Lògica en la Informàtica

Deducció en Lògica Proposicional

José Miguel Rivero Robert Nieuwenhuis

Facultad de Informàtica
Universitat Politècnica de Catalunya (UPC)

Tardor 2022

18. (dificultat 2) La resolució és completa? Demostra-ho.

Completa: qualsevol conseqüència lògica es pot arribar a obtenir mitjançant resolució.

No. Contraexemple: Sigui S el conjunt buit de clàusules.
Llavors $S \models p \vee \neg p$. (perquè $p \vee \neg p$ és una tautologia).

Però NO podem obtenir $p \vee \neg p$ a partir de S mitjançant resolució.
Per tant NO podem obtenir qualsevol conseqüència lògica mitjançant resolució.

Per tant la resolució **NO** és completa.

18. (dificultat 2) La resolució és completa? Demostra-ho.

Completa: qualsevol conseqüència lògica es pot arribar a obtenir mitjançant resolució.

Un altre contraexemple:
Sigui S qualsevol conjunt de clàusules que conté la clàusula buida.
Llavors S és insatisfactible.
I per tant tenim $S \models p$, on p és un símbol que no apareix en S.

Però NO podem obtenir p a partir de S mitjançant resolució.

18. (dificultat 2) La resolució és completa? Demostra-ho.

Completa: qualsevol conseqüència lògica es pot arribar a obtenir mitjançant resolució.

Un altre contraexemple: $S = \{p, q\}$

$S \models p \vee q$ però NO podem obtenir $p \vee q$ per resolució a partir del conjunt de clàusules $\{p, q\}$.

19. (dificultat 2) Sigui S un conjunt de clàusules insatisfactible. Per la completitud refutacional de la resolució, sabem que existeix una demostració per resolució de que $\square \in Res(S)$. És aquesta demostració única?

Un petit incís:

Hi ha un teorema que diu: S és insatisfactible **SSI** $\square$ està en $Res(S)$

$\Rightarrow$ "completitud refutacional" (és un cas particular de completitud)

$\Leftarrow$ pq $\square$ està en $Res(S) \Rightarrow Res(S)$ insat $\Rightarrow$ S insat

(l'última $\Rightarrow$ ve de l'exercici 17:
$Res(S)$ és lògicament equivalent a S)

19. (dificultat 2) Sigui S un conjunt de clàusules insatisfactible. Per la completitud refutacional de la resolució, sabem que existeix una demostració per resolució de que $\square \in Res(S)$. És aquesta demostració única?

Tornant a l'exercici:

NO és única. Contraexemple. Sigui S el conjunt amb les 4 clàusules:

$$\begin{aligned} p \vee \ \ q \\ p \vee \neg q \\ \neg p \vee \ \ q \\ \neg p \vee \neg q. \end{aligned}$$

19. (dificultat 2) Sigui S un conjunt de clàusules insatisfactible. Per la completitud refutacional de la resolució, sabem que existeix una demostració per resolució de que $\square \in Res(S)$. És aquesta demostració única?

Podem obtenir $\square$ de més d'una manera:
Per exemple així:

$$\dfrac{\dfrac{p \vee q \qquad p \vee \neg q}{p} \qquad \dfrac{\neg p \vee q \qquad \neg p \vee \neg q}{\neg p}}{\square}$$

Però també així:

$$\dfrac{\dfrac{p \vee q \qquad \neg p \vee q}{q} \qquad \dfrac{p \vee \neg q \qquad \neg p \vee \neg q}{\neg q}}{\square}$$

21. (dificultat 2) Demostra que el llenguatge de les clàusules de Horn és tancat sota resolució, és a dir, a partir de clàusules de Horn, per resolució només s'obtenen clàusules de Horn.

Una clàusula de Horn és una clàusula que té com a màxim 1 literal positiu.
Si tinc: $p \vee C$ i $\neg p \vee D$, i són de Horn, llavors hi ha 0 literals positius en C, i màxim 1 en D.

Per tant, en $C \vee D$ també hi ha com a màxim 1 literal positiu, és a dir, $C \vee D$ també és de Horn.

25. (dificultat 2) Quina és la complexitat de Horn-SAT, és a dir, del problema de determinar si un conjunt de clàusules de Horn S és satisfactible?
Ajuda: pensa si la *positive unit propagation* decideix Horn-SAT, i analitza la complexitat d'això.

unit propagation: en el meu algorisme de SAT mitjançant backtracking (veure labo 1) si tinc una clàusula de la forma $l \vee C$ i una interpretació (parcial, que estic construint) on la part C és falsa, llavors el literal l ha de ser cert.

positive unit propagation: és el mateix, però només amb l positius

25. (dificultat 2) Quina és la complexitat de Horn-SAT, és a dir, del problema de determinar si un conjunt de clàusules de Horn S és satisfactible?
    Ajuda: pensa si la *positive unit propagation* decideix Horn-SAT, i analitza la complexitat d'això.

Les clàusules d'un conjunt S de clàusules de Horn (sense la $\square$), poden ser de la forma:

a)	p	(positive unit clause)
b)	$p \vee \neg q_1 \vee \cdots \vee \neg q_n$	(1 positiu i n negatius, $n > 0$)
c)	$\neg q_1 \vee \cdots \vee \neg q_n$	(0 positius i n negatius, $n > 0$)

Un model de S HA DE satisfer totes les clàusules de tipus a) (les positive units). Si les propago, amb les clàusules de tipus b), obtinc noves positive units, que al seu torn poden propagar-se, etc, etc

25. (dificultat 2) Quina és la complexitat de Horn-SAT, és a dir, del problema de determinar si un conjunt de clàusules de Horn S és satisfactible?
    Ajuda: pensa si la *positive unit propagation* decideix Horn-SAT, i analitza la complexitat d'això.

Per exemple, S pot ser:

p	(tipus a)	
q	(tipus a)	
$r \vee \neg p \vee \neg q$	(tipus b)	$\rightarrow$ propago r
$r' \vee \neg p \vee \neg q \vee \neg r$	(tipus b)	$\rightarrow$ propago r'
$\neg r \vee \neg r'$	(tipus c)	$\rightarrow$ conflicte!!!

És clar que si em surt un conflicte, és insatisfactible. I si no em surt un conflicte? Llavors és satisfactible: hi ha un model! Quin és aquest model?

25. (dificultat 2) Quina és la complexitat de Horn-SAT, és a dir, del problema de determinar si un conjunt de clàusules de Horn S és satisfactible?
    Ajuda: pensa si la *positive unit propagation* decideix Horn-SAT, i analitza la complexitat d'això.

El model és la I tal que $I(p) = 1$ per a totes les positive units p que he anat obtenint (les inicials i les altres) i $I(p) = 0$ per a tots els altres símbols p. Per què això és un model?

És model de les clàusules de tipus:

- les a) ok
- les b) que han propagat alguna cosa: ok
- les b) que no han propagat res: ok, perquè tindrà un literal $\neg q$ on la q no està entre les positive units: $I(q) = 0$.
- les c) (que no han donat conflicte): ok, perquè tindrà un literal $\neg q$ on la q no està entre les positive units: $I(q) = 0$.

25. (dificultat 2) Quina és la complexitat de Horn-SAT, és a dir, del problema de determinar si un conjunt de clàusules de Horn S és satisfactible?
Ajuda: pensa si la *positive unit propagation* decideix Horn-SAT, i analitza la complexitat d'això.

Resumint:

sent S un conjunt de clàusules de Horn, S és insat SSI la **positive unit propagation** dona conflicte.

25. (dificultat 2) Quina és la complexitat de Horn-SAT, és a dir, del problema de determinar si un conjunt de clàusules de Horn S és satisfactible?
    Ajuda: pensa si la *positive unit propagation* decideix Horn-SAT, i analitza la complexitat d'això.

➤ Quin cost té?

Això és lineal, si uso *occur lists* (veure labo 1) i un comptador per clàusula que compta el nombre de literals negatius que li queden. Quan el comptador es posa a zero $\rightarrow$ aquesta clàusula propaga.

NOTA: aquest algorisme només posa a cert aquells símbols p que HAN DE SER certs en QUALSEVOL model de S.
És a dir, el que es calcula és el model MINIMAL de S (aquell model que té el mínim nombre de símbols a 1).

25. (dificultat 2) Quina és la complexitat de Horn-SAT, és a dir, del problema de determinar si un conjunt de clàusules de Horn S és satisfactible?
    Ajuda: pensa si la *positive unit propagation* decideix Horn-SAT, i analitza la complexitat d'això.

Un altre exemple:

p
q
$r \vee \neg p \vee \neg q$
$r' \vee \neg p \vee \neg q \vee \neg r$
$r'' \vee \neg p \vee \neg q \vee \neg r'''$ Aquí, el model *minimal* quin és?

- $I(p) = I(q) = I(r) = I(r') = 1 \qquad I(r'') = I(r''') = 0.$

Però hi ha més models:

- $I(p) = I(q) = I(r) = I(r') = I(r'') = 1 \qquad I(r''') = 0.$
- el model on $I(p) = I(q) = I(r) = I(r') = I(r'') = I(r''') = 1$ (tots).

25. (dificultat 2) Quina és la complexitat de Horn-SAT, és a dir, del problema de determinar si un conjunt de clàusules de Horn S és satisfactible?
    Ajuda: pensa si la *positive unit propagation* decideix Horn-SAT, i analitza la complexitat d'això.

Veiem que si S és de Horn i és satisfactible, llavors S té un model minimal únic.

És també cert això si S no és de Horn?

NO! Per exemple, $S = \{p \vee q\}$, llavors hi ha dos models minimals:

A) $I(p) = 1 \;\; I(q) = 0$.

B) $I(p) = 0 \;\; I(q) = 1$.

26. (dificultat 2) Les *clàusules de Krom* són aquelles que tenen com a màxim dos literals. Quantes clàusules de Krom es poden construir amb n símbols de predicat? Demostra que basta un nombre quadràtic de passos de resolució per a decidir 2-SAT, és a dir, si un conjunt de clàusules de Krom és satisfactible o no.

Hi ha $2n$ literals. Cada clàusula de Krom és un subconjunt de com a màxim 2 literals d'aquests $2n$.

$$\begin{array}{rll} \text{Hi ha } \binom{2n}{2} & \text{clàusules de Krom de dos literals} & + \\ \binom{2n}{1} = 2n & \text{clàusules de Krom de un literal} & + \\ 1 & \text{clàusula de Krom de 0 literals.} & \end{array}$$

$\binom{2n}{2} = 2n \cdot (2n-1)/2 = 4n^2 + \cdots = \mathcal{O}(n^2)$. Un número quadràtic.

26. (dificultat 2) Les *clàusules de Krom* són aquelles que tenen com a màxim dos literals. Quantes clàusules de Krom es poden construir amb n símbols de predicat? Demostra que basta un nombre quadràtic de passos de resolució per a decidir 2-SAT, és a dir, si un conjunt de clàusules de Krom és satisfactible o no.

Cada pas de resolució em donarà una altra clàusula de Krom: "el llenguatge de les clàusules de Krom està tancat sota resolució".

$$\frac{p \vee C \qquad \neg p \vee D}{C \vee D}$$

la part C té com a màxim 1 literal i la part D també. Per tant $C \vee D$ té màxim dos.

Per això, la resolució acaba després d'un nombre quadràtic de passos. Si ho implementem bé, això ens dona un algorisme quadràtic per a 2-SAT.

Resumint

Ja tenim dos problemes concrets (subcasos del problema de SAT general) que són polinòmics: Horn-SAT i 2-SAT!

27. (dificultad 3) Algorisme per a 2-SAT basat en detecció de cicles en un graf.

Sigui S un conjunt de clàusules de Krom.

Cada clàusula de Krom $l \vee l'$, en realitat representa dues implicacions:

$$\neg l \rightarrow l'$$
$$\neg l' \rightarrow l$$

Puc muntar un graf G a partir de S, amb totes aquestes arestes: cada clàusula de S em dona dues arestes en G.

Si en G tinc un camí $p \rightarrow \cdots \rightarrow \neg p$, llavors $S \models \neg p$.

Ho puc demostrar per correcció de la resolució.

27. (dificultat 3) Algorisme per a 2-SAT basat en detecció de cicles en un graf.

Si tinc:

$$p \to l \to l' \to l'' \to \cdots \neg p, \text{ llavors } S \models \neg p$$

Per resolució:

$$\dfrac{\dfrac{\dfrac{\neg p \vee l \quad \neg l \vee l'}{\neg p \vee l'} \quad \neg l' \vee l''}{\neg p \vee l''}}{\begin{array}{c}\vdots \\ \hline \neg p \vee \neg p\end{array}} \qquad \text{que és } \neg p$$

27. (dificultad 3) Algorisme per a 2-SAT basat en detecció de cicles en un graf.

Si en G tinc un camí $p \to \cdots \to \neg p$, llavors $S \models \neg p$.

Similarment, si en G tinc un camí $\neg p \to \cdots \to p$, llavors $S \models p$.

I si tinc els dos camins, és que hi ha un cicle en G que conté un símbol p i el seu negat $\neg p$, ... i llavors S és insatisfactible.

I a l'inrevés, si S és insatisfactible, llavors hi ha un cicle en G que conté un símbol i el seu negat.

Resumint

S és insatisfactible SSI hi ha un cicle en G que conté un símbol i el seu negat.

27. (dificultad 3) Algorisme per a 2-SAT basat en detecció de cicles en un graf.

➤ Quin és el cost d'aquest algorisme per a 2-SAT?

- Muntar el graf és lineal.
- Després detectar cicles, es fa amb l'algorisme de *Strongly Connected Components* (*SCCs*), (Components Fortament Connexes) que és lineal.

Resumint

Surt un algorisme lineal per a 2-SAT.

Pensem ara en SAT en general.

☞Apunts de la web de LI: "Breu resum sobre NP i NP-completitud" (Robert Nieuwenhuis)

Remembering some intuitions about NP and NP-completeness
(for more formal definitions and details, see the slides of the EDA course on this same website)

Decision problems and complexity classes
Here we focus on decision problems, the ones with output "yes" or "no", and on *classifying problems* (not algorithms!) according to the time needed to solve them (with the best of the available algorithms), and we will call problem A *harder* than problem B if solving A needs more time than solving B.

Decision problems and complexity classes (cont.)

For example, given a sequence of integers, the problem of deciding whether it contains the integer 7 can be solved in *linear* time. We say that it belongs to the *class* of problems solvable in linear time. If moreover the input sequence is *ordered*, then we can say more: it belongs to a proper subclass of the problems solvable in linear time, namely the ones solvable in *logarithmic* time (in this case, by binary search). Here we see that in fact what matters is *how fast the running time grows depending on the size of the input*.

Other problems are not linear, but harder. The class of *polynomial* problems is called P. Note that all logarithmic, linear, quadratic, cubic, etc., problems are in P.

Decision problems and complexity classes (cont.)

Some other problems are even harder, and are not in P. The class of *exponential* problems is called EXP (solvable in time with the input size n in the exponent; note that for large enough n, the number 2^n is much larger than n^2, n^3, or n^k for whatever constant k). It is known that $P \subset EXP$ (there are problems in EXP that are not in P such as "generalized chess").

The class NP, membership in NP, NP-hardness and NP completeness

There is a special class, NP, for which it is known that P $\subseteq$ NP $\subseteq$ EXP. NP is the class of problems having a Nondeterministic Polynomial algorithm. Roughly, this means that a problem A is in NP if, whenever the answer to A for a given input is "yes", there is a "witness" (a "solution") that allows one to verify this "yes" in polynomial time.

The class NP, membership in NP, NP-hardness and NP completeness (cont.)

The most famous problem in NP is SAT, the problem of deciding whether a given propositional input formula F is satisfiable or not. This problem is clearly in NP: if the answer is "yes", the witness is the model, which can be checked in polynomial (even linear) time.

Another example of problem in NP is *3-colorability*: can we color each node of a given graph G with one of three colors, such that adjacent nodes get different colors? Here the witness is the coloring, indicating each node's color.

The class NP, membership in NP, NP-hardness and NP completeness (cont.)

A problem P is called *NP-hard* if *any* other problem in NP can be polynomially *reduced to* P. SAT is NP-hard: any problem in NP can be polynomially reduced to (or solved by, or expressed as) a SAT problem.

This means that for any problem A in NP and input data D for A, we can build in polynomial time a SAT formula F that is satisfiable if, and only if, the answer to A on input D is "yes". Moreover, from a satisfiablity witness of F (i.e., a model), it is usually easy to reconstruct a witness (or a "solution") for A on input D.

The class NP, membership in NP, NP-hardness and NP completeness (cont.)

For example, we can reduce 3-colorability to SAT. Let G be a graph with n nodes. Introducing $3n$ propositional symbols x_{ic} meaning "node i gets color c", let F state,

- for each node i, that it gets at least one color (a clause $x_{i1} \vee x_{i2} \vee x_{i3}$) and,
- for each edge (i, j), that i and j do not get the same color (three clauses per edge: $\neg x_{i1} \vee \neg x_{j1}$, $\neg x_{i2} \vee \neg x_{j2}$, and $\neg x_{i3} \vee \neg x_{j3}$).

Then F is satisfiable iff G is 3-colorable, and from any model for F it is trivial to reconstruct a 3-coloring for G.

The class NP, membership in NP, NP-hardness and NP completeness (cont.)

Note that if SAT can be polynomially reduced to some problem P, then P is NP-hard too. Apart from SAT, many other problems in NP have been proved NP-hard too (doing such reductions, or chains of them).

Note that, by such reductions, if we had a polynomial algorithm for *any* single NP-hard problem, then we would have it for *all* problems in NP, that is, we would have P=NP. That would have dramatic consequences, because there are many very important real-world problems in NP. In fact, there is a million-dollar prize (search "millenium problems") for whoever proves either P=NP or $\mathrm{P} \neq \mathrm{NP}$.

The class NP, membership in NP, NP-hardness and NP completeness (cont.)

Since $P \subset EXP$, at least one of the two inclusions in $P \subseteq NP \subseteq EXP$ is strict, and it is believed that both are, i.e., $P \subset NP \subset EXP$.

A problem is called *NP-complete* if

A) it is in NP, and

B) it is NP-hard.

Exercici: Emplena la resta de la taula amb la complexitat algorítmica corresponent.

	SAT	TAUT
CNF	NP-complet	
DNF	lineal (1)	

(1) Per l'exercici 12, sabem que per una fórmula en DNF podem decidir si és satisfactible en temps lineal.

Exercici: Emplena la resta de la taula amb la complexitat algorítmica corresponent.

	SAT	TAUT
CNF	NP-complet	lineal (2)
DNF	lineal (1)	

(2) Una CNF $C_1 \wedge \cdots \wedge C_n$ és tautologia ssi totes les seves clàusules C_i són tautologias.
Sabem per l'exercici 5. que una clàusula és una tautologia ssi conté alhora p i $\neg p$.

Exercici: Emplena la resta de la taula amb la complexitat algorítmica corresponent.

	SAT	TAUT
CNF	NP-complet	lineal (2)
DNF	lineal (1)	NP-complet (3)

(3) Una DNF $Cub_1 \vee \cdots \vee Cub_N$ és tautologia ssi $\neg(Cub_1 \vee \cdots \vee Cub_N)$ és insat.
Movent les negacions cap a dins en $\neg(Cub_1 \vee \cdots \vee Cub_N)$ obtenim una CNF F.

Exercici: Emplena la resta de la taula amb la complexitat algorítmica corresponent.

	SAT	TAUT
CNF	NP-complet	lineal (2)
DNF	lineal (1)	NP-complet (3)

(3) [cont.] A l'inrevés també:
Una CNF $C_1 \wedge \cdots \wedge C_n$ és insatisfactible ssi $\neg(C_1 \wedge \cdots \wedge C_n)$ és tautologia ssi $Cub_1 \vee \cdots \vee Cub_N$ és tautologia, on $Cub_1 \vee \cdots \vee Cub_N$ és la DNF obtinguda movent les negacions cap a dins en $\neg(C_1 \wedge \cdots \wedge C_n)$.

Exercici: Emplena la resta de la taula amb la complexitat algorítmica corresponent.

	SAT	TAUT
CNF	NP-complet	lineal (2)
DNF	lineal (1)	NP-complet (3)

(3) [cont.]

Una DNF $Cub_1 \vee \cdots \vee Cub_N$ és tautologia ssi $\neg(Cub_1 \vee \cdots \vee Cub_N)$ és insat.

Per tant, decidir si una DNF és tautologia ha de ser NP-complet! Si no, tindríem una manera de fer SAT en temps polinòmic comprovant si $Cub_1 \vee \cdots \vee Cub_N$ és tautologia!

Per al proper dia:

- exercicis següents, i
- apunts de la web de LI:
  ☞ "la transformació de Tseitin a CNF equisatisfactible."