

Representación del conocimiento estructurado

José A. Alonso

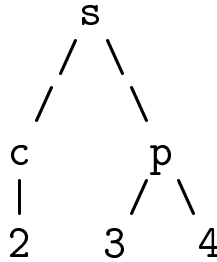
Dpto. de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE SEVILLA

Arboles como términos

- Representación de árboles por términos

Arbol:



Término: `s(c(2),p(3,4))`

- TAD árbol

```
% ?- termino_arbol(s(c(2),p(3,4)),R,S).  
% R = s  
% S = [c(2), p(3, 4)]  
% Yes  
% ?- termino_arbol(A,s,[c(2),p(3,4)]).  
% A = s(c(2), p(3, 4))  
% Yes  
termino_arbol(Arbol,Raiz,Subarboles) :-  
    Arbol=.. [Raiz|Subarboles].
```

Arboles como términos

```
% ?- termino_raiz(s(c(2),p(3,4)),R).
% R = s
% Yes
termino_raiz(Arbol,Raiz) :-
    termino_arbol(Arbol,Raiz,_S).

% ?- termino_subtermino(s(c(2),p(3,4)),S).
% S = c(2) ;
% S = p(3, 4) ;
% No
termino_subtermino(Arbol,Subarbol) :-
    termino_arbol(Arbol,_R,S),
    member(Subarbol,S).
```

● Arcos en un árbol

```
% ?- termino_arco(s(c(2),p(3,4)),Arc).
% Arc = [s, c] ;
% Arc = [s, p] ;
% Arc = [c, 2] ;
% Arc = [p, 3] ;
% Arc = [p, 4] ;
% No
termino_arco(Arbol,[Raiz,SR]) :-
    termino_raiz(Arbol,Raiz),
    termino_subtermino(Arbol,Subarbol),
    termino_raiz(Subarbol,SR).
termino_arco(Arbol,Arco) :-
    termino_subtermino(Arbol,Subarbol),
    termino_arco(Subarbol,Arco).
```

Arboles como términos

- Caminos en un árbol

```
% ?- termino_camino(s(c(2),p(3,4)),C).
% C = [s, c] ;
% C = [s, p] ;
% C = [c, 2] ;
% C = [p, 3] ;
% C = [p, 4] ;
% C = [s, c, 2] ;
% C = [s, p, 3] ;
% C = [s, p, 4] ;
% No
termino_camino(Arbol,Arco) :-
    termino_arco(Arbol,Arco).
termino_camino(Arbol,[Nodo1,Nodo2|Nodos]) :-
    termino_arco(Arbol,[Nodo1,Nodo2]),
    termino_camino(Arbol,[Nodo2|Nodos]).
```

- Escritura de árboles

```
% ?- termino_escribe(s(c(2),p(3,4))).
% -----s-----c-----2
%           -----p-----3
%                   -----4
% Yes
% ?- termino_escribe(s(cuad(2),p(3,4))).
% -----s-----cuad-----2
%           -----p-----3
%                   -----4
% Yes
```

Arboles como términos

```
termino_escribe(Arbol) :-
    termino_escribe(0,Arbol),
    nl.

termino_escribe(Pos,Arbol) :-
    termino_arbol(Arbol,Raiz,Subarboles),
    termino_escribe_nodo(Pos,Pos2,Raiz),
    termino_escribe_nodos(Pos2,Subarboles).

termino_escribe_nodo(Principio,Fin,Nodo):-
    name(Nodo,L),
    length(L,N),
    Fin is Principio+10,
    N1 is Fin-Principio-N,
    escribe_linea(N1),
    write(Nodo).

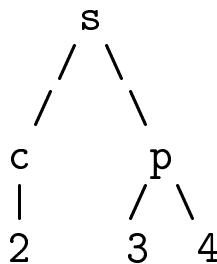
escribe_linea(0).
escribe_linea(N):-
    N>0,
    N1 is N-1,
    write('-'),
    escribe_linea(N1).

termino_escribe_nodos(_Pos, []).
termino_escribe_nodos(Pos, [Arbol]) :- !,
    termino_escribe(Pos,Arbol).
termino_escribe_nodos(Pos, [Arbol|Subarboles]) :-
    termino_escribe(Pos,Arbol),
    nl, tab(Pos),
    termino_escribe_nodos(Pos,Subarboles).
```

Grafos generados por un predicado

- Representación de grafos por predicados

- Grafo



- Representación mediante arco/2

arco(s,c).

arco(s,p).

arco(c,2).

arco(p,3).

arco(p,4).

Grafos generados por un predicado

- Caminos en el grafo

```
% ?- camino(C).
% C = [s, c] ;
% C = [s, p] ;
% C = [c, 2] ;
% C = [p, 3] ;
% C = [p, 4] ;
% C = [s, c, 2] ;
% C = [s, p, 3] ;
% C = [s, p, 4] ;
% No
camino([Nodo1,Nodo2]) :-
    arco(Nodo1,Nodo2).
camino([Nodo1,Nodo2|Nodos]) :-
    arco(Nodo1,Nodo2),
    camino([Nodo2|Nodos]).
```

- Caminos hasta las hojas

```
% ?- camino_hasta_hoja(s,C).
% C = [s, c, 2] ;
% C = [s, p, 3] ;
% C = [s, p, 4] ;
% No
camino_hasta_hoja(Hoja,[Hoja]) :-
    hoja(Hoja).
camino_hasta_hoja(Nodo1,[Nodo1|Nodos]) :-
    arco(Nodo1,Nodo2),
    camino_hasta_hoja(Nodo2,Nodos).

hoja(N) :-
    not(arco(N,_Nodo)).
```

Grafos generados por un predicado

● Grafos SLD

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

```
alumno(A,P) <- [estudia(A,C), enseña(P,C)].  
estudia(ana,ia) <- [].  
estudia(ana,pl) <- [].  
estudia(eva,ra) <- [].  
enseña(jose_a,ia) <- [].  
enseña(jose_a,ra) <- [].  
enseña(rafael,pl) <- [].
```

```
arco([A|As],L) :-  
    (A <- Cuerpo),  
    append(Cuerpo,As,L).
```

```
hoja([]).
```

```
% ?- camino_hasta_hoja([alumno(A,jose_a)],_).  
% A = ana ;  
% A = eva ;  
% No  
camino_hasta_hoja(Hoja,[Hoja]) :-  
    hoja(Hoja).  
camino_hasta_hoja(Nodo1,[Nodo1|Nodos]) :-  
    arco(Nodo1,Nodo2),  
    camino_hasta_hoja(Nodo2,Nodos).
```


Redes de clasificación

● Ejemplo

```
Personas [ciudad: Sevilla]
  * Alumnos [estado: soltero]
    * Juan [edad: 19]
    * Luis [edad: 24, estado: casado]
  * Profesores [estado: casado]
    * Pablo [edad: 44, ciudad: Mairena]
    * Pedro [edad: 47]
```

● Representación

```
% Relaciones entre clases:
persona(X) :- alumno(X).
persona(X) :- profesor(X).

% Instancias:
alumno(juan).      alumno(luis).
profesor(pablo).  profesor(pedro).

% Propiedades:
ciudad(pablo,mairena).
ciudad(X,sevilla) :- persona(X).

estado(luis,casado).
estado(X,soltero) :- alumno(X).
estado(X,casado) :- profesor(X).

edad(juan,19).
edad(luis,24).
edad(pablo,44).
edad(pedro,47).

atributos([ciudad,estado,edad]).
```

Redes de clasificación

- Razonamiento

- Sesión

```
?- propiedades(juan,P).  
P = [ciudad = sevilla, estado = soltero, edad = 19]  
?- propiedades(luis,P).  
P = [ciudad = sevilla, estado = casado, edad = 24]  
?- propiedades(pablo,P).  
P = [ciudad = mairena, estado = casado, edad = 44]  
?- propiedades(pedro,P).  
P = [ciudad = sevilla, estado = casado, edad = 47]
```

- Definición

```
propiedades(Inst,Props) :-  
    atributos(Atrs),  
    propiedades(Atrs,Inst,Props).  
  
propiedades([],_Inst,[]).  
propiedades([Atr|Atrs],Inst,[Atr=Valor|Props]) :-  
    valor(Atr,Inst,Valor), !,  
    propiedades(Atrs,Inst,Props).  
  
valor(A,I,V) :-  
    P =.. [A,I,V],  
    call(P).
```

Redes de clasificación

- Elementos de la representación
 - Las instancias se representan por constantes
 - Las clases se representan por predicados unarios
 - Las relaciones clase–superclase se representan por cláusulas
 - Las relaciones instancia–clase se representan por hechos básicos
 - Cada propiedad se representa por un predicado binario cuyo argumentos son una instancia y el valor correspondiente
 - La lista de atributos se representa por un hecho de la forma
atributos([<atributo 1>, ..., <atributo n>])
 - Las propiedades de una instancia es una lista de pares atributo–valor

Redes de clasificación

- Comentarios

- Conflicto entre propiedades

```
?- estado(luis,X).
```

```
X = casado ;
```

```
X = soltero ;
```

```
No
```

```
?- propiedades(luis,_P),member(estado=X,_P).
```

```
X = casado ;
```

```
No
```

- Dependencia del orden de las cláusulas en las propiedades específicas

- Dificultad de razonamiento sobre clases, p.e.

“¿Cuáles son las subclases de persona”

Redes semánticas

● Representación

% Relaciones entre clases:

es_un(persona, inicio).

es_un(alumno, persona).

es_un(profesor, persona).

% Relaciones entre instancias y clases:

inst(juan, alumno).

inst(luis, alumno).

inst(pablo, profesor).

inst(pedro, profesor).

% Propiedades de clases:

prop(persona, ciudad, sevilla).

prop(alumno, estado, soltero).

prop(profesor, estado, casado).

% Propiedades de instancias:

prop(juan, edad, 19).

prop(luis, edad, 24).

prop(luis, estado, casado).

prop(pablo, edad, 44).

prop(pablo, ciudad, mairena).

prop(pedro, edad, 47).

Redes semánticas

- Razonamiento

- Sesión

```
?- propiedades_rs(luis,P).  
P = [ciudad = sevilla, edad = 24, estado = casado]
```

- Definición

```
propiedades_rs(Inst,Props) :-  
    props(Inst,P_Especificas),  
    inst(Inst,Clase),  
    herencia_rs(Clase,P_Especificas,Props).  
  
props(X,Props) :-  
    findall(Atr=Valor,prop(X,Atr,Valor),Props).  
  
herencia_rs(inicio,Props,Props).  
herencia_rs(Clase,P_Actuales,Props) :-  
    props(Clase,P_Generales),  
    actualiza(P_Actuales,P_Generales,N_P_Actuales),  
    es_un(Clase,Super_clase),  
    herencia_rs(Super_clase,N_P_Actuales,Props).  
  
actualiza(Props,[],Props).  
actualiza(P_Actuales,[Atr=_Valor|P_Generales],Props) :-  
    member(Atr=_V,P_Actuales),  
    actualiza(P_Actuales,P_Generales,Props).  
actualiza(P_Actuales,[Atr=Valor|P_Generales],  
          [Atr=Valor|Props]) :-  
    not(member(Atr=_V,P_Actuales)),  
    actualiza(P_Actuales,P_Generales,Props).
```

Redes semánticas

- Elementos de la representación

- Las instancias se representan por constantes
- Las clases se representan por constantes
- Las relaciones clase–superclase se representan por hechos de la forma

`es_un(<clase>, <super-clase>)`

- Las relaciones instancia–clases se representan por hechos de la forma

`inst(<instancia>, <clase>)`

- Cada propiedad se representa por un predicado binario de la forma

`prop(<instancia o clase>, <propiedad>, valor)`

- La constante inicio representa la clase inicial de la jerarquía
- Las propiedades de una instancia es una lista de pares atributo–valor

Redes semánticas

- **Comentarios**

- Independencia del orden de las cláusulas en las propiedades específicas

- Razonamiento sobre clases, p.e.

“¿Cuáles son las subclases de persona”

?- es_un(X, persona).

X = alumno ;

X = profesor ;

No

- Especificación declarativa de la estrategia de herencia
- Posibilidad de implementar estrategias de herencia múltiple

Marcos

- **Representación**

```
% Clases;
clase(persona, inicio, [ciudad=sevilla]).
clase(alumno, persona, [estado=soltero]).
clase(profesor, persona, [estado=casado]).

% Instancias:
instancia(juan, alumno, [edad=19]).
instancia(luis, alumno, [edad=24, estado=casado]).
instancia(pablo, profesor, [edad=44, ciudad=mairena]).
instancia(pedro, profesor, [edad=47]).
```

- **Razonamiento**

- **Sesión**

```
?- propiedades_marco(luis, P).
P = [ciudad = sevilla, edad = 24, estado = casado]
```

- **Definición**

```
propiedades_marco(Inst, Props) :-
    instancia(Inst, Clase, PropsInst),
    herencia_marco(Clase, PropsInst, Props).

herencia_marco(inicio, Props, Props).
herencia_marco(Clase, P_Actuales, Props) :-
    clase(Clase, Super_clase, P_Generales),
    actualiza(P_Actuales, P_Generales, N_P_Actuales),
    herencia_marco(Super_clase, N_P_Actuales, Props).
```

Marcos

- Elementos de la representación

- Las instancias se representan por constantes
- Las clases se representan por constantes
- Cada propiedad se representa por una igualdad de la forma

`<atributo> = <valor>`

- Las relaciones clase–superclase se representan por hechos de la forma

`clase(<clase>, <sup-clase>, [<prop-1>, . . . , <prop-n>])`

- Las relaciones instancia–clase se representan por hechos de la forma

`instancia(<clase>, <sup-clase>, [<prop-1>, . . . , <prop-n>])`

- La constante inicio representa la clase inicial de la jerarquía
- Las propiedades de una instancia es una lista de pares atributo–valor

Marcos

- **Comentarios**

- Independencia del orden de las cláusulas en las propiedades específicas

- Razonamiento sobre clases, p.e.

“¿Cuáles son las subclases de persona”

```
?- clase(X,persona,_).
```

```
X = alumno ;
```

```
X = profesor ;
```

```
No
```

- Especificación declarativa de la estrategia de herencia
- Posibilidad de implementar estrategias de herencia múltiple

Bibliografía

- Flach, P. “Simply Logical (Intelligent Reasoning by Example)” (John Wiley, 1994)
 - Cap. 3: “Representing structured knowledge”.
- Lucas, P. y Gaag, L.v.d. “Principles of Expert Systems” (Addison–Wesley, 1991).
 - Cap. 4: “Frames and inherance”.
- Mira, J.; Delgado, A.E.; Boticario, J.G. y Díez, F.J. “Aspectos básicos de la Inteligencia Artificial” (Sanz y Torres, 1995)
 - Cap. 7: “Redes asociativas”.
 - Cap. 8: “Marcos y guiones”.
- Rich, E. y Knight, K. “Inteligencia artificial (segunda edición)” (McGraw–Hill Interamericana, 1994).
 - Cap. 9: “Estructuras de ranura y relleno débiles”.
- Winston, P.R. “Inteligencia artificial (3a. ed.)” (Addison–Wesley, 1994).
 - Cap. 9: “Plantillas y herencia”.