Tema 9: Razonamiento por defecto y razonamiento explicativo

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

Razonamiento por defecto

- **Ejemplo de razonamiento por defecto**

```
El animal_1 es un pájaro
Normalmente, los pájaros vuelan.
Por tanto, el animal_1 vuela.
```

- **Programa** `P1`
  - **Programa** `P1`

```
pajaro(animal_1).
vuela(X) :-
   pajaro(X),
   normal(X).
```

  - **Modelos del programa** `P1`

```
{pajaro(animal_1)}
{pajaro(animal_1), vuela(animal_1)}
{pajaro(animal_1), normal(animal_1), vuela(animal_1)}
```

  - **Consecuencia**

```
P1 |=/= vuela(animal_1)
```

Razonamiento por defecto

- **Programa `P2` con `anormal/1`**
  - **Programa `P2`**

```
:- dynamic anormal/1.

pajaro(animal_1).
vuela(X) :-
   pajaro(X),
   not anormal(X).
```

  - **Sesión**

```
?- vuela(animal_1).
Yes
```

Razonamiento por defecto

- **Traza**

```
?- vuela(animal_1).
   Call:  (  7) vuela(animal_1) ?
   Call:  (  8) pajaro(animal_1) ?
   Exit:  (  8) pajaro(animal_1) ?
 ^ Call:  (  8) not anormal(animal_1) ?
   Call:  (  9) anormal(animal_1) ?
   Fail:  (  9) anormal(animal_1) ?
 ^ Exit:  (  8) not anormal(animal_1) ?
   Exit:  (  7) vuela(animal_1) ?

Yes
```

Razonamiento por defecto

- **Extensión del conocimiento**
  - **Nuevo conocimiento**

```
El animal_1 es un avestruz.
Los avestruces son pájaros que no vuelan.
```

  - **Programa extendido**

```
:- dynamic anormal/1.

pajaro(animal_1).
avestruz(animal_1).
vuela(X) :-
   pajaro(X),
   not anormal(X).
anormal(X) :- avestruz(X).
```

Razonamiento por defecto

- **Traza**

```
?- vuela(animal_1).
   Call:  (  7) vuela(animal_1) ?
   Call:  (  8) pajaro(animal_1) ?
   Exit:  (  8) pajaro(animal_1) ?
 ^ Call:  (  8) not anormal(animal_1) ?
   Call:  (  9) anormal(animal_1) ?
   Call:  ( 10) avestruz(animal_1) ?
   Exit:  ( 10) avestruz(animal_1) ?
   Exit:  (  9) anormal(animal_1) ?
 ^ Fail:  (  8) not anormal(animal_1) ?
   Fail:  (  7) vuela(animal_1) ?

No
```

Razonamiento por defecto

- **Cancelación reglas por defectos mediante reglas específicas**
  - **Regla por defecto: "Normalmente, los pájaros vuelan"**
  - **Regla específica: "Los avestruces no vuelan"**
- **Razonamiento monótono y no–monótono**
  - **Razonamiento monótono**

    `P1 |= C y P2 extiende a P1, entonces P2 |= C.`
  - **Razonamiento monótono**

    `P1 |= C y P2 extiende a P1, es posible P2 |=/= C.`

Razonamiento por defecto

- **Programa con reglas y reglas con excepciones (defectos)**
  - **Programa objeto**

```
:- op(1100,xfx,<-).

defecto(vuela(X) <- pajaro(X)).

regla(pajaro(X) <- avestruz(X)).
regla(not(vuela(X)) <- avestruz(X)).
regla(avestruz(animal_1) <- verdad).
regla(pajaro(animal_2) <- verdad).
```

  - **Sesión**

```
?- explica(vuela(X),E).
X = animal_2
E = [defecto((vuela(animal_2) <- pajaro(animal_2))),
     regla((pajaro(animal_2) <- verdad))] ;
No
?- explica(not(vuela(X)),E).
X = animal_1
E = [regla((not(vuela(animal_1)) <- avestruz(animal_1))),
     regla((avestruz(animal_1) <- verdad))] ;
No
```

Razonamiento por defecto

- **Metaprograma para explicaciones**

```
explica(A,E) :- explica(A,[],E).

explica(verdad,E,E)                 :- !.
explica((A,B),E0,E)                 :- !, explica(A,E0,E1), explica(B,E1,E).
explica(A,E0,E)                     :- prueba(A,E0,E).
explica(A,E0,[defecto(A<-B)|E])     :- defecto(A<-B),
                                       explica(B,E0,E),
                                       \+ contradiccion(A,E).

prueba(verdad,E,E)                :- !.
prueba((A,B),E0,E)                :- !, prueba(A,E0,E1), prueba(B,E1,E).
prueba(A,E0,[regla(A<-B)|E])      :- regla(A<-B), prueba(B,E0,E).

contradiccion(not(A),E) :- !, prueba(A,E,_E1).
contradiccion(A,E)      :- prueba(not(A),E,_E1).
```

Razonamiento por defecto

- **Explicaciones de hechos contradictorios**
  - **Programa objeto**

```
defecto(not(vuela(X))  <- mamifero(X)).
defecto(vuela(X)       <- vampiro(X)).
defecto(not(vuela(X))  <- muerto(X)).

regla(mamifero(X)      <- vampiro(X)).
regla(vampiro(dracula) <- verdad).
regla(muerto(dracula)  <- verdad).
```

Razonamiento por defecto

- **Sesión**

```
?- explica(vuela(dracula),E).
E = [defecto(vuela(dracula) <- vampiro(dracula)),
     regla(vampiro(dracula) <- verdad)] ;
No

?- explica(not(vuela(dracula),E)).
E = [defecto(not(vuela(dracula)) <- mamifero(dracula)),
     regla(mamifero(dracula) <- vampiro(dracula)),
     regla(vampiro(dracula) <- verdad)] ;
E = [defecto(not(vuela(dracula)) <- muerto(dracula)),
     regla(muerto(dracula) <- verdad)] ;
No
```

Razonamiento por defecto

- **Cancelación entre defectos mediante nombres**
  - **Programa objeto**

```
defecto(mamiferos_no_vuelan(X), (not(vuela(X)) <- mamifero(X))).
defecto(vampiros_vuelan(X),     (vuela(X) <- vampiro(X))).
defecto(muertos_no_vuelan(X),   (not(vuela(X)) <- muerto(X))).

regla(mamifero(X)      <- vampiro(X)).
regla(vampiro(dracula) <- verdad).
regla(muerto(dracula)  <- verdad).

regla(not(mamiferos_no_vuelan(X)) <- vampiro(X)).
regla(not(vampiros_vuelan(X))     <- muerto(X)).
```

Razonamiento por defecto

- **Modificación de** `explica`

```
explica(A,E0,[defecto(A<-B)|E]) :-
   defecto(Nombre,(A<-B)),
   explica(B,E0,E),
   \+ contradiccion(Nombre,E),
   \+ contradiccion(A,E).
```

- **Sesión**

```
?- explica(vuela(dracula),E).
No

?- explica(not vuela(dracula),E).
E = [defecto((not(vuela(dracula))<-muerto(dracula))),
     regla((muerto(dracula) <- verdad))]
Yes
```

Razonamiento explicativo

- **Problema de la explicación**

```
Dados     P un programa lógico y
          O una observación (un hecho básico en el lenguaje de P)
Encontrar E una explicación (una lista de hechos atómicos en el lenguaje de P
            cuyos predicados no son cabezas de cláusulas de P) tal que
            P U E |- O)
```

- **Explicación para programas definidos**
  - **Programa objeto**

```
europeo(X) <- español(X).
español(X) <- andaluz(X).
europeo(X) <- italiano(X).
```

  - **Sesión**

```
?- explicación(europeo(juan),E).
E = [andaluz(juan)] ;
E = [italiano(juan)] ;
No
```

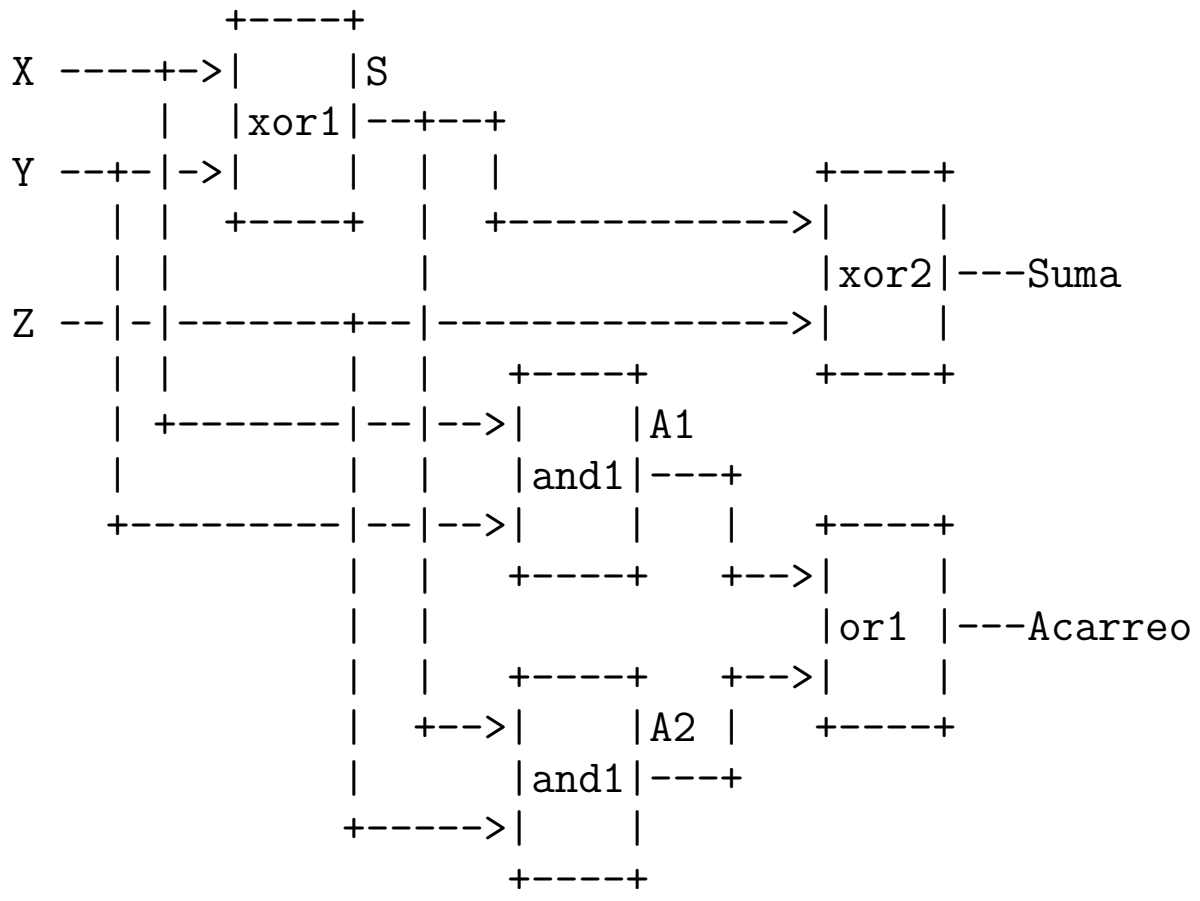
Razonamiento explicativo

- **Programa**

```
:- op(1200,xfx,<-).

explicación(O,E) :-
   explicación(O,[],E).

explicación(verdad,E,E) :- !.
explicación((A,B),E0,E) :- !,
   explicación(A,E0,E1),
   explicación(B,E1,E).
explicación(A,E0,E) :-
   (A <- B),
   explicación(B,E0,E).
explicación(A,E,E) :-
   member(A,E).
explicación(A,E,[A|E]) :-
   not member(A,E),
   explicable(A).

explicable(A) :-
   not (A <- _B).
```

Diagnóstico mediante explicación

- **Representación de un sumador**



Diagnóstico mediante explicación

- **Definición del sumador**

```
sumador(X,Y,Z,Acarreo,Suma) :-
   xor(X,Y,S),
   xor(Z,S,Suma),
   and(X,Y,A1),
   and(Z,S,A2),
   or(A1,A2,Acarreo).

and(1,1,1).  and(1,0,0). and(0,1,0).  and(0,0,0).
or(1,1,1).   or(1,0,1).  or(0,1,1).   or(0,0,0).
xor(1,0,1).  xor(0,1,1). xor(1,1,0).  xor(0,0,0).

tabla :-
   format('X Y Z A S~n'),
   tabla2.
tabla2 :-
   member(X,[0,1]), member(Y,[0,1]), member(Z,[0,1]),
   sumador(X,Y,Z,A,S),
   format('~a ~a ~a ~a ~a~n',[X,Y,Z,A,S]),
   fail.
tabla2.
```

Diagnóstico mediante explicación

- **Sesión con el sumador**

```
?- tabla.
X Y Z A S
0 0 0 0 0
0 0 1 0 1
0 1 0 0 1
0 1 1 1 0
1 0 0 0 1
1 0 1 1 0
1 1 0 1 0
1 1 1 1 1
Yes
```

Diagnóstico mediante explicación

- Modelo de fallo del sumador

```
sumador(X,Y,Z,Acarreo,Suma) <-
        xorg(xor1,X,Y,S),       xorg(xor2,Z,S,Suma),
        andg(and1,X,Y,A1),      andg(and2,Z,S,A2),
        org(or1,A1,A2,Acarreo).

xorg(_N,X,Y,Z) <- xor(X,Y,Z).
xorg(N,1,1,1)  <- fallo(N=f1).
xorg(N,0,0,1)  <- fallo(N=f1).
xorg(N,1,0,0)  <- fallo(N=f0).
xorg(N,0,1,0)  <- fallo(N=f0).
andg(_N,X,Y,Z) <- and(X,Y,Z).
andg(N,0,0,1)  <- fallo(N=f1).
andg(N,1,0,1)  <- fallo(N=f1).
andg(N,0,1,1)  <- fallo(N=f1).
andg(N,1,1,0)  <- fallo(N=f0).
org(_N,X,Y,Z) <- or(X,Y,Z).
org(N,0,0,1)  <- fallo(N=f1).
org(N,1,0,0)  <- fallo(N=f0).
org(N,0,1,0)  <- fallo(N=f0).
org(N,1,1,0)  <- fallo(N=f0).
```

Diagnóstico mediante explicación

- **Diagnóstico mediante explicación**

```
diagnostico(Observacion,Diagnostico) :-
   explicación(Observacion,Diagnostico).

:- abolish(explicable,2).
explicable(fallo(_X)).
```

Diagnóstico mediante explicación

- **Sesión de diagnóstico**

```
?- diagnostico(sumador(0,0,1,1,0),D).
D = [fallo(or1 = f1),fallo(xor2 = f0)] ;
D = [fallo(and2 = f1),fallo(xor2 = f0)] ;
D = [fallo(and1 = f1),fallo(xor2 = f0)] ;
D = [fallo(and2 = f1),fallo(and1 = f1),fallo(xor2 = f0)] ;
D = [fallo(xor1 = f1)] ;
D = [fallo(or1 = f1),fallo(and2 = f0),fallo(xor1 = f1)] ;
D = [fallo(and1 = f1),fallo(xor1 = f1)] ;
D = [fallo(and2 = f0),fallo(and1 = f1),fallo(xor1 = f1)] ;
No
```

Diagnóstico mediante explicación

- **Diagnóstico mínimo**

```
diagnostico_minimo(O,D) :-
   diagnostico(O,D),
   not((diagnostico(O,D1),
       subconjunto_propio(D1,D))).

subconjunto_propio([],Ys):-
   Ys \= [].
subconjunto_propio([X|Xs],Ys):-
   select(Ys,X,Ys1),
   subconjunto_propio(Xs,Ys1).
```

- **Diagnóstico mínimo del sumador**

```
?- diagnostico_minimo(sumador(0,0,1,1,0),D).
D = [fallo(or1 = f1),fallo(xor2 = f0)] ;
D = [fallo(and2 = f1),fallo(xor2 = f0)] ;
D = [fallo(and1 = f1),fallo(xor2 = f0)] ;
D = [fallo(xor1 = f1)] ;
No
```

Bibliografía

- **Flach, P. "Simply Logical (Intelligent Reasoning by Example)" (John Wiley, 1994)**
  - **Cap. 8: "Reasoning incomplete information"**

- **Poole, D.; Mackworth, A. y Goebel, R. *Computational Intelligence (A Logical Approach)* (Oxford University Press, 1998)**
  - **Cap. 9: "Assumption–Based Reasoning"**

- **Rich, E. y Knight, K. "Inteligencia artificial (segunda edición)" (McGraw–Hill Interamericana, 1994).**
  - **Cap. 7: "Razonamiento simbólico bajo incertidumbre"**