Satisfação Booleana

Prof^a Jerusa Marchi

Departamento de Informática e Estatística

Universidade Federal de Santa Catarina

e-mail: jerusa.marchi@ufsc.br

Satisfação Booleana

- Também chamado de Satisfazibilidade ou Satisfatibilidade Booleana
- Do inglês *Satisfiability*
- Problema da Satisfação Booleana
  - Importante problema para o estudo da tratabilidade (o que é possível resolver em um computador utilizando um tempo "razoável"[a])

[a] tempo polinomial

Sumário

- Parte 1 - O que é o problema SAT?
  - Lógica Booleana
  - Formas Normais Canônicas
  - O problema SAT
- Parte 2 - Classes de Complexidade e a Importância do SAT
  - Classes P e NP
  - Teorema de Cook-Levin

Lógica Booleana

- Descreve proposições simples
  - Está chovendo.
  - O carro é branco.
- Uso de conectivos lógicos para construir proposições compostas
  - A vassoura não está no canto ou está chovendo

Lógica Booleana

- As proposições simples podem ser representadas por meio de variáveis lógicas, como p, q, r,... ou ainda p_1, p_2,...
- A sintaxe da lógica booleana é a seguinte:
  - Símbolos proposicionais: $p_1, p_2, ..., p_n$ (alfabeto infinito, mas contável)
  - Constantes: verdadeiro $(\top)$ e falso $(\bot)$
  - Conectivos lógicos: o conectivo pode ser unário - negação $(\neg)$ ou binários - e $(\wedge)$, ou $(\vee)$ e implicação $(\rightarrow)$
  - Parênteses.

Lógica Booleana

- A sintaxe da lógica proposicional é definida em linguagem formal BNF (Backus-Naur Formalism) como:

$$
\begin{aligned}
\langle \text{Sentença} \rangle &::= \langle \text{Sentença Atômica} \rangle \mid \langle \text{Sentença Complexa} \rangle \\
\langle \text{Sentença Atômica} \rangle &::= \top \mid \bot \\
&\quad \mid p_1 \mid p_2 \mid \ldots \mid p_n \\
\langle \text{Sentença Complexa} \rangle &::= (\langle \text{Sentença} \rangle) \\
&\quad \mid \langle \text{Sentença} \rangle \ \langle \text{Conectivo} \rangle \ \langle \text{Sentença} \rangle \\
&\quad \mid \neg \langle \text{Sentença} \rangle \\
\langle \text{Conectivo} \rangle &::= \vee \mid \wedge \mid \rightarrow
\end{aligned}
$$

Lógica Booleana

- A lógica booleana lida com dois valores: verdadeiro ou falso
- A semântica da lógica booleana é dada pela interpretação da fórmula de acordo com o valor verdade dos símbolos proposicionais e com a aplicação dos conectivos lógicos
  - Tabelas Verdade

p_1	$\neg p_1$	p_2	$p_1 \wedge p_2$	$p_1 \vee p_2$	$p_1 \rightarrow p_2$
F	V	F	F	F	V
F	V	V	F	V	V
V	F	F	F	V	F
V	F	V	V	V	V

Lógica Booleana

- A construção da tabela verdade apresenta o conjunto de **interpretações** possíveis para uma dada fórmula ou proposição complexa
- Se para todas as interpretações, o valor da fórmula é verdadeiro, diz-se que esta fórmula é *teorema* ou uma *tautologia*
- Se há pelo menos uma interpretação que torne a fórmula verdadeira, então a fórmula é dita *válida* ou *satisfazível* (do inglês satisfiable)
- Se não existe uma atribuição de valores verdade que tornem a fórmula verdadeira, então esta fórmula é *insatisfazível* ou uma *contradição*.

Lógica Booleana

- Podemos representar as fórmulas em lógica proposicional em uma forma normalizada utilizando unicamente os operadores $\{\neg, \vee, \wedge\}$
- Formas Normais Canônicas:
  - Simplificar fórmulas complexas
  - Em geral, primeira etapa dos procedimentos de demonstração automática de teoremas

Formas normais canônicas

- Forma normal conjuntiva (ou forma clausal) - CNF
  - conjunção de disjunções

$$(p \vee q) \wedge (\neg q \vee r)$$

  - Formalmente
    - a fórmula está na forma

$$C_1 \wedge C_2 \wedge \ldots C_n$$

    - onde cada cláusula C_i é uma disjunção de literais

$$L_1 \vee L_2 \vee \ldots L_n$$

    - onde cada literal L_i é um símbolo de predicado ou sua negação

Formas normais canônicas

- Forma normal disjuntiva (ou forma clausal dual) - DNF
  - disjunção de conjunções

$$(r \wedge \neg p) \vee (q \wedge r)$$

  - Formalmente
    - a fórmula esta na forma

$$D_1 \vee D_2 \vee \ldots D_n$$

    - onde cada cláusula D_i é uma conjunção de literais

$$L_1 \wedge L_2 \wedge \ldots L_n$$

Formas normais canônicas

- As fórmulas são transformadas utilizando as relações de equivalência

$$A \rightarrow B \equiv \neg A \vee B \text{ (eliminação da implicação)}$$

$$\neg\neg A \equiv A \text{ (eliminação da dupla negação)}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \text{ (lei de De Morgan)}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \text{ (lei de De Morgan dual)}$$

$$A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C) \text{ (distributividade)}$$

$$A \wedge (B \vee C) \Leftrightarrow (A \wedge B) \vee (A \wedge C) \text{ (distributividade)}$$

- As representações em formas normais são equivalentes às fórmulas originais

$$W \equiv CNF_W \equiv DNF_W$$

Formas normais canônicas

- Algoritmo Forma Normal Conjuntiva
  - Eliminar todas as ocorrências de $A \rightarrow B$ em W, substituindo-as por $\neg A \vee B$.
  - Reduzir o escopo das negações de maneira que só restem negações aplicadas a fórmulas atômicas. Para isto usar as regras:

$$\neg(A \vee B) \Rightarrow (\neg A \wedge \neg B)$$
$$\neg(A \wedge B) \Rightarrow (\neg A \vee \neg B)$$
$$\neg(\neg(A)) \Rightarrow A$$

  - Converter a fórmula para a forma de uma conjunção de disjunções usando a propriedade distributiva do operador $\vee$ sobre o operador $\wedge$:

$$A \vee (B \wedge C) \Rightarrow (A \vee B) \wedge (A \vee C).$$

Forma normal conjuntiva

- Exemplo 1:
  - “Se chove então eu não saio. Mas, se não chove, eu saio e eu tomo sorvete. Eu saio e tomo sorvete.”

representamos como:	transformando em CNF:
$c \rightarrow \neg s$	$\neg c \vee \neg s$
$\neg c \rightarrow s \wedge t$	$c \vee s$
$s \wedge t$	$c \vee t$
	s
	t

Forma normal conjuntiva

- Exemplo 2:
  - $(p \wedge q) \rightarrow \neg r$

$$\neg(p \wedge q) \vee \neg r$$

$$\neg p \vee \neg q \vee \neg r$$

Forma normal conjuntiva

- Exemplo 3:
  - $p \rightarrow (q \wedge \neg(r \vee s))$

$$\neg p \vee (q \wedge \neg(r \vee s))$$

$$\neg p \vee (q \wedge \neg r \wedge \neg s)$$

$$(\neg p \vee q) \wedge (\neg p \vee \neg r) \wedge (\neg p \vee \neg s)$$

Problema da Satisfação Booleana

- O problema da satisfazibilidade ou satisfação (SAT) em formas normais conjuntivas consiste em determinar se a fórmula

$$W_{CNF} = C_1 \wedge C_2 \wedge, ..., C_n$$

é satisfazível (do inglês “satisfiable”), ou seja, se há uma combinação de valores para as variáveis tal que a fórmula seja avaliada como *verdadeira*.

Problema da Satisfação Booleana

- Exemplos:

$$W_1 = \{(\neg c \vee \neg s), (c \vee s), (c \vee t), (s), (t)\}$$

W_1 é satisfazível fazendo-se $c = \bot$, $s = \top$, $t = \top$

$$W_2 = \{(\neg p \vee \neg q \vee \neg r)\}$$

W_2 é satisfazível fazendo-se $p = \bot$ ou $q = \bot$ ou $r = \bot$

$$W_3 = \{(\neg p \vee q), (\neg p \vee \neg r), (\neg p \vee \neg s)\}$$

W_3 é satisfazível fazendo-se $p = \bot$

Problema da Satisfação Booleana

- Mais um Exemplo: Dada a fórmula:

$W_4 = \{(p_1 \vee p_2 \vee p_3), (\neg p_1 \vee p_2), (\neg p_2 \vee p_3), (\neg p_3 \vee p_1), (\neg p_1 \vee \neg p_2 \vee \neg p_3)\}$

W_4 é satisfazível?

Classes de Complexidade

- Em Computação a classificação de um problema é de suma importância para determinar estratégias de solução
  - Se o problema é simples e "algoritmizável"com baixo custo, implementa-se a solução
  - Se o problema cresce de forma exponencial, técnicas heurísticas ou estocásticas precisam ser aplicadas

Complexidade de Tempo

função	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 60$
n	.00001s	.00002s	.00003s	.00004s	.00005s	.00006s
n^2	.0001s	.0004s	.0009s	.0016s	.0025s	.0036s
n^3	.001s	.008s	.027s	.064s	.125s	.216s
n^5	.1s	3.2s	24.3s	1.7 min	5.2 min	13.0 min
2^n	.001s	1.0s	17.9 min	12.7 dias	35.7 anos	366 séculos
3^n	.059s	58 min	6.5 anos	3855 séculos	2×10^8 séculos	1.3×10^{13} séculos

Classes de Problemas

- Um problema é dito tratável se é decidível por Máquinas de Turing em **tempo polinomial** em relação ao tamanho da entrada
  - MT decidem linguagens respondendo **sim** ou **não**
  - É preciso então, dado um problema, modificar a sua formulação para se torne um **Problema de Decisão**

Problema de Decisão

- SAT como um problema de Decisão
  - dado um conjunto de cláusulas $W = \{C_1, ..., C_n\}$ onde C_i é uma disjunção de literais, há alguma atribuição de valores verdade que torne a fórmula verdadeira?

Classes de Complexidade

- Contudo tanto máquina de Turing Determinística quanto máquinas de Turing não Determinísticas podem computar em tempo polinomial
  - P $\times$ NP

Classes de Complexidade

- Classe $\mathcal{P}$
  - Classe de problemas para os quais há uma máquina de Turing Determinística $M = (K, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ que computa em até $p(n)$ passos. Onde p é um polinômio e n é o tamanho da entrada.

2-SAT está na Classe $\mathcal{P}$

- Um algoritmo que computa 2-SAT em tempo linear foi apresentado em 1979 por B. Aspvall, M. F. Plass, and R. E. Tarjan;
- O algoritmo consiste em representar a fórmula W de entrada como um grafo dirigido G, construído como segue:
  - O conjunto de vértices de G consiste de $2n$ vértices que representam as variáveis de W e suas negações
  - Cada cláusula $(a \vee b)$ em W pode ser vista como duas implicaçãoes: $\neg a \rightarrow b$ e $\neg b \rightarrow a$, produzindo arestas $(\neg a, b)$ e $(\neg b, a)$ no grafo
  - W é insatisfazível sse há uma variável x tal que G tem um ciclo contendo x e $\neg x$
  - para computar, basta computar as componentes fortementes conexas do grafo.

2-SAT está na Classe $\mathcal{P}$

- Há outro método baseado em resolução que toma $O(n^2)$
  - 1. procure por cláusulas unitárias ou por alguma variável livre (que aparece unicamente na forma positiva ou negativa)
    - a. Faça $\phi(V) = \top$
    - b. Omita da fórmula tais cláusulas, pois todas elas já estão satisfeitas
  - 2. Procure por cláusulas que tenham o literal oposto ao das cláusulas unitárias, remova estes literais das cláusulas pois sua interpretação é falsa ($\phi(\overline{V}) = \bot$), caso alguma cláusula fique vazia, retorne falha (a fórmula é insatisfazível)
  - 3. repita o processo (passo 1) até não ser possível realizar mais nenhuma eliminação. Se o conjunto de cláusulas ficou vazio, retorne sucesso. Caso contrário:
    - a. escolha uma variável V qualquer e faça $\phi(V) = \top$, volte ao passo 1.b. Se retornar falha, restaure a fórmula e tente fazer $\phi(V) = \bot$. Se retornar falha, retorne falha (a fórmula é insatisfazível)

Classes de Complexidade

- Classe $\mathcal{NP}$
  - Classe de problemas para os quais há uma máquina de Turing não Determinística $M = (K, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ que computa em até $p(n)$ passos. Onde p é um polinômio e n é o tamanho da entrada.
- Definição Alternativa
  - Classe de problemas que pode ser verificada em tempo polinomial (por uma Máquina de Turing Determinística)

Classe $\mathcal{NP}$

- Recordando: Uma MT não determinística decide uma linguagem L se:
  - Para cada sentença de entrada que não pertença à linguagem L, todas as computações possíveis da MT devem rejeitar tal entrada
  - Para cada sentença de entrada pertencente a L, exige-se que haja pelo menos uma computação que aceite tal entrada
- a computação não deteminística forma uma árvore, onde:
  - os vértices representam configurações e arestas representam os passos
  - escolhas não deteminísticas representam mais de uma aresta partindo de um vértice
  - a altura da árvore corresponde ao tempo (número de passos)

Classe $\mathcal{NP}$

- Toda MT determinística é um caso particular de uma MT não determinística que não possui múltiplas escolhas em seus movimentos, portanto

$$\mathcal{P} \subseteq \mathcal{NP}$$