Lógica
Aula 12

Renata Wassermann

renata@ime.usp.br

2020

O problema SAT

"Dada uma fórmula, decidir se ela é satisfatível."

O problema SAT

"Dada uma fórmula, decidir se ela é satisfatível."

- SAT é NP-completo [Cook 1971]

O problema SAT

"Dada uma fórmula, decidir se ela é satisfatível."

- SAT é NP-completo [Cook 1971]
- Competição desde 2002:
  `http://www.satcompetition.org/`

NaiveSAT

Entrada: φ, em CNF
Saída: v, se $v(\varphi) = T$; "não", caso contrário

```
Para toda valoração v sobre os átomos de φ faça:
     se v(φ) = T então devolva v
Devolva ''não''
```

Resolução

Única regra:

$$\frac{\phi \vee p \qquad \psi \vee \neg p}{\phi \vee \psi}$$

Robinson, J. Alan (1965). "A Machine-Oriented Logic Based on the Resolution Principle".

Resolução

Única regra:

$$\frac{\phi \vee p \qquad \psi \vee \neg p}{\phi \vee \psi}$$

Robinson, J. Alan (1965). "A Machine-Oriented Logic Based on the Resolution Principle".

Completa para *refutação*!

Refutação por resolução

Raciocínio por contradição:

1. Para provar $\varphi_1, \ldots, \varphi_n \vdash \psi$, transforme $\chi = \varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\psi$ em CNF.

Refutação por resolução

Raciocínio por contradição:

1. Para provar $\varphi_1, \ldots, \varphi_n \vdash \psi$, transforme $\chi = \varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\psi$ em CNF.
2. Seja C o conjunto de cláusulas obtido.

Refutação por resolução

Raciocínio por contradição:

1. Para provar $\varphi_1, \ldots, \varphi_n \vdash \psi$, transforme $\chi = \varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\psi$ em CNF.
2. Seja C o conjunto de cláusulas obtido.
3. Aplique resolução ao conjunto quantas vezes for possível.

Refutação por resolução

Raciocínio por contradição:

1. Para provar $\varphi_1, \ldots, \varphi_n \vdash \psi$, transforme $\chi = \varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\psi$ em CNF.
2. Seja C o conjunto de cláusulas obtido.
3. Aplique resolução ao conjunto quantas vezes for possível.
4. Se em algum momento gerar uma cláusula vazia, devolva "sequente verdadeiro" (χ não é SAT).

Refutação por resolução

Raciocínio por contradição:

1. Para provar $\varphi_1, \ldots, \varphi_n \vdash \psi$, transforme $\chi = \varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\psi$ em CNF.
2. Seja C o conjunto de cláusulas obtido.
3. Aplique resolução ao conjunto quantas vezes for possível.
4. Se em algum momento gerar uma cláusula vazia, devolva "sequente verdadeiro" (χ não é SAT).
5. Senão, devolva "sequente falso" (χ é SAT).

Refutação por resolução

Raciocínio por contradição:

1. Para provar $\varphi_1, \ldots, \varphi_n \vdash \psi$, transforme $\chi = \varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\psi$ em CNF.
2. Seja C o conjunto de cláusulas obtido.
3. Aplique resolução ao conjunto quantas vezes for possível.
4. Se em algum momento gerar uma cláusula vazia, devolva "sequente verdadeiro" (χ não é SAT).
5. Senão, devolva "sequente falso" (χ é SAT).

Refutação por resolução

Raciocínio por contradição:

1. Para provar $\varphi_1, \ldots, \varphi_n \vdash \psi$, transforme $\chi = \varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\psi$ em CNF.
2. Seja C o conjunto de cláusulas obtido.
3. Aplique resolução ao conjunto quantas vezes for possível.
4. Se em algum momento gerar uma cláusula vazia, devolva "sequente verdadeiro" (χ não é SAT).
5. Senão, devolva "sequente falso" (χ é SAT).

Exemplo:

- $p \to q, r \to s \vdash (p \vee r) \to (q \vee s)$

O Algoritmo DPLL

Davis & Putnam, 1960; Davis, Longemann & Loveland, 1962

- Evita construir valorações completas.

O Algoritmo DPLL

Davis & Putnam, 1960; Davis, Longemann & Loveland, 1962

- Evita construir valorações completas.
- *Símbolo puro:* Aparece só positivo ou só negativo.

O Algoritmo DPLL

Davis & Putnam, 1960; Davis, Longemann & Loveland, 1962

- Evita construir valorações completas.
- *Símbolo puro:* Aparece só positivo ou só negativo.
- *Propagação Unitária:* Preferência por cláusulas com um só literal.

O Algoritmo DPLL

Davis & Putnam, 1960; Davis, Longemann & Loveland, 1962

- Evita construir valorações completas.
- *Símbolo puro:* Aparece só positivo ou só negativo.
- *Propagação Unitária:* Preferência por cláusulas com um só literal.

O Algoritmo DPLL

Davis & Putnam, 1960; Davis, Longemann & Loveland, 1962

- Evita construir valorações completas.
- *Símbolo puro:* Aparece só positivo ou só negativo.
- *Propagação Unitária:* Preferência por cláusulas com um só literal.

(Tseitin, 1966: DPLL é exponencial)

O Algoritmo DPLL

1. Começa com modelo vazio.

O Algoritmo DPLL

1. Começa com modelo vazio.
2. Se alguma cláusula é F, devolve F.

O Algoritmo DPLL

1. Começa com modelo vazio.
2. Se alguma cláusula é F, devolve F.
3. Se todas as cláusulas são T, devolve T.

O Algoritmo DPLL

1. Começa com modelo vazio.
2. Se alguma cláusula é F, devolve F.
3. Se todas as cláusulas são T, devolve T.
4. Remove todas as cláusulas com símbolo puro.

O Algoritmo DPLL

1. Começa com modelo vazio.
2. Se alguma cláusula é F, devolve F.
3. Se todas as cláusulas são T, devolve T.
4. Remove todas as cláusulas com símbolo puro.
5. Quando tem cláusulas unitárias, acrescenta ao modelo.

O Algoritmo DPLL

1. Começa com modelo vazio.
2. Se alguma cláusula é F, devolve F.
3. Se todas as cláusulas são T, devolve T.
4. Remove todas as cláusulas com símbolo puro.
5. Quando tem cláusulas unitárias, acrescenta ao modelo.
6. Se não, **escolhe** literal l e acrescenta.

O Algoritmo DPLL

1. Começa com modelo vazio.
2. Se alguma cláusula é F, devolve F.
3. Se todas as cláusulas são T, devolve T.
4. Remove todas as cláusulas com símbolo puro.
5. Quando tem cláusulas unitárias, acrescenta ao modelo.
6. Se não, **escolhe** literal l e acrescenta.
7. Apaga as cláusulas contendo l.

O Algoritmo DPLL

1. Começa com modelo vazio.
2. Se alguma cláusula é F, devolve F.
3. Se todas as cláusulas são T, devolve T.
4. Remove todas as cláusulas com símbolo puro.
5. Quando tem cláusulas unitárias, acrescenta ao modelo.
6. Se não, **escolhe** literal l e acrescenta.
7. Apaga as cláusulas contendo l.
8. Apaga o oposto de l das outras cláusulas

O Algoritmo DPLL

1. Começa com modelo vazio.
2. Se alguma cláusula é F, devolve F.
3. Se todas as cláusulas são T, devolve T.
4. Remove todas as cláusulas com símbolo puro.
5. Quando tem cláusulas unitárias, acrescenta ao modelo.
6. Se não, **escolhe** literal l e acrescenta.
7. Apaga as cláusulas contendo l.
8. Apaga o oposto de l das outras cláusulas
9. Repete até achar contradição, neste caso, refaz a escolha, ou não ter mais o que aplicar.

O Algoritmo DPLL

1. Começa com modelo vazio.
2. Se alguma cláusula é F, devolve F.
3. Se todas as cláusulas são T, devolve T.
4. Remove todas as cláusulas com símbolo puro.
5. Quando tem cláusulas unitárias, acrescenta ao modelo.
6. Se não, **escolhe** literal l e acrescenta.
7. Apaga as cláusulas contendo l.
8. Apaga o oposto de l das outras cláusulas
9. Repete até achar contradição, neste caso, refaz a escolha, ou não ter mais o que aplicar.
10. Se o modelo tem contradição, a fórmula não é SAT. Se não, é uma valoração.

Exercício 1

Dada a tabela verdade abaixo, escreva uma fórmula equivalente a φ usando as variáveis proposicionais p, q e r e os conectivos da lógica proposicional. Prove, usando resolução, que $\varphi, r \vdash p$

p	q	r	φ
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	T

Exercício 2

(a) Prove o seguinte teorema, usando dedução natural:

$$(p \rightarrow q) \rightarrow ((\neg p \rightarrow q) \rightarrow q)$$

Exercício 2

(a) Prove o seguinte teorema, usando dedução natural:

$$(p \rightarrow q) \rightarrow ((\neg p \rightarrow q) \rightarrow q)$$

(b) Prove usando resolução.

Exercício 3

Determinar se o seqüente abaixo é válido ou não pelo *método que você achar mais conveniente*. Em caso positivo, dar uma prova pelo método de Dedução Natural; caso contrário, exibir uma valoração que refuta o seqüente.

$\varphi \vdash \psi$ onde $\varphi = \begin{array}{l}(p_{01} \vee p_{02}) \wedge \\ (p_{11} \vee p_{12}) \wedge \\ (p_{21} \vee p_{22})\end{array}$ e $\psi = \begin{array}{l}(p_{01} \wedge p_{11}) \vee \\ (p_{02} \wedge p_{12}) \vee \\ (p_{11} \wedge p_{21}) \vee \\ (p_{12} \wedge p_{22})\end{array}$.