2.8.5 RESOLUÇÃO

Newton José Vieira

UFMG

18 de setembro de 2008

Observações iniciais

- DPLL pode ser visto como um refinamento de resolução.
- A maioria dos sistemas para processamento de conhecimento expresso em lógica de predicados de primeira ordem é baseada em resolução.
- Para lógica proposicional resolução é mais simples.

Sistema dedutivo de **Resolução**:

- Linguagem: fórmulas na FNC.
- Sem axiomas.
- Uma única regra de inferência: a regra de resolução.

A regra de resolução

Regra de resolução:

$$\frac{C_1 \quad C_2}{R}$$

onde:

- C_1, C_2 e R são cláusulas;
- há um literal l tal que $l \in C_1$ e $\bar{l} \in C_2$;
- $R = C_1 - \{l\} \cup C_2 - \{\bar{l}\}$. Tal cláusula é denominada **resolvente**.

Note que $\{C_1, C_2\} \models C_1 - \{l\} \cup C_2 - \{\bar{l}\}$.

Uma pequena dedução em resolução

Todo resolvente obtido de B é consequência lógica de B.

Exemplo: $\{\neg p \vee q, p\} \models q$, pois:

1. $\neg p \vee q$ (B)
2. p (B)
3. q (res 1,2)

Observações

- Se C_1 contém apenas l e C_2 apenas $\bar{l}$ o resolvente é $\bot$!
- Uma dedução de $\bot$ a partir de um conjunto de cláusulas, B, é dita ser uma refutação de B.
- Há uma refutação de B se, e somente se B é insatisfatível.

Um exemplo de refutação

Seja $B = \{p \vee \neg q, \neg p \vee \neg q, q\}$. Uma refutação de B:

1. $p \vee \neg q$ (B)
2. $\neg p \vee \neg q$ (B)
3. q (B)
4. $\neg q$ (1,2)
5. $\bot$ (3,4)

Logo, B é insatisfatível.

Completude de resolução

O sistema dedutivo de resolução não é completo com relação a consequência lógica:

$\emptyset \nvdash p \vee \neg p$.

O sistema dedutivo de resolução é refutacionalmente completo:

se um conjunto de cláusulas B é insatisfatível, então $B \vdash \bot$.

Correção e completude de resolução:

Um conjunto de cláusulas B é insatisfatível se, se somente se $B \vdash \bot$.

$H \models \alpha$?

Para se provar que $H \models \alpha$:

- Provar que $H \cup \{\neg\alpha\}$ é insatisfatível assim:
  1. obter cláusulas equivalentes a $H \cup \{\neg\alpha\}$;
  2. obter uma refutação.

Exemplo

Seja H o conjunto das fórmulas:

1. inoc $\vee$ culp (o mordomo é inocente ou culpado)
2. inoc $\rightarrow$ tmen (se o mordomo é inocente, a testemunha mentiu)
3. cump $\rightarrow \neg$tmen (se o mordomo tem um cúmplice, a testemunha não mentiu)
4. cump (o mordomo tem um cúmplice)

Para mostrar que $H \models$ culp, obtém-se primeiro o conjunto de cláusulas B:

1. inoc $\vee$ culp (de 1)
2. $\neg$inoc $\vee$ tmen (de 2)
3. $\neg$cump $\vee \neg$tmen (de 3)
4. cump (de 4)

e também o conjunto de cláusulas nc a partir da negação da consulta:

5. $\neg$culp (da negação da consulta)

Segue uma dedução de $\perp$ a partir de $B \cup$ nc.

Refutação de $B \cup$ nc

Dedução de $\perp$ a partir de $B \cup$ nc:

1. inoc $\vee$ culp (B)
2. $\neg$inoc $\vee$ tmen (B)
3. $\neg$cump $\vee \neg$tmen (B)
4. cump (B)
5. $\neg$culp (nc)
6. $\neg$tmen (4,3)
7. $\neg$inoc (6,2)
8. culp (7,1)
9. $\perp$ (8,5)

Logo, $B \cup$ nc é insatisfatível e, portanto, $H \models$ culp.

Algumas observações

- Todas as fórmulas devem ser "transformadas" em cláusulas.
- Para isto, pode-se usar os algoritmos já vistos para FNC.
- Embora a transformação possa ser exponencial no pior caso, em muitas situações isso não constitui empecilho:
  - cada fórmula (da base de conhecimento e da negação da consulta) pode ser transformada isoladamente;
  - a transformação de cada fórmula inserida na base de conhecimento é feita uma única vez.

Resolução por saturação

O algoritmo mais simples para tentar encontrar uma refutação a partir de um conjunto de cláusulas B (ou determinar que não existe uma) é o de saturação:

```
proc res-satura(conjunto de cláusulas B) retorna {verdadeiro, falso}:
    conjunto de cláusulas N, R;
    N := B;
    enquanto N ≠ ∅  e  ⊥ ∉ N faça
        R := {resolventes de C1 e C2 | C1 ∈ N e C2 ∈ B};
        N := R − B; B := B ∪ N
    fimenquanto;
    retorne ⊥ ∈ N
fim res-satura
```

Esse algoritmo é extremamente ineficiente, como mostra o próximo exemplo.

- N_0: valor inicial de N; N_1: valor seguinte, no final do corpo do enquanto.
- N_2: próximo valor, na segunda iteração do enquanto, e assim por diante.

Algoritmo de saturação/exemplo

Refutando $\{p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q\}$, inicialmente N recebe:

N_0: 1. $p \vee q$ (B)
2. $p \vee \neg q$ (B)
3. $\neg p \vee q$ (B)
4. $\neg p \vee \neg q$ (B)

Ao final da primeira iteração do corpo do **enquanto**, tem-se as novas cláusulas:

N_1: 5. p (1,2)
6. q (1,3)
7. $q \vee \neg q$ (1,4)(2,3)
8. $p \vee \neg p$ (1,4)(2,3)
9. $\neg q$ (2,4)
10. $\neg p$ (3,4)

Em seguida, gera 22 resolventes, só que 20 deles idênticos a cláusulas de $N_0 \cup N_1$; e $\perp$ é gerada duas vezes, resolvendo as cláusulas 5 e 10 e resolvendo 6 e 9. Como $N_2 = \{\perp\}$, o algoritmo retorna **verdadeiro**.

Refinamentos no algoritmo de saturação

Eliminação de cláusulas sem afetar completude:

- **Eliminação de tautologias**:

  se C é tautológica, $B \cup \{C\}$ é insatisfatível se e somente se B é insatisfatível.

- **Eliminação de cláusulas puras**:

  se $l \in C$ e $\bar{l}$ não ocorre em B, $B \cup \{C\}$ é insatisfatível se e somente se B é insatisfatível.

- **Eliminação de cláusulas subjugadas**. A cláusula C_1 **subjuga** C_2 se, e somente se, $C_1 \subseteq C_2$.

  para quaisquer cláusulas C_1 e C_2 tais que $C_1 \in B$, $C_2 \subset C_1$ e $B \models C_2$, B é insatisfatível se e somente se $(B - \{C_1\}) \cup \{C_2\}$ é insatisfatível.

Exemplo/eliminação de tautologias e subjugação

No algoritmo de saturação:

Se uma cláusula, resolvente ou não, é tautologia ou subjugada, é descartada.

Refutação de $\{p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q\}$:

N_0: 1. $p \vee q$ (B)
2. $p \vee \neg q$ (B)
3. $\neg p \vee q$ (B)
4. $\neg p \vee \neg q$ (B)

N_1: 5. p (1,2) [subjuga 1 e 2]
6. $\neg p$ (3,4) [subjuga 3 e 4]

N_2: 7. $\perp$ (5,6)

O algoritmo pode gerar algumas tautologias, dependendo da ordem de geração dos resolventes (como $q \vee \neg q$ (1,4)), que são imediatamente descartadas.

Uma estratégia

Estratégia **conjunto de suporte**:

Uma das premissas deve estar no conjunto de suporte S. Inicialmente, ele é um subconjunto de B, normalmente o conjunto de cláusulas relativas à negação da consulta. Depois inclui também os resolventes.

Essa estratégia é completa se, e somente se, $B - S$ é satisfatível.

(A insatisfabilidade tem que ter relação com S.)

Usando a negação da consulta como conjunto de suporte

- Há completude se o conjunto original é consistente.
- É uma forma de "isolar" uma eventual inconsistência da base de conhecimento, não permitindo uma refutação fundamentada na inconsistência da base de conhecimento.
- Tem-se um "comportamento paraconsistente": uma inconsistência nunca é usada como justificativa, a menos que já esteja presente na própria negação da consulta (neste último caso, a fórmula correspondente à consulta seria tautológica).

Exemplos de uso da negação da consulta como conjunto de suporte

Seja $H = \{p \rightarrow q, p, \neg q\}$, inconsistente, que leva ao conjunto B:

1. $\neg p \vee q$ (B)
2. p (B)
3. $\neg q$ (B)

À consulta $H \models p$, corresponde a dedução imediata de $\perp$:

4. $\neg p$ (nc)
5. $\perp$ (4,2)

$\Rightarrow$ p segue de H, não porque H é inconsistente, mas porque está afirmada em B.

Exemplos de uso da negação da consulta como conjunto de suporte (cont.)

Analogamente, $H \models \neg p$, pois há a refutação:

1. $\neg p \vee q$ (B)
2. p (B)
3. $\neg q$ (B)
4. p (nc)
5. q (4,1)
6. $\perp$ (5,3)

$\Rightarrow$ A resposta afirmativa não apela para a inconsistência: $\neg p$ segue de 1 e 3.

De forma similar, refutações que mostram

$H \models p \rightarrow q$, $H \models p \rightarrow \neg q$ etc.

nunca apelam para a inconsistência de H.

Exemplos de uso da negação da consulta como conjunto de suporte (cont.)

1. $\neg p \vee q$ (B)
2. p (B)
3. $\neg q$ (B)

A consulta $H \models r$ recebe a resposta negativa:

$B \cup \{\neg r\}$ não é refutável utilizando-se $\neg r$ como conjunto de suporte.

A consulta $H \models r \rightarrow r$ recebe a resposta positiva:

utilizando-se $\{r, \neg r\}$ como conjunto de suporte obtém-se $\perp$ imediatamente.

Outra estratégia

Estratégia linear: só produz deduções lineares.

Uma dedução linear de uma cláusula C_n, a partir do conjunto de cláusulas B, começando na cláusula C_0, é tal que:

- $C_0 \in B$;
- uma das premissas para obtenção do resolvente C_i, $i \geq 1$, é C_{i-1}.
- a outra premissa para obtenção de C_i é:
  - cláusula de B ou
  - um resolvente prévio.

Em uma dedução linear de entrada:

- a segunda premissa é sempre uma cláusula de B

Dedução linear é completa, mas linear de entrada não é.

Exemplo de dedução linear

Uma dedução linear de $\bot$, refutação de $B = \{p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q\}$, seria (escolhendo a cláusula 4 como inicial):

1. $p \vee q$ (B)
2. $p \vee \neg q$ (B)
3. $\neg p \vee q$ (B)
4. $\neg p \vee \neg q$ (B)
5. $\neg q$ (4,2)
6. p (5,1)
7. q (6,3)
8. $\bot$ (7,5)

Não existe refutação linear de entrada para B:

Todas as cláusulas de B têm dois literais. Resolvendo-se uma delas com uma outra obtém-se um resolvente de um ou mais literais; este, portanto, não pode ser a cláusula vazia.

Estratégia linear com conjunto de suporte

- Na estratégia linear qualquer cláusula pode ser escolhida como inicial.
  $\Rightarrow$ Importante para completude.
- Combinando com a de conjunto de suporte:

  apenas o conjunto de suporte inicial supre cláusulas para iniciar uma dedução.

Prolog usa dedução linear de entrada

- Estratégia linear de entrada admite uma implementação eficiente, como ilustra a linguagem de programação em lógica Prolog.
- Ela é completa porque um programa em Prolog consta apenas de cláusulas de Horn.
- Uma cláusula de Horn contém no máximo um literal positivo.

Em Prolog:

- Um programa consta de cláusulas com um literal positivo:

  É impossível que seja inconsistente, pois é satisfatível para a interpretação que satisfaz todo literal positivo.
- Consulta: conjunção de literais positivos; .
  negando: cláusula com apenas literais negativos
- Tal cláusula é usada como conjunto (unitário) de suporte.
- Todos os resolventes têm apenas literais negativos (ou é $\bot$).

Exemplo de processamento de programa em Prolog

Seja o seguinte conjunto de cláusulas, B, com um literal positivo cada uma:

1. $p \vee \neg q \vee \neg r$
2. $q \vee \neg s$
3. r
4. s

As cláusulas não unitárias podem ser lidas como condicionais:

$q \wedge r \rightarrow p$ e $s \rightarrow q$.

Negando-se a consulta $p \wedge s$ (para saber se $B \models p \wedge s$), obtém-se:

5. $\neg p \vee \neg s$ (nc)

Exemplo de processamento de programa em Prolog (cont.)

Segue uma dedução linear de entrada de $\perp$:

1.	$p \vee \neg q \vee \neg r$	(B)
2.	$q \vee \neg s$	(B)
3.	r	(B)
4.	s	(B)
5.	$\neg p \vee \neg s$	(nc)
6.	$\neg q \vee \neg r \vee \neg s$	(5,1)
7.	$\neg r \vee \neg s$	(6,2)
8.	$\neg s$	(7,4)
9.	$\perp$	(8,3)

FIM