Lógica informática (2011–12)

Tema 16: Implementación en Prolog de la normalización

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 16: Implementación en Prolog de la normalización

1. Forma normal negativa
2. Forma normal conjuntiva
3. Transformación a cláusulas

Tema 16: Implementación en Prolog de la normalización

1. Forma normal negativa
2. Forma normal conjuntiva
3. Transformación a cláusulas

Cálculo de la forma normal negativa

Aplicando a una fórmula F los siguientes pasos se obtiene una forma normal negativa de F:

1. Eliminar las equivalencias usando la relación
$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \quad (1)$$
2. Eliminar las implicaciones usando la equivalencia
$$A \rightarrow B \equiv \neg A \vee B \quad (2)$$
3. Interiorizar las negaciones usando las equivalencias
$$\neg(A \wedge B) \equiv \neg A \vee \neg B \quad (3)$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B \quad (4)$$
$$\neg\neg A \equiv A \quad (5)$$

Cálculo de la forma normal negativa

- `fnn(+F,?G)` se verifica si `G` es una forma normal negativa de la fórmula `F`. Por ejemplo,

```
?- fnn(p <=> q,F).
F = (-p v q) & (-q v p)
?- fnn(p v -q => r, F).
F = (-p & q) v r
?- fnn(p & (q => r) => s, F).
F = (-p v (q & -r)) v s
```

```
fnn(F, G) :-
    elimina_equivalencias(F,F1),
    elimina_implicaciones(F1,F2),
    interioriza_negaciones(F2,G).
```

Eliminación de equivalencias

- `elimina_equivalencias(+F,?G)` se verifica si `G` es la fórmula obtenida eliminando las equivalencias de la fórmula `F`, usando la reducción (1).

```
elimina_equivalencias(A <=> B, (A1 => B1) & (B1 => A1)) :- !,
   elimina_equivalencias(A, A1),
   elimina_equivalencias(B, B1).
elimina_equivalencias(A, B) :-
   A =.. [Op|L1], !,
   maplist(elimina_equivalencias,L1,L2),
   B =.. [Op|L2].
elimina_equivalencias(A, A).
```

Eliminación de las implicaciones

- `elimina_implicaciones(+F,?G)` se verifica si `G` es la fórmula obtenida eliminando las implicaciones de la fórmula `F`, usando la reducción (2).

```
elimina_implicaciones(A => B, -A1 v B1) :- !,
   elimina_implicaciones(A, A1),
   elimina_implicaciones(B, B1).
elimina_implicaciones(A, B) :-
   A =.. [Op|L1], !,
   maplist(elimina_implicaciones,L1,L2),
   B =.. [Op|L2].
elimina_implicaciones(A, A).
```

Interiorización de las negaciones

- `interioriza_negaciones(+F,?G)` se verifica si `G` es la fórmula obtenida interiorizando las negaciones de la fórmula `F` (que no tiene => ni <=>) de forma que las negaciones se apliquen sólo sobre símbolos proposicionales. Las reglas de interiorización son las reducciones (3), (4) y (5).

```
interioriza_negaciones(-(A & B), A1 v B1) :- !,
   interioriza_negaciones(-A, A1),
   interioriza_negaciones(-B, B1).
interioriza_negaciones(-(A v B), A1 & B1) :- !,
   interioriza_negaciones(-A, A1),
   interioriza_negaciones(-B, B1).
interioriza_negaciones((-(-A)),  A1) :- !,
   interioriza_negaciones(A, A1).
interioriza_negaciones(A, B) :-
   A =.. [Op|L1], !,
   maplist(interioriza_negaciones,L1,L2),
   B =.. [Op|L2].
interioriza_negaciones(A, A).
```

Tema 16: Implementación en Prolog de la normalización

1. Forma normal negativa
2. Forma normal conjuntiva
3. Transformación a cláusulas

Forma normal conjuntiva

- Aplicando a una fórmula F los siguientes pasos se obtiene una forma normal conjuntiva de F:
  1. Calcular una forma normal negativa de F.
  2. Interiorizar las disyunciones usando la propiedad distributiva de la disyunción sobre la conjunción

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C) \quad (6)$$
$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \quad (7)$$

- `fnc(+F,?G)` se verifica si `G` es una forma normal conjuntiva de la fórmula `F`. Por ejemplo,

```
?- fnc(p & (q => r), F).
F = p& (-q v r)
?- fnc(-(p & (q => r)), F).
F = (-p v q)& (-p v-r)
```

```
fnc(F,G) :-
   fnn(F,F1),
   interioriza_disyunciones(F1,G).
```

Interiorización de las disyunciones

```
interioriza_disyunciones(A v (B & C), ABC) :- !,
   interioriza_disyunciones(A v B, AB),
   interioriza_disyunciones(A v C, AC),
   interioriza_disyunciones(AB & AC, ABC).
interioriza_disyunciones((A & B) v C, ABC) :- !,
   interioriza_disyunciones(A v C, AC),
   interioriza_disyunciones(B v C, BC),
   interioriza_disyunciones(AC & BC, ABC).
interioriza_disyunciones(A, B) :-
   A =.. [Op|L],
   maplist(interioriza_disyunciones,L,L1),
   ( L1 = L ->
     B =.. [Op|L1]
   ; % not(L1 = L) ->
     B1 =.. [Op|L1],
     interioriza_disyunciones(B1,B)).
interioriza_disyunciones(A, A).
```

Tema 16: Implementación en Prolog de la normalización

1. Forma normal negativa
2. Forma normal conjuntiva
3. Transformación a cláusulas

Transformación de fórmulas clausales en cláusulas

- Las fórmulas clausales son las fórmulas obtenidas mediante las siguientes reglas
  - Si F es un literal, entonces F es una fórmula clausal.
  - Si F y G son fórmulas clausales, entonces $F \vee G$ es una fórmula clausal.

  Por ejemplo,
  - p, $\neg p$ y $\neg p \vee (q \vee \neg r)$ son fórmulas clausales.
  - $\neg p \vee (q \wedge \neg r)$ no es una fórmula clausal
- `cláusula(+F,-C)` se verifica si `C` es una cláusula equivalente a la fórmula clausal `F`. Por ejemplo,

```
?- cláusula(p,C).                    =>  C = [p]
?- cláusula(-p,C).                   =>  C = [-p]
?- cláusula((-p v r) v (-p v q),C).  =>  C = [q, r, -p]
```

```
cláusula(F1 v F2, S) :- !,
   cláusula(F1, S1),
   cláusula(F2, S2),
   append(S1, S2, S3),
   sort(S3,S).
cláusula(L, [L]).
```

Transformación de fórmulas en FNC a conjunto de cláusulas

- `cláusulas_FNC(+F,?S)` se verifica si `S` es un conjunto de cláusulas equivalente a la fórmula en forma normal conjuntiva `F`. Por ejemplo,

```
?- cláusulas_FNC(p & (-q v r), S).
S = [[p], [r, -q]]
?- cláusulas_FNC((-p v q) & (-p v -r), S).
S = [[q, -p], [-p, -r]]
```

```
cláusulas_FNC(A1 & A2, S) :- !,
   cláusulas_FNC(A1, S1),
   cláusulas_FNC(A2, S2),
   union(S1, S2, S).
cláusulas_FNC(A, [S]) :-
   cláusula(A, S).
```

Transformación de fórmulas a conjunto de cláusulas

- `cláusulas(+F,?S)` se verifica si `S` es un conjunto de cláusulas equivalente a la fórmula `F`. Por ejemplo,

```
?- cláusulas(p & (q => r),S).
S = [[p], [r, -q]]
```

```
cláusulas(F,S) :-
   fnc(F,F1),
   cláusulas_FNC(F1,S).
```

Transformación de conjuntos de fórmulas a conjunto de cláusulas

- `cláusulas_conjunto(+CF,-CC)` se verifica si `CC` es un conjunto de cláusulas equivalente al conjunto de fórmulas `CF`. Por ejemplo,

```
?- cláusulas_conjunto([p => q, q => r],CC).
CC = [[q, -p], [r, -q]]
```

```
cláusulas_conjunto([],[]).
cláusulas_conjunto([F|CF],CC) :-
   cláusulas(F,CC1),
   cláusulas_conjunto(CF,CC2),
   union(CC1,CC2,CC).
```

Bibliografía

- Alonso, J.A. y Borrego, J. *Deducción automática (Vol. 1: Construcción lógica de sistemas lógicos)* (Ed. Kronos, 2002)
  - Cap. 4.2: Cláusulas.
- Fitting, M. *First-Order Logic and Automated Theorem Proving (2nd ed.)* (Springer, 1995)
- Schöning, U. *Logic for Computer Scientists* (Birkäuser, 1989)