PROBLEMA DA SATISFABILIDADE

Diego S. Costa, Talles B.de Assunção

Universidade Federal de Roraima (UFRR)
Boa Vista – RR – Brasil

Resumo. *Este artigo descreve o problema de Satisfabilidade Booleana (SAT – Boolean Satifiability problem)*

1. O problema SAT

O problema SAT foi o primeiro problema a ser identificado como NP-completo. O problema consiste em, dada um expressão booleana formada por conectivos AND (∧), OR (∨) e NOT (¬), com n variáveis, verificar se existe alguma solução que torne essa expressão verdadeira. Caso exista, essa expressão e dita como satisfatível.

Exemplos:

Dada a expressão booleana com variáveis x1, x2, x3 e x4, existem valores para x1, x2, x3 e x4 que torne essa expressão verdadeira?

(x1 ∨ x2) ∧ (¬x1 ∧ x3) ∧ (¬x1 ∧ x3 ∧ ¬x4) ∧ (x2 ∨ x4)

Se existir alguma atribuição para x1, x2, x3 e x4 que torne essa expressão verdadeira ela é considerada satisfativel. No caso, é possível notar que a expressão tem solução com x1 = 0, x2 = 1, x3 = 1 e x4 = 0.

Dada a expressão booleana com variáveis x1, x2, x3 e x4, existem valores para x1, x2, x3 e x4 que torne essa expressão verdadeira?

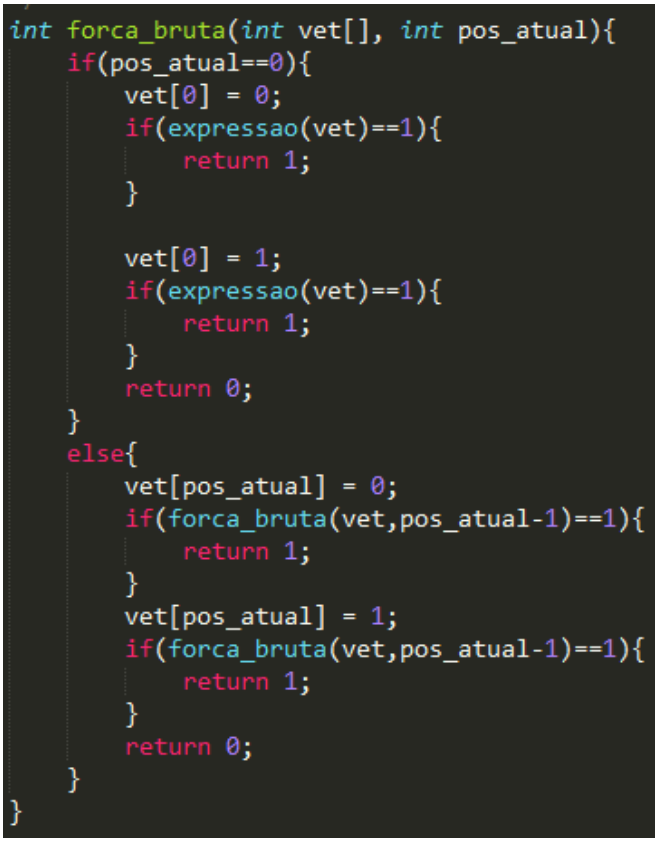
(x1 ∧ ¬x1) ∨ (x2 ∧ ¬x2) ∨ (x3 ∧ ¬x3) ∨ (x4 ∧ ¬x4)

Neste caso, é possível notar que não é possível expressão ser verdadeira com nenhum valor para x1, x2, x3 e x4, então é dito que a expressão é insatisfatível.

2. Versão exata do SAT

O algoritmo da versão exata do SAT, disponível no arquivo sat-exata.c, busca uma solução por meio da força bruta, gerando todas as possíveis combinações de valores para as variáveis da expressão booleana de maneira recursiva, caso encontre uma, encerra a busca e exibe os valores da variáveis que torna a expressão verdadeira. Dada uma expressão booleana para a verificação, é criado um vetor com n posições, onde cada posição é uma variável da expressão, o algoritmo percorre o vetor de trás para frente atribuindo 0 em cada posição. Chegando na primeira posição, nela é atribuido o valor 0 e verifaca se a expressão é verdadeira, caso não seja, o valor atribuido agora é 1 e verifica novamente, se também não for, volta uma posição e atribui 1 nela e percorre o vetor novamente.

Podemos identificar que o pior caso desse algoritmo será se a única solução para a expressão ser verdadeira se todas as variáveis possuirem o valor, pois será a última combinação gerada pelo algoritmo. Outro pior caso também é se não existir solução, pois todas as combinações terão que ser verificadas.



```
int forca_bruta(int vet[], int pos_atual){
    if(pos_atual==0){
        vet[0] = 0;
        if(expressao(vet)==1){
            return 1;
        }

        vet[0] = 1;
        if(expressao(vet)==1){
            return 1;
        }
        return 0;
    }
    else{
        vet[pos_atual] = 0;
        if(forca_bruta(vet,pos_atual-1)==1){
            return 1;
        }
        vet[pos_atual] = 1;
        if(forca_bruta(vet,pos_atual-1)==1){
            return 1;
        }
        return 0;
    }
}
```

Figura 1. Função forca_bruta()

Esse algoritmo possui complexidade $O(2^n)$, cada atribuição de valor e if() da função possui custo 1, assim é possível notar que com uma variável no pior caso, o custo seria O(4), e criar o sistema para calcular a complexidade:

$$\begin{cases} 0 & \text{, para } n = 0 \\ 2T(n-1) + 4 & \text{, para } n > 0 \end{cases}$$

$2T(n-1) + 4$, para n=1

$2(2T(n-2) + 4) + 4 = 4T(n-2) + 12$, para n=2

$4(2T(n-3) + 4) + 12 = 8T(n-3) + 28$, para n=3

$8(2T(n-4) + 4) + 28 = 16T(n-4) + 60$, para n=4

$2^kT(n-k) + 2^{k+2} - 4$

Com k = n,

$2^nT(n-n) + 2^{n+2} - 4 = 2^nT(0) + 2^{n+2} - 4 = 2^{n+2} - 4$

Complexidade $O(2^n)$.

7. References

Bibliographic references must be unambiguous and uniform. We recommend giving the author names references in brackets, e.g. [Knuth 1984], [Boulic and Renault 1991]; or dates in parentheses, e.g. Knuth (1984), Smith and Jones (1999).

The references must be listed using 12 point font size, with 6 points of space before each reference. The first line of each reference should not be indented, while the subsequent should be indented by 0.5 cm.

References

Boulic, R. and Renault, O. (1991) "3D Hierarchies for Animation", In: New Trends in Animation and Visualization, Edited by Nadia Magnenat-Thalmann and Daniel Thalmann, John Wiley & Sons ltd., England.

Dyer, S., Martin, J. and Zulauf, J. (1995) "Motion Capture White Paper", http://reality.sgi.com/employees/jam_sb/mocap/MoCapWP_v2.0.html, December.

Holton, M. and Alexander, S. (1995) "Soft Cellular Modeling: A Technique for the Simulation of Non-rigid Materials", Computer Graphics: Developments in Virtual Environments, R. A. Earnshaw and J. A. Vince, England, Academic Press Ltd., p. 449-460.

Knuth, D. E. (1984), The TeXbook, Addison Wesley, 15th edition.

Smith, A. and Jones, B. (1999). On the complexity of computing. In *Advances in Computer Science*, pages 555–566. Publishing Press.