CNF och DNF

- Konjunktiv normalform (CNF)
- Omskrivning av en formel till CNF
- Disjunktiv normalform (DNF)
- Lite om komplexitet

Konjunktiv normalform

Definitioner

- En litteral är en atom eller en negerad atom.
- En klausul är en formel på formen $l_1 \vee \cdots \vee l_m$ där $m \in \mathbb{N}$ and $l_1, \ldots, l_m$ är litteraler.
- En formel är på konjunktiv normalform (*conjunctive normal form*) om den är på formen $\varphi_1 \wedge \cdots \wedge \varphi_n$ där $n \in \mathbb{N}$ och $\varphi_1, \ldots, \varphi_n$ är klausuler.

En formel i CNF är alltså en konjunktion av disjunktioner av litteraler.

Exempel: $(A \vee C \vee \neg D) \wedge (B \vee \neg B) \wedge (A \vee \neg C \vee \neg D \vee \neg E) \wedge B$.

Uppgift 1

Vilka av följande formler är på konjunktiv normalform?

$\varphi_1 = A \wedge B$
$\varphi_2 = (A \vee C) \wedge B$
$\varphi_3 = (A \wedge C) \vee B$
$\varphi_4 = A \vee B$
$\varphi_5 = A$
$\varphi_6 = A \vee C \wedge B$
$\varphi_7 = A \wedge B \wedge C$
$\varphi_8 = (\neg A \vee A) \wedge (A \vee A)$

Lösning 1

- φ_1 är på CNF då den har två klausuler med en litteral i varje.
- φ_2 är på CNF.
- φ_3 är inte på CNF då den är en disjunktion av konjunktioner.
- φ_4 är på CNF då den består av en enda klausul.
- φ_5 är på CNF då även den består av en enda klausul.
- φ_6 är inte på CNF, även om de implicit parenteserna läggs till (för då får man $A \vee (C \wedge B)$ eftersom $\wedge$ binder hårdare än $\vee$.
- φ_7 är på CNF med tre klausuler om vardera en litteral.
- φ_8 är på CNF, en klausul kan innehåller samma litteral flera gånger, även om det är onödigt.

Omskrivning till CNF (Exempel)

Varje formel kan skrivas om till en ekvivalent formel på CNF.

1. Originalformel (exempel): $(A \rightarrow (\neg(B \wedge C))) \wedge (\neg(C \wedge (A \vee \neg D)))$
2. $\varphi \rightarrow \varphi' \equiv \neg\varphi \vee \varphi'$ ger $(\neg A \vee (\neg(B \wedge C))) \wedge (\neg(C \wedge (A \vee \neg D)))$
3. Använd de Morgans regler för att flytta alla negationer in till atomerna (och förkasta dubbla negationer pga $\neg\neg\varphi \equiv \varphi$):
   $\cdots \equiv (\neg A \vee (\neg B \vee \neg C)) \wedge (\neg C \vee \neg(A \vee \neg D))$
   $\equiv (\neg A \vee (\neg B \vee \neg C)) \wedge (\neg C \vee (\neg A \wedge D))$
4. Använd distributivlagen för att eliminera disjunktioner av konjunktioner
   $\cdots \equiv (\neg A \vee (\neg B \vee \neg C)) \wedge ((\neg C \vee \neg A) \wedge (\neg C \vee D))$
5. Stryk överflödiga parenteser: $(\neg A \vee \neg B \vee \neg C) \wedge (\neg C \vee \neg A) \wedge (\neg C \vee D)$

Omskrivning till CNF i allmänhet

1. Använd ekvivalenserna $\varphi \leftrightarrow \varphi' \equiv (\varphi \wedge \varphi') \vee (\neg\varphi \wedge \neg\varphi')$ samt $\varphi \rightarrow \varphi' \equiv \neg\varphi \vee \varphi'$ för att få en formel som endast innehåller $\neg$, $\vee$ och $\wedge$.
2. Använd de Morgans regler samt $\neg\neg\varphi \equiv \varphi$ för att flytta alla negationer in till atomerna.
3. Använd distributivlagen $\varphi \vee (\varphi' \wedge \varphi'') \equiv (\varphi \vee \varphi') \wedge (\varphi \vee \varphi'')$ för att eliminera disjunktioner av konjunktioner.
4. Snygga till formeln genom att ta bort överflödiga parenteser. Formeln kan ev också förenklas, t ex genom att ta bort dubbla litteraler i klausuler.

OBS! Kom ihåg att upprepa stegen 2 och 3 tills formeln är på den önskade formen (dvs tills det inte längre går att fortsätta).

Disjunktiv normalform

Definition En formel är på disjunktiv normalform (DNF) om den är på formen $\varphi_1 \vee \cdots \vee \varphi_n$ där $n \in \mathbb{N}$ and $\varphi_1, \ldots, \varphi_n$ är konjunktioner av litteraler (dvs på formen $l_1 \wedge \cdots \wedge l_m$ där $m \in \mathbb{N}$ and $l_1, \ldots, l_m$ är litteraler).

Exempel: $(A \wedge C \wedge \neg D) \vee (B \wedge \neg B) \vee (A \wedge \neg C \wedge \neg D \wedge \neg E) \vee B$.

Omskrivning av godtyckliga formler till DNF

Funkar på samma sätt som när det gäller CNF men i steg 3 används den duala distributivlagen $\varphi \wedge (\varphi' \vee \varphi'') \equiv (\varphi \wedge \varphi') \vee (\varphi \wedge \varphi'')$ istället.

Uppgift 2

Vilka av följande formler är på disjunktiv normalform?

$\varphi_1 = A \wedge B$
$\varphi_2 = (A \vee C) \wedge B$
$\varphi_3 = A \wedge C \wedge B$
$\varphi_4 = A \wedge B \vee C$
$\varphi_5 = \neg A$
$\varphi_6 = (A \wedge B) \vee C$
$\varphi_7 = A \vee C$
$\varphi_8 = (\neg A \wedge A) \vee (A \wedge A)$

Lösning 2

- φ_1 är på DNF då den består av en enda konjunktion.
- φ_2 är inte på DNF då den är en konjunktion av disjunktioner.
- φ_3 är på DNF eftersom den består av en enda konjunktion med tre litteraler.
- φ_4 är på DNF om de implicita parenteserna som följer av prioritetsreglerna läggs till kring $A \wedge B$.
- φ_5 är på DNF och består av en enda konjunktion.
- φ_6 är DNF.
- φ_7 är på DNF och har två konjunktioner, A och C.
- φ_8 är på DNF; en konjunktion kan innehåller samma litteral flera gånger.

Lite om komplexitet (1)

- Satisfierbarhet av formler i CNF (problemet SAT) är NP-komplett.
- Satisfierbarhet av formler i DNF (problemet DNF-SAT) är mycket enklare: Säg att två litteraler l, l' är komplementära om de är varandras komplement, dvs om en av dem är en atom och den andra är dess negation. En konjunktion av litteraler är satisfierbar om och endast om den inte innehåller komplementära litteraler. Dessutom är en disjunktion $\varphi_1 \vee \cdots \vee \varphi_n$ satisfierbar om och endast om en av $\varphi_1, \ldots, \varphi_n$ är satisfierbar. En formel $\varphi_1 \vee \cdots \vee \varphi_n$ på DNF är alltså satisfierbar omm någon φ_i inte innehåller komplementära litteraler.

  Slutsats: Satisfierbarhet av formler i DNF kan avgöras inom linjär tid.

Lite om komplexitet (2)

Varför kan SAT inte lösas genom att först konvertera formeln till DNF och sedan lösa DNF-SAT (dvs med hjälp av en reduktion)?

- Jovisst, det går men...
- Problemet är att distributivlagen fördubblar den del av formeln som distribueras: $\boxed{\varphi} \wedge (\varphi' \vee \varphi'') \equiv (\boxed{\varphi} \wedge \varphi') \vee (\boxed{\varphi} \wedge \varphi'')$. Dess iterativa tillämpning kan därmed leda till en formel vars längd är 2^n (exaktare: $\Omega(2^n)$), där n är formelns ursprungliga längd. Den här algoritmen skulle alltså ta exponentiell tid.
- Det här innebär inget bevis på att det inte går att lösa SAT inom polynomisk tid. Det enda vi för närvarande kan säga är att det inte går på det här viset (samt att P=NP om det skulle gå på något annat sätt).