

Lògica en la Informàtica

Definició de la Lògica Proposicional

Deducció en Lògica Proposicional

José Miguel Rivero Robert Nieuwenhuis

Dept. Ciències de la Computació
Facultat de Informàtica
Universitat Politècnica de Catalunya (UPC)

Tardor 2022



Temari:

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

26. (dificultat 2) Suposem que $|\mathcal{P}| = 100$ i que ens interessa determinar si una fórmula F construïda sobre \mathcal{P} és satisfactible o no. Si l'algorisme està basat en una anàlisi de la taula de veritat, i avaluar F en una interpretació I donada costa un microsegon (10^{-6} segons), quants anys trigarà?

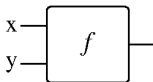
Més endavant veurem tècniques que moltes vegades funcionen millor.

Quantes interpretacions hi ha? hi ha 2^{100}

$$\begin{array}{rcl} 2^{10} & = & 1024 \approx 10^3 \\ 2^{100} & & \approx 10^{30} \end{array}$$

Avaluar-les totes trigarà $2^{100} \cdot 10^{-6}$ segons $\approx 10^{30} \cdot 10^{-6}$ segons
 $\approx 10^{24}$ segons $\approx 10^{24} / (365 \cdot 24 \cdot 3600)$ anys $\approx 4 \cdot 10^{16}$ anys aprox.

27. (dificultat 2) Una funció booleana de n entrades és una funció $f : \{0, 1\}^n \rightarrow \{0, 1\}$, 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^{2^n} : 2 elevat a (2 elevat a n): hi ha tantes funcions com tires de 2^n bits

Definició de Lògica Proposicional

27. (dificultat 2) Una funció booleana de n entrades és una funció $f : \{0, 1\}^n \rightarrow \{0, 1\}$, 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^{2^n} : 2 elevat a (2 elevat a n): hi ha tantes funcions com tirs de 2^n bits

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

| x | y | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | ... |
|---|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |

| x | y | ... | (9) | (10) | (11) | (12) | (13) | (14) | (15) | (16) |
|---|---|-----|-----|------|------|------|------|------|------|------|
| 0 | 0 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Definició de Lògica Proposicional

27. (dificultat 2) Una funció booleana de n entrades és una funció $f : \{0, 1\}^n \rightarrow \{0, 1\}$, 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^{2^n} : 2 elevat a (2 elevat a n): hi ha tantes funcions com tirs de 2^n bits

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

| x | y | 0 | and | $\neg(x \rightarrow y)$ | x | $\neg(y \rightarrow x)$ | y | xor | or | ... |
|-----|-----|-----|-----|-------------------------|----------|-------------------------|----------|-------------------|------|-----|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |
| x | y | ... | nor | = | $\neg y$ | $y \rightarrow x$ | $\neg x$ | $x \rightarrow y$ | nand | 1 |
| 0 | 0 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

27. (dificultat 2) Una funció booleana de n entrades és una funció $f : \{0, 1\}^n \rightarrow \{0, 1\}$, 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^{2^n} : 2 elevat a (2 elevat a n): hi ha tantes funcions com tirs de 2^n bits

Exemple $n = 3$: hi ha $2^8 = 256$

| x | y | z |
|-----|-----|-----|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| ... | | |
| 1 | 1 | 1 |

Si $n = 4$, hi ha $2^{16} \approx 65000$

28. (dificultat 2) Cada fórmula F representa una única funció booleana: la que retorna 1 exactament per a aquelles cadenes de bits I tals que $eval_I(F) = 1$. Per aixó, dues fórmules són lògicament equivalents si i només si representen la mateixa funció booleana. Quantes funcions booleanes (o quantes fórmules lògicament no-equivalents) hi ha en funció de $n = |\mathcal{P}|$?

Per exemple, la funció booleana “and” (de 2 entrades x, y) la podem representar mitjançant les fórmules

$$x \wedge y$$

$$(x \wedge y) \wedge y$$

$$\neg(\neg x \vee \neg y)$$

$$\neg(\neg x \vee \neg y) \wedge y$$

...

28. (dificultat 2) Cada fórmula F representa una única funció booleana: la que retorna 1 exactament per a aquelles cadenes de bits I tals que $eval_I(F) = 1$. Per aixó, dues fórmules són lògicament equivalents si i només si representen la mateixa funció booleana. Quantes funcions booleanes (o quantes fórmules lògicament no-equivalents) hi ha en funció de $n = |\mathcal{P}|$?

Hi ha 2^{2^n} [2 elevat a (2 elevat a n)]: hi ha tantes funcions com tires de 2^n bits

31. (dificultat 3) 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 connectives lògiques, reben algun nom? Ja sabem que podem expressar qualsevol funció booleana amb el conjunt de tres connectives $\{\wedge, \vee, \neg\}$, és a dir, qualsevol funció booleana és equivalent a una fórmula construïda sobre aquestes tres connectives. És cert això també para algun conjunt de només dues de les 16 funcions? (Hi ha diverses maneres, però basta amb donar una sola.)

Definició de Lògica Proposicional

31. (dificultat 3) 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 connectives lògiques, reben algun nom?

| x | y | 0 | and | $\neg(x \rightarrow y)$ | x | $\neg(y \rightarrow x)$ | y | xor | or | ... |
|---|---|---|-----|-------------------------|---|-------------------------|---|-----|----|-----|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |

| x | y | ... | nor | = | $\neg y$ | $y \rightarrow x$ | $\neg x$ | $x \rightarrow y$ | nand | 1 |
|---|---|-----|-----|---|----------|-------------------|----------|-------------------|------|---|
| 0 | 0 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

31. (dificultat 3) Escriu en una taula de veritat les 16 funcions booleanes de 2 entrades. Quantes de elles només depenen d'una de les dues entrades? Quantes depenen de zero entrades? Les altres, vistes com connectives lògiques, reben algun nom? Ja sabem que podem expressar qualsevol funció booleana amb el conjunt de tres connectives $\{\wedge, \vee, \neg\}$, és a dir, qualsevol funció booleana és equivalent a una fórmula construïda sobre aquestes tres connectives. És cert això també para algun conjunt 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: $x \wedge y \equiv \neg(\neg x \vee \neg y)$

32. (dificultat 3) Demuestra que qualsevol funció booleana de dues entrades es pot expressar amb només *nor* o bé amb només *nand*, on $nor(F, G)$ és $\neg(F \vee G)$, i $nand(F, G)$ és $\neg(F \wedge G)$.

Ho fem per *nand*:

$$\neg F \quad \equiv \quad F \text{ nand } F$$

$$F \vee G \quad \equiv \quad \neg(\neg F \wedge \neg G) \quad \equiv \quad \neg F \text{ nand } \neg G \quad \equiv \\ (F \text{ nand } F) \text{ nand } (G \text{ nand } G)$$

$$F \wedge G \quad \equiv \quad \neg(F \text{ nand } G) \quad \equiv \quad (F \text{ nand } G) \text{ nand } (F \text{ nand } G)$$

37. (dificultat 3) 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 substituint els valors de retorn per un sol valor de retorn que sigui una expressió booleana en $i > 0$, a i b :

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

37. (dificultat 3) 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)
```

$\text{if_then_else}(C, F, G) \equiv (C \rightarrow F) \wedge (\neg C \rightarrow G)$

37. (dificultat 3) 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 (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) |

33. (dificultat 3) 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 contradictòries?
- b) Assumint que tots son innocents, qui o quins van mentir en la declaració?
- c) Assumint que ningú va mentir, qui és innocent i qui és culpable?

33. (dificultat 3) 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: $b \wedge \neg c$
- B diu: $a \rightarrow c$
- C diu: $\neg c \wedge (a \vee b)$

33. (dificultat 3) 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: $b \wedge \neg c$
- B diu: $a \rightarrow c$
- C diu: $\neg c \wedge (a \vee b)$

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 model I : $I(a) = 0$, $I(b) = 1$, $I(c) = 0$.

Per tant, **no són contradictòries**.

33. (dificultat 3) 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: $b \wedge \neg c$
- B diu: $a \rightarrow c$
- C diu: $\neg c \wedge (a \vee b)$

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

Si tots són innocents, A i C van mentir.

33. (dificultat 3) 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: $b \wedge \neg c$
- B diu: $a \rightarrow c$
- C diu: $\neg c \wedge (a \vee b)$

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

Si ningú ha mentit, llavors B és culpable.

34. (dificultat 2) 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 binària: 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).

Podem fer-ho així, de manera raonable, suposant que \perp en la nostra aplicació modela “no ho sé”:

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

Definició de Lògica Proposicional

34. (dificultat 2) 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 binària: 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).

| | | | | | | | | |
|--------------------|------|---------|------------------------|---------|------|----------------------|---------|---------|
| | | | $eval_I(F \wedge G) =$ | | | $eval_I(F \vee G) =$ | | |
| | | | 0 | 0 | dona | 0 | 0 | 0 |
| | | | 0 | 1 | | 0 | 1 | 1 |
| $eval_I(\neg F) =$ | | | 0 | \perp | | 0 | \perp | \perp |
| 0 | dona | 0 | 1 | 0 | | 0 | 1 | 1 |
| 1 | | 1 | 1 | 1 | | 1 | 1 | 1 |
| \perp | | \perp | 1 | \perp | | 1 | \perp | 1 |
| | | | \perp | 0 | | \perp | 0 | \perp |
| | | | \perp | 1 | | \perp | 1 | 1 |
| | | | \perp | \perp | | \perp | \perp | \perp |

34. (dificultat 2) 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 binària: 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(...) ) {  
}
```


Definició de Lògica Proposicional

34. (dificultat 2) Inventa i defineix formalment alguna altre lògica diferent a la lògica proposicional. Per exemple, si les interpretacions són funcions $I : \mathcal{P} \rightarrow \{0, 1, \perp\}$ que també poden donar “indefinit” \perp , es pot adaptar la noció de satisfacció de manera raonable, tot i que la resposta ja no serà binària: 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).

| | | | | | | | | |
|--------------------|------|---------|------------------------|---------|------|----------------------|---------|---------|
| | | | $eval_I(F \wedge G) =$ | | | $eval_I(F \vee G) =$ | | |
| | | | 0 | 0 | dona | 0 | 0 | 0 |
| | | | 0 | 1 | | 0 | 0 | 1 |
| $eval_I(\neg F) =$ | | | 0 | \perp | | 0 | \perp | \perp |
| 0 | dona | 0 | 1 | 0 | | 0 | 1 | 1 |
| 1 | | 1 | 1 | 1 | | 1 | 1 | 1 |
| \perp | | \perp | 1 | \perp | | 1 | \perp | 1 |
| | | | \perp | 0 | | \perp | 0 | \perp |
| | | | \perp | 1 | | \perp | 1 | \perp |
| | | | \perp | \perp | | \perp | \perp | \perp |

Definició de Lògica Proposicional

35. (dificultat 2) Com l'exercici anterior, però considerant $I : \mathcal{P} \rightarrow [0 \dots 1]$, 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 (remotament) 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?

$$eval_I(\neg F) = 1 - eval_I(F)$$


$$eval_I(F \wedge G) = eval_I(F) \cdot eval_I(G)$$

$$eval_I(F \vee G) = (eval_I(F) + eval_I(G)) - (eval_I(F) \cdot eval_I(G))$$

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.

Temari:

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

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

1. Formes normals i clàusules

- Fórmules com a conjunts
- Literals
- CNF i DNF
- Clàusula
- Conjunt de clàusules
- Clàusula buida
- Clàusula de Horn

Exercicis del tema 3

- Exercicis 2,3
- Exercici 3
- Exercici 4

5. (dificultat 2) Demostra que una clàusula és una tautologia si, i només si, conté alhora p i $\neg p$ per un cert símbol proposicional p .

Demostrem que una clàusula de la forma

$p_1 \vee \dots \vee p_m \vee \neg q_1 \vee \dots \vee \neg q_n$ NO és tautologia ssi NO conté alhora p i $\neg p$.

$p_1 \vee \dots \vee p_m \vee \neg q_1 \vee \dots \vee \neg q_n$ NO és tautologia ssi

$\exists I$, tal que I no és model de $p_1 \vee \dots \vee p_m \vee \neg q_1 \vee \dots \vee \neg q_n$ ssi

$\exists I$, tal que $I \not\models p_1 \vee \dots \vee p_m \vee \neg q_1 \vee \dots \vee \neg q_n$ ssi

$\exists I$, tal que $\max(\text{eval}_I(p_1), \dots, \text{eval}_I(p_m), \text{eval}_I(\neg q_1), \dots, \text{eval}_I(\neg q_n)) = 0$ ssi

$\exists I$, tal que $I(p_i) = 0$ per a totes les p_i i $I(q_j) = 1$ per a totes les q_j ssi

$p_i \neq q_j$ per a tota i, j

6. (dificultat 2) Sigui S un conjunt de clàusules amb $\square \notin S$.
Demuestra que S és satisfactible (donant un model per a S) en cadascuna de les següents situacions:

- a) Tota clàusula C de S té algún literal positiu.
- b) Tota clàusula C 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 .

6. (dificultat 2) 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 C de S té algún literal positiu.

C és de la forma: $p_1 \vee \dots \vee p_m \vee \neg q_1 \vee \dots \vee \neg q_n$ ($m > 1, n \geq 0$)

Sigui $S = \{C_1, C_2, \dots\}$. Cada clàusula C_i 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 .

6. (dificultat 2) Sigui S un conjunt de clàusules amb $\square \notin S$.
Demuestra que S és satisfactible (donant un model per a S) en cadascuna de les següents situacions:

b) Tota clàusula C de S té algún literal negatiu.

Sigui $S = \{C_1, C_2, \dots\}$. Cada clàusula C_i 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 .

6. (dificultat 2) Sigui S un conjunt de clàusules amb $\square \notin S$.
Demuestra 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 $p_1 \vee \dots \vee p_m \vee \neg q_1 \vee \dots \vee \neg q_n$ on

- les p_i 's només apareixen en literals positius en la resta de les clàusulas
- les q_i 's només apareixen en literals negatius en la resta de les clàusulas

Un model que satisfarà totes les clàusulas de S és la I tal que

$$I(p) = 0 \text{ per tot símbol } p \text{ tal que } p \text{ només apareix negatiu}$$
$$I(p) = 1 \text{ per tot símbol } p \text{ tal que } p \text{ només apareix positiu}$$


6. (dificultat 2) Sigui S un conjunt de clàusules amb $\square \notin S$.
Demuestra 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 satisfarà totes les clàusules 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 satisfarà TOTS els literals de TOTES les clàusules (quan en realitat en tenia prou amb complir UN literal de cada clàusula).

Continguts del capítol p3.pdf per al proper dia:

- Exercicis del 7 en endavant
-  Resolució (pàg. 5)