Lógica informática (2011–12)

Tema 15: Implementación en Prolog de los tableros semánticos

José A. Alonso Jiménez
Andrés Cordón Franco
María J. Hidalgo Doblado

Grupo de Lógica Computacional
Departamento de Ciencias de la Computación e I.A.
Universidad de Sevilla

Tema 15: Implementación en Prolog de los tableros semánticos

1. Notación uniforme
2. Procedimiento de completación de tableros
3. Teorema por tableros
4. Deducción por tableros

Tema 15: Implementación en Prolog de los tableros semánticos

1. Notación uniforme

2. Procedimiento de completación de tableros

3. Teorema por tableros

4. Deducción por tableros

Notación uniforme

- `literal(+F)` se verifica si la fórmula `F` es un literal.

```
literal(F)  :- atom(F).
literal(-F) :- atom(F).
```

- `alfa(+A,-A1,-A2)` se verifica si `A` es una fórmula alfa y sus componentes son `A1` y `A2`.

```
alfa(A1 & A2,      A1,  A2).
alfa(-(A1 => A2), A1,  -A2).
alfa(-(A1 v A2),  -A1, -A2).
```

- `beta(+B,-B1,-B2)` se verifica si `B` es una fórmula beta y sus componentes son `B1` y `B2`.

```
beta(B1 v B2,       B1,        B2).
beta(B1 => B2,      -B1,       B2).
beta(-(B1 & B2),    -B1,       -B2).
beta(B1 <=> B2,     B1 & B2,   -B1 & -B2).
beta(-(B1 <=> B2), B1 & -B2, -B1 & B2).
```

Tema 15: Implementación en Prolog de los tableros semánticos

1. Notación uniforme
2. Procedimiento de completación de tableros
3. Teorema por tableros
4. Deducción por tableros

Tablero del conjunto de fórmulas S

Un tablero del conjunto de fórmulas S es un árbol construido mediante las reglas:

- El árbol cuyo único nodo tiene como etiqueta S es un tablero de S.
- Sea $\mathcal{T}$ un tablero de S y S_1 la etiqueta de una hoja de $\mathcal{T}$.
  1. Si S_1 contiene una fórmula y su negación, entonces el árbol obtenido añadiendo como hijo de S_1 el nodo etiquetado con $\{\bot\}$ es un tablero de S.
  2. Si S_1 contiene una doble negación $\neg\neg F$, entonces el árbol obtenido añadiendo como hijo de S_1 el nodo etiquetado con $(S_1 \setminus \{\neg\neg F\}) \cup \{F\}$ es un tablero de S.
  3. Si S_1 contiene una fórmula alfa F de componentes F_1 y F_2, entonces el árbol obtenido añadiendo como hijo de S_1 el nodo etiquetado con $(S_1 \setminus \{F\}) \cup \{F_1, F_2\}$ es un tablero de S.
  4. Si S_1 contiene una fórmula beta F de componentes F_1 y F_2, entonces el árbol obtenido añadiendo como hijos de S_1 los nodos etiquetados con $(S_1 \setminus \{F\}) \cup \{F_1\}$ y $(S_1 \setminus \{F\}) \cup \{F_2\}$ es un tablero de S.

Completación de tableros

- `tablero_completo(+S,-Tab)` se verifica si `Tab` es un tablero completo del conjunto `S`. Por ejemplo,

```
?- tablero_completo([- (-p v -q => - (p & r))],T).
T = t([- (-p v -q => - (p & r))],
      t([-p v -q, - -(p & r)],
        t([p & r, -p v -q],
          t([p, r, -p v -q],
            t([-p, p, r], cerrado, vacio),
            t([-q, p, r], abierto, vacio)),
          vacio),
        vacio),
      vacio)
Yes
```

- Representación: `t(S,Izq,Dcha)` representa el tablero de raíz `S`, rama izquierda `Izq` y rama derecha `Dcha`.

Completación de tableros

- Def. de `tablero_completo`:

```
tablero_completo(S,Tab) :-
    completacion(t(S,_Izq,_Dcha),Tab).
```

- `completacion(+Tab1,-Tab2)` se verifica si Tab2 es una completación (i.e. un tablero completo) del tablero Tab1.

```
completacion(t(Flas,Izq1,Dcha1),t(Flas,Izq3,Dcha3)) :-
    paso(t(Flas,Izq1,Dcha1),t(Flas,Izq2,Dcha2)), !,
    completacion(Izq2,Izq3),
    completacion(Dcha2,Dcha3).
completacion(Tab,Tab).
```

Completación de tableros

- `paso(+Tab1,-Tab2)` se verifica si `Tab2` es un tablero obtenido aplicando una regla de completación al tablero `Tab1`.

```
paso(t(S1,_,_),t(S1, cerrado, vacio)) :-
   cerrada(S1), !.
paso(t(S1,_,_),t(S1, abierto, vacio)) :-
   lista_de_literales(S1), !.
paso(t(S1,_,_),t(S1, Izq, vacio)) :-
   regla_doble_negacion(S1, S2), !,
   Izq = t(S2, _, _).
paso(t(S1,_,_),t(S1, Izq, vacio)) :-
   regla_alfa(S1, S2), !,
   Izq = t(S2, _, _).
paso(t(S1,_,_),t(S1, Izq, Dcha)) :-
   regla_beta(S1, S2, S3),
   Izq  = t(S2, _, _),
   Dcha = t(S3, _, _).
```

Completación de tableros

- `cerrada(+S)` se verifica si S es una lista cerrada (i.e. que contiene una fórmula y su negación).

```
cerrada(S) :-
   member(-F,S),
   member(F,S).
```

- `lista_de_literales(+S)` se verifica si S es una lista de literales.

```
lista_de_literales([]).
lista_de_literales([F|S]) :-
   literal(F),
   lista_de_literales(S).
```

Completación de tableros

- `regla_doble_negacion(+S1,-S2)` que se verifica si `S1` es una lista de fórmulas que contiene una doble negación `--F` y `S2` es la lista de fórmulas obtenidas sustituyendo en `S1` la fórmula `--F` por `F`. Por ejemplo,

```
?- regla_doble_negacion([- -(q v r), p => r],L).
L = [q v r, p => r]
?- regla_doble_negacion([p v (q v r), p => r],L).
No
```

```
regla_doble_negacion(S1, [F|S2]) :-
   member(- -F, S1), !,
   delete(S1, - -F, S2).
```

Completación de tableros

- `regla_alfa(+S1,-S2)` que se verifica si `S1` es una lista de fórmulas que contiene una fórmula alfa `A` y `S2` es la lista de fórmulas obtenidas sustituyendo en `S1` la fórmula `A` por sus componentes. Por ejemplos,

```
?- regla_alfa([p & (q v r), p => r],L).
L = [p, q v r, p => r]
?- regla_alfa([p v (q v r), p => r],L).
No
```

```
regla_alfa(S1, [A1,A2|S2]) :-
    member(A, S1),
    alfa(A, A1, A2), !,
    delete(S1, A, S2).
```

Completación de tableros

- `regla_beta(+S1,-S2,-S3)` que se verifica si `S1` es una lista de fórmulas que contiene al menos una fórmula beta `B` y `S2` es la lista de fórmulas obtenidas sustituyendo en `S1` la fórmula `B` por una de sus componentes y `S3`, por la otra. Por ejemplo,

```
?- regla_beta([p & (q v r), p => r],L1,L2).
L1 = [-p, p & (q v r)]
L2 = [r, p & (q v r)]
?- regla_beta([p & (q v r), -(p => r)],L1,L2).
No
```

```
regla_beta(S1, [B1|RS1], [B2|RS1]) :-
   member(B, S1),
   beta(B, B1, B2),
   delete(S1, B, RS1).
```

Tema 15: Implementación en Prolog de los tableros semánticos

1. Notación uniforme

2. Procedimiento de completación de tableros

3. Teorema por tableros

4. Deducción por tableros

Tableros cerrados

- `es_cerrado(+Tab)` se verifica si `Tab` es un tablero cerrado.

```
es_cerrado(t(_,Izq,Dcha)) :-
    es_cerrado(Izq),
    es_cerrado(Dcha).
es_cerrado(cerrado).
es_cerrado(vacio).
```

Prueba mediante tableros

- `prueba(+F,-Tab)` se verifica si `Tab` es una prueba de la fórmula `F`; es decir, un tablero completo cerrado de $\{\neg F\}$. Por ejemplo,

```
?- prueba(-p v -q => -(p & q),T).
T = t([- (-p v-q => - (p & q))],
      t([-p v-q, - -(p & q)],
        t([p & q, -p v -q],
          t([p, q, -p v -q],
            t([-p, p, q], cerrado, vacio),
            t([-q, p, q], cerrado, vacio)),
          vacio),
        vacio),
      vacio)
?- prueba(-p v -q => -p,T).
No
```

```
prueba(F,Tab) :-
   tablero_completo([-F],Tab), !,
   es_cerrado(Tab).
```

Teorema mediante tableros

- `es_teorema(+F)` se verifica si la fórmula `F` es teorema (mediante tableros semánticos). Por ejemplo,

```
?- es_teorema(-p v -q => -(p & q)).
Yes
?- es_teorema(-p v -q => -(p & r)).
No
```

```
es_teorema(F) :-
   prueba(F,_Tab).
```

Tema 15: Implementación en Prolog de los tableros semánticos

1. Notación uniforme

2. Procedimiento de completación de tableros

3. Teorema por tableros

4. Deducción por tableros

Deducción mediante tableros

- `prueba_deducible_tab(+S,+F,-Tab)` se verifica si `Tab` es una prueba por tableros semánticos de que la fórmula `F` es deducible a partir del conjunto de fórmulas `S`; es decir, existe un tablero completo cerrado de $S \cup \{\neg F\}$. Por ejemplo,

```
?- prueba_deducible_tab([p => q, q => r], p => r,T).
T = t([- (p => r), p => q, q => r],
      t([p, -r, p => q, q => r],
        t([-p, p, -r, q => r], cerrado, vacio),
        t([q, p, -r, q => r],
          t([-q, q, p, -r], cerrado, vacio),
          t([r, q, p, -r], cerrado, vacio))),
      vacio)
Yes
?- prueba_deducible_tab([p => q, q => r], p <=> r,T).
No
```

```
prueba_deducible_tab(S,F,Tab) :-
   tablero_completo([-F|S],Tab), !,
   es_cerrado(Tab).
```

Deducibilidad mediante tableros

- `es_deducible_tab(+S,+F)` se verifica si la fórmula `F` es deducible (mediante tableros) a partir del conjunto de fórmulas `S`.

```
es_deducible_tab(S,F) :-
   prueba_deducible_tab(S,F,_Tab).
```

Bibliografía

- Alonso, J.A. y Borrego, J. *Deducción automática (Vol. 1: Construcción lógica de sistemas lógicos)* (Ed. Kronos, 2002)
  - Cap. 4.1: Tableros semánticos
- Ben–Ari, M. *Mathematical Logic for Computer Science (2nd ed.)* (Springer, 2001)
  - Cap. 2: Propositional Calculus: Formulas, Models, Tableaux
- Fitting, M. *First-Order Logic and Automated Theorem Proving (2nd ed.)* (Springer, 1995)
- Nerode,A. y Shore, R.A. *Logic for Applications* (Springer, 1997)