Cenni di Soddisfacibilità Booleana

Domenico Salvagnin

2011-12-18

1 Logica delle proposizioni

La logica delle proposizioni è lo studio di formule la cui verità è determinata dal valore di verità delle loro parti. Tali formule sono ottenute combinando formule più piccole con connettivi logici, quali *and*, *or* e *not*. I componenti di base sono detti *proposizioni atomiche*. Una formula booleana ottenuta combinando le proposizioni atomiche $x_1, \dots, x_n$ può pertanto essere vista come una funzione booleana $f(x_1, \dots, x_n) : \{T, F\}^n \to \{T, F\}$, dove abbiamo indicato con T e F i valori *True* e *False*.

Più formalmente, le formule logiche possono essere definite ricorsivamente come:

- una formula vuota (per definizione falsa)
- proposizione atomiche x_j, che sono formule "di base" che possono essere vere o false
- espressioni quali $\neg(F)$, $(F) \vee (G)$ e $(F) \wedge G$, dove F e G sono formule.

Il valore di una formula composta dipende dal valore di verità delle formule costituenti e dalle tabelle di verità associate ai connettivi logici usati. Oltre ai connettivi di base *and*, *or* e *not*, sono di solito usati anche i connettivi di implicazione ($\to$) e di equivalenza ($\equiv$). $F \to G$ è equivalente a $\neg F \vee G$, mentre $F \equiv G$ è equivalente a $(F \to G) \wedge (G \to F)$.

2 Problema di soddisfacibilità booleana

Un tipo particolare di formula logica è la *clausola* (*clause*), cioè una disgiunzione (*or*) di *literals* (proposizioni atomiche o loro negazioni). Una clause è ad esempio $x_1 \vee \neg x_2 \vee x_3$.

Una formula è detta in forma congiuntiva normale (*conjunctive normal form*, CNF) se è una congiunzione (*and*) di clauses. Il problema classico della soddisfacibilità booleana (SAT) è determinare se una formula in CNF è soddisfacibile o meno, cioè se è possibile assegnare un valore di verità alle proposizione atomiche in modo che la formula sia vera.

Le clauses godono delle seguenti proprietà:

- una clause F implica un'altra clause G se e solo se la assorbe, cioè ogni literal di F è anche un literal di G. Pertanto, due clauses sono equivalenti se e solo se sono uguali.
- ogni formula logica può essere portata in forma CNF e la trasformazione è polinomiale, pertanto è possibile considerare le formule in CNF senza perdita di generalità.

Una particolare classe di clausole sono le cosiddette clausole *Horn*. Una clausola si dice Horn se ha al più un literal positivo (cioè non negato). L'interpretazione di una clausola Horn

$$x_k \vee (\bigvee_j \neg x_j)$$

è la seguente

$$(\bigwedge_j x_j) \rightarrow x_k$$

Tali clausole sono molto usate per modellare sistemi esperti.

3 Algoritmi risolutivi per SAT

Esistono due principali algoritmi per risolvere problemi SAT:

1. la regola di inferenza di risoluzione (*resolution*)
2. un'algoritmo enumerativo ad albero, detto schema di Davis–Putnam–Logemann–Loveland (DPLL)

Entrambi i metodi hanno complessità esponenziale al caso peggiore, cosa che non sorprende essendo il problema SAT NP-completo.

3.1 Resolution

La regola di risoluzione è una regola molto semplice di inferenza logica che deriva a partire da due clausole generatrici, che contengono una un literal e l'altra la sua negazione, una nuova clausola, detta risolvente (*resolvent*). Ad esempio, da

$$x_1 \vee x_2 \vee x_3$$

$$\neg x_1 \vee x_2 \vee \neg x_4$$

si ottiene

$$x_2 \vee x_3 \vee \neg x_4$$

Il metodo di risoluzione ragiona per casi. La proposizione atomica x_1 è vera o falsa. Se è vera, allora $x_2 \vee \neg x_4$ deve essere vera. Se è falsa, allora deve essere vera $x_2 \vee x_3$. In entrambi i casi, il risolvente è vero.

L'algoritmo di risoluzione consiste nell'applicare ripetutamente tale regola all'insieme corrente di clausole, fino a che non è possibile derivare niente di nuovo. La formula di partenza è soddisfacibile se e solo se durante il processo di risoluzione non viene derivata la clausola vuota. Uno schema generale dell'algoritmo è in Figura 1.

L'algoritmo di risoluzione è un metodo di inferenza puro (non effettua enumerazione) e come tale è l'equivalente in ambito SAT dell'approccio a piani di taglio puro in MIP. Come l'approccio a piani di taglio, non è in genere lo stato dell'arte per quanto riguarda la risoluzione pratica dei problemi SAT, che viene invece ottenuta applicando varianti dell'algoritmo DPLL.

Le implementazioni dei risolutori SAT implementano solo una forma ristretta di risoluzione, detta *unit resolution*, in cui almeno una delle clausole generatrici è una clausola unitaria (cioè

Algorithm 1: Schema generale algoritmo di risoluzione.

Sia S l'insieme di clauses di partenza e $S' = S$
while S' *contiene due clauses con risolvente* R *non implicato da clausole in* S' **do**
  Rimuovi da S' tutte le clauses implicate da R
  $S' \leftarrow S' \cup \{R\}$
end
S è soddisfacibile $\Leftrightarrow$ S' non contiene la clause vuota.

costituita da un solo literal). Tale forma ristretta di risoluzione è ampiamente diffusa in quanto può essere applicata in modo particolarmente efficiente (la complessità è lineare nel numero di clausole, ed esistono tecniche implementative molto efficienti, come i *watch literals*) e non aumenta il numero di clausole del problema (il risolvente R implica sempre almeno una delle due clausole generatrici).

A differenza del metodo di risoluzione generale, unit resolution *non* è un metodo di inferenza completo, cioè potrebbe non derivare la clausola vuota anche quando il sistema non è soddisfacibile. Ad esempio il seguente sistema

$$x_1 \vee x_2$$

$$\neg x_1 \vee x_2$$

$$x_1 \vee \neg x_2$$

$$\neg x_1 \vee \neg x_2$$

è impossibile, ma unit resolution non deriva niente, in quanto nessuna clause è unitaria.

Per alcuni casi particolari però, unit resolution è sufficiente. Ad esempio, un insieme di clausole Horn è soddisfacibile se e solo se unit resolution genera la clausola vuota.

4 Algoritmo DPLL

Lo schema DPLL è un algoritmo di ricerca ad albero, in cui ogni branching consiste nel fissare il valore di verità di un literal. Le caratteristiche principali del metodo sono:

- le uniche tecniche di inferenza usate sono *unit resolution* e *pure literal elimination*, cioè il fissaggio dei literal che compaiono in tutte le clause con lo stesso segno. Si noti come fissare una variabile sia equivalente ad aggiungere una clause unitaria al problema (x_i nel branch di sinistra e $\neg x_i$ in quello di destra) e quindi unit resolution è naturalmente adatta a tale scenario.
- l'albero viene visitato in depth-first-search (DFS), e questo garantisce una occupazione di memoria lineare nella dimensione del problema.

Le implementazioni stato dell'arte dell'algoritmo usano ulteriori tecniche quali *conflict learning* e *restart*.