Lògica en la Informàtica

Deducció en Lògica Proposicional

José Miguel Rivero   Robert Nieuwenhuis

Facultad de Informàtica
Universitat Politècnica de Catalunya (UPC)

Tardor 2022

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

- Formes normals i clàusules
- Nocions informals de decidibilitat i complexitat
- Resolució. Correcció i completesa
- Resoldre problemes pràctics amb la 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/~rivero/Teaching/LI`) sobre la transformació de Tseitin d'una fórmula qualsevol a una CNF equisatisfactible.

Quick Introduction to Propositional Logic

Albert Oliveras and Enric Rodríguez-Carbonell

Logic and Algebra in Computer Science

Session 1

Fall 2009, Barcelona

Departament de Llenguatges i Sistemes Informàtics

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Tranformation to CNF via distributivity

1. Apply the three transformation rules up to completion:
   - $\neg\neg F \Rightarrow F$
   - $\neg(F \wedge G) \Rightarrow \neg F \vee \neg G$
   - $\neg(F \vee G) \Rightarrow \neg F \wedge \neg G$

   After that, the formula is in Negation Normal Form (NNF)
2. Now apply the distributivity rule up to completion:
   - $F \vee (G \wedge H) \Rightarrow (F \vee G) \wedge (F \vee H)$

EXAMPLE: let F be $(p \wedge q) \vee \neg(\ \neg p \wedge (q \vee \neg r)\ )$

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

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

Perquè la regla de distributivitat

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

Exemple de cas pitjor: si F és una DNF $Cub_1 \vee \cdots \vee Cub_n$, 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:

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

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: $\neg a \vee p \quad \neg b \vee p$

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.

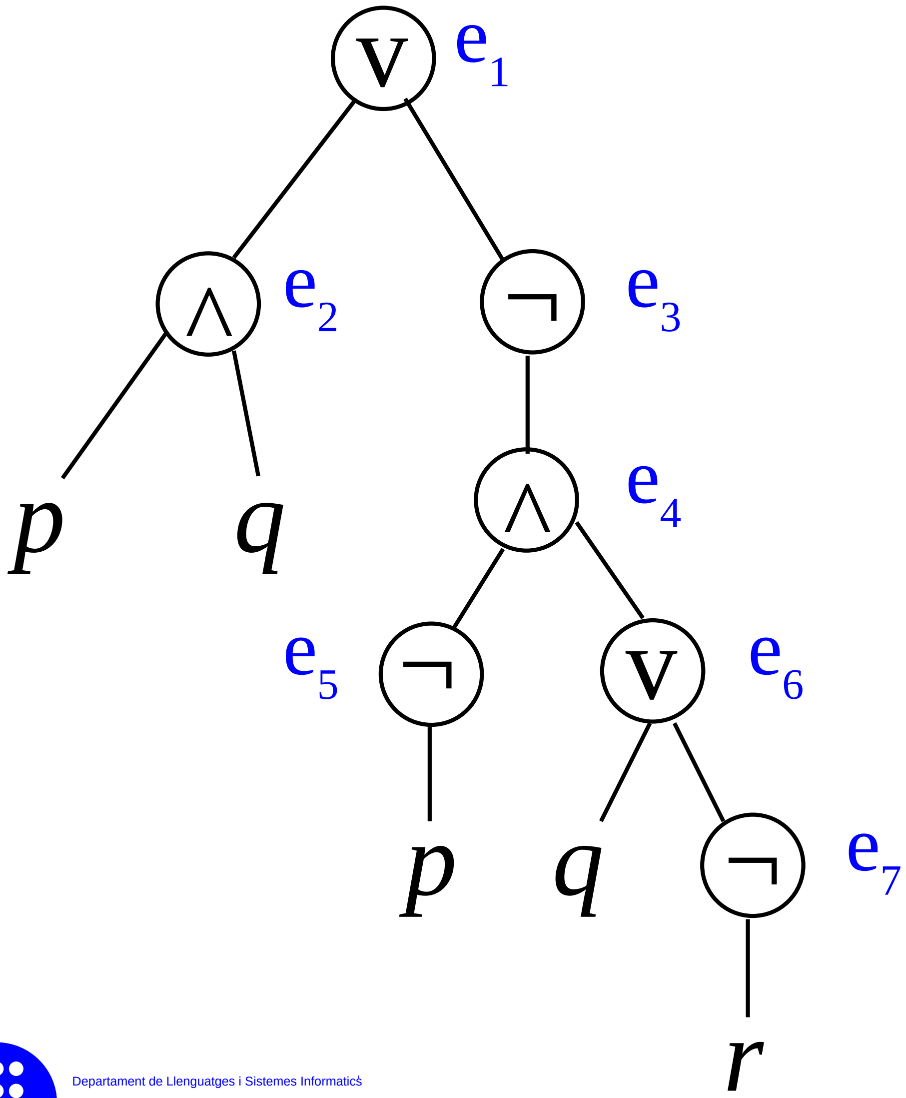
Per a expressar que p és el símbol d'un node AND de dos fills amb símbols a, b, necessitem $p \Leftrightarrow a \wedge b$.

Per a això:

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

Tranformation to CNF via Tseitin

Let F be $(p \wedge q) \vee \neg(\ \neg p \wedge (q \vee \neg r)\ )$



- e_1
- $e_1 \leftrightarrow e_2 \vee e_3$

$$\begin{array}{lllll} \neg e_1 & \vee & e_2 & \vee & e_3 \\ \neg e_2 & \vee & e_1 & & \\ \neg e_3 & \vee & e_1 & & \end{array}$$

- $e_2 \leftrightarrow p \wedge q$

$$\begin{array}{lllll} \neg p & \vee & \neg q & \vee & e_2 \\ \neg e_2 & \vee & p & & \\ \neg e_2 & \vee & q & & \end{array}$$

- $e_3 \leftrightarrow \neg e_4$

$$\begin{array}{lll} \neg e_3 & \vee & \neg e_4 \\ e_3 & \vee & e_4 \end{array}$$

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

Departament de Llenguatges i Sistemes Informàtics

UNIVERSITAT POLITÈCNICA DE CATALUNYA

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:

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

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 $e6 \Leftrightarrow q \vee e7$.

Quins resultats obtenim?

Sigui F una fórmula.
Sigui Tseitin(F) la CNF de (el conjunt de les clàusules generades per) la transformació de Tseitin de F.

Llavors:

1. 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 ($e1$ en l'exemple).
2. F i Tseitin(F) són EQUISATISFACTIBLES: F és satisfactible SSI Tseitin(F) és satisfactible
3. F i Tseitin(F) NO són logicamente equivalents
4. La mida de Tseitin(F) és lineal en la mida de F (3 clàusules per cada connectiva AND o OR de F) + l'arrell
5. Podem obtenir Tseitin(F) en temps lineal a partir de F
6. Podem reconstruir fàcilment un model de F a partir d'un model de Tseitin(F) ("oblidant-nos" dels símbols auxiliars)

1. 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 ($e1$ en l'exemple).
2. F i Tseitin(F) són EQUISATISFACTIBLES: F és satisfactible SSI Tseitin(F) és satisfactible
3. F i Tseitin(F) NO són logicamente equivalents
4. La mida de Tseitin(F) és lineal en la mida de F (3 clàusules per cada connectiva AND o OR de F) + l'arrell
5. Podem obtenir Tseitin(F) en temps lineal a partir de F
6. Podem reconstruir fàcilment un model de F a partir d'un model de 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.

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:

    $$l_1 + \cdots + l_n \leq K$$
    $$l_1 + \cdots + l_n \geq K$$
    $$l_1 + \cdots + l_n = K$$

  - pseudo-Boolean constraints:

    $$a_1 l_1 + \cdots + a_n l_n \leq K$$
    $$a_1 l_1 + \cdots + a_n l_n \geq K$$
    $$a_1 l_1 + \cdots + a_n l_n = K$$

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 $i \in 1 \ldots N$ needs K_i trucks, and has a list $L_i \subseteq \{1 \ldots H\}$ 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 $d \in 1 \ldots D$ there is a list of blockings $B_d \subseteq \{1 \ldots H\}$ of hours at which driver D can *not* work.

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 many 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.