Lògica en la Informàtica

Tardor 2022. Teoria classe 3

Lògica en la Informàtica

Temari

1. Introducció i motivació
2. Definició de Lògica Proposicional (Lprop)
3. Deducció en Lògica Proposicional
4. Definició de Lògica de Primer Ordre (LPO)
5. Deducció en Lògica de Primer Ordre
6. Programació Lògica (Prolog)

Lògica en la Informàtica. Definició de Lògica Proposicional

Aplicacions directes de la lògica en la informàtica:

- Verificació de hardware i de software
  - demostració de correcció (terminació, etc.)
  - testing
- Aplicacions "crítiques" en:
  - vides humanes: centrals nuclears, químicas, avions, trànsit, cotxes, trens,... "safety"
  - confidencialitat: diners electrònics, signatura electrònic, dades bancaris... "security"
  - economia: la borsa, la telefonia, el sistema eléctric...
- Intel·ligència artificial, web semàntica (representaciò del coneixement: ontologías, description logics, sistemes experts, ...)
- Bases de dades
- Programació lògica (prolog)
- Ús de lògica per a resoldre problemes d'optimització, planificació...: per exemple, https://barcelogic.com/
  - especificació/formalització fent servir lògica
  - "solvers" lògics, per exemple, SAT solvers.

Lògica en la Informàtica. Definició de Lògica Proposicional

En la pràctica ens interessa sempre, esbrinar aquest tipus de propietats:

F és **satisfactible** si F té algun model

F és **insatisfactible** si F no té models

F és **tautología** si tota I és model de F

G és **conseqüència lògica** de F si tot model de F satisfà G (es denota F |= G)

F i G són **lògicamente equivalents** si F i G tenen els mateixos models (F ≡ G)

Com ho podem fer, si l'única cosa que tenim és un SAT solver?

F és tautologia ssi -F és insatisfactible. [Ex. 6]

G és conseqüència lògica de F ssi $F \land \neg G$ és insatisfactible. [Ex. 7]

F y G són lògicamente equivalentes ssi $(G \land \neg F) \lor (F \land \neg G)$ és insatisfactible. [Ex. 8]

Lògica en la Informàtica. Definició de Lògica Proposicional

Exercicis del capítol 2 dels apunts

16. Siguin F i G dues fórmules qualsevols.
    Si F -> G és satisfactible i F és satisfactible, llavors G és satisfactible?
    Demostra-ho fent servir tan sols la definició de la LProp.

Farem un *intent de demostració*:

a) F -> G és satisfactible ssi [per def. de ->]
…
E I, max(eval_I(-F), eval_I(G)) = 1 ssi [per def. de eval_I(-)]
E I, max(1-eval_I(F), eval_I(G)) = 1 ssi [per def. de max]
E I, 1-eval_I(F) = 1 o bé eval_I(G) = 1 ssi [per aritmètica]
E I, eval_I(F) = 0 **o bé** eval_I(G) = 1

b) F és satisfactible ssi [per def. de satisfactible]
…
E I', eval_I'(F) = 1

Lògica en la Informàtica. Definició de Lògica Proposicional

Exercicis del capítol 2 dels apunts

21. Demostra que l'equivalència lògica és realment una relació d'**equivalència**.

Una relacion binària R sobre un conjunt S és un subconjunt del producte cartesià S x S.
És a dir R ens diu quines parelles estan relacionades, quines parelles (e,e') estan en R
(on e i e' són elements de S).

R és **reflexiva**	si (e,e) està en R		per a tot e	de S.
R és **simètrica**	si (e,e') en R	implica (e',e) en R	per a tot e,e'	de S.
R és **transitiva**	si (e,e') en R i (e',e'') en R	implica (e, e'') en R	per a tot e,e',e''	de S.

I si R compleix les tres propietats llavors R és una relació d'**equivalència**.

Lògica en la Informàtica. Definició de Lògica Proposicional

Exercicis del capítol 2 dels apunts

18. Demostra les següents equivalències entre fórmules:

$F \land F$	$\equiv \; F$	idempotència de $\land$
$F \lor F$	$\equiv \; F$	idempotència de $\lor$
$F \land G$	$\equiv \; G \land F$	conmutativitat de $\land$
$F \lor G$	$\equiv \; G \lor F$	conmutativitat de $\lor$
$(F \land G) \land H$	$\equiv \; F \land (G \land H)$	associativitat de $\land$
$(F \lor G) \lor H$	$\equiv \; F \lor (G \lor H)$	associativitat de $\lor$

Aquestes tres propietats (idempotència, commutativitat, asociatividad de & i de v) ens indiquen que a vegades podem escriure les fórmules de manera més "relaxada", ometent alguns parèntesis. I també, que podem veure una CNF com un CONJUNT (and) de clàusules, i podem veure una clàusula com un CONJUNT (un or) de literals.

Lògica en la Informàtica. Definició de Lògica Proposicional

Exercicis del capítol 2 dels apunts

18. Demostra les següents equivalències entre fórmules:

$\neg\neg F$	$\equiv$	F	doble negació
$\neg(F \land G)$	$\equiv$	$\neg F \lor \neg G$	llei de De Morgan 1
$\neg(F \lor G)$	$\equiv$	$\neg F \land \neg G$	llei de De Morgan 2

Aquestes tres propietats ens serveixen per a transformar fórmules "movent les negacions cap a dins", fins que només hi hagi negacions aplicades a símbols de predicat.

Lògica en la Informàtica. Definició de Lògica Proposicional

Exercicis del capítol 2 dels apunts

18. Demostra les següents equivalències entre fórmules:

$(F \land G) \lor H \equiv (F \lor H) \land (G \lor H)$ distributivitat 1
$(F \lor G) \land H \equiv (F \land H) \lor (G \land H)$ distributivitat 2

Una vegada les negacions estan aplicades als símbols de predicat, aplicant distributivitat 1
$(F \land G) \lor H \Longrightarrow (F \lor H) \land (G \lor H)$ d'esquerra a dreta obtenim una CNF.

Hi ha un detall: Demostra que $p \land (q \lor q) \equiv p \land q$.
Podem "aplicar" alegrement la idempotència del v sobre la subfórmula q v q?
No! Cal demostrar primer el següent *Lema de Substitució*:

Lògica en la Informàtica. Definició de Lògica Proposicional

Exercicis del capítol 2 dels apunts

23. Lema de Substitució.

    Siguin F, G, G' fórmules qualsevols, amb $G \equiv G'$.

    Si en F substituïm una aparició de una subfórmula G per G' obtenim una nova fórmula F’ amb $F \equiv F'$.

    En l’exemple anterior:

    F és $p \land (q \lor q)$
    G és $(q \lor q)$
    G' es q
    F' és $p \land q$.

Lògica en la Informàtica. Definició de Lògica Proposicional

Exercicis del capítol 2.

26. Suposem que |P| = 100 i que ens interessa determinar si una fórmula F construïda sobre P és satisfactible o no. Si l'algorisme està basat en un anàlisi de la taula de veritat i avaluar F en una interpretació I donada triga un microsegon (10^−6 segons), quants anys trigarà (si F és insatisfactible)?

- quantes I's hi ha?   n'hi ha 2^100
- 2^10 = 1024 ~~ 10^3

  2^100 ~~ 10^30

avaluar-les totes trigarà 2^100 * 10^-6 segons = 10^30 * 10^-6 segons
= 10^24 segons = 10^24 / (365*24*3600) anys ~~ 4 * 10^16 anys aprox.

Lògica en la Informàtica. Definició de Lògica Proposicional

27. Una *funció booleana* de *n* entrades és una funció f : {0, 1}^n → {0, 1}, és a dir, una funció que prem com entrada una cadena de *n* bits i retorna un bit. Quantes funcions booleanes de n entrades hi ha?

```
x —+——+
   |  f  |—
y —+——+
```

hi ha 2 elevat a (2 elevat a n): hi ha tantes funcions com tires de 2^n bits

Lògica en la Informàtica. Definició de Lògica Proposicional

27. Una *funció booleana* de *n* entrades és una funció f : {0, 1}^n → {0, 1}, és a dir, una funció que prem com entrada una cadena de *n* bits i retorna un bit. Quantes funcions booleanes de n entrades hi ha?

hi ha 2 elevat a (2 elevat a n): hi ha tantes funcions com tires de 2^n bits

Exemple n=2: tantes funcions com tires de 2^n bits = tires de 4 bits = 2^4

x y	0	and	-(x->y)	x	-(y->x)	y	xor	or	nor	=	-y	y->x	-x	x->y	nand	1
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Lògica en la Informàtica. Definició de Lògica Proposicional

27. Una *funció booleana* de *n* entrades és una funció f : {0, 1}^n → {0, 1}, és a dir, una funció que prem com entrada una cadena de *n* bits i retorna un bit. Quantes funcions booleanes de n entrades hi ha?

hi ha 2 elevat a (2 elevat a n): hi ha tantes funcions com tires de 2^n bits

Exemple: n=3; hi ha 2^8 = 256
x y z
0 0 0
...
1 1 1

si n=4, hi ha 2^64 =   ~~ 65000

Lògica en la Informàtica. Definició de Lògica Proposicional

28. Cada fórmula F representa una única funció booleana: la que retorna 1 exactament per a aquellas cadenes de bits I tals que eval_I(F) = 1.
Per això, dues fórmules són lògicament equivalents si i només si representen la mateixa funció booleana. Quantes funcions booleanes (o quantes fórmules lògicamente no-equivalents) hi ha en funció de n = |P|?

Per exemple, la funció booleana "and" (de 2 entrades x, y) la podem representar mitjançant les fórmules
x & y
(x & x) & y
-(-x v -y)
-(-x v -y) & y
...

hi ha 2 elevat a (2 elevat a n): hi ha tantes funcions com tires de 2^n bits

Lògica en la Informàtica. Definició de Lògica Proposicional

31. Escriu en una taula de veritat les 16 funcions booleanes de 2 entrades. Quantes de elles només depenen d'una de las dues entrades? Quantes depenen de zero entrades? Les altres, vistes com conectives lògiques, reben algun nom? Ja sabem que podem expressar qualsevol funció booleana amb el conjunt de tres conectivas $\{\wedge, \vee, \neg\}$, és a dir, qualsevol funció booleana és equivalent a una fórmula construïda sobre aquestes tres conectivas. És cert això també para algún conjunto de només dues de les 16 funcions? (Hi ha diverses maneres, però basta amb donar una sola.)

Lògica en la Informàtica. Definició de Lògica Proposicional

31. Escriu en una taula de veritat les 16 funcions booleanes de 2 entrades. Quantes de elles només depenen d'una de las dues entrades? Quantes depenen de zero entrades? Les altres, vistes com conectives lògicas, reben algun nom?

x y	0	and	no x->y	x	no y->x	y	xor	or	nor	=	-y	y->x	-x	x->y	nand	1
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Lògica en la Informàtica. Definició de Lògica Proposicional

31. Escriu en una taula de veritat les 16 funcions booleanes de 2 entrades. Quantes de elles només depenen d'una de las dues entrades? Quantes depenen de zero entrades? Les altres, vistes com conectives lògicas, reben algun nom?

Ja sabem que podem expressar qualsevol funció booleana
amb el conjunt de tres conectivas $\{\wedge, \vee, \neg\}$, és a dir, qualsevol funció booleana és equivalent a una fórmula construïda sobre aquestes tres conectivas. És cert això també para algún conjunto de només dues de les 16 funcions? (Hi ha diverses maneres, però basta amb donar una sola.)

Sí, amb només or i not, per exemple: $x \wedge y$ === $\neg(\neg x \vee \neg y)$

Lògica en la Informàtica. Definició de Lògica Proposicional

32. Demostra que qualsevol funció booleana de dues entrades es pot expressar amb només nor o bé amb només nand, on nor(F, G) és ¬(F ∨ G), i nand(F, G) és ¬(F ∧ G).

ho fem per nand:

- not F == F nand F

- F or G == not(not(F) and not(G)) == not(F) nand not(G)
  == (F nand F) nand (G nand G)

- F and G == not(F nand G) == (F nand G) nand (F nand G)

Lògica en la Informàtica. Definició de Lògica Proposicional

37. Considera el següent fragment de codi, que retorna un booleà:

```
int i;
bool a, b;
...
if (a and i>0)        return b;      // (1)
else if (a and i<=0) return false; // (2)
else if (a or b)      return a;      // (3)
else                  return (i>0); // (4)
```

Simplifíca'l sustituïnt els valors de retorn per un sol valor de retorn que sigui una expressió booleana en els predicats *i > 0*, *a* i *b*:

```
int i;
bool a, b;
return ...;
```

Lògica en la Informàtica. Definició de Lògica Proposicional

37. Considera el següent fragment de codi, que retorna un booleà:

```
int i;
bool a, b;
...
if (a and i>0)          return b;       // (1)
else if (a and i<=0) return false; // (2)
else if (a or b)        return a;       // (3)
else                    return (i>0); // (4)
```

```
return ((not a) and (not b) and i>0) or
       (a and b and i>0);
return ((not a and not b) or (a or b)) and i>0:

return (a==b) and i>0;
```

Tindrem tres símbols de predicat; a, b, i>0.

a,	b,	i>0	return	
0	0	0	0	(4)
0	0	1	1	(4)
0	1	0	0	(3)
0	1	1	0	(3)
1	0	0	0	(2)
1	0	1	0	(1)
1	1	0	0	(2)
1	1	1	1	(1)

Lògica en la Informàtica. Definició de Lògica Proposicional

33. Tres estudiants A, B i C són acusats d'introduir un virus en les sales d’ordinadors de la FIB. Durant l’interrogatori, les declaracions són les següents:

- A diu: “B ho va fer i C és innocent”
- B diu: “Si A és culpable llavors C també ho és”
- C diu: “Jo no ho vaig fer, ho va fer almenys un dels altres dos”

a) Són les tres declaracions contradictories?
b) Assumint que tots son innocents, qui o quins van mentir en la declaració?
c) Assumint que ningún va mentir, qui és innocent i qui és culpable?

Lògica en la Informàtica. Definició de Lògica Proposicional

33. Tres estudiants A, B i C són acusats d'introduir un virus en les sales d'ordinadors de la FIB. Durant l'interrogatori, les declaracions són les següents:

- A diu: "B ho va fer i C és innocent"
- B diu: "Si A és culpable llavors C també ho és"
- C diu: "Jo no ho vaig fer, ho va fer almenys un dels altres dos"

Introduïm símbols de predicat: a,b,c que signifiquen: "A ho va fer", "B ho va fer", "C ho va fer".

Les tres declaracions són:

A diu: b & -c
B diu: a -> c
C diu: -c & (a v b)

Lògica en la Informàtica. Definició de Lògica Proposicional

33. Tres estudiants A, B i C són acusats d'introduïr un virus en les sales d'ordinadors de la FIB. Durant l'interrogatori, les declaracions són les següents:

Les tres declaracions són:

A diu: b & -c
B diu: a -> c
C diu: -c & (a v b)

a) Són les tres declaracions contradictòries?

Per saber si poden ser veritat les tres declaracions, formalment, hem de veure si és satisfactible la conjunció (and) de les tres fórmules.

Només hi ha un modelo I: I(a)=0, I(b)=1, I(c)=0.
Per tant, no són contradictòries

Lògica en la Informàtica. Definició de Lògica Proposicional

33. Tres estudiants A, B i C són acusats d'introduir un virus en les sales d'ordinadors de la FIB. Durant l'interrogatori, les declaracions són les següents:

Les tres declaracions són:

A diu: b & -c
B diu: a -> c
C diu: -c & (a v b)

b) Assumint que tots són innocents, qui o quins van mentir en la declaració?

Si tots són innocents, I(a) = I(b) = I(c) = 0, i per tant A i C van mentir.

Lògica en la Informàtica. Definició de Lògica Proposicional

33. Tres estudiants A, B i C són acusats d'introduir un virus en les sales d'ordinadors de la FIB. Durant l'interrogatori, les declaracions són les següents:

Les tres declaracions són:

A diu: b & -c
B diu: a -> c
C diu: -c & (a v b)

c) Assumint que ningún va mentir, qui és innocent i qui és culpable?

Si ningú ha mentit, llavors B és culpable, tal com es comprova en l'únic model trobat abans: I(a)=0, I(b)=1, I(c)=0.

Lògica en la Informàtica. Definició de Lògica Proposicional

34. Inventa i defineix formalment alguna altre lògica diferent a la lògica proposicional. Per exemple, si les interpretacions són funcions I: P → {0, 1, ⊥} que també poden donar “indefinit” ⊥, es pot adaptar la noció de satisfacció de manera raonable, tot i que la resposta ja no serà binaria: l’“avaluació” d’una fórmula F en una interpretació I pot donar 1 (I satisfà F) o 0 (I no satisfà F) o ⊥ (indefinit).

Podem fer-ho així, de manera raonable, suposant que ⊥ en la nostra aplicació modela "no ho sé":

El valor d’una variable pot estar “indefinit”:

```
if ( x and y )  {
}
```

Lògica en la Informàtica. Definició de Lògica Proposicional

34. Inventa i defineix formalment alguna altre lògica diferent a la lògica proposicional. Per exemple, si les interpretacions són funcions I: P → {0, 1, ⊥} que també poden donar "indefinit" ⊥, es pot adaptar la noció de satisfacció de manera raonable, tot i que la resposta ja no serà binaria: l'"avaluació" d'una fórmula F en una interpretació I pot donar 1 (I satisfà F) o 0 (I no satisfà F) o ⊥ (indefinit).

eval_I(not F) =

F	dona:
1	0
0	1
⊥	⊥

eval_I(F and G) =

F	G	dona:
0	0	0
0	1	0
0	⊥	0
1	0	0
1	1	1
1	⊥	⊥
⊥	0	0
⊥	1	⊥
⊥	⊥	⊥

eval_I(F or G) =

F	G	dona:
0	0	0
0	1	1
0	⊥	⊥
1	0	1
1	1	1
1	⊥	1
⊥	0	⊥
⊥	1	1
⊥	⊥	⊥

Lògica en la Informàtica. Definició de Lògica Proposicional

34. Inventa i defineix formalment alguna altre lògica diferent a la lògica proposicional. Per exemple, si les interpretacions són funcions $I: P \rightarrow \{0, 1, \perp\}$ que també poden donar “indefinit” $\perp$, es pot adaptar la noció de satisfacció de manera raonable, tot i que la resposta ja no serà binaria: l’“avaluació” d’una fórmula F en una interpretació I pot donar 1 (I satisfà F) o 0 (I no satisfà F) o $\perp$ (indefinit).

I si $\perp$ modela "no termina"? Per exemple, en un programa com:

```
if ( not f(...) ) {
}

if ( f(...) and g(...) ) {
}

if ( f(...) or g(...) ) {
}
```

Lògica en la Informàtica. Definició de Lògica Proposicional

34. Inventa i defineix formalment alguna altre lògica diferent a la lògica proposicional. Per exemple, si les interpretacions són funcions I: P → {0, 1, ⊥} que també poden donar "indefinit" ⊥, es pot adaptar la noció de satisfacció de manera raonable, tot i que la resposta ja no serà binaria: l'"avaluació" d'una fórmula F en una interpretació I pot donar 1 (I satisfà F) o 0 (I no satisfà F) o ⊥ (indefinit).

eval_I(not F) =

F		
1	dona:	0
0		1
⊥		⊥

eval_I(F and G) =

F	G		
0	0	dona:	0
0	1		0
0	⊥		0
1	0		0
1	1		1
1	⊥		⊥
⊥	**0**		**⊥**
⊥	1		⊥
⊥	⊥		⊥

eval_I(F or G) =

F	G		
0	0	dona:	0
0	1		1
0	⊥		⊥
1	0		1
1	1		1
1	⊥		1
⊥	0		⊥
⊥	**1**		**⊥**
⊥	⊥		⊥

Lògica en la Informàtica. Definició de Lògica Proposicional

35. Com l'exercici anterior, però considerant I: P → [0…1], és a dir, l'interpretació d'un símbol p és una probabilitat (un número real entre 0 i 1). En aquest cas, l'avaluació d'una fórmula F en una interpretació I pot donar quelcom (remotamente) semblant a la probabilitat de satisfacció de F en I. En la lògica que has definit, l'avaluació de F en una I determinada, i la de F $\wedge$ F en aquesta mateixa I donen el mateix resultat?

```
eval_I( not F)        =  1 - eval_I( F )
eval_I( F and G )  =  eval_I( F ) * eval_I( G )
eval_I( F or  G )    =  (eval_I(F) + eval_I(G)) -  (eval_I(F) * eval_I(G))
```

Això ens ho hem inventat, però és incorrecte en general perquè les probabilitats de les subfórmules no són independents. Per exemple, l'avaluació de F $\wedge$ F hauria de donar el mateix que la de F i aquí no és així.

Lògica en la Informàtica. Deducció en Lògica Proposicional

Col·lecció d'apunts bàsics de lògica. Fitxer p3.pdf

1. Formes normals i clàusules
    - Fórmules com a conjunts
    - Literals
    - CNF
    - DNF
    - Clàusules
    - Conjunt de clàusules
    - Clàusula buida
    - Clàusula de Horn

Lògica en la Informàtica. Deducció en Lògica Proposicional

Exercicis del tema 3

- Exercicis 1,2
- Exercici 3
- Exercici 4

Lògica en la Informàtica. Deducció en Lògica Proposicional

Exercicis del tema 3

5. Demostra que una clàusula és una tautologia si, i només si, conté alhora p i $\neg p$ per un cert símbol proposicional p.

Lògica en la Informàtica. Deducció en Lògica Proposicional

Exercicis del tema 3

5. Demostra que una clàusula és una tautologia si, i només si, conté alhora p i ¬p per un cert símbol proposicional p.

Demostrarem que una clàusula de la forma $p_1 \vee \ldots \vee p_m \vee \neg q_1 \vee \ldots \vee \neg q_n$ NO és tautologia ssi NO conté alhora p i ¬p.

Lògica en la Informàtica. Deducció en Lògica Proposicional

Exercicis del tema 3

5. Demostra que una clàusula és una tautologia si, i només si, conté alhora p i ¬p per un cert símbol proposicional p.

Demostrarem que una clàusula de la forma p1 v … v pm v ¬q1 v … v ¬qn NO és tautologia ssi NO conté alhora p i ¬p.

p1 v...v pm v -q1 v...v -qn NO es tautologia ssi

Lògica en la Informàtica. Deducció en Lògica Proposicional

Exercicis del tema 3

5. Demostra que una clàusula és una tautologia si, i només si, conté alhora p i ¬p per un cert símbol proposicional p.

Demostrarem que una clàusula de la forma p1 v … v pm v ¬q1 v … v ¬qn NO és tautologia ssi NO conté alhora p i ¬p.

p1 v...v pm v -q1 v...v -qn NO es tautologia ssi
E I, tal que I no és model de p1 v...v pm v -q1 v...v -qn ssi

Lògica en la Informàtica. Deducció en Lògica Proposicional

Exercicis del tema 3

5. Demostra que una clàusula és una tautologia si, i només si, conté alhora p i ¬p per un cert símbol proposicional p.

Demostrarem que una clàusula de la forma $p1 \vee \ldots \vee pm \vee \neg q1 \vee \ldots \vee \neg qn$ NO és tautologia ssi NO conté alhora p i ¬p.

p1 v...v pm v -q1 v...v -qn NO es tautologia ssi
E I, tal que I no és model de p1 v...v pm v -q1 v...v -qn ssi
E I, tal que NO I |= p1 v...v pm v -q1 v...v -qn ssi

Lògica en la Informàtica. Deducció en Lògica Proposicional

Exercicis del tema 3

5. Demostra que una clàusula és una tautologia si, i només si, conté alhora p i ¬p per un cert símbol proposicional p.

Demostrarem que una clàusula de la forma p1 v … v pm v ¬q1 v … v ¬qn NO és tautologia ssi NO conté alhora p i ¬p.

p1 v...v pm v -q1 v...v -qn NO es tautologia ssi
E I, tal que I no és model de p1 v...v pm v -q1 v...v -qn ssi
E I, tal que NO I |= p1 v...v pm v -q1 v...v -qn ssi
E I, tal que max(eval_I(p1)... eval_I(pm), eval_I(-q1), ... eval_I(-qn)) = 0 ssi

Lògica en la Informàtica. Deducció en Lògica Proposicional

Exercicis del tema 3

5. Demostra que una clàusula és una tautologia si, i només si, conté alhora p i ¬p per un cert símbol proposicional p.

Demostrarem que una clàusula de la forma p1 v … v pm v ¬q1 v … v ¬qn NO és tautologia ssi NO conté alhora p i ¬p.

p1 v...v pm v -q1 v...v -qn NO es tautologia ssi
E I, tal que I no és model de p1 v...v pm v -q1 v...v -qn ssi
E I, tal que NO I |= p1 v...v pm v -q1 v...v -qn ssi
E I, tal que max(eval_I(p1)... eval_I(pm), eval_I(-q1), ... eval_I(-qn)) = 0 ssi
E I, tal que I(pi)=0 per a totes les pi i I(qj)=1 per a totes les qj ssi

Lògica en la Informàtica. Deducció en Lògica Proposicional

Exercicis del tema 3

5. Demostra que una clàusula és una tautologia si, i només si, conté alhora p i ¬p per un cert símbol proposicional p.

Demostrarem que una clàusula de la forma p1 v … v pm v ¬q1 v … v ¬qn NO és tautologia ssi NO conté alhora p i ¬p.

p1 v...v pm v -q1 v...v -qn NO es tautologia ssi
E I, tal que I no és model de p1 v...v pm v -q1 v...v -qn ssi
E I, tal que NO I |= p1 v...v pm v -q1 v...v -qn ssi
E I, tal que max(eval_I(p1)... eval_I(pm), eval_I(-q1), ... eval_I(-qn)) = 0 ssi
E I, tal que I(pi)=0 per a totes les pi i I(qj)=1 per a totes les qj ssi
pi != qj per a tota i,j

Lògica en la Informàtica. Deducció en Lògica Proposicional

6. Sigui S un conjunt de clàusules amb $\square \notin S$. Demostra que S és satisfactible (donant un model per a S) en cadascuna de les següents situacions:

   a) Tota clàusula de S té algún literal positiu.

   b) Tota clàusula de S té algún literal negatiu.

   c) Per tot símbol de predicat p es compleix que: o bé p apareix només en literals positius en S, o bé p apareix només en literals negatius en S.

Lògica en la Informàtica. Deducció en Lògica Proposicional

6. Sigui S un conjunt de clàusules amb $\square \notin S$. Demostra que S és satisfactible (donant un model per a S) en cadascuna de les següents situacions:

   a) Tota clàusula de S té algún literal positiu.

Sigui S = { C1, C2, … }. Cada clàusula Ci te almenys un literal positiu (un símbol de predicat sense negar).
Un model que satisfarà totes les clàusules de S és la I tal que I(p)=1 per tot símbol de predicat p.

Lògica en la Informàtica. Deducció en Lògica Proposicional

6. Sigui S un conjunt de clàusules amb $\Box \notin S$. Demostra que S és satisfactible (donant un model per a S) en cadascuna de les següents situacions:

   b) Tota clàusula de S té algún literal negatiu.

Sigui S = { C1, C2, … }. Cada clàusula Ci te almenys un literal negatiu (un símbol de predicat negat).
Un model que satisfarà totes les clàusules de S és la I tal que I(p)=0 per tot símbol de predicat p.

Lògica en la Informàtica. Deducció en Lògica Proposicional

6. Sigui S un conjunt de clàusules amb $\square \notin S$. Demostra que S és satisfactible (donant un model per a S) en cadascuna de les següents situacions:

   c) Per tot símbol de predicat p es compleix que: o bé p apareix només en literals positius en S, o bé p apareix només en literals negatius en S.

Lògica en la Informàtica. Deducció en Lògica Proposicional

6. Sigui S un conjunt de clàusules amb $\square \notin S$. Demostra que S és satisfactible (donant un model per a S) en cadascuna de les següents situacions:

    c) Per tot símbol de predicat p es compleix que: o bé p apareix només en literals positius en S, o bé p apareix només en literals negatius en S.

És a dir, cada clàusula és de la forma $p1 \lor \ldots \lor pm \lor -q1 \lor \ldots \lor -qn$ on

- les pi's només apareixen en literals positius en la resta de les clàusulas
- las qi's només apareixen en literals negatius en la resta de les clàusulas

Lògica en la Informàtica. Deducció en Lògica Proposicional

6. Sigui S un conjunt de clàusules amb $\square \notin S$. Demostra que S és satisfactible (donant un model per a S) en cadascuna de les següents situacions:

   c) Per tot símbol de predicat p es compleix que: o bé p apareix només en literals positius en S, o bé p apareix només en literals negatius en S.

És a dir, cada clàusula és de la forma p1 v … v pm v -q1 v … v -qn on

- les pi's només apareixen en literals positius en la resta de les clàusulas
- las qi's només apareixen en literals negatius en la resta de les clàusulas

Un model que va a satisfer totes les clàusulas de S és la I tal que

I(p)=0 per tot símbol p tal que p només apareix negatiu

I(p)=1 per tot símbol p tal que p només apareix positiu

Lògica en la Informàtica. Deducció en Lògica Proposicional

6. Sigui S un conjunt de clàusules amb $\square \notin S$. Demostra que S és satisfactible (donant un model per a S) en cadascuna de les següents situacions:

    c) Per tot símbol de predicat p es compleix que: o bé p apareix només en literals positius en S, o bé p apareix només en literals negatius en S.

Un model que va a satisfer totes les clàusulas de S és la I tal que

    I(p)=0 per tot símbol p tal que p només apareix negatiu

    I(p)=1 per tot símbol p tal que p només apareix positiu

Nota: en realitat aquesta I va a satisfer TOTS els literals de TOTES les clàusules (quan en realitat en tenia prou amb complir UN literal de cada clàusula).

Lògica en la Informàtica. Deducció en Lògica Proposicional

Pròxim dia: exercicis del 7 en endavant i Resolució.