

Lògica en la Informàtica

Deducció en Lògica Proposicional

José Miguel Rivero

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

Primavera 2023




Crèdits

El material utilitzat en aquesta presentació ha estat extret del elaborat pel professor Robert Nieuwenhuis (Dept. CS, UPC) per l'assignatura *Lògica en la Informàtica* de la FIB.


En particular, del llibre *Lógica para informáticos* - Farré, R. [et al.], Marcombo, 2011. ISBN: 9788426716941.



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)

Deducció en Lògica Proposicional

Racó: Col·lecció d'apunts bàsics de lògica.  p3.pf


- Formes normals i clàusules
- Resolució. Correcció i completesa
- Nocions informals de decidibilitat i complexitat
- Resoldre problemes pràctics amb la lògica proposicional

- 1 Transformació de fórmules qualssevol a CNF
- 2 Transformació via distributivitat
- 3 Transformació via Tseitin
 - Descripció
 - Connectiva NOT
 - Conclusions
 - Connectives AND/OR niuades
- 4 Vídeos: *The Transportation Company*, i Restriccions numèriques
- 5 Exercici: *The Transportation Company*
- 6 Codificació de restriccions numèriques en SAT

Deducció en Lògica Proposicional

Necessitem decidir SAT per a fórmules qualssevol, però els SAT solvers només treballen amb CNFs (conjunts de clàusules).

Per tant, necessitem poder transformar fórmules qualssevol en CNFs.

Tseitin. Veure la presentació (en la web de Lògica en la Informàtica  <https://www.cs.upc.edu/~li>) sobre la transformació de Tseitin d'una fórmula qualsevol a una CNF **equisatisfactible**.

ATENCIÓ: ERRATA

Hi ha un error en la presentació de la web de LI sobre Tseitin: on diu $e6 \Leftrightarrow q \vee \neg e7$, ha de dir $e6 \Leftrightarrow q \vee e7$.

Transformació a CNF via distributivitat

1. Aplica les tres regles de transformació **mentre sigui possible**:

- $\neg\neg F \Rightarrow F$
- $\neg(F \wedge G) \Rightarrow \neg F \vee \neg G$
- $\neg(F \vee G) \Rightarrow \neg F \wedge \neg G$

Després d'això, la fórmula està en **Negation Normal Form (NNF)**

2. Ara aplica la regla de **distributivitat mentre sigui possible**:

- $F \vee (G \wedge H) \Rightarrow (F \vee G) \wedge (F \vee H)$

EXAMPLE: sigui F la fórmula $(p \wedge q) \vee \neg(\neg p \wedge (q \vee \neg r))$

1. $(p \wedge q) \vee \neg(\neg p \wedge (q \vee \neg r)) \Rightarrow (p \wedge q) \vee (\neg\neg p \vee \neg(q \vee \neg r)) \Rightarrow (p \wedge q) \vee (p \vee (\neg q \wedge \neg\neg r)) \Rightarrow (p \wedge q) \vee (p \vee (\neg q \wedge r))$
2. $(p \wedge q) \vee (p \vee (\neg q \wedge r)) \Rightarrow (p \vee p \vee (\neg q \wedge r)) \wedge (q \vee p \vee (\neg q \wedge r)) \Rightarrow (p \vee p \vee \neg q) \wedge (p \vee p \vee r) \wedge (q \vee p \vee \neg q) \wedge (q \vee p \vee r) \Rightarrow (p \vee \neg q) \wedge (p \vee r) \wedge (q \vee p \vee r)$



Deducció en Lògica Proposicional

Per què la transformació via distributivitat pot fer créixer exponencialment la fórmula?

Perquè la regla de distributivitat

$F \vee (G \wedge H) \Rightarrow (F \vee G) \wedge (F \vee H)$ DUPLICA la subfórmula F .

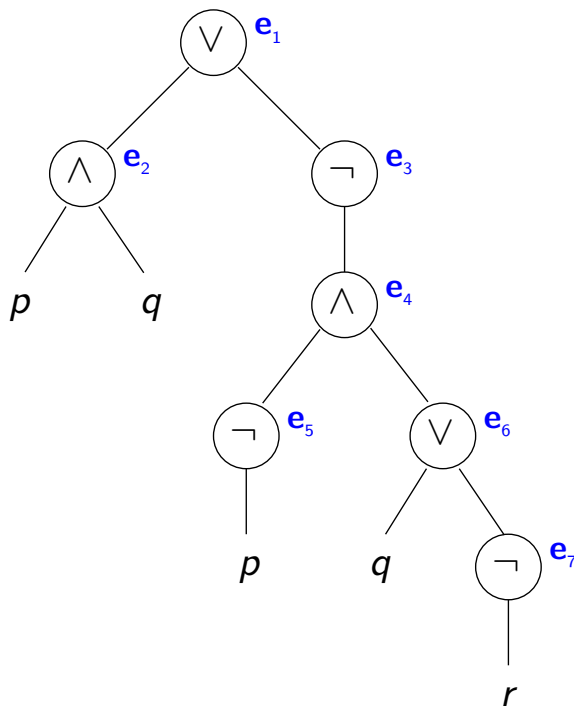
Exemple de cas pitjor: si F és una DNF $Cub_1 \vee \dots \vee Cub_n$, on cada cub és un AND de k literals, la CNF tindrà TOTES les clàusules possibles amb un literal de cada cub, és a dir k^n clàusules (el literal del primer cub es pot triar de k maneres, el del segon també, etc.).

Exemple: $(p \wedge q \wedge r) \vee (p' \wedge q' \wedge r')$ donaria:

$$\begin{array}{lll} p \vee p', & p \vee q', & p \vee r', \\ q \vee p', & q \vee q', & q \vee r', \\ r \vee p', & r \vee q', & r \vee r' \end{array}$$



Sigui F la fórmula $(p \wedge q) \vee \neg(\neg p \wedge (q \vee \neg r))$



- e_1
- $e_1 \leftrightarrow e_2 \vee e_3$
 - $\neg e_1 \vee e_2 \vee e_3$
 - $\neg e_2 \vee e_1$
 - $\neg e_3 \vee e_1$
- $e_2 \leftrightarrow p \wedge q$
 - $\neg p \vee \neg q \vee e_2$
 - $\neg e_2 \vee p$
 - $\neg e_2 \vee q$
- $e_3 \leftrightarrow \neg e_4$
 - $\neg e_3 \vee \neg e_4$
 - $e_3 \vee e_4$
- $e_4 \leftrightarrow e_5 \wedge e_6$
- $e_5 \leftrightarrow \neg p$
- $e_6 \leftrightarrow q \vee e_7$
- $e_7 \leftrightarrow \neg r$

Dedució en Lògica Proposicional

Per això fem TSEITIN:

Introduïm un símbol nou per cada connectiva de la fórmula.
I generem les clàusules que "defineixen" el paper que juguen aquests símbols nous en la fórmula.

Per exemple, per a expressar que p és el símbol d'un node OR de dos fills amb símbols a , b , necessitem $p \leftrightarrow a \vee b$.

Per a això:

- expressem $p \rightarrow a \vee b$ mitjançant una clàusula de tres literals:
 $\neg p \vee a \vee b$
- expressem $p \leftarrow a \vee b$ que és $a \rightarrow p$ i $b \rightarrow p$, amb dues clàusules: $\neg a \vee p$ i $\neg b \vee p$

Per això fem TSEITIN:

Introduïm un símbol nou per cada connectiva de la fórmula.
I generem les clàusules que "defineixen" el paper que juguen aquests símbols nous en la fórmula.

Per a expressar que p és el símbol d'un node AND de dos fills amb símbols a , b , necessitem $p \leftrightarrow a \wedge b$.

Per a això:

- expressem $p \rightarrow a \wedge b$ que és $p \rightarrow a$ i $p \rightarrow b$, amb dues clàusules: $\neg p \vee a$ i $\neg p \vee b$
- expressem $p \leftarrow a \wedge b$ mitjançant una clàusula de tres literals: $\neg a \vee \neg b \vee p$.

En la presentació de la web de LI s'introdueixen també símbols i clàusules per als nodes NOT, però això no és necessari.

Per exemple, per a evitar el primer node NOT i el seu símbol $e3$, podem expressar directament que $e1 \leftrightarrow e2 \vee \neg e4$, generant les clàusules:

$$\begin{aligned} &e1 \vee e2 \vee \neg e4, \\ &\neg e2 \vee e1, \\ &e4 \vee e1. \end{aligned}$$

Quins resultats obtenim?

Sigui F una fórmula.

Sigui $\text{Tseitin}(F)$ la CNF de (el conjunt de les clàusules generades per) la transformació de Tseitin de F .

Llavors:

1. $\text{Tseitin}(F)$ té clàusules de fins a 3 literals.
Compte!: hi ha una clàusula unitària (d'1 només literal) que és el símbol auxiliar de l'arrel (e_1 en l'exemple).
2. F i $\text{Tseitin}(F)$ són EQUISATISFACTIBLES: F és satisfactible SSI $\text{Tseitin}(F)$ és satisfactible
3. F i $\text{Tseitin}(F)$ NO són logicamente equivalents
4. La mida de $\text{Tseitin}(F)$ és lineal en la mida de F (3 clàusules per cada connectiva AND o OR de F) + l'arrel
5. Podem obtenir $\text{Tseitin}(F)$ en temps lineal a partir de F
6. Podem reconstruir fàcilment un model de F a partir d'un model de $\text{Tseitin}(F)$ ("oblidant-nos" dels símbols auxiliars)

Deducció en Lògica Proposicional

1. $\text{Tseitin}(F)$ té clàusules de fins a 3 literals.
Compte!: hi ha una clàusula unitària (d'1 només literal) que és el símbol auxiliar de l'arrel (e_1 en l'exemple).
2. F i $\text{Tseitin}(F)$ són EQUISATISFACTIBLES: F és satisfactible SSI $\text{Tseitin}(F)$ és satisfactible
3. F i $\text{Tseitin}(F)$ NO són logicamente equivalents
4. La mida de $\text{Tseitin}(F)$ és lineal en la mida de F (3 clàusules per cada connectiva AND o OR de F) + l'arrel
5. Podem obtenir $\text{Tseitin}(F)$ en temps lineal a partir de F
6. Podem reconstruir fàcilment un model de F a partir d'un model de $\text{Tseitin}(F)$ ("oblidant-nos" dels símbols auxiliars)

Nota:

Sabent que SAT per a fórmules F qualssevol és NP-complet, els punts 1,2,5 impliquen que 3-SAT també és NP-complet.

Nota:

Si tenim una subfórmula amb ORs (o ANDs) niats, com a $p \vee (q \vee r)$ podem fer Tseitin com sempre, introduint per cada OR binari un símbol auxiliar i tres clàusules. Però també podem considerar que és una OR de tres entrades $p \vee q \vee r$, i generar un sol símbol auxiliar i quatre clàusules per a expressar $a \Leftrightarrow p \vee q \vee r$:

$$\neg a \vee p \vee q \vee r$$

$$\neg p \vee a$$

$$\neg q \vee a$$

$$\neg r \vee a$$

Això pot fer-se similarment per a ORs i ANDs de qualsevol nombre d'entrades.

Deducció en Lògica Proposicional

Ara: veure els vídeos de la web de LI sobre:

👉 *the Transportation Company*

👉 Codificació de restriccions numèriques en SAT

- ALO, AMO, exactly one
- cardinality constraints en general:

$$l_1 + \dots + l_n \leq K$$

$$l_1 + \dots + l_n \geq K$$

$$l_1 + \dots + l_n = K$$

- pseudo-Boolean constraints:

$$a_1 l_1 + \dots + a_n l_n \leq K$$

$$a_1 l_1 + \dots + a_n l_n \geq K$$

$$a_1 l_1 + \dots + a_n l_n = K$$

The Transportation Company

We need to plan the activities of a transportation company during a period of H hours. The company has T trucks, D drivers and there are N transportation tasks to be done, each one of which lasts one hour and needs one driver per truck.

Each task $i \in 1 \dots N$ needs K_i trucks, and has a list $L_i \subseteq \{1 \dots H\}$ of hours at which this task i can take place. For example, if $L_7 = \{3, 4, 8\}$ this means that task 7 can take place at hour 3, at hour 4 or at hour 8. For each driver $d \in 1 \dots D$ there is a list of blockings $B_d \subseteq \{1 \dots H\}$ of hours at which driver D can *not* work.

The Transportation Company

Explain how to use a SAT solver for planning this: for each task, when does it take place, and using which drivers. Clearly indicate which types of propositional variables you are using, and how many of each type, using the following format:

variables $t_{i,h}$ meaning "task i takes place at hour h "

for all tasks $i \in 1 \dots N$ and for all hours $h \in 1 \dots H$

Total: $N \cdot H$ variables.

Since H , D and N may be large, it is not allowed to use $O(H \cdot D \cdot N)$ variables (but using such a large number of clauses is fine).

Hint: you may use several types of variables, for example one type with $N \cdot H$ variables and another one with $N \cdot D$.

Also clearly indicate which clauses you need, and how many of each type, and how many literals each type of clause has. If you use any AMO, cardinality or pseudo-Boolean constraints, it is not necessary to convert these into CNF.

The Transportation Company

- H hours
- T trucks
- D drivers
- N tasks 1 hour 1 driver/truck
- each task $i \in 1 \dots N$
 - needs K_i trucks
 - $L_i \subseteq \{1 \dots H\}$ ok hours
- each driver $d \in 1 \dots D$
 - has list of blockings

The Transportation Company

Solution:

- Variables $t_{i,h}$ meaning “task i takes place at hour h ”
for all tasks $i \in 1 \dots N$ and for all hours $h \in 1 \dots H$
Total: $N \cdot H$ variables.
- Variables $dr_{i,d}$ meaning “task i has driver d among its drivers”
for all tasks $i \in 1 \dots N$ and for all drivers $d \in 1 \dots D$
Total: $N \cdot D$ variables.

- a) Clauses to express that each task i is at exactly one hour from its list L_i :
for each i , if $L_i = \{h_1, \dots, h_k\}$, one cardinality constraint $t_{i,h_1} + \dots + t_{i,h_k} = 1$ (or one AMO+ALO).
- b) Clauses to express that no driver d works on two different tasks i and j at the same hour h :
for each driver d and hour h and for each pair of tasks i, j such that $h \in L_i$ and $h \in L_j$,
a four-literal clause $\neg dr_{i,d} \vee \neg dr_{j,d} \vee \neg t_{i,h} \vee \neg t_{j,h}$.
- c) Clauses to express that each task i needs K_i drivers:
one cardinality constraint $dr_{i,1} + \dots + dr_{i,D} = K_i$ per task (here \geq would suffice instead of $=$).

- d) Clauses to express that at no time point we are using more than T trucks:
for each hour h , a pseudo-Boolean constraint $K_1 t_{1,h} + \dots + K_N t_{N,h} \leq T$.
- e) Clauses to express that a driver d cannot work at hour h :
for each task i , a two-literal clause $\neg t_{i,h} \vee \neg dr_{i,d}$
i.e., for each such a blocking of a driver d at hour h there are N two-literal clauses.

- Extend your solution to take into account that no driver can work more than 50 hours in total during the whole period of H hours. Hint: this may require another type of variables.

- **Solution:**

Clauses to express that no driver works more than 50 hours:
for each driver d , one cardinality constraint

$$dr_{i,d} + \dots + dr_{N,d} \leq 50$$

Codificació de restriccions numèriques en SAT

- 👉 Vídeos de la web de LI sobre:
Codificació de restriccions numèriques en SAT
- 👉 Apunts de la web de LI:
Breu resum en escrit sobre això

- ALO, AMO, exactly one
- Cardinality constraints en general:

$$l_1 + \dots + l_n \leq K$$

$$l_1 + \dots + l_n \geq K$$

$$l_1 + \dots + l_n = K$$

- Pseudo-Boolean constraints:

$$a_1 l_1 + \dots + a_n l_n \leq K$$

$$a_1 l_1 + \dots + a_n l_n \geq K$$

$$a_1 l_1 + \dots + a_n l_n = K$$

Codificació de restriccions numèriques en SAT

- ▶ ALO (at least one), AMO (at most one), exactly one

Per exemple, exactly 1 of $\{l_1, l_2, \dots, l_9\}$ és equivalent a:

- at least 1 of $\{l_1, l_2, \dots, l_9\}$ que pot ser codificat amb una sola clàusula: $l_1 \vee l_2 \vee \dots \vee l_9$
- at most 1 of $\{l_1, l_2, \dots, l_9\}$ per $\binom{9}{2} = \frac{9 \cdot 8}{2} = 36$ clàusules binaries: $\neg l_1 \vee \neg l_2, \neg l_1 \vee \neg l_3, \dots, \neg l_8 \vee \neg l_9$.

- ▶ ALO (at least one), AMO (at most one), exactly one

En lloc de $\text{AMO}(l_1, \dots, l_n)$ escriurem: $l_1 + \dots + l_n \leq 1$.

Nota: aquí l_1, \dots, l_n poden ser literals positius o negatius (variables negades).

Codificació	num vars auxiliars	num clausulas
Quadràtica	0	$\binom{n}{2} = n(n-1)/2$
Ladder	n	$3n$
Heule 3	$n/2$	$3n$
Heule 4	$n/3$	$3.3n$
Log	$\log n$	$n \log n$

► Cardinality constraints en general:

$$l_1 + \dots + l_n \leq K$$

$$l_1 + \dots + l_n \geq K$$

$$l_1 + \dots + l_n = K$$

🔗 Pràctica 3: miSudoku.pl

```
exactly(K, Lits) :- atLeast(K, Lits), atMost(K, Lits), !.
```

```
atMost(K, Lits) :- % l1+...+ln <= k: in all subsets of size k+1,
                  % at least one is false:
                  negateAll(Lits, NLits),
                  K1 is K+1, subsetOfSize(K1, NLits, Clause), writeClause(Clause), fail.
atMost(_, _).
```

```
atLeast(K, Lits) :- % l1+...+ln >= k: in all subsets of size n-k+1,
                  % at least one is true:
                  length(Lits, N),
                  K1 is N-K+1, subsetOfSize(K1, Lits, Clause), writeClause(Clause), fail.
atLeast(_, _).
```

► Cardinality constraints en general:

$$l_1 + \dots + l_n \leq K$$

$$l_1 + \dots + l_n \geq K$$

$$l_1 + \dots + l_n = K$$

🔗 Pràctica 3: miSudoku.pl

```
atMost(K, Lits) :- % l1+...+ln <= k: in all subsets of size k+1,
                  % at least one is false:
                  negateAll(Lits, NLits),
                  K1 is K+1, subsetOfSize(K1, NLits, Clause), writeClause(Clause), fail.
atMost(_, _).
```

Per exemple: el constraint `atMost(2, [x,y,z,u])` que representa $x+y+z+u \leq 2$ genera 4 clàusules

($K1 = K+1 = 2+1 = 3$, $i \binom{4}{3} = 4$):

```
-x v -y v -z
-x v -y      v -u
-x      v -z v -u
      -y v -z v -u
```

Codificació de restriccions numèriques en SAT

► Cardinality constraints en general:

$$l_1 + \dots + l_n \leq K$$

$$l_1 + \dots + l_n \geq K$$

$$l_1 + \dots + l_n = K$$

🔗 Pràctica 3: miSudoku.pl

```

atLeast(K, Lits) :- % l1+...+ln >= k: in all subsets of size n-k+1,
                    %                               at least one is true:
                    length(Lits, N),
                    K1 is N-K+1, subsetOfSize(K1, Lits, Clause), writeClause(Clause), fail
atLeast(_, _).

```

Per exemple: el constraint `atLeast(3, [x,y,z,u])` que representa $x+y+z+u \geq 3$ genera 6 clàusules

($K1 = N-K+1 = 4-3+1 = 2$, i $\binom{4}{2} = 6$):

```

x v y
x      v z
x          v u
      y v z
      y      v u
          z v u

```



Codificació de restriccions numèriques en SAT

► Cardinality constraints en general:

$$l_1 + \dots + l_n \leq K$$

$$l_1 + \dots + l_n \geq K$$

$$l_1 + \dots + l_n = K$$

Exemples sense variables auxiliars:

$$N = 10, K = 5, \binom{10}{5} = 252$$

$$N = 20, K = 10, \binom{20}{10} = 184756$$

$$N = 30, K = 15, \binom{30}{15} = 155117520$$

Codificacions més compactes fent servir variables auxiliars:

Codificació	num vars auxiliars	num clausulas
Polinòmic	0	$\binom{n}{k}$
Ladder encoding	n	$3n$
Sorting networks	$n \log^2 n$	$n \log^2 n$
Cardinality networks	$n \log^2 k$	$n \log^2 k$



► Pseudo-Boolean constraints:

$$a_1 l_1 + \dots + a_n l_n \leq K$$

$$a_1 l_1 + \dots + a_n l_n \geq K$$

$$a_1 l_1 + \dots + a_n l_n = K$$

Es codifiquen en SAT fent servir BDD's, i definint una nova variable auxiliar i quatre clàusules per cada node del BDD.

Deducció en Lògica Proposicional

Per al proper dia de classe:

- **Lògica de Primer Ordre (LPO)**
Capítol 4 dels apunts: fitxer p4.pdf
- Exercicis del tema 4 : 5, 6, ...