binaria([]).
lista_binaria([X|Y]) :- binario(X), lista_binaria(Y).

binario(0).
binario(1).
```

Ejemplo con Progol en espacios infinitos

- **Sesión:**

```
> progol par
CProgol Version 4.4
...
[Testing for contradictions]
[No contradictions found]
[Generalising p([]).]
[Most specific clause is]

p(A) :- A=[].

[C:0,16,15,0 p(A).]
[1 explored search nodes]
f=0,p=16,n=15,h=0
[No compression]
```

Ejemplo con Progol en espacios infinitos

```
[Generalising p([0]).]
[Most specific clause is]

p(A) :- A=[B|C], B=0, C=[], p(C).

[C:0,16,15,0 p(A).]
[C:-2,15,15,0 p(A) :- A=[B|C].]
[C:-4,8,7,0 p(A) :- A=[B|C], B=0.]
[C:-6,8,8,0 p(A) :- A=[B|C], p(C).]
[C:8,8,0,0 p(A) :- A=[B|C], B=0, p(C).]
[5 explored search nodes]
f=8,p=8,n=0,h=0
[Result of search is]

p([0|A]) :- p(A).

[8 redundant clauses retracted]
```

Ejemplo con Progol en espacios infinitos

```
[Generalising p([1,1]).]
[Most specific clause is]

p(A) :- A=[B|C], B=1, C=[B|D], not(p(C)), D=[], p(D).

[C:-9,10,15,0 p(A).]
[C:-14,9,15,0 p(A) :- A=[B|C].]
[C:-9,7,8,0 p(A) :- A=[B|C], B=1.]
[C:-12,4,3,0 p(A) :- A=[B|C], B=1, C=[B|D].]
[C:-9,7,7,0 p(A) :- A=[B|C], B=1, C=[D|E].]
[C:5,7,0,0 p(A) :- A=[B|C], B=1, C=[D|E], not(p(C)).]
[C:7,7,0,0 p(A) :- A=[B|C], B=1, not(p(C)).]
[C:-20,4,6,0 p(A) :- A=[B|C], C=[B|D].]
[C:-23,7,14,0 p(A) :- A=[B|C], C=[D|E].]
[C:-7,7,7,0 p(A) :- A=[B|C], not(p(C)).]
```

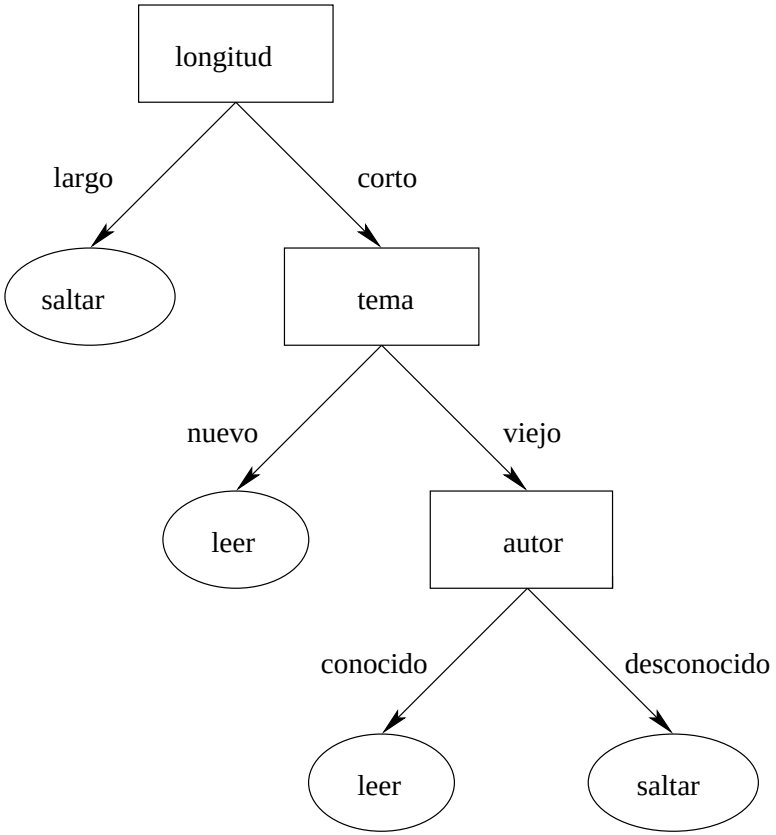
Ejemplo con Progol en espacios infinitos

```
[C:-12,4,3,0 p(A) :- A=[B|C], C=[B|D], not(p(C)).]
[C:0,4,0,0 p(A) :- A=[B|C], C=[B|D], p(D).]
[C:-9,7,7,0 p(A) :- A=[B|C], C=[D|E], not(p(C)).]
[C:-32,4,8,0 p(A) :- A=[B|C], C=[D|E], p(E).]
[14 explored search nodes]
f=7,p=7,n=0,h=0
[Result of search is]

p([1|A]) :- not(p(A)).

[7 redundant clauses retracted]
p([]).
p([0|A]) :- p(A).
p([1|A]) :- not(p(A)).
[Total number of clauses = 3]

[Time taken 0.030s]
```

Árboles de decisión en Progol

Ejemplo	Acción	Autor	Tema	Longitud	Sitio
e1	saltar	conocido	nuevo	largo	casa
e2	leer	desconocido	nuevo	corto	trabajo
e3	saltar	desconocido	viejo	largo	trabajo
e4	saltar	conocido	viejo	largo	casa
e5	leer	conocido	nuevo	corto	casa
e6	saltar	conocido	viejo	largo	trabajo
e7	saltar	desconocido	viejo	corto	trabajo
e8	leer	desconocido	nuevo	corto	trabajo
e9	saltar	conocido	viejo	largo	casa
e10	saltar	conocido	nuevo	largo	trabajo
e11	saltar	desconocido	viejo	corto	casa
e12	saltar	conocido	nuevo	largo	trabajo
e13	leer	conocido	viejo	corto	casa
e14	leer	conocido	nuevo	corto	trabajo
e15	leer	conocido	nuevo	corto	casa
e16	leer	conocido	viejo	corto	trabajo
e17	leer	conocido	nuevo	corto	casa
e18	leer	desconocido	nuevo	corto	trabajo

Árboles de decisión en Progol



Árboles de decisión en Progol

- **Representación** `softbot.pl`**:**
  - **Parámetros:**

```
:- set(r,1000)?
:- set(posonly)?
```

  - **Modos:**

```
:- modeh(1,accion(+ejemplo,#t_accion))?
:- modeb(1,autor(+ejemplo,#t_autor))?
:- modeb(1,tema(+ejemplo,#t_tema))?
:- modeb(1,longitud(+ejemplo,#t_longitud))?
:- modeb(1,sitio(+ejemplo,#t_sitio))?
```

Árboles de decisión en Progol

- **Tipos:**

```
ejemplo(e1).        ejemplo(e2).           ...   ejemplo(e18).
t_accion(saltar).   t_accion(leer).
t_autor(conocido).  t_autor(desconocido).
t_tema(nuevo).      t_tema(viejo).
t_longitud(largo).  t_longitud(corto).
t_sitio(casa).      t_sitio(trabajo).
```

- **Determinación:**

```
:- determination(accion/2, autor/2)?
:- determination(accion/2, tema/2)?
:- determination(accion/2, longitud/2)?
:- determination(accion/2, sitio/2)?
```

Árboles de decisión en Progol

- **Conocimiento base:**

```
autor(e1,conocido).   autor(e2,desconocido).   ... autor(e18,desconocido).
tema(e1,nuevo).       tema(e2,nuevo).          ... tema(e18, nuevo).
longitud(e1,largo).   longitud(e2,corto).      ... longitud(e18,corto).
sitio(e1,casa).       sitio(e2,trabajo).       ... sitio(e18, trabajo).
```

- **Ejemplos positivos:**

```
accion(e1,saltar).    accion(e2,leer).         ... accion(e18,leer).
```

- **Restricciones:**

```
:- hypothesis(Cabeza,Cuerpo,_),
   accion(A,C),
   Cuerpo,
   Cabeza = accion(A,B),
   B \= C.
```

Árboles de decisión en Progol

- **Sesión:**

```
> progol softbot
CProgol Version 4.4
...
[Testing for contradictions]
[No contradictions found]
[Generalising accion(e1,saltar).]
[Most specific clause is]

accion(A,saltar) :-
  autor(A,conocido), tema(A,nuevo), longitud(A,largo), sitio(A,casa).
```

Árboles de decisión en Progol

```
[Learning accion/2 from positive examples]
[C:-39932,18,10000,0 accion(A,saltar).]
[C:-39936,18,10000,0 accion(A,saltar) :- autor(A,conocido).]
[C:-39936,18,10000,0 accion(A,saltar) :- tema(A,nuevo).]
[C:34,28,13,0 accion(A,saltar) :- longitud(A,largo).]
[C:13,12,7,0 accion(A,saltar) :- longitud(A,largo), sitio(A,casa).]
[C:-39936,18,10000,0 accion(A,saltar) :- sitio(A,casa).]
[C:-39940,18,10000,0 accion(A,saltar) :- autor(A,conocido), tema(A,nuevo).]
[C:31,24,11,0 accion(A,saltar) :- autor(A,conocido), longitud(A,largo).]
[C:7,12,7,0 accion(A,saltar) :- autor(A,conocido), longitud(A,largo), sitio(A,casa).]
[C:-39940,18,10000,0 accion(A,saltar) :- autor(A,conocido), sitio(A,casa).]
[C:25,12,5,0 accion(A,saltar) :- tema(A,nuevo), longitud(A,largo).]
[C:-39940,18,10000,0 accion(A,saltar) :- tema(A,nuevo), sitio(A,casa).]
[C:19,12,5,0 accion(A,saltar) :- autor(A,conocido), tema(A,nuevo), longitud(A,largo).]
[C:-39944,18,10000,0 accion(A,saltar) :- autor(A,conocido), tema(A,nuevo), sitio(A,casa)
[14 explored search nodes]
f=34,p=28,n=13,h=0
[Result of search is]

accion(A,saltar) :- longitud(A,largo).
```

Árboles de decisión en Progol

```
[7 redundant clauses retracted]
[Generalising accion(e2,leer).]
[Most specific clause is]

accion(A,leer) :-
  autor(A,desconocido), tema(A,nuevo), longitud(A,corto), sitio(A,trabajo).
```

Árboles de decisión en Progol

```
[Learning accion/2 from positive examples]
[C:-39932,18,10000,0 accion(A,leer).]
[C:-39936,18,10000,0 accion(A,leer) :- autor(A,desconocido).]
[C:-39936,18,10000,0 accion(A,leer) :- tema(A,nuevo).]
[C:-39936,18,10000,0 accion(A,leer) :- longitud(A,corto).]
[C:-39936,18,10000,0 accion(A,leer) :- sitio(A,trabajo).]
[C:-39940,18,10000,0 accion(A,leer) :- longitud(A,corto), sitio(A,trabajo).]
[C:34,28,12,0 accion(A,leer) :- tema(A,nuevo), longitud(A,corto).]
[C:19,16,8,0 accion(A,leer) :- tema(A,nuevo), longitud(A,corto), sitio(A,trabajo).]
[C:-39940,18,10000,0 accion(A,leer) :- tema(A,nuevo), sitio(A,trabajo).]
[C:13,12,7,0 accion(A,leer) :- autor(A,desconocido), tema(A,nuevo).]
[C:7,12,7,0 accion(A,leer) :- autor(A,desconocido), tema(A,nuevo), longitud(A,corto).]
[C:7,12,7,0 accion(A,leer) :- autor(A,desconocido), tema(A,nuevo), sitio(A,trabajo).]
[C:1,12,7,0 accion(A,leer) :- autor(A,desconocido), tema(A,nuevo), longitud(A,corto), si
[C:-39940,18,10000,0 accion(A,leer) :- autor(A,desconocido), longitud(A,corto).]
[C:-39940,18,10000,0 accion(A,leer) :- autor(A,desconocido), sitio(A,trabajo).]
[C:-39944,18,10000,0 accion(A,leer) :- autor(A,desconocido), longitud(A,corto), sitio(A,
[16 explored search nodes]
f=34,p=28,n=12,h=0
[Result of search is]
accion(A,leer) :- tema(A,nuevo), longitud(A,corto).
```

Árboles de decisión en Progol

```
[7 redundant clauses retracted]
[Generalising accion(e7,saltar).]
[Most specific clause is]

accion(A,saltar) :-
   autor(A,desconocido), tema(A,viejo), longitud(A,corto), sitio(A,trabajo).
```

Árboles de decisión en Progol

```
[Learning accion/2 from positive examples]
[C:-39932,18,10000,0 accion(A,saltar).]
[C:-39936,18,10000,0 accion(A,saltar) :- autor(A,desconocido).]
[C:-39936,18,10000,0 accion(A,saltar) :- tema(A,viejo).]
[C:-39936,18,10000,0 accion(A,saltar) :- longitud(A,corto).]
[C:-39936,18,10000,0 accion(A,saltar) :- sitio(A,trabajo).]
[C:-39940,18,10000,0 accion(A,saltar) :- longitud(A,corto), sitio(A,trabajo).]
[C:10,8,4,0 accion(A,saltar) :- autor(A,desconocido), tema(A,viejo).]
[C:1,8,4,0 accion(A,saltar) :- autor(A,desconocido), tema(A,viejo), longitud(A,corto).]
[C:-39940,18,10000,0 accion(A,saltar) :- autor(A,desconocido), longitud(A,corto).]
[C:-39940,18,10000,0 accion(A,saltar) :- autor(A,desconocido), sitio(A,trabajo).]
[C:-39940,18,10000,0 accion(A,saltar) :- tema(A,viejo), longitud(A,corto).]
[C:-39940,18,10000,0 accion(A,saltar) :- tema(A,viejo), sitio(A,trabajo).]
[C:-39944,18,10000,0 accion(A,saltar) :- tema(A,viejo), longitud(A,corto), sitio(A,traba
[C:-39944,18,10000,0 accion(A,saltar) :- autor(A,desconocido), longitud(A,corto), sitio(
[14 explored search nodes]
f=10,p=8,n=4,h=0
[Result of search is]
accion(A,saltar) :- autor(A,desconocido), tema(A,viejo).
```

Árboles de decisión en Progol

```
[2 redundant clauses retracted]
[Generalising accion(e13,leer).]
[Most specific clause is]

accion(A,leer) :-
   autor(A,conocido), tema(A,viejo), longitud(A,corto), sitio(A,casa).
```

Árboles de decisión en Progol

```
[Learning accion/2 from positive examples]
[C:-39932,18,10000,0 accion(A,leer).]
[C:-39936,18,10000,0 accion(A,leer) :- autor(A,conocido).]
[C:-39936,18,10000,0 accion(A,leer) :- tema(A,viejo).]
[C:-39936,18,10000,0 accion(A,leer) :- longitud(A,corto).]
[C:-39936,18,10000,0 accion(A,leer) :- sitio(A,casa).]
[C:-39940,18,10000,0 accion(A,leer) :- tema(A,viejo), longitud(A,corto).]
[C:-39940,18,10000,0 accion(A,leer) :- tema(A,viejo), sitio(A,casa).]
[C:-39940,18,10000,0 accion(A,leer) :- longitud(A,corto), sitio(A,casa).]
[C:-39940,18,10000,0 accion(A,leer) :- autor(A,conocido), tema(A,viejo).]
[C:10,8,4,0 accion(A,leer) :- autor(A,conocido), longitud(A,corto).]
[C:-39940,18,10000,0 accion(A,leer) :- autor(A,conocido), sitio(A,casa).]
[C:1,8,4,0 accion(A,leer) :- autor(A,conocido), tema(A,viejo), longitud(A,corto).]
[C:-39944,18,10000,0 accion(A,leer) :- autor(A,conocido), tema(A,viejo), sitio(A,casa).]
[C:-39944,18,10000,0 accion(A,leer) :- tema(A,viejo), longitud(A,corto), sitio(A,casa).]
[14 explored search nodes]
f=10,p=8,n=4,h=0
[Result of search is]
accion(A,leer) :- autor(A,conocido), longitud(A,corto).
```

Árboles de decisión en Progol

```
[2 redundant clauses retracted]
accion(A,saltar) :- longitud(A,largo).
accion(A,leer) :- tema(A,nuevo), longitud(A,corto).
accion(A,saltar) :- autor(A,desconocido), tema(A,viejo).
accion(A,leer) :- autor(A,conocido), longitud(A,corto).
[Total number of clauses = 4]

[Time taken 0.090s]
```

Algoritmo de Progol

- **Algoritmo de Progol:**
  1. **Empezar con la teoría vacía: $T = \emptyset$.**
  2. **Seleccionar un ejemplo para generalizarlo: E**
  3. **Construir la cláusula más específica que implica el ejemplo seleccionado y cumple las restricciones impuestas: $T \cup \{C_1\} \models E$.**
  4. **Buscar la mejor clásula que generaliza la anterior y añadirla a la teoría: $T := T \cup \{C_2\}$**
  5. **Borrar los ejemplos positivos cubiertos por la teoría.**
  6. **Si quedan ejemplos positivos, volver a 1; en caso contrario, devolver la teoría construida.**

Aplicaciones con PLI

- **Procesamiento de lenguaje natural:**
  - **aprender reglas gramaticales.**
- **Diseño asistido por ordenador:**
  - **aprender relaciones entre objetos.**
- **Diseño de medicamentos:**
  - **predecir actividad a apartir de propiedades moleculares.**
- **Musicología:**
  - **aprender el estilo de un músico.**

Aplicaciones con PLI

- **Predicción de la estructura secundaria de las proteinas.**
  - **Dada la estructura primaria de una proteina (secuencia de aminoácidos),**
    - **Encontrar la estructura secundaria.**
    - **Predecir si los residuos individuales forman una hélice levógira.**
  - **Ejemplos: 12 proteinas no homólogas (1612 residuos).**
  - **Conocimiento base: Propiedades físicas y químicas de los residuos individuales y su posición relativa dentro de la proteina.**
  - **Sistema: GOLEM.**
  - **Resultados:**
    - **21 cláusula producidas, cada una de unos 15 literales.**
    - **Su precisión sobre un test independiente fue del 82%, mientras que la precisión del mejor método convencional fue del 73%.**

Bibliografía

- **Flach, P. *Simply Logical (Intelligent Reasoning by Example)* (John Wiley, 1994).**
  - **Cap. 9: "Inductive reasoning".**
- **Fürnkranz, J. *Inductive Logic Programming* (`http://www.ai.univie.ac.at/~juffi/Ilp-Vu/ilp-vu-program.html`)**
- **Markov,Z. *Machine larning course* (Faculty of Mathematics and Informatics, Univ. of Sofia, 1998)**
- **Gómez, A.J. *Inducción de conocimiento con incertidumbre en bases de datos relacionales borrosas* (Tesis doctoral, Univ. Politécnica de Madrid, 1998).**
- **Mitchell, T.M. *Machine learning* (McGraw–Hill, 1997).**
  - **Cap. 3: "Decision tree learning".**
  - **Cap. 11: "Learning sets of rules".**

Bibliografía

- Muggleton, S. y Firth, J. *CProgol4.4: a tutorial introduction* (En "Inductive Logic Programming and Knowledge Discovery in Databases". Springer–Verlag, 2000).
- Poole, D.; Mackworth, A. y Goebel, R. *Computational Intelligence (A Logical Approach)* (Oxford University Press, 1998).
  - Cap. 11: "Learning".
- Russell, S. y Norvig, P. *Inteligencia artificial (Un enfoque moderno)* (Prentice–Hall Hispanoamericana, 1996).
  - Cap. 18: "Aprendizaje a partir de la observación".
  - Cap. 21: "El conocimiento en el aprendizaje".
- The Online School on Inductive Logic Programing and Knowledge Discovery in Databases (`http://www-ai.ijs.si/SasoDzeroski/ILP2/ilpkdd`).
- MLnet Online Information Service (`http://www.mlnet.org`).