https://eventos.utfpr.edu.br//sicite/sicite2020

Software para minimização de expressões lógicas utilizando Mapas de Karnaugh

Software for minimum expression generation using Karnaugh Maps

Luana Cristina Guerreiro Campos
luanacampos@alunos.utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Toledo, Paraná, Brasil

Gustavo Henrique Paetzold
ghpaetzold@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Toledo, Paraná, Brasil

RESUMO

Expressões da lógica proposicional são amplamente utilizadas em áreas como circuitos digitais. Torna-se vantajoso reduzir expressões sem alterar o seu valor semântico, uma vez que obter o mesmo resultado com menos componentes torna o processo mais eficiente. Existem alguns métodos de minimização existentes, como o método de Quine-McCluskey e mapas de Karnaugh. Esse trabalho buscou a implementação do método de minimização dos mapas de Karnaugh utilizando a linguagem de programação Python e o *framework* PyQt5. O programa desenvolvido faz minimização de expressões com duas, três e quatro variáveis. Como o *software* tem o objetivo de ser didático, também foi implementada uma função que mostra o mapa de Karnaugh resultante, bem como realça os trechos do mapa que geram a variável de saída realçada.

PALAVRAS-CHAVE: Mapa de Karnaugh. Minimização. Lógica proposicional.

ABSTRACT

Propositional logical expressions are widely used in fields such as digital circuits. It is advantageous to reduce these expressions without altering their meaning, because having the same results with less componentes makes manufacturing more cost-effective. There are some minimization methods, such as the Quine-McCluskey algorithm and Karnaugh Maps. This paper focuses on implementing the Karnaugh Map minimization method using the Python programming language and the PyQt5 *framework*. The software developed minimizes expressions with two, three, and four variables. Since our software was built for educational purposes, the resulting Karnaugh Map is also shown through a visual interface, and it's possible to select an exit variable to highlight its corresponding part in the map.

KEYWORDS: Karnaugh Maps. Minimization. Propositional logic.

Recebido: 19 ago. 2020.
Aprovado: 01 out. 2020.

Direito autoral: Este trabalho está licenciado sob os termos da Licença Creative Commons-Atribuição 4.0 Internacional.

4 EDUCAÇÃO DE QUALIDADE

INTRODUÇÃO

Expressões lógicas, também chamadas de expressões booleanas, são compostas por literais e conectivos lógicos. As variáveis podem assumir dois valores: verdadeiro ou falso. O conjunto de todas as combinações de verdadeiro-falso resulta em uma tabela-verdade (KLOCK, 2003).

Uma das aplicações de expressões da lógica proposicional é na área de circuitos digitais. Circuitos digitais, ou circuitos lógicos, operam de modo que o valor verdadeiro represente algum valor de tensão, enquanto o valor falso é indicado pela ausência dessa tensão. Assim, é possível descrever o circuito em uma expressão lógica. Com o intuito de deixar o processo de fabricação mais eficiente, é conveniente o estudo de métodos de minimização de expressões lógicas para gerar circuitos menores, mas equivalentes ao original (TOCCI, 2007).

Existem algumas formas de reduzir expressões, como o método de Quine McCluskey. Como o propósito é desenvolver uma ferramenta didática e visto que não existem softwares acessíveis, didático e de fácil uso, foi criada uma ferramenta didática em Python que emprega o método de Mapas de Karnaugh. Esse método é gráfico e mais simples que o de Quine McCluskey (GERSTING, 1993).

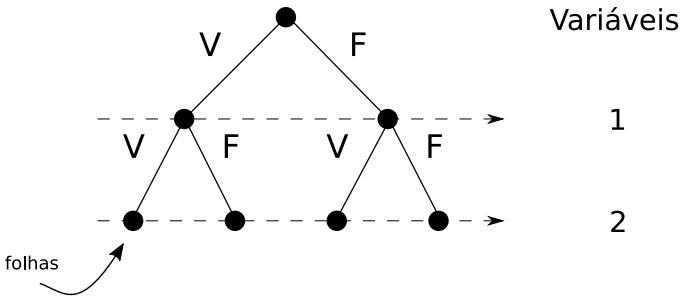
Foi feita a minimização de expressões de duas até quatro variáveis. A codificação contou com a linguagem Python, e o desenvolvimento da interface gráfica foi feito utilizando o *framework* PyQt5.

MATERIAIS E MÉTODOS

A álgebra booleana é o ramo da álgebra em que os valores das variáveis são os valores-verdade verdadeiro ou falso, normalmente denotados como 1 e 0, respectivamente. Uma expressão é uma tautologia se todos os valores-verdade resultam em uma saída verdadeira e, se todos resultam em uma saída falsa, é chamado de contradição. As principais operações são a conjunção (·), a disjunção (+) e a negação, denotada por uma barra em cima da variável ($\overline{A}$) (DA SILVA, 2017).

À união de todas as combinações de valores possíveis para as variáveis é dado o nome de tabela-verdade. O número de linhas da tabela depende da quantidade de variáveis de entrada. Podemos ver a representação das combinações na árvore da Figura 1.

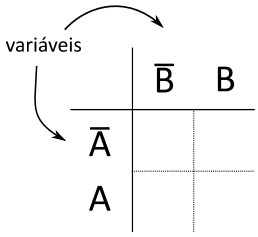
Figura 1 – Árvore de combinações



Fonte: Tocci (2007), adaptado

Cada caminho até as folhas representa uma combinação possível. O número total de folhas (linhas) será de 2^n, onde *n* representa o número de variáveis de entrada (TOCCI, 2007). O Mapa de Karnaugh contém as mesmas informações da tabela-verdade, porém apresentadas em um formato diferente. A montagem do mapa se inicia com o preenchimento das variáveis e suas negações de forma que cada quadrado gere uma combinação. A Figura 2 aborda essa etapa.

Figura 2 – Montagem do mapa de Karnaugh



Fonte: autoria própria

O próximo passo é preencher com “1” as combinações que resultam em saídas verdadeiras e preencher com “0” caso contrário. Isso pode ser visto na Figura 3.

Figura 3 – Preenchimento dos valores das combinações no mapa de Karnaugh

	B̄	B
Ā	1	0
A	1	1

Fonte: autoria própria

Combinando adequadamente os quadrados do mapa cujos valores são verdadeiros, obtemos a expressão mínima. Esse processo de combinação é denominado de agrupamentos. No escopo desse trabalho existem três tipos de agrupamento: pares, quartetos e octetos. Esses são formados por cédulas adjacentes entre si (GERSTING, 2007). O software também reconhece tautologias e contradições.

Figura 4 – Exemplo de agrupamento par

	B̄	B
Ā	1	0
A	1	1

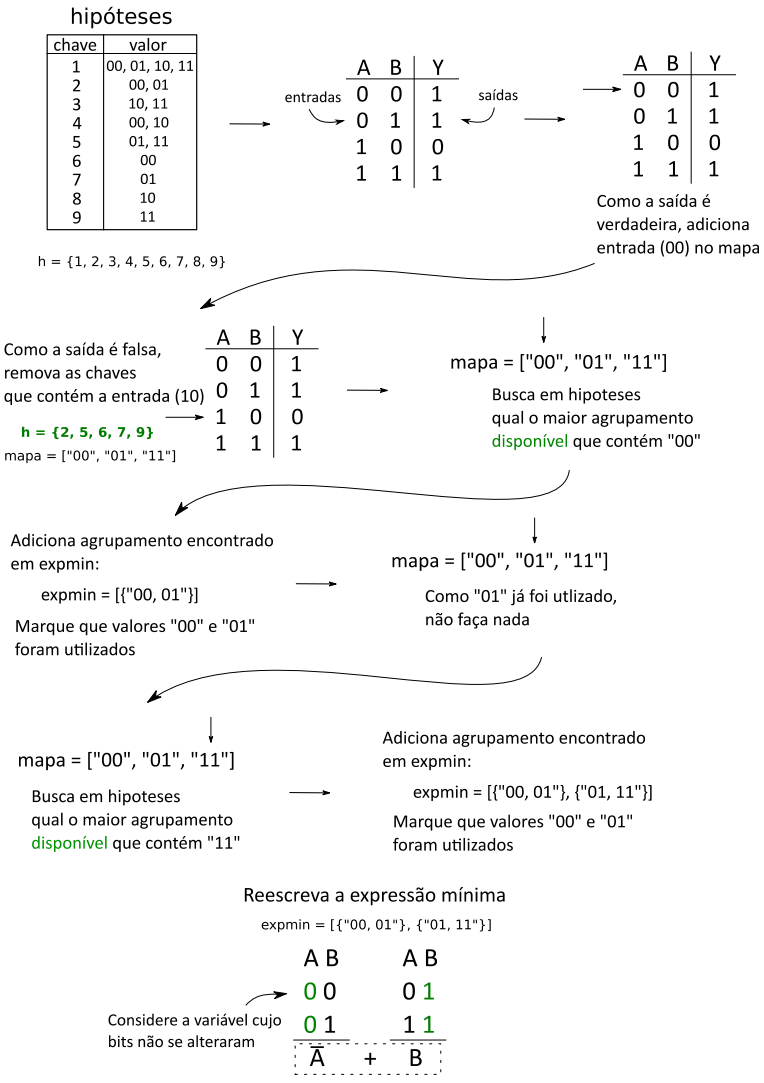
Fonte: autoria própria

Foi implementado o método de Mapas de Karnaugh na linguagem de programação Python. A entrada é uma tabela verdade e a saída é a expressão mínima em forma de soma de produtos. Inicialmente, o algoritmo desenvolvido possui um dicionário de hipóteses. Esse dicionário contém todos os agrupamentos possíveis em ordem decrescente levando em consideração o tamanho dos agrupamentos.

As chaves dos agrupamentos são armazenadas em uma estrutura de dados nativa da linguagem Python chamada *set*. Esse *set* é responsável por remover chaves de agrupamentos que não são válidos para a formação da expressão mínima.

Caso a combinação “00”, por exemplo, seja falsa, serão removidos do *set* todos os agrupamentos que contém essa combinação. Isso garante que apenas chaves de combinações verdadeiras estarão presentes. Um exemplo de execução pode ser visto na Figura 5. O pseudocódigo pode ser visto na Tabela 1.

Figura 5 – Exemplo de execução



Fonte: autoria própria

Tabela 1 – Pseudocódigo do algoritmo

Pseudocódigo

```
mapa = []
expmin = []
set = {}

Para cada linha da tabela:
        x = entrada
        y = saida
        Se y é falso:
                remove todos agrupamentos que contem x
        Se não:
                adiciona y em mapa

Para cada valor do mapa:
        busca(valor)

busca(x):
        k = []
        Se x não foi utilizado:
                Para cada chave do set:
                        Se x está em hipoteses[chave]:
                                Adiciona agrupamento encontrado na expmin
                                k = agrupamento
                                pare

                Para cada valor de k:
                        Marque que valor foi utilizado
```

Fonte: autoria própria.

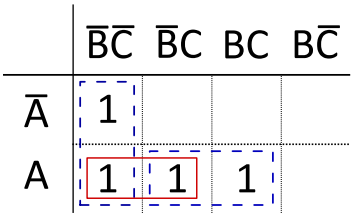
Também foi desenvolvido um processo de validação do algoritmo. O objetivo do processo é gerar expressões mínimas aleatórias para que estas sejam passadas ao minimizador criado para que seja analisada sua corretude. Para isso, foi criado um *set* que possui todos os literais possíveis de uma expressão (AB, $\overline{A}C$, $\overline{A}$), e também um dicionário de regras. As chaves do dicionário de regras são os literais mencionados anteriormente, e cada chave está atrelada ao um conjunto de literais que não podem existir na expressão mínima caso o literal da chave pertença a expressão. Para construir o dicionário de regras foram utilizadas as propriedades da álgebra booleana apresentadas nas Equações 1 e 2 (GERSTING, 2007).

$$x_1x_2 + x'_1x_2 = x_2 \quad (1)$$

$$x_1 + x'_1x_2 = x_1 + x_2 \quad (2)$$

Durante a validação foi descoberta uma falha de implementação. A falha consiste na geração de agrupamentos onde todos os elementos pertencem a outros agrupamentos. Apesar de não alterar o valor semântico, faz com que o algoritmo não gere a expressão mínima. Isso pode ser visto na Figura 6.

Figura 6 – Azul: agrupamentos gerados corretamente. Vermelho: agrupamento gerado devido à falha de implementação



Fonte: autoria própria

Para concertar essa falha foi utilizada uma função de pós-processamento para a resposta. Essa função remove os agrupamentos incorretos. O pseudocódigo pode ser visto na Tabela 2.

Tabela 2 – Pseudocódigo da função de pós-processamento

Pseudocódigo da função de pós-processamento

posProcessamento():
  Para cada termo da expmin:
  contador = 0
  Para cada variável do termo:
  Se variável foi utilizada:
  contador += 1
  Se contador == tamanho(termo):
  Retira termo da expmin

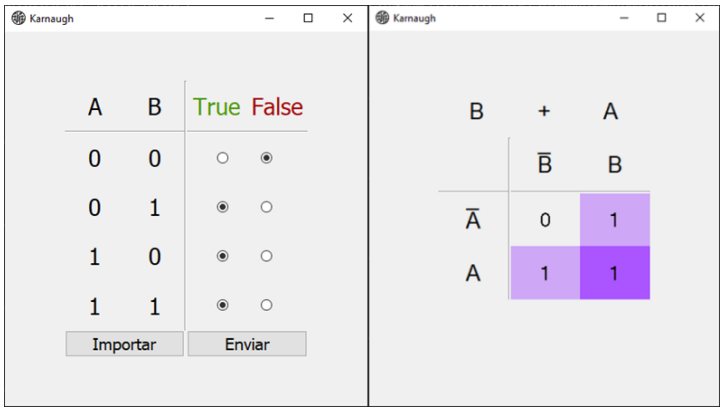
Fonte: autoria própria.

RESULTADOS E DISCUSSÕES

O programa projetado foi concebido para minimizar expressões de até quatro variáveis. Dois tipos de entrada são suportados: via botões (na interface gráfica) ou importando arquivos de extensão *csv* ou *tsv* que contém a tabela-verdade de entrada.

Foram feitos testes com duas, três e quatro variáveis. A Figura 7 mostra a tabela-verdade de entrada para duas variáveis e também a saída do programa. A Figura 8 mostra exemplos de saída do programa para três e quatro variáveis.

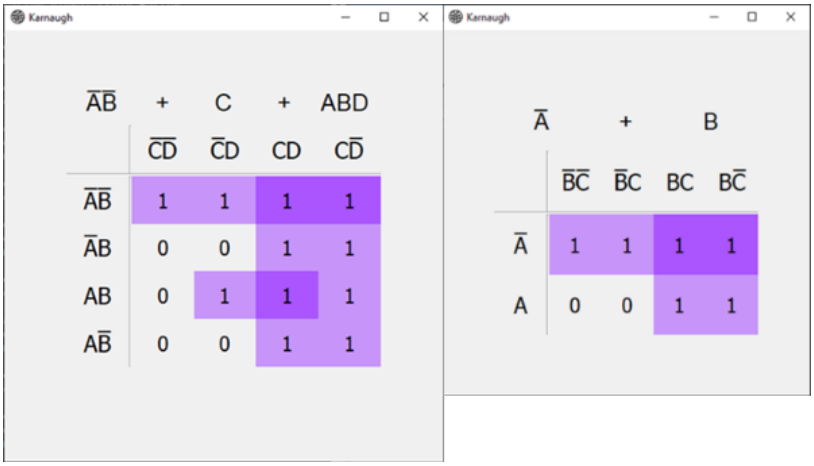
Figura 7 – Entrada e saída para duas variáveis



Fonte: autoria própria

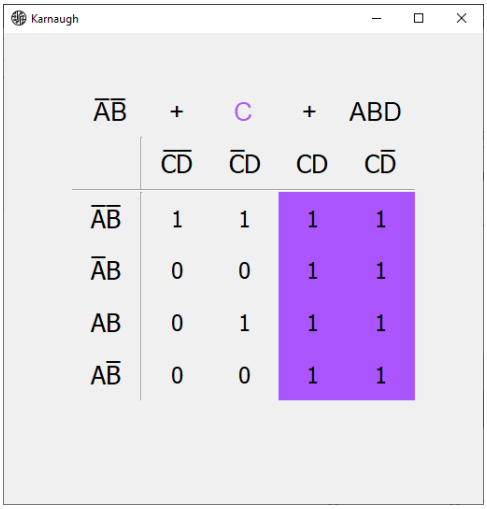
Como o *software* projetado busca ser o mais didático possível, foi feito uma função que realça os agrupamentos que formam a expressão mínima. Essa função pode ser vista na Figura 9.

Figura 8 – Esquerda: saída de quatro variáveis. Direita: saída de três variáveis



Fonte: autoria própria

Figura 9 – Função que realça agrupamento selecionado



Fonte: autoria própria

CONCLUSÃO

Expressões da lógica proposicional são ferramentas úteis no âmbito de circuitos digitais. A minimização dessas expressões, então, contribui para projetos de menor custo de produção e mais eficientes.

Nesse trabalho foi feito um minimizador de expressões booleanas utilizando a linguagem de programação Python. Uma interface gráfica também acompanha o programa. Os resultados obtidos foram validados e a corretude foi evidenciada por meio de um experimento de validação. Como sugestão de trabalhos futuros fica a implementação do método computacional mais tradicional de minimização, o método de Quine-McCluskey, o qual é capaz de resolver mais facilmente expressões com mais de 4 variáveis.

REFERÊNCIAS

GERSTING, Judith L. **Fundamentos Matemáticos para a Ciência da Computação**. 3ª. ed. Rio de Janeiro, RJ: LTC, 1993.

TOCCI, Ronald J. *et al*. **Sistemas Digitais**: Princípios e Aplicações. 10ª. ed. [*S. l.*: *s. n.*], 2007.

KLOCK, Carlos Eduardo; SCHNEIDER, Felipe R.; RIBAS, Renato P. Ferramenta para visualização de mapas de karnaugh de duas a oito variáveis. **Salão de iniciação Científica (15.: 2003: Porto Alegre, RS). Livro de resumos. Porto Alegre: UFRGS, 2003.**, 2003.

DA SILVA, Flávio Soares Corrêa *et al*. **Lógica para Computação**. 2ª. ed. [*S. l.*: *s. n.*], 2017.