Fondamenti di Informatica 2

Linguaggi e Complessità : Lezione 3

Clausole

♢ I *letterali* sono simboli proposizionali (*atomi*) o simboli proposizionali negati *atomi negati*.

♢ Una *formula* è detta in *forma normale negativa* se il segno di negazione compare solo davanti agli atomi.

♢ Una *clausola* è una disgiunzione di letterali $L_1 \vee L_2 \vee \cdots \vee L_n$.

♢ Una *formula* è detta in *forma normale congiuntiva* (CNF) o in *forma clausale* oppure si dice che essa è un *insieme di clausole* se è in forma normale negativa e inoltre ha la forma $C_1 \wedge C_2 \wedge \cdots \wedge C_n$ (oppure equivalentemente $\{C_1, C_2, \cdots, C_n\}$) dove le C_i sono clausole.

♢ Poiché la disgiunzione è commutativa, una clausola generica può essere scritta come:

$$A_1 \vee A_2 \vee \cdots \vee A_n \vee \neg B_1 \vee \neg B_2 \vee \cdots \vee \neg B_m$$

dove gli A_i sono gli n atomi e B_j sono gli m atomi negati.

Clausole

♢ Se $n = m = 0$ si ha la *clausola vuota* e si scrive $\{\}$.

♢ Una clausola verrà nel seguito anche indicata come l'insieme dei suoi letterali e cioè $\{L_1, L_2, \ldots, L_p\}$, omettendo il simbolo di disgiunzione $\vee$.

♢ Se L è un letterale e $C = \{L_1, L_2, \ldots, L_p\}$ una clausola, talvolta scriveremo $L \cup C$ per indicare la clausola $C' = \{L\} \cup C$,
cioè $C' = \{L, L_1, L_2, \ldots, L_p\}$.

♢ Se $n = 1$, cioè se la clausola ha un solo letterale positivo e quindi la forma:

$$A_1 \vee \neg B_1 \vee \ldots, \vee \neg B_m$$

si parla di *clausola definita*.

Trasformazione in clausole: $Convert1$

Si può facilmente mettere una formula qualunque in forma CNF attraverso l'applicazione ripetuta dei seguenti 3 passi:

- Eliminando i connettivi $\rightarrow$ e $\leftrightarrow$:

$$(\alpha \leftrightarrow \beta) \equiv (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$$
$$(\alpha \rightarrow \beta) \equiv (\neg\alpha \vee \beta)$$

- Mettendo in forma normale per la negazione (NNF):

$$\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$$
$$\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$$
$$\neg\neg\alpha \equiv \alpha$$

- Distribuendo la disgiunzione $\vee$ sulla congiunzione $\wedge$:

$$\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$$

A questo punto dovremo solo raggruppare tutti i letterali in disgiunzione in clausole per ottenere una formula in CNF.

Problemi di $Convert1$

Il problema dellalgoritmo $Convert1$ è che la dimensione della formula CNF ottenuta potrebbe essere molto maggiore di quella originaria. Esempio:

$((A_1 \wedge A_2) \vee (B_1 \wedge B_2)) \vee (C_1 \wedge C_2))$
$(((A_1 \wedge A_2) \vee B_1) \wedge ((A_1 \wedge A_2) \vee B_2) \vee (C_1 \wedge C_2))$
$((A_1 \vee B_1) \wedge (A_2 \vee B_1) \wedge ((A_1 \vee B_2) \wedge (A_2 \vee B_2) \vee (C_1 \wedge C_2))$
$((A_1 \vee B_1) \wedge (A_2 \vee B_1) \wedge ((A_1 \vee B_2) \wedge (A_2 \vee B_2) \vee C_1) \wedge ((A_1 \vee B_1) \wedge$
$(A_2 \vee B_1) \wedge ((A_1 \vee B_2) \wedge (A_2 \vee B_2) \vee C_2))$
$\ldots$
$\ldots$
$(A_1 \vee B_1 \vee C_1) \wedge (A_2 \vee B_1 \vee C_1) \wedge (A_1 \vee B_2 \vee C_1) \wedge (A_2 \vee B_2 \vee C_1) \wedge$
$(A_1 \vee B_1 \vee C_2) \wedge (A_2 \vee B_1 \vee C_2) \wedge (A_1 \vee B_2 \vee C_2) \wedge (A_2 \vee B_2 \vee C_2)$

Trasformazione in clausole: $Convert2$

Per ovviare a questi problemi possiamo usare una diversa trasformazione, dove i passi 1 e 2 sono uguali al precedente producendo una formula F, mentre il passo 3 produce una formula CNF F_1 e consiste in:

- Se A è una lettera proposizionale, $Convert2(A) = A$ e $Convert2(\neg A) = \neg A$.
- Per ogni sottoformula $\gamma = \alpha \wedge \beta$ di F, crea tre nuove lettere $A_\gamma, A_\alpha, A_\beta$, aggiungi ad F_1 le clausole $\neg A_\gamma \vee A_\alpha$ e $\neg A_\gamma \vee A_\beta$ e continua con la conversione di α e di β.
- Per ogni sottoformula $\gamma = \alpha \vee \beta$ di F, crea tre nuove lettere $A_\gamma, A_\alpha, A_\beta$, aggiungi ad F_1 la clausola $\neg A_\gamma \vee A_\alpha \vee A_\beta$ e continua con la conversione di α e di β.
- Aggiungi la clausola unitaria A_F ad F_1.

Esempio di *Convert2*

Usiamo la formula della slide precedente $((A_1 \wedge A_2) \vee (B_1 \wedge B_2)) \vee (C_1 \wedge C_2))$. Diamo un nome a tutte le sottoformule $D_1 = A_1 \wedge A_2$, $D_2 = B_1 \wedge B_2$ e $D_3 = C_1 \wedge C_2$, $D_4 = D_1 \vee D_2$ e $F = D_4 \vee D_3$ ed inizializzamo $F_1 = \{\}$

$((A_1 \wedge A_2) \vee (B_1 \wedge B_2)) \vee (C_1 \wedge C_2))$	$F_1 = \{\}$
$((D_3 \vee (B_1 \wedge B_2)) \vee (C_1 \wedge C_2))$	$F_1 = F_1 \cup \{(\neg D_3 \vee A_1), (\neg D_3 \vee A_2)\}$
$((D_3 \vee D_4) \vee (C_1 \wedge C_2))$	$F_1 = F_1 \cup \{(\neg D_4 \vee B_1), (\neg D_4 \vee B_2)\}$
$((D_3 \vee D_4) \vee D_2)$	$F_1 = F_1 \cup \{(\neg D_2 \vee C_1), (\neg D_2 \vee C_2)\}$
$(D_1 \vee D_2)$	$F_1 = F_1 \cup \{(\neg D_1 \vee D_3 \vee D_4)\}$
F	$F_1 = F_1 \cup \{(\neg A_F \vee D_1 \vee D_2)\}$
	$F_1 = F_1 \cup \{A_F\}$

$F_1 = \{(\neg D_3 \vee A_1), (\neg D_3 \vee A_2), (\neg D_4 \vee B_1), (\neg D_4 \vee B_2), (\neg D_2 \vee C_1), (\neg D_2 \vee C_2), (\neg D_1 \vee D_3 \vee D_4), (\neg D_1 \vee D_3 \vee D_4), (\neg A_F \vee D_1 \vee D_2), A_F\}$

Proprietà

- Data una formula α, si può facilmente dimostrare che $\alpha \equiv Convert1(\alpha)$
- Data una formula α, si noti che $\alpha \not\equiv Convert2(\alpha)$, infatti non hanno nemmeno lo stesso alfabeto. Si può però dimostrare che α è soddisfacibile se e solo se $Convert2(\alpha)$ è soddisfacibile. Si dice che la trasformazione $Convert2$ preserva la soddisfacibilità

Algoritmo DPLL: Idea

Si applica ad un insieme di clausole ed è basato sulla tecnica del backtracking, infatti prova ricorsivamente tutte le assegnazioni possibili per vedere se una soddisfa la formula. Usa 2 controlli per ridurre (significativamente) il numero di assegnazioni da provare:

- **Unit propagation**. Se una clausola è **unitaria** (contiene un solo letterale) essa può essere soddisfatta solo assegnandogli il valore corretto (1 se il letterale è positivo, 0 se è negativo).
- **Pure literal elimination**.Se un letterale appare sempre o positivo o negativo viene chiamato **puro**. I letterali puri possono essere resi tutti veri, senza bisogno di fare scelte.

Algoritmo DPLL: Funzioni di appoggio

- **unit-propagate(letterale l, formula F)** Per ogni clausola $c \in F$, se l compare in c con lo stesso segno allora cancella la clausola c, se l compare in c con il segno opposto allora cancella l da c. Restituisce la formula semplificata.

- **pure-literal-assign(letterale l, formula F)** Cancella tutte le clausole che contengono il letterale l. Restituisce la formula semplificata.

- **choose-literal(formula F)** Seleziona (casualmente) un letterale che compare in F. Restituisce il letterale scelto.

Algoritmo DPLL: Codice

```
function DPLL(F)
   if F is empty
       then return true;
   if F contains an empty clause
       then return false;
   for every unit clause l in F
      F = unit-propagate(l, F);
   for every literal l that occurs pure in F
      F = pure-literal-assign(l, F);
   l := choose-literal(F);
   return DPLL(F Union {l}) OR DPLL(F Union {NOT l});
```

Esercizi

1. Verificare che $\neg A \rightarrow (A \rightarrow B)$ è una tautologia.
2. Mostrare che dall'insieme di formule $\Gamma = \{(P \wedge Q) \rightarrow R, P, P \rightarrow Q\}$ si deriva $Q \wedge R$.

Riformulate i problemi come problemi di soddisfacibilità , poi usate l'algoritmo $Convert1$ (o $Convert2$) per mettere in forma CNF ed infine DPLL per verificare la soddisfacibilità .

Esercizi: Svolgimento 1

$\neg A \rightarrow (A \rightarrow B)$ è una tautologia se e solo se $\neg(\neg A \rightarrow (A \rightarrow B))$ non è soddisfacibile. Usando $Convert1$ otteniamo:

$\neg(\neg\neg A \vee (\neg A \vee B))$	elimino le implicazioni
$\neg(A \vee (\neg A \vee B))$	elimino doppia negazione
$(\neg A) \wedge \neg(\neg A \vee B))$	spingo dentro la negazione
$\neg A \wedge A \wedge \neg B$	spingo dentro la negazione
$F = \{A, \neg A, B\}$	metto in forma a clausole

Applicando la DPLL ottengo:

$F =$ unit-propagate(A, F)	seleziono la clausola unitaria$A$
$F = \{\}$	applicando unit-propagate la clausola$\neg A$ diventa vuota

Quindi la formula $\neg A \rightarrow (A \rightarrow B)$ è una tautologia.

Esercizi: Svolgimento 2

Da $\Gamma = \{(P \wedge Q) \rightarrow R, P, P \rightarrow Q\}$ si deriva $Q \wedge R$ se e solo se $\Gamma \cup \{\neg(Q \wedge R)\}$ non è soddisfacibile. Usando $Convert1$ otteniamo:

$\neg(P \wedge Q) \vee R, P, \neg P \vee Q, \neg(Q \wedge R)$	elimino le implicazioni
$(\neg P \vee \neg Q) \vee R, P, \neg P \vee Q, \neg Q \vee \neg R$	spingo dentro la negazione
$F = \{\neg P \vee \neg Q \vee R, P, \neg P \vee Q, \neg Q \vee \neg R\}$	metto in forma a clausole

Applicando la DPLL ottengo:

$F =$ unit-propagate(P, F)	seleziono la clausola unitaria P
$F = \{\neg Q \vee R, Q, \neg Q \vee \neg R\}$	applico unit-propagate(P, F)
$F =$ unit-propagate(Q, F)	seleziono la clausola unitaria Q
$F = \{R, \neg R\}$	applico unit-propagate(Q, F)
$F =$ unit-propagate(R, F)	seleziono la clausola unitaria R
$F = \{\}$	applico unit-propagate(R, F) ottenendo la clausola vuota

Quindi dalla formula $\Gamma\{(P \wedge Q) \rightarrow R, P, P \rightarrow Q\}$ si deriva $Q \wedge R$.

Esercizio: Vacanza

Se parto e vado in vacanza allora sono contento
Se parto allora vado in vacanza
Parto
posso concludere che: vado in vacanza e sono contento??

$\Gamma = \{(P \wedge V) \rightarrow C, P \rightarrow V, P\} \vdash_T (V \wedge C)$

Riscriviamo in termini di soddisfacibilità usando la regola $\Gamma \vdash \alpha$ se e solo se $\Gamma \wedge \neg\alpha$ è insoddisfacibile.

Si!!!!

Esercizio: La lotteria

Sono felice se e solo se ho vinto la lotteria, o la mia ragazza esce con me
Se piove la mia ragazza non esce con me
Piove e sono felice
posso concludere che: sono felice se e solo se ho vinto la lotteria??

$$\Gamma = \{F \leftrightarrow (L \vee E), P \rightarrow \neg E, P \wedge F\} \vdash_T (F \leftrightarrow L)$$

Esercizio: in viaggio

Se nevica non si va in macchina
Se tira vento non si va in aereo
Se i piloti e i ferrovieri scioperano non si va né in aereo né in treno
Se c'è bel tempo non nevica né tira vento

Dire se è consistente con le affermazioni fatte che:
A. Se è bel tempo si viaggia come si vuole
B. Se è bel tempo e non c'è nessuno sciopero si viaggia come si vuole
C. Se nevica si va in aereo
D. Se nevica non si va in aereo

Esercizio sui modelli 1

Se Giovanni studia canto e non ha una brutta voce allora avrà successo alla Scala.

Siano:
$C = studiacanto$
$B = bruttavoce$
$S = successo$

Dire quali tra i seguenti insiemi sono modelli:
$\{C, B, S\}$
$\{\mathsf{C},\mathsf{S}\}$
$\{B, S\}$
$\{\mathsf{C}\}$
$\{B\}$
$\{\}$

Esercizio sui modelli 2

Verificare se

$$\{(C \wedge \neg B) \rightarrow S, C, S\} \models B$$

per via semantica, cioè ragionando sui modelli e non facendo deduzione.

La risposta è no

Esercizio sui modelli 3

Una formula costituita dalla congiunzione di n letterali distinti quanti modelli ha?

$$A_1 \wedge \ldots \wedge A_n$$

Una formula costituita dalla disgiunzione di n letterali distinti quanti modelli ha?

$$A_1 \vee \ldots \vee A_n$$