

1 OPIS PROGRAMU

Program „Minimisation 2.1” służy do minimalizacji funkcji logicznych metodą ekspansji w sposób zachłanny. To znaczy, że ten program nie gwarantuje rozwiązania najlepszego.

Jako dane wejściowe program przyjmuje dwa zbiory liczb w systemie dziesiętnym: zbiór prawdy i zbiór nieprawdy. Program również daje możliwość wprowadzić jako jeden z wektorów zbiór wartości nieokreślonych, wtedy pozostałe wartości zostaną wygenerowane automatycznie, lub od razu automatycznie wygenerować wektor na podstawie drugiego.

Na wyjściu program wypisuje funkcji zminimalizowane i proces minimalizacji. Istnieje możliwość zapisywania wyniku do pliku.

2 OPIS DZIAŁANIA PROGRAMU

2.1 Opis danych

Nazwa typu danych	Pola	Opis
<code>enum matrix_type</code> {F, R, Z, D}	Typ macierzy: F – macierz jedynek, R – macierz zer, Z – wypełnić na podstawie drugiej macierzy, D – macierz wartości nieokreślonych.	
<code>vector_with_indikator</code>	<code>set<int> vec</code>	Zbiór wejściowych wartości.
	<code>matrix_type</code> indikator	Typ zbioru wejściowego.
<code>matrix_b</code>	<code>matrix</code> matr	Kostka.
	<code>vector<vector<int> ></code> implicants	Implikanty z tej kostki.
<code>vec_of_impl</code>	<code>int</code> num_of_line_from_F	Numer kolumny zbioru implikant w macierzy jedynkowej.
	<code>vector<int> vec</code>	Zbiór implikant z kostki.
<code>matrix</code>	<code>vector<QBitArray> array</code>	Tablica dwuwymiarowa, składająca się z wektora tablic bitowych. Wykorzystanie tablic bitowych oszczędza pamięć, używaną przez macierze do 64 razy odnośnie tablic integerów.

	<code>int numOfRows</code>	Liczba wierszy macierzy.
	<code>int numOfColumns</code>	Liczba kolumn macierzy.

2.2 Opis funkcji

Prototyp funkcji	Opis
<code>bool cmp_for_same_vec_of_impl</code> (<code>vec_of_impl A</code> , <code>vec_of_impl B</code>)	Zwraca true, jeżeli wektor <i>A</i> jest równy wektorowi <i>B</i> .
<code>bool cmp_vec_of_impl</code> (<code>vec_of_impl A</code> , <code>vec_of_impl B</code>)	Zwraca true, jeżeli wektor <i>A</i> jest mniejszy od wektora <i>B</i> .
<code>bool current_b_has_this_implicant</code> (<code>const vector<vec_of_impl></code> & <code>vector_of_implicants</code> , <code>const vector<matrix_b></code> & <code>vector_of_b</code> , <code>int row</code> , <code>int column</code>)	Zwraca true, jeżeli kostka o indeksie <i>row</i> z <i>vector_of_implicants</i> zawiera implikantę, która znajduje się w <i>vector_of_b</i> pod indeksem <i>column</i> .
<code>bool vectors_have_collision</code> (<code>vector_with_indikator &vec1</code> , <code>vector_with_indikator &vec2</code>)	Zwraca true, jeżeli dwa podanych wektora mają wspólne wartości.
<code>int invert</code> (<code>const int val</code>)	Zwraca 1, jeżeli podana wartość jest równa 0 i odwrotnie.
<code>matrix createMatrix</code> (<code>int numOfLines</code> , <code>int numOfColumns</code>)	Tworzy macierz o <i>numOfLines</i> wierszach i <i>numOfColumns</i> kolumnach.
<code>matrix get_result_matrix</code> (<code>const vector<vec_of_impl></code> & <code>vector_of_implicants</code> , <code>const vector<matrix></code> & <i>fr</i> , <code>const vector<matrix_b></code> & <code>vector_of_b</code>)	Zwraca macierz końcową implikant, zbudowaną na podstawie wejściowych parametrów(algorytm 'get_result_matrix').
<code>matrix make_bin_matrix_by_dec_vec</code> (<code>const set<int></code> & <i>vec</i> , <code>int max</code>)	Zwraca macierz, wypełnioną wartościami ze zbioru <i>vec</i> w postaci dwójkowej. <i>max</i> To maksymalna wartość z dwóch wektorów wejściowych.
<code>string print_implicant_as_string</code> (<code>vector<matrix_b></code> <i>mat_b</i> , <code>int n_line</code> , <code>int j</code> , <code>vector<matrix></code> <i>fr</i>)	Zwraca napis z implkantą <i>j</i> z linii <i>n_line</i> kostki <i>mat_b</i> .
<code>string show_matrix_as_string</code> (<code>const matrix</code> & <i>array</i>)	Zwraca wypisaną macierz <i>array</i> w postaci napisu.

set<int> make_by_other (const set<int> &vec1, const set<int> &vec2)	Tworzy zbiór wartości od 0 do maksymalnej wartości z <i>vec1</i> i <i>vec2</i> , nie włączając do niego wartości, zawartych w tych wektorach na podstawie algorytmu 'make_by_other'.
set<int> make_by_other (const set<int> &vec1)	Tworzy zbiór wartości od 0 do maksymalnej wartości z <i>vec1</i> , nie włączając do niego wartości, zawartych w tym wektorze na podstawie algorytmu 'make_by_other'.
vector<int> toBinary (int <i>dec</i>)	Zwraca liczbę <i>dec</i> w postaci dwójkowej jako wektor zero-jedynkowy.
vector<matrix> get_f_and_r (vector_with_indikator &vec1, vector_with_indikator &vec2)	Zwraca wektor, zawierający macierzy jedynek i zer, pobudowane na podstawie wektorów wejściowych (algorytm 'get_f_and_r').
vector<matrix> get_matrix_b (const vector<matrix> &fr)	Zwraca wektor kostek dla tablicy prawdy, podaną jako parametr wejściowy (algorytm 'get_matrix_b').
vector<string> get_finally_functions (const vector<matrix> &fr, const matrix &result_matr, vector<vec_of_impl> &vector_of_implicants)	Zwraca wektor napisów, każdy z których jest jedną z funkcji wynikowych, pobudowane na podstawie wejściowych parametrów (algorytm 'get_finally_functions').
vector<vec_of_impl> get_vector_of_implicants (IN const vector<matrix> &fr, OUT vector<matrix_b> &mat_b)	Zwraca wektor numerów minimalnych implikant z macierzy funkcji o tablicę prawdy <i>fr</i> i zapisuje do <i>mat_b</i> kostki.
vector<vector<int> > cross_out_implicants (vector<vector<int> > <i>in_ones</i>)	Zwraca wektor implikant <i>in_ones</i> z wykreślonymi implikantami, niepotrzebnymi do wyznaczenia minimalnego pokrycia kolumnowego (algorytm 'cross_out_implicants').
vector<vector<int> > get_ones_positions_from_matrix (const matrix &B)	Zwraca wektor, zawierający numery kolumn, które mają wartość równą 1 dla każdego wierszu macierzy <i>B</i> ('algorytm get_ones_positions_from_matrix').

vector<vector<int> > only_minimal_implicants (vector<vector<int> > <i>in_im</i>)	Zwraca wektor implikant minimalnych z wektora wejściowego.
vector<vector<int> > open_brackets (vector<vector<int> > <i>in_ones</i>)	Zwraca wektor implikant <i>in_ones</i> po otwarciu nawiasów, to znaczy przekształca funkcji z wektora do postaci dysjunkcji (algorytm 'open_brackets').
void print_implicant (vector<matrix_b> <i>mat_b</i> , int <i>n_line</i> , int <i>j</i> , vector<matrix> <i>fr</i>)	Wypisuje implikantę <i>j</i> z linii <i>n_line</i> kostki <i>mat_b</i> .
void show_finally_functions (const set<int> & <i>vec1</i> , const set<int> & <i>vec2</i> , char * <i>chose</i>)	Wypisuje wynik działania programu dla wejściowych danych do standardowego potoku wyjścia.
void showMatrix (const matrix & <i>array</i>)	Wypisuje macierz <i>array</i> do standardowego potoku wyjścia.
void write_to_file (const string & <i>in_string</i> , const char * <i>filename</i>)	Zapisuje <i>in_string</i> do pliku o nazwie <i>filename</i> .
void write_to_file (const set<int> & <i>vec1</i> , const set<int> & <i>vec2</i> , char * <i>chose</i> , const char * <i>filename</i>)	Zapisuje wynik działania programu dla wejściowych danych do pliku o nazwie <i>filename</i> .