UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS VII – GOVERNADOR ANTÔNIO MARIZ
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS
CURSO DE LICENCIATURA PLENA EM COMPUTAÇÃO

ROMULO JOSÉ PEREIRA DA SILVA

AVALIAÇÃO DA HEURISTICA WINDOW NA GERAÇÃO DE DIAGRAMAS DE DECISÃO BINARIO ORDENADO E REDUZIDO DE ESPECIFICAÇÕES BOOLEANAS DE SISTEMAS INDUSTRIAIS

PATOS – PB
2014

ROMULO JOSÉ PEREIRA DA SILVA

AVALIAÇÃO DA HEURISTICA WINDOW NA GERAÇÃO DE DIAGRAMAS DE DECISÃO BINARIO ORDENADO E REDUZIDO DE ESPECIFICAÇÕES BOOLEANAS DE SISTEMAS INDUSTRIAIS

Monografia apresentada ao Curso de Licenciatura plena em Computação da Universidade Estadual da Paraíba em cumprimento à exigência para obtenção do grau de licenciado em computação.

Orientadora: Dr.(a) Kézia de Vasconcelos Oliveira Dantas

PATOS – PB
2014

UEPB - SIB - Setorial - Campus VII

S586a Silva, Romulo José Pereira da
Avaliação da Heurística Window na geração de diagramas de decisão binário ordenado e reduzido de especificações Booleanas de Sistemas Industriais [manuscrito] / Romulo José Pereira da Silva. - 2014.
45 p. : il. color.

Digitado.
Trabalho de Conclusão de Curso (Graduação em Computação) - Centro de Ciências Exatas e Sociais Aplicadas, Universidade Estadual da Paraíba, 2014.
"Orientação: Profa. Dra. Kézia de Vasconcelos Oliveira Dantas, CCEA".

1. ROBDD. 2. Funções booleanas. 3. Heurística Window. 4. Sistemas industriais. I. Título.

21. ed. CDD 005

Rômulo José Pereira da Silva

AVALIAÇÃO DA HEURISTICA WINDOW NA GERAÇÃO DE DIAGRAMAS DE DECISÃO BINARIO ORDENADO E REDUZIDO DE ESPECIFICAÇÕES BOOLEANAS DE SISTEMAS INDUSTRIAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Licenciatura em Computação da Universidade Estadual da Paraíba, em cumprimento à exigência para obtenção do grau de Licenciado em Computação

Aprovado em 04 de dezembro de 2014

BANCA EXAMINADORA

Kézia de Vasconcelos Oliveira Dantas
(Orientadora)

Pablo Ribeiro Suárez
(Examinador)

Rodrigo Farias Herculano Mendes
(Examinador)

Dedico aos meus pais José *e* Terezinha, as minhas irmãs Renata e Rayane pelo incentivo *e apoio* que foram constantes durante todo o curso.

AGRADECIMENTOS

Quero agradecer, em primeiro lugar, a Deus, pela força e coragem durante toda esta longa caminhada.

Aos meus pais José e Terezinha, as minhas irmãs por sempre estarem presentes me apoiando e incentivando durante toda a minha caminhada acadêmica.

À professora Doutora Kézia de Vasconcelos Oliveira Dantas, pela dedicação em suas orientações prestadas na elaboração deste trabalho, me incentivando e colaborando no desenvolvimento.

A todos os amigos e colegas do curso de licenciatura em computação e aos de trabalho da empresa QIX Soluções computacionais pelo apoio e força que sempre me deram.

"É melhor tentar e falhar, que preocupar-se e ver a vida passar.
É melhor tentar, ainda que em vão que sentar-se, fazendo nada até o final.
Eu prefiro na chuva caminhar, que em dias frios em casa me esconder.
Prefiro ser feliz embora louco, que em conformidade viver."

(Martin Luther king)

RESUMO

Diagrama de decisão Binário (BDD) é um grafo acíclico direcionado, usado para representar as funções booleanas. ROBDD (Diagramas de Decisão Binária Ordenada e Reduzida) é um tipo de BDD que utiliza a ordenação de variáveis para obter uma forma canônica que melhore a representação das funções. Desta forma, existirá apenas um único BDD que representará uma função. Com esta propriedade, podemos facilmente verificar a equivalência de duas funções booleana apenas verificando se as estruturas apresentam a mesma forma. A ordenação de variáveis pode reduzir substancialmente o tamanho do ROBDD. As ordenações do tipo dinâmicas tendem a estabelecer uma ordem melhor para as variáveis por que este ajusta a ordem durante a geração do ROBDD. Neste trabalho foi desenvolvida uma ferramenta gráfica para geração de ROBDDs utilizando a heurística window para a ordenação das variáveis dinamicamente. Foram utilizados estudos de casos que utilizam especificações booleanas de sistemas industriais. Os ROBDDs gerados pela ferramenta desenvolvida foram comparados aos ROBDDs obtidos por OLIVEIRA (2014), a fim de verificar a eficiência da heurística window. De acordo com os resultados obtidos no presente trabalho, pode se destacar que a ferramenta desenvolvida obteve resultados semelhantes, ou melhor, na representação dos ROBDDs nos estudos de casos do que os obtidos por OLIVEIRA (2014).

Palavras-chaves: ROBDD, funções booleanas, heurística window, sistemas industriais.

ABSTRACT

Binary decision diagram (BDD) is a directed acyclic graph, used to represent Boolean functions. ROBDD (Ordered Binary Decision diagrams and reduced) is a type of BDD that uses the ordering of variables to obtain a canonical form that improves the representation of functions. In this way, there is only a single BDD that represents a function. With this property, we can easily verify the equivalence of two Boolean functions just checking if the structures have the same shape. The ordering of variables can substantially reduce the size of the ROBDD. Dynamic type Ordinances tend to establish a better order for variables for which this adjusts the order during the generation of the ROBDD. In this work, we developed a tool for generating graphics ROBDDs using heuristics window for the ordering of the variables dynamically. We used case studies that use Boolean specifications of industrial systems. The ROBDDs generated by the tool developed were compared to those obtained by ROBDDs OLIVEIRA (2014), in order to verify the efficiency of heuristic window. According to the results obtained in the present work, can stand out the tool developed similar results, or better, in representation of ROBDDs us case studies than those obtained by OLIVEIRA (2014).

Keywords: ROBDD, Boolean functions, heuristics window, industrial systems.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE SIGLAS

BDD	Diagrama de Decisão Binário
BSD	Berkeley Software Distribution
CAD	Computer Aided Design
IDE	Integrated Development Environment
ITE	If-Then-Else
OBDD	Diagrama de Decisão Binário Ordenado
ROBDD	Diagrama de Decisão Binário Ordenado e Reduzido
SIS	Sistemas Instrumentados de Segurança
VLSI	Very Large Scale Integration
JVM	Java Virtual Machine

SUMÁRIO

1. INTRODUÇÃO

Diagramas de decisão binária (do inglês Binary Decision Diagram - BDD) são grafos usados para representar funções booleanas. É definido como um grafo acíclico direcionado com dois nós terminais que são chamados de nó 0 terminal e nó 1 terminal. Cada nó não terminal tem um índice para identificar uma variável de entrada da função booleana e têm duas arestas, aresta-0 e aresta-1 que são direcionados a outros nós do BDD (MINATO, 1996).

Com o uso de BDDs é possível aumentar a estabilidade e a confiabilidade dos hardwares e dos softwares desenvolvidos (MARQUES, 2003). Os mesmos são empregados com sucesso em: ferramentas de CAD (Computer Aided Design) e VLSI (Very Large Scale Integration); aplicações de sínteses; testes, simulação, entre outras. A utilização de BDDs tem se expandido a áreas de projeto de sistemas concorrentes, lógica matemática e inteligência artificial. Em muitos domínios de aplicação os BDDs apresentam características próprias que podem ser aproveitadas para reduzir o tamanho da representação e o seu tempo de manipulação (FLORES, 1996).

As aplicações de BDDs na checagem de modelos (do inglês model checking) tornaram úteis as representações de estados, transições de estados e especificações lógicas temporais das propriedades de um sistema. Praticamente todos os hardwares passam por checagens de modelos antes de serem lançados, especialmente após 1994, época em que ocorreu o "Pentium Disaster", problema este ocasionado pela falta de checagem que atingia operações de cálculo de ponto flutuante do processador (MIRANDA, 2006).

Existem diversos tipos de BDDs, estes são variações que buscam melhorar as formas de representação de funções booleanas. Por exemplo, diagrama de decisão binário ordenado (do inglês Ordered Binary Decision Diagram - OBDD) utiliza-se de uma ordenação fixa para melhorar a representação. Outro exemplo é o Diagrama de Decisão Binário Ordenado e Reduzido (ROBDD), introduzidos por Bryant (1986), estes são formas canônicas para representar funções booleanas. Desta forma, existirá apenas um único BDD que representará uma função desde que a ordem das variáveis seja fixa.

A ordenação de variáveis sempre foi um problema crítico na construção de BDDs, para esta problemática pesquisadores desenvolveram técnicas e abordagens mais eficientes para ordenar as variáveis. Existem duas técnicas de ordenação de variáveis as estáticas e as

dinâmicas. A técnica de ordenação estática estabelece a ordem das variáveis antes da construção do BDD, enquanto a técnica de ordenação dinâmica ajusta-se durante a própria construção, ou seja, permite ajustar a ordem das variáveis durante a sua construção.

Dentre as técnicas de ordenação estática as principais são: busca em profundidade (appending) (MALIK et. al, 1988), Fine Heuristc (MALIK et. al, 1988), algoritmo de entrelaçamento (Interleaving based algorithm) (FUJII, 1993), Esquema de amostras (JAIN, 1998), Meta Heurística (DRECHSLER, 2002). As principais técnicas dinâmicas são: Permutação de variáveis adjacentes (RUDELL, 1993), Sifting (RUDELL, 1993), Window (FUJITA et. al, 1991).

Diante deste contexto, o presente trabalho visa desenvolver uma ferramenta que utiliza as propriedades do ROBBD, com o uso da heurística window para ordenar as variáveis para obter a representação de funções booleanas. A escolha de uma técnica dinâmica para ordenar as variáveis, é em razão de que ao ajustar a ordenação durante a geração do BDD, esta utiliza sempre a melhor ordenação possível diferente das técnicas de ordenação estática. VIDAL (2002) afirma que, as técnicas de ordenações dinâmicas podem diminuir o espaço alocado, otimizar e acelerar os algoritmos de manipulação de BDDs.

Para avaliar a heurística window, foram feitos estudos de casos, que utilizaram especificações booleanas de sistemas industriais. O primeiro é SIS (Sistemas Instrumentados de Segurança) para detectar a presença de fogo e gás em uma zona e o segundo é um SIS para controlar o nível de um tanque (OLIVEIRA, 2014).

OBJETIVOS

Objetivo Geral

Desenvolver uma ferramenta gráfica para geração de ROBDDs utilizando a heurística Window para ordenar as variáveis dinamicamente.

Objetivos Específicos

Fazer uma pesquisa na literatura acerca de ROBDDs.

Investigar as técnicas de ordenação de variáveis em ROBDDs, com foco na ordenação dinâmica window.

Aplicar estudos de caso utilizando as especificações booleanas de sistemas industriais para avaliar a heurística.

2. FUNDAMENTAÇÃO TEÓRICA

Nesta seção apresentará conceitos a cerca de funções booleanas, diagramas de decisão binário, diagramas de decisão binário ordenado, diagramas de decisão binário ordenado e reduzido, operador ITE (If-The-Else) e a heurística window. Embasado em autores que já possuem trabalhos desenvolvidos nesta área

.

2.1 FUNÇÕES BOLEANAS

A álgebra booleana é de fundamental importância para a ciência da computação e designer de sistemas digitais, BRYANT (1986) destaca que, muitos dos problemas na lógica de designer e testes digitais, inteligência artificial, otimização combinatória podem ser expressas por uma sequência de operações sobre as funções booleanas.

MINATO (1996) destaca que a manipulação de funções booleanas é uma importante técnica para muitos dos problemas da ciência da computação, como os sistemas CAD e VLSI. Com o recente avanço na tecnologia VLSI, os problemas têm crescido além do escopo do projeto, tornando os sistemas CAD amplamente utilizados. O desempenho destes sistemas depende muito da manipulação eficiente das funções booleanas

O cálculo clássico para lidar com as constantes de valores booleanos verdade e falso (1 e 0), utiliza-se de operadores lógicos: conjunção, disjunção, negação , implicação e bi-implicação, são representados respectivamente pelos símbolos: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$, que juntos formam as expressões booleanas, que são chamados de variáveis proposicionais ou letras proposicionais, e as expressões booleanas são conhecidas como Lógica Proposicional (ANDERSEN, 1997).

ANDERSEN (1997) explica que as expressões booleanas são geradas a partir da gramática:

$$\varepsilon ::= 0 \mid 1 \mid x \mid \neg\varepsilon \mid \varepsilon \wedge \varepsilon \mid \varepsilon \vee \varepsilon \mid \varepsilon \rightarrow \varepsilon \mid \varepsilon \leftrightarrow \varepsilon,$$

Sendo que x denota a variável proposicional, que varia ao longo de um conjunto de variáveis booleanas. Utilizam-se parênteses para resolver ambiguidades e alterar a ordem de prioridade dos operadores que segue essa ordem decrescente: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, Por exemplo:

$$\neg x_1 \wedge x_2 \vee x_3 \rightarrow x_4 = \Big(\big((\neg x_1) \wedge x_2\big) \vee x_3\Big) \rightarrow x_4.$$

Uma expressão booleana com variáveis $x_1, \ldots, x_n$ denota para cada atribuição de valor verdade a própria variável, de acordo com a tabela verdade padrão (Figura 1), as atribuições verdade são escritas como sequências de valores para as variáveis, por exemplo, [0/x_1 , 1/ x_2, 0/x_3, 1/x_4] atribui 0 a x_1 e x_3 , 1 a x_2 e x_4. Para esta atribuição, a expressão acima teria o valor 1, e para [0/x_1 , 1/ x_2, 0/x_3, 0/x_4] a expressão teria valor 0 (ANDERSEN, 1997).

Figura 1: Tabelas verdade.

	¬
0	1
1	0

∧	0	1
0	0	0
1	0	1

∨	0	1
0	0	1
1	1	1

⇒	0	1
0	1	1
1	0	1

⇔	0	1
0	1	0
1	0	1

Fonte: ANDERSEN (1997).

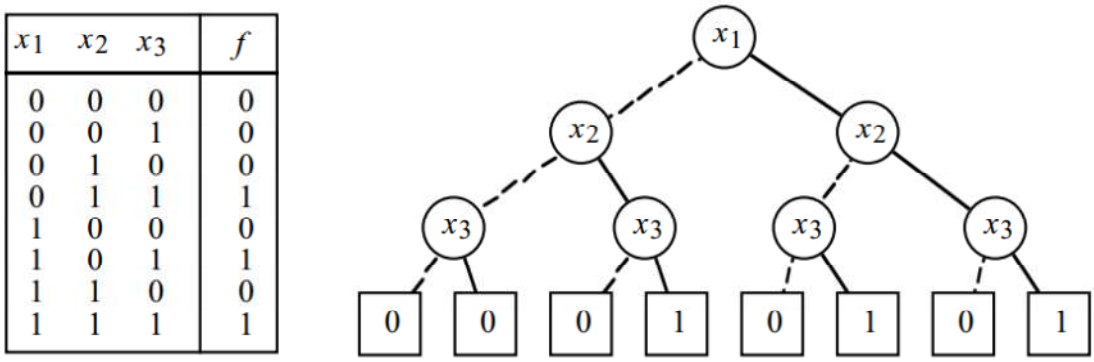
O conjunto de valores booleanos denotado por $\mathbb{B} = \{0,1\}$, em uma expressão booleana t, quando fixado a ordem das variáveis, pode-se ver t como a definição de uma função de $\mathbb{B}^n$ para $\mathbb{B}$, em que n é o número de variáveis. Observe, que a ordem escolhida para as variáveis é essencial para a função que está definida. Considere, por exemplo, a expressão $x \rightarrow y$. Se escolher a ordenação $x < y$, então esta função f (x, y) = $x \rightarrow y$ é verdadeira se o primeiro argumento implica o segundo, mas se escolhermos a ordenação $x > y$ então a função f (x, y) = $x \rightarrow y$, é verdadeira se o segundo argumento implica o primeiro. Quando considerar as representações das expressões booleanas compactadas, tais ordenações de variáveis desempenharam um papel crucial (ANDERSEN, 1997).

Na literatura, existem diversos métodos para representar e manipular funções booleanas. Métodos com base em representações clássicas, tais como tabelas verdade, mapas de Karnaugh (SENA e TORRES, 2008), ou soma de produtos canônicos, são bastante impraticáveis, pois sua representação é de ordem exponencial. Abordagens mais práticas utilizam representações que não são de tamanho exponencial, para muitas das funções. Estes tipos de representação têm ainda a desvantagem de não terem uma forma canônica (única) de representar uma função, ou seja, uma mesma função pode ter várias representações válidas, mas diferentes. Isto faz com que os problemas de equivalência de funções ou verificação de tautologias sejam muito difíceis de resolver. (BRYANT, 1986; FLORES, 1996).

Funções booleanas também podem ser expressas por um grafo direcionado acíclico

denominado de diagrama de decisão binária (BDD). Os BDDs têm atraído a atenção de muitos pesquisadores devido a suas boas propriedades para representar funções booleanas. Um BDD dá uma forma canônica (única) para uma função booleana, de modo que podemos facilmente verificar a equivalência das duas funções. No BDD tamanho varia de acordo com as funções, ao contrário da tabela verdade que é de ordem exponencial como pode ser visto na Figura 2 que ilustra uma função f = (x_1, x_2, x_3) (BRYANT, 1986; MINATO, 1996). Na Seção 2.2 apresentam-se os principais conceitos relativos à BDD.

Figura 2: Tabela verdade e árvore de decisão binária função f = $(\boldsymbol{x_1, x_2, x_3})$.



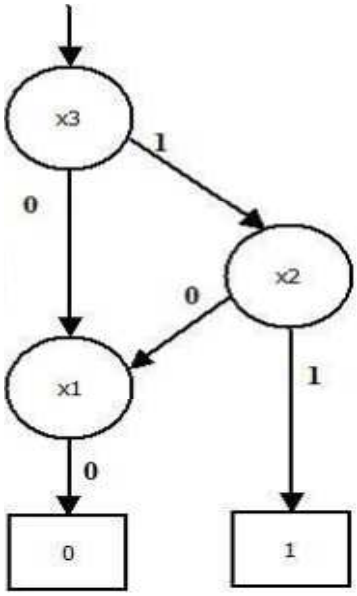
Fonte: BRYANT (1992).

2.2 DIAGRAMAS DE DECISÃO BINÁRIA

BDD é um grafo usado representar funções booleanas, é definido como um gráfico acíclico (sem ciclos) direcionado com um nó de entrada (raiz), os nós intermediários representam as variáveis, Cada nó possui duas arestas (saídas), aresta-0 e aresta-1 que são direcionados a outros nós do BDD e os nós terminais representam os valores da função (0 e 1), como é ilustrado na Figura 3. (MINATO, 1996).

Atribuindo-se valores as variáveis, o valor produzido pela função é determinado traçando um caminho da raiz até um nó terminal, seguindo os ramos indicados pelos valores atribuídos às variáveis. O valor da função é indicado pelo nó terminal. Devido à forma como os ramos são ordenados como ilustrado na Figura 3 (BRYANT, 1992).

Figura 3: BDD que representa a expressão ($x_3 \cdot x_2 \vee x_1$).



Fonte: MINATO (1996).

Cada nó de um BDD representa uma decomposição de Shannon, para a função booleana f:

$$\mathrm{f} = (\neg x_i \wedge f_0) \vee (x_i \wedge f_1),$$

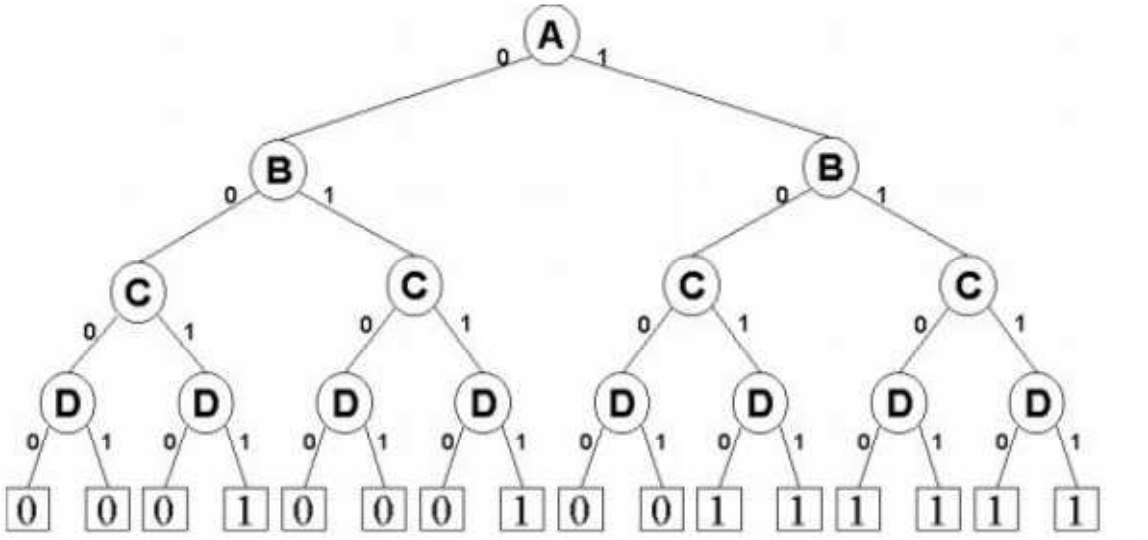
onde i é o índice do nó. f_0 e f_1 são as funções dos nós apontados para as arestas 0 e 1, respectivamente (MYNATO, 1996). Considerando um conjunto de variáveis booleanas X = $\{x_1, x_2, \ldots, x_n\}$ e sendo $\pi \rightarrow \{1, 2, \ldots, |X| = n\}$ sendo o mapeamento bijetivo dos índices das variáveis. Referimos que uma variável x_i é de uma ordem inferior a x_j se x_i estiver mais perto da raiz do grafo (BRYANT, 1992).

2.2.1 Diagrama de decisão binário ordenado

Um OBDD (Diagrama de Decisão Binário Ordenado) é um BDD em que as variáveis de entrada aparecem em uma ordem fixa em todos os caminhos do grafo, e que nenhuma variável aparece mais de uma vez em um caminho, ou seja, ela não é avaliada mais de uma vez (MINATO, 1996).

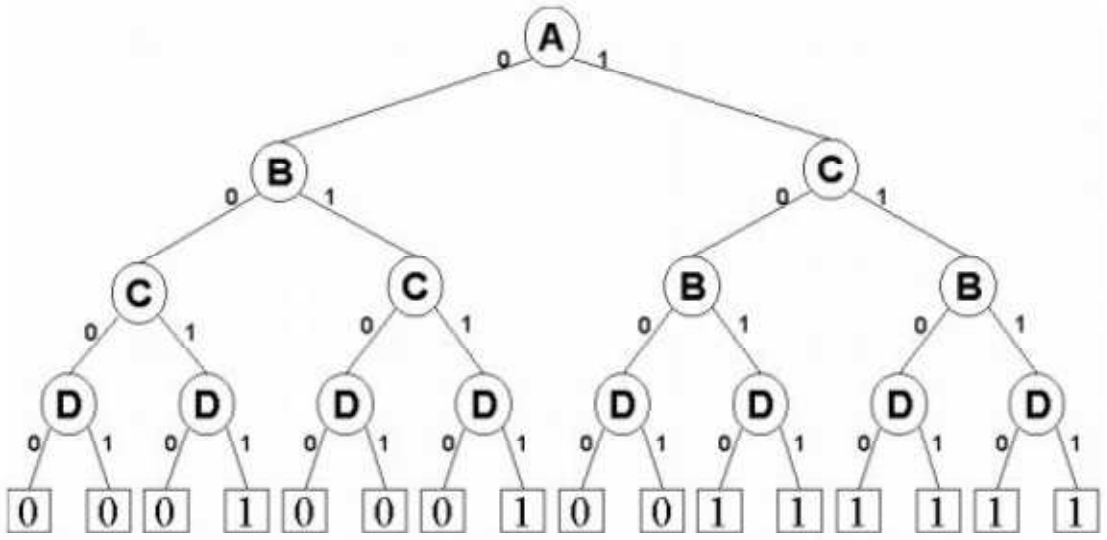
A Figura 4.a ilustra um OBDD que representa a função $\mathrm{f} = \big(A \wedge (B \vee C)\big) \vee (C \wedge D)$, já a Figura 4.b, ilustra um BDD que representa a mesma função f, sendo que este é um BDD não ordenado.

Figura 4: BDD Ordenado e BDD não ordenado.



a) BDD Ordenado.

.



b) BDD não ordenado

Fonte: MARQUES (2003).

2.2.2 Diagrama de decisão binário ordenado reduzido

ROBDDs, introduzidos por BRYANT (1986), são formas canônicas para representar funções booleanas. Desta forma, existirá apenas um único BDD que representará uma função desde que a ordem das variáveis seja fixa. Esta propriedade é muito importante para

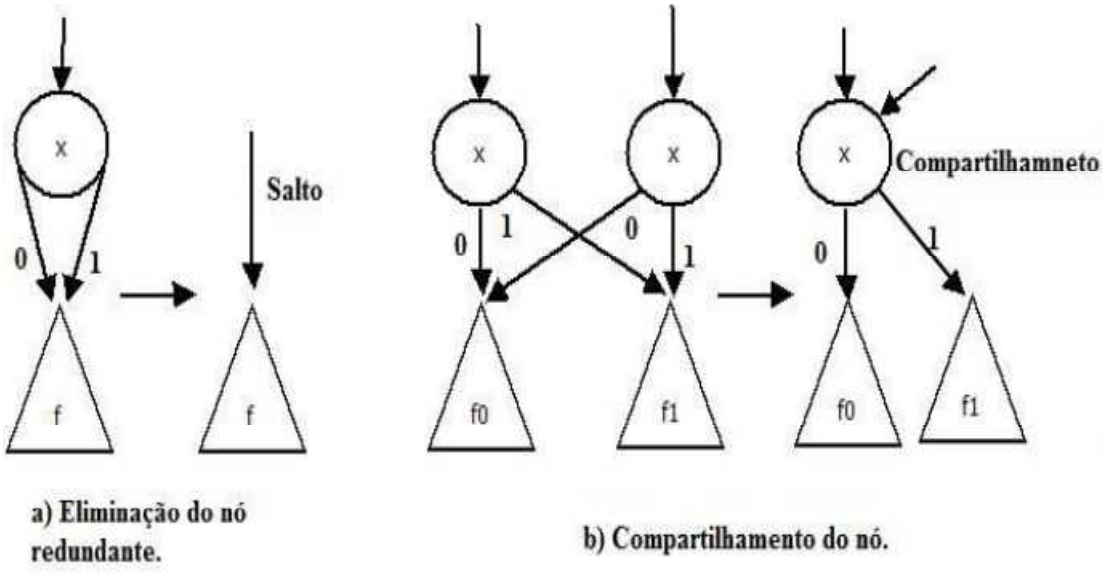
aplicações práticas, por exemplo, podemos facilmente verificar a equivalência de duas funções booleana apenas verificando se as estruturas apresentam a mesma forma (isomorfismo) de seus ROBDDs. A maioria dos trabalhos sobre BDDs são baseados na técnica dos ROBDDs (MINATO, 1996).

Para transformar um dado BDD em um ROBDD aplicam-se as seguintes regras:

1. Eliminar todos os nós redundantes, ou seja, os nós que estejam com suas arestas apontando para o mesmo nó. (Figura 5.a).

2. Compartilhar todos os sub-grafos equivalentes. (Figura 5.b).

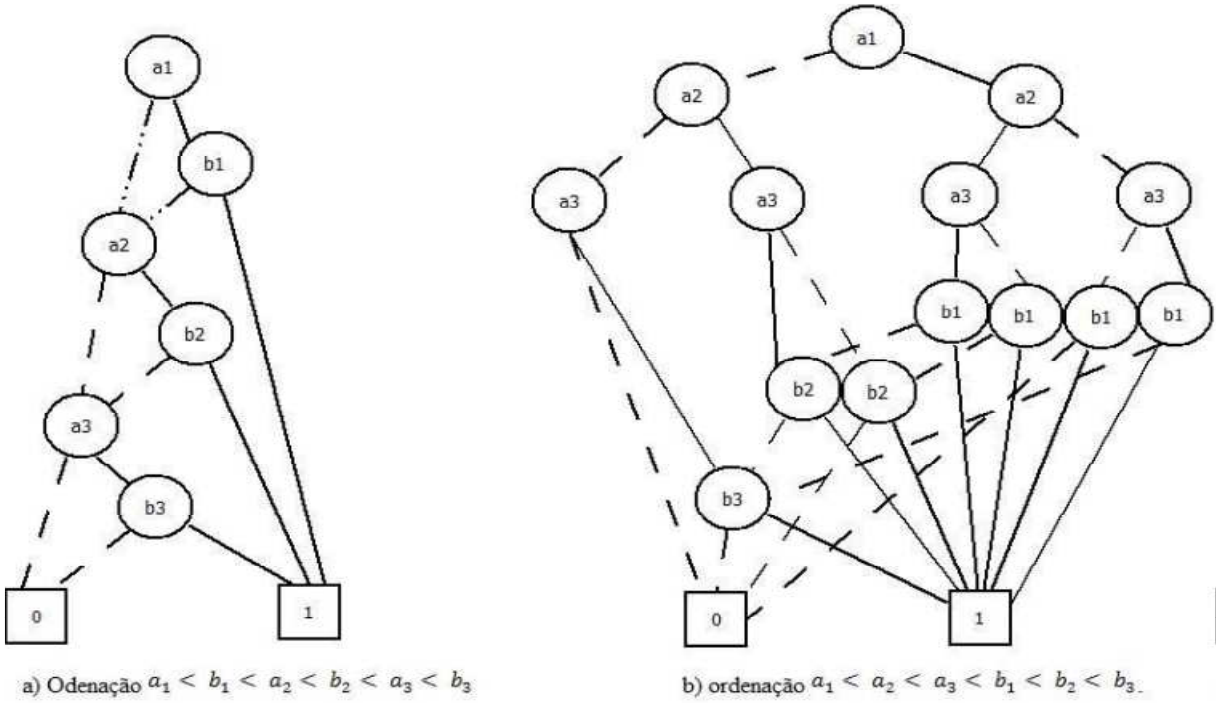
Figura 5: Regras de redução de BDDs.



Fonte: MINATO (1996).

A forma e tamanho do ROBDD de uma função dependem das ordenações das variáveis. Por exemplo, na Figura 5 mostra-se o ROBDD da função denotado pela expressão booleana $a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3$, onde $\cdot$ denota a operação AND e OR é denotada pela operação +. Para o caso da Figura 6.a, as variáveis são ordenadas $a_1 < b_1 < a_2 < b_2 < a_3 < b_3$, enquanto para o caso da Figura 6.b, são ordenadas $a_1 < a_2 < a_3 < b_1 < b_2 < b_3$ (BRYANT, 1992).

Figura 6: Representação de um ROBDD de uma mesma função, mas com duas ordenações diferentes.



Fonte: BRYANT (1992).

Podemos generalizar esta função sobre as variáveis $a_1, \cdots, a_n$ e $b_1, \cdots, b_n$ dada pela expressão:

$$a_1 \cdot b_1 + a_2 \cdot b_2 + \cdots + a_n \cdot b_n.$$

Generalizando a primeira ordenação de variáveis $a_1 < b_1 < \cdots < a_3 < b_3$, resulta em um ROBDD com 2n para um nó não terminal e para cada variável. Generalizando a segunda ordenação para $a_1 < \cdots < a_n < b_1 < \cdots < b_n$, por outro lado, resulta em um OBDD com $2(2^n - 1)$ nós não terminais. Para n com grandes valores, a diferença entre o crescimento linear da primeira ordenação contra o crescimento exponencial da segunda tem um efeito sobre os requisitos de armazenamento e a eficiência dos algoritmos de manipulação (BRYANT, 1992).

É um problema quase sempre crítico encontrar uma ordem adequada para as variáveis no BDD. Pesquisadores desenvolveram técnicas e abordagens mais eficientes para ordenações de variáveis para a construção de BDDs. Existem dois tipos de ordenações de variáveis, ordenação estática e dinâmica (FERNANDES e NASCIMENTO, 2002; RICE e KULHARI, 2008).

As técnicas de ordenação estáticas tentam estabelecer a ordem das variáveis antes da construção do BDD. Esta técnica pode ser automática ou manual, na ordenação automática o sistema utiliza-se de algoritmos para ordenar as variáveis analisando a estrutura do problema em questão. Já na ordenação manual é o usuário que informa qual a ordem das variáveis (VIDAL, 2002). Dentre os algoritmos de ordenação estáticos os principais são: busca em profundidade (MALIK et. al, 1988), Fine Heuristc (MALIK et. al, 1988), algoritmo de entrelaçamento (FUJII, 1993), esquema de amostras (JAIN, 1998), meta heurística (DRECHSLER, 2002). Na Seção 2.3 apresentam-se os principais conceitos relativos à heurística Window.

As técnicas de ordenação de dinâmicas, ou reordenação, tentam ajustar a ordenação do BDD durante a construção do mesmo, os métodos de reordenação mais simples tentam diminuir o número de nós BDDs alterando a posição das variáveis e identificando em que posição aquela variável estava quando o número de nós era o menor possível (VIDAL, 2002; RICE e KULHARI, 2008). Dentre os algoritmos de ordenação dinâmicos os principais são: Permutação de variáveis adjacentes (RUDELL, 1993), Sifting (RUDELL, 1993), Window (FUJITA et. al, 1991).

2.2.3 Operador ITE (If-Then-Else)

O operador If-Then-Else ou ITE é uma função booleana definida para três entradas F, G, H, que calcula: se F então G mais H. Isto é equivalente a:

$$ITE\ (F, G, W) = F \cdot G + \neg F \cdot H.$$

O ITE pode ser utilizado para implementar todas as operações booleanas de duas variáveis, veja na Tabela 1. ITE executa a função em cada nó do ROBDD, que é um bloco de construção eficiente para muitas outras operações no ROBDD (BRACE et. al., 1990).

Tabela 1: Funções de duas variáveis descritas usando o ITE.

Função	Formula Equivalente
F · G	ITE(F, G, 0)
F · ¬ G	ITE(F, ¬ G, 0)
¬ F · G	ITE(F, 0, G)
F + G	ITE(F, 1, G)
F + ¬ G	ITE(F, 1, ¬ G)
¬ F + G	ITE(F, G, 1)

Fonte: BRACE et. al (1990).

A formulação recursiva é a chave para a computação ITE (F, G, H) para as funções de F, G, H, representada pelo ROBDD. Os casos terminais para recursão são: ITE (1, F, G) = ITE (0, G, F) = ITE (F, 1, O) = F. Nota-se que esta formulação é válida para qualquer função booleana de qualquer número de variáveis Usa-se uma função de memória para melhorar o desempenho do ITE (BRACE et. al., 1990).

No Algoritmo 1, desenvolvido por BRACE et. al. (1990), apresenta-se a operação ITE, para o algoritmo as entradas são três nós de um ROBDD e a saída é um nó que representa a operação.

Tabela única é uma tabela que impõe ao ROBDD uma forma canônica forte. De modo que cada nó no ROBDD representa uma função lógica exclusiva. Assim, esta tabela hash é chamada de tabela única. Ela mapeia um tripla (v, G, H) a um nó de ROBDD F = (v, G, H). Cada nó no ROBDD tem uma entrada na tabela única. Antes de um novo nó é adicionado à ROBDD, uma pesquisa na tabela-única determina se um nó para essa função já existe. Se assim for, o nó existente é utilizado. Caso contrário, o novo nó é adicionado ao ROBDD e uma nova entrada é feita na tabela única (BRACE et. al., 1990).

Algoritmo 1: Operação ITE.

```
1   ITE(F, G, H){
2       if (caso terminal da execução) {
3           return resultado;
4       } else if (tabela calculada tem entrada (F, G, H}) {
5           return resultado;
6       }else{
7           v = topo_da_variável(F, G, H);
8           T = ITE (F_v, G_v, H_v);
9           E = ITE(F_v̄, G_v̄, H_v̄);
10          if (T == E){
11              return T;
12          }
13          R = ache_ou_adicione_a_tabela_única(v, T, E);
14          insere_na_tabela_calculada((F, G, H), R);
15          return R,
16      }
17  }
```

Fonte: BRACE et. al (1990).

2.3 HEURÍSTICA DE ORDENAÇÃO WINDOW

A heurística window utiliza a permutação de variáveis como base (FUJITA et. al, 1991). Consiste em criar janelas (windows) de k variáveis e fazer todas as possíveis k! combinações, permutando as variáveis adjacentes de k. Isto é feito usando k! – 1 permutações entre as variáveis, armazenando a melhor ordem, e no pior dos casos, terá que fazer mais k(k − 1) / 2 permutações para restaurar a melhor ordem para o BDD.

Uma janela de tamanho k no nível i é composta das variáveis do nível i até o nível i + k (FUJITA et. al, 1991). A Tabela 2 mostra as permutações de variáveis que são exploradas quando se aplica uma janela de tamanho k = 3 a partir da variável x_2. São necessárias seis

permutações e mais 3 permutações adjacentes adicionais (pior caso) são utilizadas para restaurar a melhor permutação (RUDELL, 1993; VIDAL, 2002).

Tabela 2: Exemplo de permutação window.

$\mathbf{x_1, x_2, x_3, x_4, x_5, x_6, x_7}$	**Inicial**
$\mathbf{x_1, x_3, x_2, x_4, x_5, x_6, x_7}$	Permutar (x_2, x_3)
$\mathbf{x_1, x_3, x_4, x_2, x_5, x_6, x_7}$	Permutar (x_2, x_4)
$\mathbf{x_1, x_4, x_3, x_2, x_5, x_6, x_7}$	Permutar (x_3, x_4)
$\mathbf{x_1, x_4, x_2, x_3, x_5, x_6, x_7}$	Permutar (x_3, x_2)
$\mathbf{x_1, x_2, x_4, x_2, x_5, x_6, x_7}$	Permutar (x_4, x_2)
$\mathbf{x_1, x_2, x_3, x_4, x_5, x_6, x_7}$	Permutar (x_4, x_3)
$\mathbf{x_1, x_3, x_2, x_4, x_5, x_6, x_7}$	Permutar (x_2, x_3)
$\mathbf{x_1, x_3, x_4, x_2, x_5, x_6, x_7}$	Permutar (x_2, x_4)

Fonte: RUDELL (1993).

O algoritmo window é bastante rápido, como utiliza a permutação de variáveis adjacentes. Porém este tem sua capacidade limitada a usar janelas de tamanho k (testes realizados mostram que o tamanho da janela deve ficar entre 3 e 6), assim ele não encontra ordens ótimas para as variáveis (RUDELL, 1993; VIDAL, 2002).

3. METODOLOGIA

Inicialmente foi realizada uma pesquisa bibliográfica com o objetivo de compreender melhor sobre diagramas de decisão binários ordenados e reduzidos (ROBDD), qual a sua utilização e importância. Foram realizadas também pesquisas sobre as técnicas para ordenação de variáveis (estáticas e dinâmicas) para construção de ROBDDs. Para a construção da fundamentação teórica foram feitas buscas na internet de artigos e trabalhos acadêmicos utilizando a ferramenta de pesquisas Google Acadêmico.

Na segunda etapa foi para desenvolver a ferramenta, a linguagem de programação que foi utilizada para o desenvolvimento da ferramenta foi à linguagem JAVA (ORACLE Corporation), a escolha desta linguagem é devido já ter experiência na utilização, além possuir uma grande quantidade de bibliotecas e frameworks. Foi utilizado uma biblioteca para manipulação de grafos chamada de JGraphX (JGRAPH Ltd).

A terceira etapa consistia em aplicar estudos de casos extraídos de OLIVEIRA (2014), para avaliar os ROBDDs gerados a partir da ferramenta desenvolvida usando a heurística window, com os de OLIVEIRA (2014) que utilizou uma ordenação estática em ordem crescente.

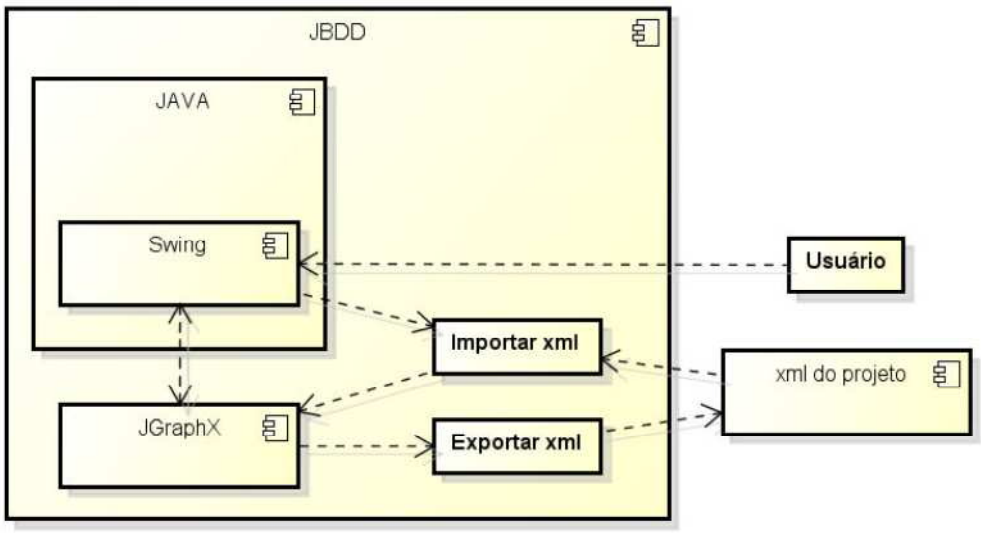
4. FERRAMENTA DESENVOLVIDA

A ferramenta desenvolvida foi em razão à escassez de ferramentas para geração de ROBDDs com algoritmos de ordenação de variáveis dinâmica. Existem aplicações para a estes fins, mas sempre utilizando as técnicas de ordenação estáticas, um exemplo é a ferramenta visBDD (MEINEL et. al, 2002) que gera o ROBDD de acordo com a ordenação passada pelo usuário. Então a ideia é construir uma aplicação que use uma heurística dinâmica para a ordenação das variáveis que seja automática, no qual o usuário irá informar somente a função booleana e a ferramenta irá ordenar e gerar o ROBDD automaticamente.

4.1 ARQUITETURA

A aplicação foi desenvolvida na linguagem de programação JAVA (ORACLE Corporation). O que motivou o uso desta linguagem foi à portabilidade e a plataforma mais popular, ou seja, podendo ser executado na maioria dos sistemas operacionais devido a JVM (Java Virtual Machine), que é um programa que executa os aplicativos JAVA. Para a codificação da aplicação, contou com o auxilio da IDE (Integrated Development Environment) eclipse Luna (ECLIPSE Foundation), e da biblioteca JGraphX (JGRAPH Ltd), conforme a Figura 7 ilustra a arquitetura.

Figura 7: Arquitetura da ferramenta Desenvolvida.



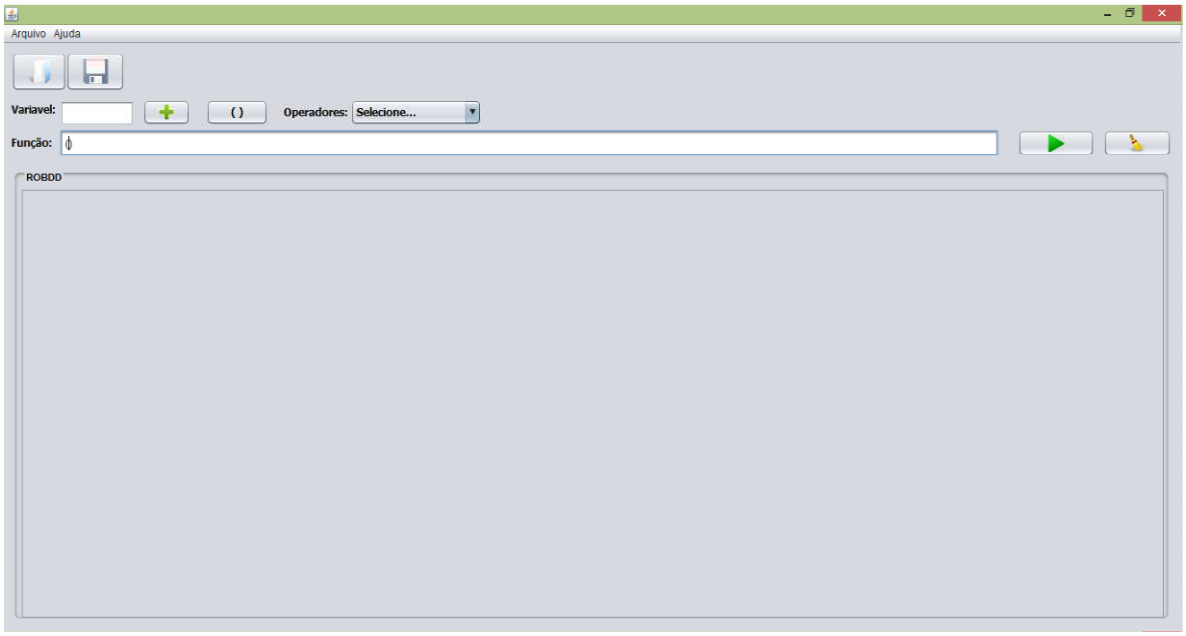
Fonte: Autoria própria.

JGraphX foi desenvolvida pela JGraph Ltd, é uma biblioteca Java Swing para criar diagramações, ela fornece funcionalidades para visualização e interação com gráficos. Esta é distribuída sob a licença BSD (Berkeley Software Distribution) (OPEN SOURCE INITIATIVE, 2014).

4.2 DESCRIÇÃO DA FERRAMENTA

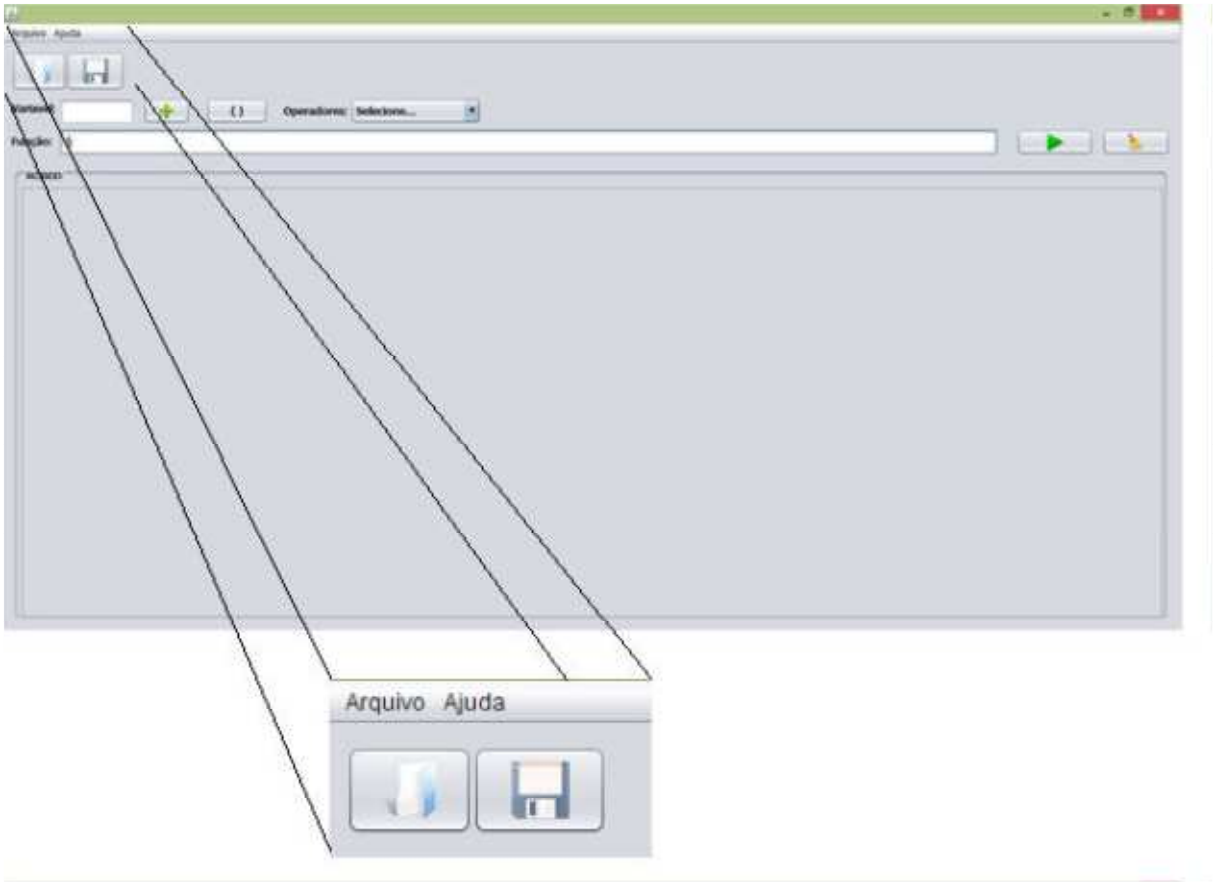
A ferramenta teve sua tela baseada na tela da ferramenta visBDD (MEINEL et. al, 2002). Esta possui uma tela inicial de acordo com a Figura 8, onde todo e qualquer recurso da ferramenta pode ser acessado a partir desta tela. Esta tela é composta de um menu chamado "Arquivo" com as opções de salvar, importar projetos e outro menu de ajuda Figura 9. Esta figura também mostra dois botões de atalho para abrir e salvar um projeto em uso.

Figura 8: Tela inicial da ferramenta.



Fonte: Autoria própria.

Figura 9: Barra de menu e atalhos para salvar e alterar projeto



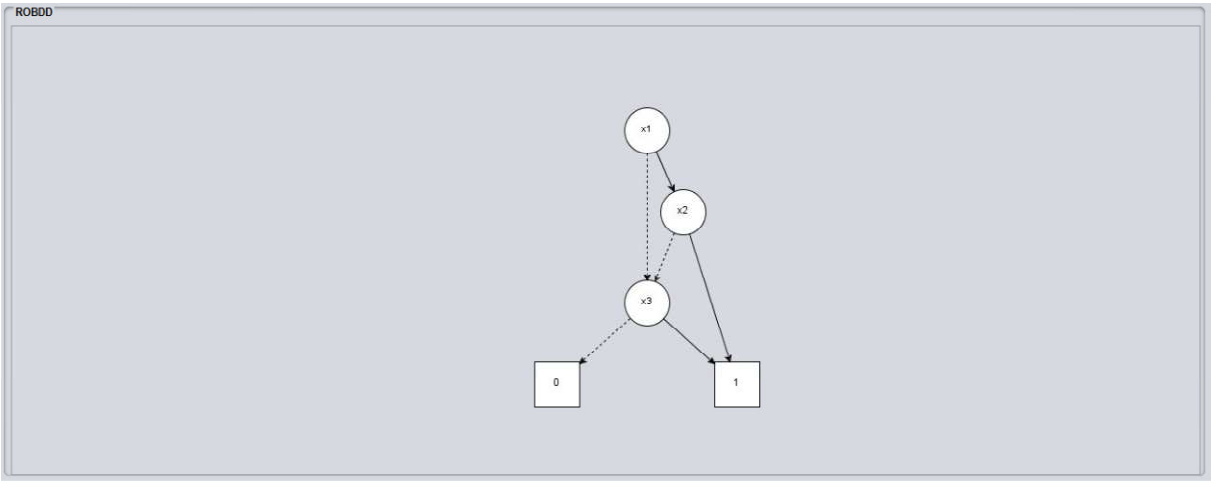
Fonte: Autoria própria.

Para a manipulação das expressões booleanas a ferramenta possui um campo de texto para digitar a expressão, além de possuir também outros campos para auxiliar o usuário na inserção desta, como pode ser visto na Figura 10. Após a inserção da expressão é só clicar no botão executar que a ferramenta irá gerar o ROBDD em um painel mostrado na Figura 11.

Figura 10: Componentes para inserção e manipulação da função booleana.

Fonte: Autoria própria.

Figura 11: Área para desenhar o ROBDD.



Fonte: Autoria própria.

Na seção 5 a seguir, serão apresentados dois estudos de casos: um sistema para detecção de fogo ou gás em uma zona e sistema de controle de nível de tanque que foram utilizados como parâmetros com objetivo de comparar os resultados obtidos com a ferramenta desenvolvida com os de OLIVEIRA (2014).

5. ESTUDOS DE CASOS

Neste capítulo a ferramenta desenvolvida será submetida a dois estudos de casos, com o objetivo de avaliar os resultados obtidos com a heurística window, através da comparação com os ROBDDs gerados por OLIVEIRA (2014) usando uma ordenação estática dada em ordem crescente. Na seção 5.1, o primeiro estudo de caso é um SIS para detectar a presença de fogo e gás em uma zona. Já na seção 5.2, apresenta o segundo estudo, onde é apresentado um SIS para controlar o nível de um tanque. Estes estudos são extraídos de OLIVEIRA (2014).

5.1 SISTEMA PARA DETECÇÃO DE FOGO OU GÁS

O sistema utilizado neste estudo de caso é para detectar fogo ou gás em uma determinada zona. O sistema é composto por cinco sensores, sendo dois de fumaça (SF1 e SF2) e três de gás (SG1, SG2 e SG3), um tanque, um dispositivo que libera gás CO_2 na confirmação de fogo na zona ($DispCO_2$), dois alarmes sonoros um que indica presença de fogo (AlaFDZ) o outro para indicar presença de gás (AlaGDZ) e duas válvulas, que são utilizadas para controlar o nível do gás no tanque. Uma das válvulas é chamada de auxiliar, que é acionada quando ocorre o vazamento de gás na válvula principal, a válvula auxiliar é utilizada para manter o sistema estável, evitando assim possíveis danos às instalações da zona. As seguintes operações são realizadas no sistema:

- se algum dos sensores de fumaça for acionado então, foi detectado fogo na zona;
- se o sensor de gás 1 e o sensores de gás 2 ou 3 forem acionados então, foi detectado presença de gás na zona;
- se o sensor de gás 2 e o sensor de gás 3 forem acionados então, foi detectado presença de gás na zona;
- se for detectado fogo na zona então, um alarme sonoro que indica presença de fogo é acionado e após 2 segundos o dispositivo que libera gás CO_2 é acionado e a válvula principal é desligada. O gás CO_2 liberado é utilizado para sanar o fogo na zona. O alarme sonoro será desligado quando não houver mais presença de fogo na zona;

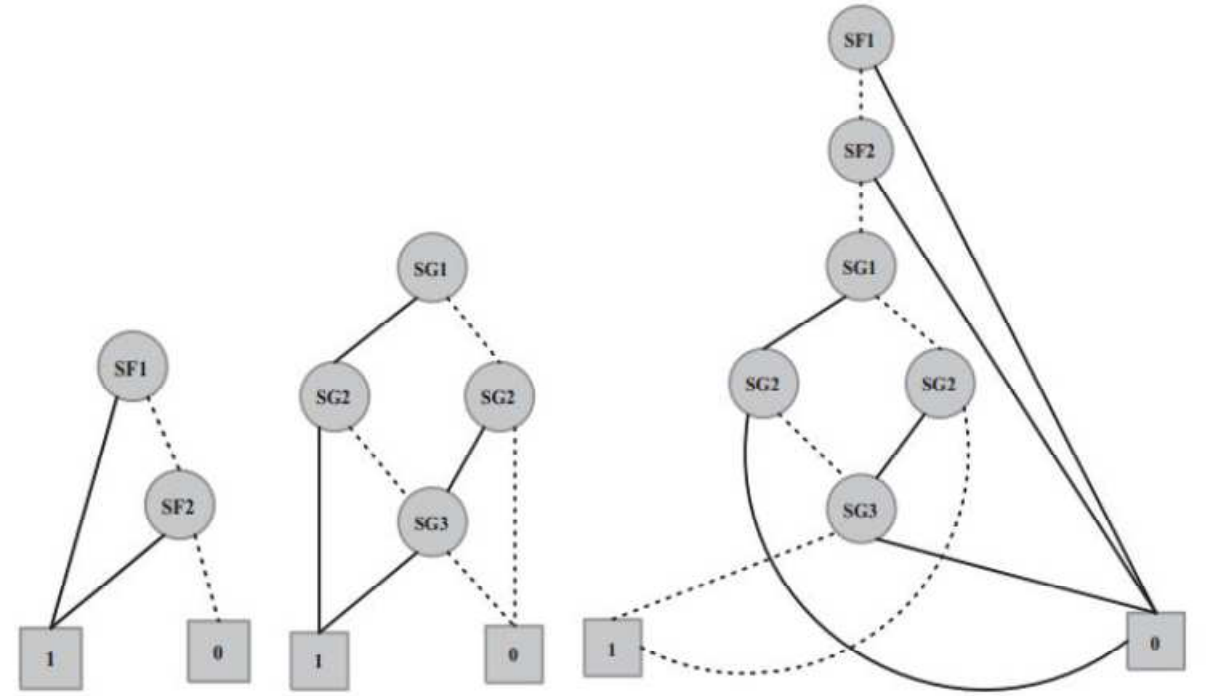
- se for detectado gás na zona então, o alarme sonoro que indica presença de gás é acionado e após 4 segundos a válvula que controla o nível de gás no tanque é desligada e a válvula que mantém o sistema estável é aberta. Esta válvula é utilizada para controlar a quantidade de gás no tanque de forma que não haja danificações às instalações e nem desperdício de material. O alarme sonoro será desligado quando não houver mais presença de gás na zona.

São definidos dois conjuntos de entradas E = {SF1, SF2, SG1, SG2, SG3}, e outro de saídas S = {s_1, s_2, s_3, s_4, s_5}. Cada saída é uma tupla da forma (Y, f), em que Y é o conjunto de variáveis de entrada e f é a função Booleana que define a saída; para este sistema.

- s 1 = DispCO_2 = ({SF1, SF2}, (SF1 ∨ SF2));
- s_2 = AlaFDZ = ⟨{SF1, SF2}, (SF1 ∨ SF2)⟩;
- s_3 = AuxiliaryValve = ⟨{SG1, SG2, SG3}, ((SG1∧ (SG2 ∨ SG3)) ∨(SG2 ∧ SG3)));
- s_4 = AlaGDZ = ⟨{SG1, SG2, SG3}, ((SG1 ∧ (SG2 ∨ SG3)) ∨ (SG2 ∧ SG3))⟩;
- s_5 = Valve = ({SF1, SF2, SG1, SG2, SG3}, (∼(SF1 ∨ SF2) ∧ ∼((SG1 ∧ (SG2 ∨ SG3)) ∨ (SG2 ∧ SG3)))).

São gerados apenas três ROBDDs para as cinco saídas, isto acontece porque a função booleana para as saídas DispCO_2 e AlaFDZ é a mesma. O mesmo ocorre com a função para as saídas AuxiliaryValve e AlaGDZ.

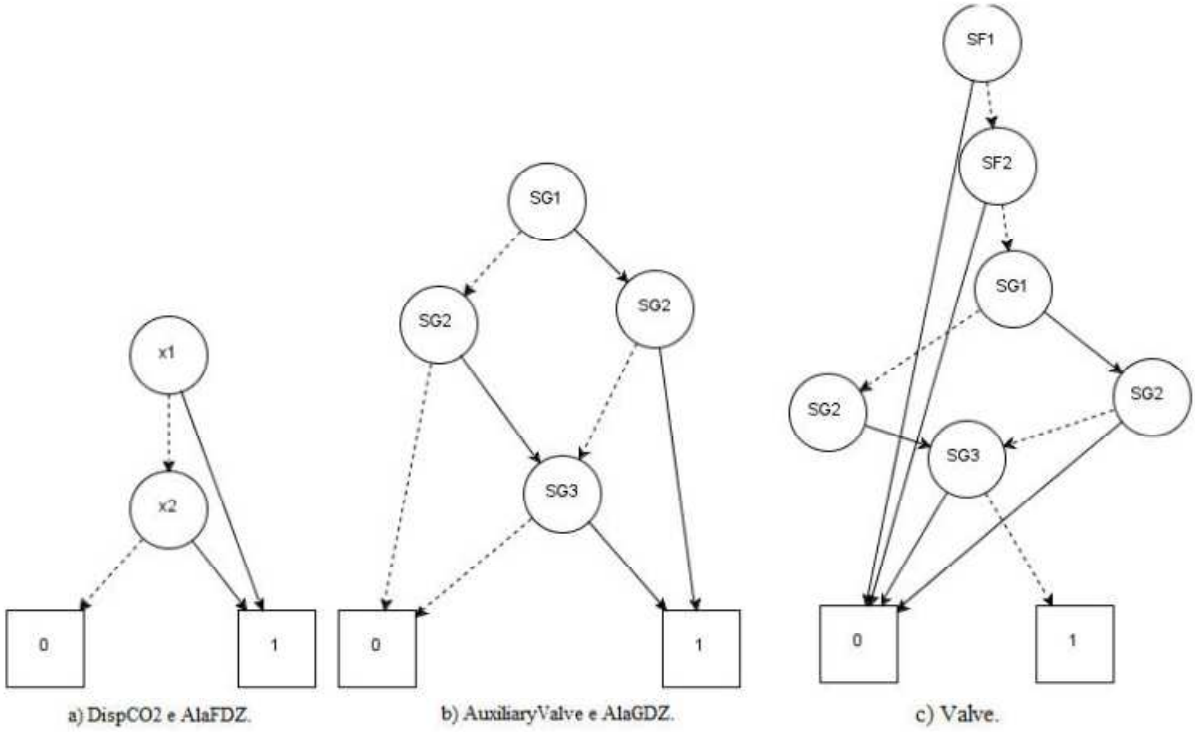
Figura 12: ROBDDs para as saídas do sistema de detecção de fogo ou gás.



(a) DispCO2 e AlaFDZ (b) AuxiliaryValve e AlaGDZ (c) Valve

Fonte: Oliveira (2014)

Figura 13: ROBDDs para as saídas do sistema de detecção de fogo ou gás, gerados pela ferramenta desenvolvida.



Fonte: Autoria própria.

Na Figura 12, é visualizado os ROBDDs para as saídas do sistema de detecção de fogo ou gás gerados em ordenação estática, dada em ordem crescente das variáveis, e no ROBDD gerado pela a ferramenta desenvolvida, com ordenação dinâmica, na Figura 13, percebe-se semelhança nos grafos. Mesmo utilizando uma heurística de ordenação dinâmica (window) a ferramenta obteve ROBDDs com a mesma quantidade de caminhos da raiz até os nós terminais. Tomando como exemplo a representação da AuxiliaryValve e AlaGDZ a ferramenta fez a permutação para as variáveis (SG1<SG2<SG3, SG1<SG3<SG2, SG3<SG1<SG2, SG3<SG2<SG1, SG2<SG1<SG3, SG2<SG3<SG1), após a permutação, é gerada para cada ordenação um ROBDD, a fim de comparar e escolher aquele que tiver o menor número de caminhos da raiz aos nós terminais, consequentemente a melhor representação, que neste caso foi o ROBDD com a ordenação SG1<SG2<SG3.

5.2 SISTEMA DE CONTROLE DE NÍVEL DE TANQUE

Para este estudo de caso, segue estas especificações: o controle de nível de tanque é realizado através do acionamento de um sistema de drenagem composto por uma válvula de escape e uma bomba de sucção. O sistema pode funcionar de quatro maneiras:

- modo 1: completamente manual;
- modo 2: completamente automático;
- modo 3: bomba manual e válvula automática;
- modo 4: bomba automática e válvula manual.

O modo de funcionamento desejado é escolhido pela combinação dos estados de duas chaves, Automatico e Automatico2, que são utilizados para selecionar, o modo automático ou manual para o acionamento da bomba e da válvula respectivamente. Com a planta funcionando no modo 1, completamente manual, tanto a válvula quanto a bomba são comandadas independentemente pelo operador. Isto é feito através das chaves Abre e Fecha, para a válvula, e Liga e Desliga para a bomba. No modo 2, completamente automático, o estado dos sensores de nível determinam as ações que serão realizadas.

Os sinais dos sensores são Nivel_Muito_Baixo, Nivel_Baixo e Nivel_Muito_Alto. Nos modos em que a bomba ou a válvula estão no modo manual as ações realizadas são função da combinação do estado dos sensores e dos comandos do operador. Existem também ações de emergência que são tomadas independentemente do modo selecionado, e que são combinadas com as ações de qualquer modo escolhido para determinar o estado final do sistema. Uma descrição das possíveis combinações para os sinais de entrada e as correspondentes ações que devem ser efetuadas pelo sistema é apresentada a seguir.

Independente do modo de funcionamento selecionado, se Nivel_Muito_Baixo apresentar o valor verdadeiro por 5s consecutivos a ação tomada pelo SIS deve ser, desligar a bomba e fechar a válvula. Se Nivel_Muito_Alto apresentar o valor verdadeiro por 5s consecutivos, abrir a válvula. Na descrição dos modos seguintes, os estados apresentados serão combinações entre as ações dos respectivos modos e das ações de emergência. Para diferenciar as ações de emergência elas serão apresentadas em negrito.

No Modo 1, o comportamento dos dois dispositivos é determinado pelo operador. No entanto, se for verificado alguma das combinações de sinais consideradas críticas, o sistema

executará as ações de emergência programadas.

No Modo 2, se Nivel_Baixo apresentar o valor verdadeiro por 5s consecutivos, desligar a bomba e fechar a válvula. Caso Nivel_Muito_Alto apresentar o valor verdadeiro por 5s consecutivos, ligar a bomba e **abrir a válvula**.

No Modo 3, se Nivel_Muito_Baixo for verdadeiro por 5s consecutivos, **desligar a bomba e fechar a válvula**. Caso Nivel_Baixo for verdadeiro por 5s consecutivos, a ação do sistema será fechar a válvula e o estado da bomba será determinado pelo operador. Se Nivel_Muito_Alto for verdadeiro por 5s consecutivos, o estado da bomba será determinado pelo operador e a ação do sistema será **abrir a válvula**.

No Modo 4, se Nivel_Muito_Baixo for verdadeiro por 5s consecutivos, **desligar a bomba e fechar a válvula**. Se Nivel_Baixo for verdadeiro por 5s consecutivos, a ação do sistema será desligar a bomba e o estado da válvula será determinado pelo operador. Se Nivel_Muito_Baixo apresentar o valor verdadeiro por 5s consecutivos, ligar a bomba e abrir a válvula.

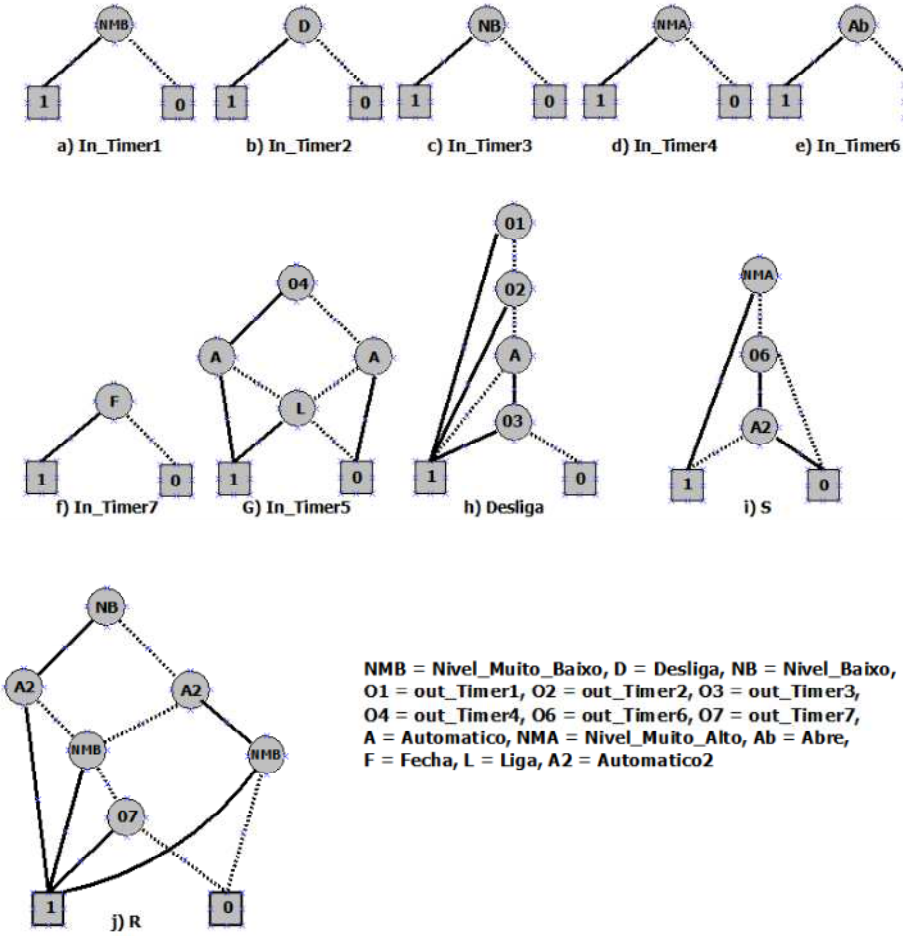
São especificadas para este sistema um conjunto de entradas E = {Nivel_Muito_Baixo, Desliga, Automatico, Nivel_Baixo, Nivel_Muito_Alto, Liga, Abre, Automatico2, Fecha} e um conjunto de saídas S = {s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}}. Cada saída é uma tupla da forma (Y, f), em que Y é o conjunto de variáveis de entrada e f é a função Booleana que define a saída.

- s_1 = In_Timer1 = ({Nivel_Muito_Baixo}, (Nivel_Muito_Baixo));
- s_2 = In_Timer2 = ({Desliga}, (Desliga));
- s_3 = In_Timer3 = ({Nivel_Baixo}, (Nivel_Baixo));
- s_4 = Desliga = ({Nivel_Muito_Baixo, Desliga, Automatico, Nivel_Baixo}, ((out_Timer1) ∨ (out_Timer2 ∨ ~Automatico) ∨ (out_Timer3 ∧ Automatico)));
- s_5 = In_Timer4 = ({Nivel_Muito_Alto}, (Nivel_Muito_Alto));
- s_6 = In_Timer5 = ({Nivel_Muito_Alto, Automatico, Liga}, (((out_Timer4 ∧ Automatic) ∨ (!Automatic ∧ Liga))));
- s_7 = In_Timer6 = ({Abre}, (Abre));
- s_8 = In_Timer7 = ({Fecha}, (Fecha));
- s_9 = S = ({Nivel_Muito_Alto, Abre, Automatico2}, ((Nivel_Muito_Alto) or (out_Timer6 and !Automatico2)));

– s_{10} = R = ((Nivel_Baixo, Automatico2, Nivel_Muito_Baixo, Fecha}, ((Nivel_Baixo ∧ Automatico2) ∨ (out_Timer7 ∧ ~ Automatico2) ∨ (Nivel_Muito_Baixo))).

Para cada função booleana que determina uma saída, são gerados um ROBDD que são apresentados.

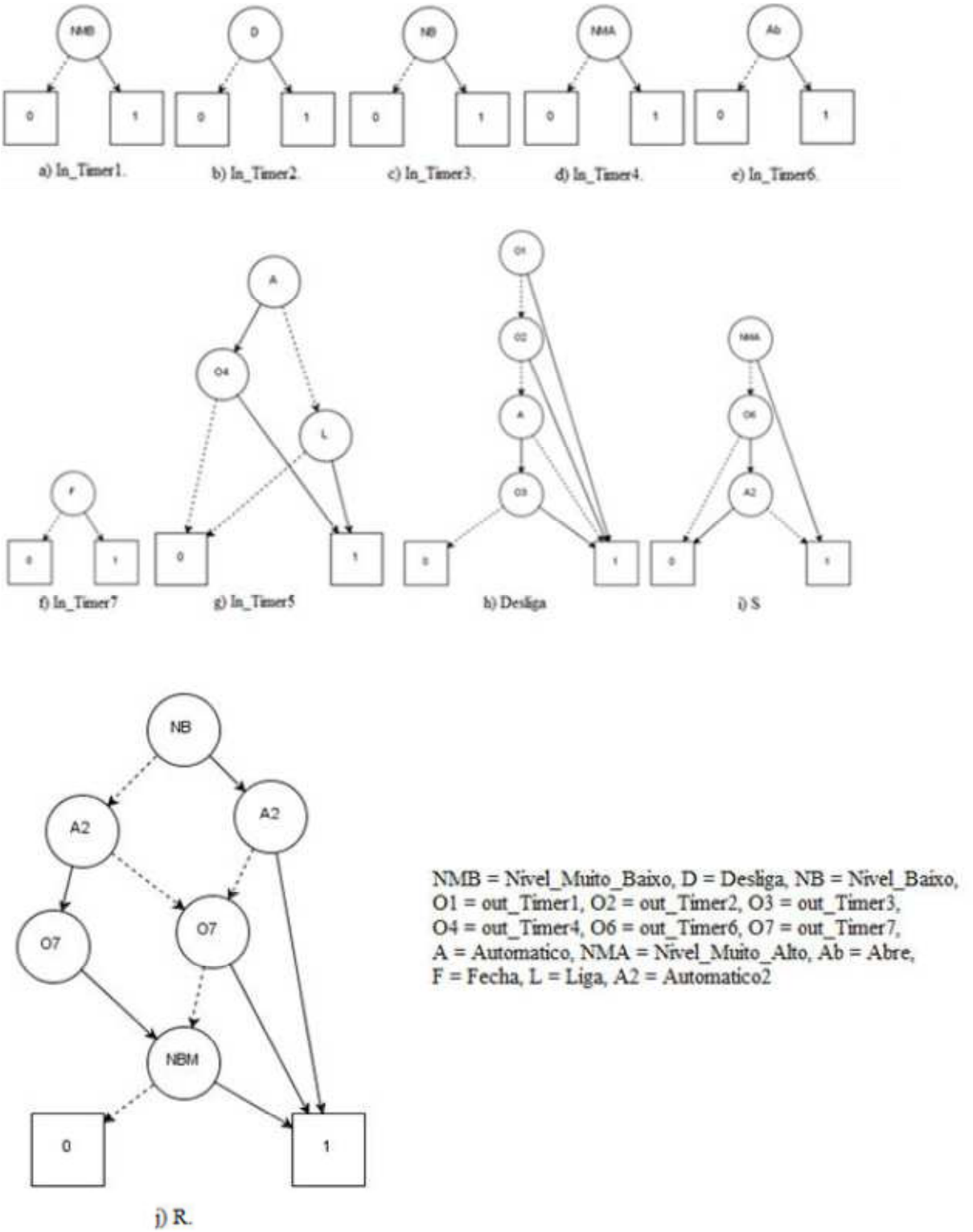
Figura 14: ROBDDs para as saídas do sistema de controle de nível de um tanque.



Fonte: Oliveira (2014).

Na Figura 14 é visualizado os ROBDDs para as saídas sistema de controle de nível de um tanque, com a ordenação estática, dada em ordem crescente das variáveis, e na Figura 15 os ROBDDs gerados pela a ferramenta desenvolvida, percebe-se semelhança nos grafos menos no grafo da saída In_Timer5 na Figura 15(g), este tem uma redução no número de caminhos da raiz até os nós terminais (de 6 para 4) do que na Figura 14(g), cona Figura 16. Nas outras saídas não há alteração no tamanho do BDD.

Figura 15: ROBDDs para as saídas do sistema de controle de nível de um tanque obtido pela ferramenta desenvolvida.



Fonte: Autoria própria.

Figura 16: Todos os caminhos possíveis para os ROBDDs das figuras 14(g) e 15(g) .

f (A, O4, L)	*In_Timer5*
f(1,1, -)	1
f(1,0, 1)	1
f(1, 0, 0)	0
f(0, 1 , -)	0
f(0, 0 , 1)	1
f(0, 0 , 0)	0

a) *In_Timer5* da Figura 14(g) .

f (A, O4, L)	*In_Timer5*
f(1, 1, -)	1
f(1, 0, -)	0
f(0, -, 1)	1
f(0, - , 0)	0

b) *In_Timer5* da Figura 15(g).

Fonte: Autoria própria.

CONSIDERAÇÕES FINAIS

Este trabalho se propôs a desenvolver uma ferramenta gráfica para a geração de ROBDDs que utilizasse a heurística window, com o intuito de avaliar esta heurística. Para tanto foram feitas pesquisas na literatura acerca de ROBDDs, e a heurística window. Para avaliar esta foram realizados estudos de casos de especificações booleanas de sistemas industriais, a fim de comparar os resultados obtidos pela ferramenta desenvolvida com os de OLIVEIRA (2014), que utilizou ordenação estática, dada em ordem crescente das variáveis para gerar os ROBDDs.

Apesar da ferramenta só ter sido aplicada nestes estudos de caso, esta mostrou potencial, uma vez que apresentou semelhanças e melhoria em um dos casos, onde houve uma redução no número de caminhos de 6 para 4 no ROBDD gerado pela ferramenta da saída. Os estudos de casos realizados neste trabalho foram:

- No primeiro estudo de caso, no sistema de segurança para detecção de fogo ou gás: a ferramenta obteve resultados semelhantes, não houve uma redução no tamanho dos ROBDDs.
- No segundo estudo de caso, no sistema de controle do nível de um tanque: a ferramenta obteve em uma das saídas (In_Timer5), redução no número de caminhos e na quantidade de nós do ROBDD gerado.

No que este trabalho se propôs, que era desenvolver uma ferramenta gráfica para geração de ROBDDs usando a heurística window, para avaliar a heurística window, através dos estudos de casos citados a cima, mostra que os objetivos foram atingidos. Deixando como sugestões como trabalhos futuros:

- A realização de mais estudos de casos para avaliar a viabilidade da heurística window, com funções com uma grande quantidade de variáveis.
- Extrair casos de teste a partir dos ROBDD gerados.
- Aplicar a heurística window em conjunto com uma heurística estática, para tentar obter uma melhor ordenação.

REFERÊNCIAS

ANDERSEN, H. R. **Introduction to Binary Decision Diagrams**, Department of Information Technology, Technical University of Denmark Building 344, DK-2800 Lyngby, Denmark, 1997.

BRACE, Karl S. et. al. **Efficient implementation of a BDD package.** In: Proceedings of the 27th ACM/IEEE design automation conference. ACM, 1990. pp. 40-45.

BRYANT, Randal E. **Graph-Based Algorithms for Boolean Function Manipulation**. IEEE Transactions on Computers, C-35-8, pp. 677-691, August, 1986.

_____. **Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams**, School of Computer Science Carnegie Mellon University Pittsburgh, 1992.

DRECHSLER, Rolf. **Evaluation of static variable ordering heuristics for MDD construction [multi-valued decision diagrams]**. In: Multiple-Valued Logic, 2002. ISMVL 2002. Proceedings 32nd IEEE International Symposium on. IEEE, 2002. p. 254-260.

ECLIPSE FOUNDATION. **Eclipse IDE for Java Developers**. Disponível em: https://eclipse.org/downloads/packages/eclipse-ide-java-developers/lunasr1. Acesso em: 21 de novembro de 2014.

FERNANDES, Ricardo Hernandes; NASCIMENTO, Francisco Assis Moreira do. **Processamento Distribuído de Diagramas de Decisão Binária.** ERAD 2002 São Leopoldo.

FLORES, Paulo. **Utilização de Diagramas de Decisão Binária (BDDs) para Representar Máquinas de Estado (FSMs) e Verificar Fórmulas de Lógica Temporal (CTL).** INESC/IST 1996.

FUJITA, Masahiro; MATSUNAGA, Yusuke; KAKUDA, Taeko. **On variable ordering of binary decision diagrams for the application of multi-level logic synthesis**. In: Proceedings of the conference on European design automation. IEEE Computer Society Press, 1991. p. 50-54.

FUJII, Hiroshige; OOTOMO, Goichi; HORI, Chikahiro. **Interleaving based variable ordering methods for ordered binary decision diagrams.** In: Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design. IEEE Computer Society Press, 1993. p. 38-41.

JAIN, Jawahar; ADAMS, William; FUJITA, Masahiro. **Sampling schemes for computing OBDD variable orderings**. In: Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design. ACM, 1998. p. 631-638.

JGRAPH Ltd. **JGraphX - Open Source Java Component**. Disponivel em: https://github.com/jgraph/jgraphx. Acesso em: 21 de novembro de 2014.

MALIK, S. et. al. **Logic Verification Using Binary Decision Diagrams in a Logic Synthesis Environment**. International Conference on CAD: pp.6-9, 1988.

MARQUES, Felipe de Souza**. Um Algoritmo Formal para remoção de redundâncias**. Dissertação de mestrado. Porto Alegre, PPGC da UFRGS, 2003.

MEINEL, Christoph; SACK, Harald; SCHILLINGS, Volker. **VisBDD-A Web-based Visualization Framework for OBDD Algorithms**. In: Proceedings of the IEEE/ACM Int. Workshop on logic Syntheses (IWLS) 2002. p. 385-390.

MINATO, Shin-Ichi. **Binary decision diagrams and applications for VLSI CAD**. Kluwer International series in Engineering and Computer Science. Boston, Dordrecht, London: Kluwer academic publishers 1996.

MIRANDA, Leonardo de et. al.. **BDD's (Diagramas de Decisão Binária).** Porto Alegre, Universidade Federal do Rio Grande do Sul – UFRGS, 2006.

OLIVEIRA, Kézia Vasconcelos. **Geração e execução automática de testes para p programas de controladores lógicos programáveis para sistemas instrumentados de s segurança.** Cidade**:** Campina Grande – PB, Universidade Federal de Campina Grande Centro de Engenharia Elétrica, 2014.

OPEN SOURCE INITIATIVE. **The BSD 3-Clause License.** Acesso em: 17 de novembro de 2014, disponível em: http://opensource.org/licenses/BSD-3-Clause.

ORACLE Corporation. **Java Platform (JDK) 8u25**. Disponível em: http://www.oracle.com/technetwork/java/javase/downloads/index.html acesso em: 21 de novembro de 2014.

RICE, Michael; KULHARI, Sanjay. **A Survey of Static Variable Ordering Heuristics for Efficient BDD/MDD Construction**. 2008. Disponível em: http://alumni.cs.ucr.edu/~skulhari/StaticHeuristics.pdf. Acesso em: 07 de julho de 2014.

RUDELL, Richard. **Dynamic variable ordering for ordered binary decision diagrams**. In: Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design. IEEE Computer Society Press, 1993. p. 42-47.

SENA, Adriano; TORRES, Martha. **EasyKarnaugh 3.0 Uma ferramenta computacional para o auxílio no ensino de Mapas de Karnaugh em Lógica Digital**. In: WEIXVII Workshop sobre Educação em Computação. 2008.

VIDAL, Jorgiano Márcio Bruno. **Ordenação inicial de BDDs para a verificação automática de sistemas de transição finita**. Natal, Rio Grande do Norte, Fevereiro de 2002.