WYŻSZA SZKOŁA INFORMATYKI STOSOWANEJ I ZARZĄDZANIA
pod auspicjami Polskiej Akademii Nauk

WYDZIAŁ INFORMATYKI

STUDIA I STOPNIA
(INŻYNIERSKIE)

Kierunek INFORMATYKA

PRACA DYPLOMOWA

Andrzej Kisiel

UOGÓLNIANIE REGUŁ DECYZYJNYCH BINARNYCH TABLIC DANYCH METODĄ UZUPEŁNIANIA FUNKCJI BOOLOWSKICH

DISCOVERING DECISION RULES OF BINARY DATA TABLES USING COMPLEMENT OF BOOLEAN FUNCTIONS

Praca wykonana pod kierunkiem:

prof. dr hab. Tadeusza Łuby

WARSZAWA, 2012 r.

Autor: **Andrzej Kisiel**

Tytuł: **UOGÓLNIANIE REGUŁ DECYZYJNYCH BINARNYCH TABLIC DANYCH METODĄ UZUPEŁNIANIA FUNKCJI BOOLOWSKICH**

Rok akademicki: **2011/2012**

Dziekan Wydziału: **dr inż. Jarosław Sikorski**

Kierunek: **INFORMATYKA**

Specjalność: **BAZY DANYCH**

Opiekun pracy: **prof. dr hab. Tadeusz Łuba**

Konsultant: .....

Kwalifikuję do złożenia jako dyplomową pracę inżynierską na Wydziale Informatyki WSISiZ.

Ocena pracy (słownie): .....
(skala ocen: bardzo dobra, dobra i pół, dobra, dostateczna i pół, dostateczna)

....
(data) (podpis opiekuna pracy)

Pracę złożono w ..... **egzemplarzach dnia** .....

.....
(pieczątka wydziału) (podpis kierownika dziekanatu)

Tytuł naukowy, imię i nazwisko recenzenta pracy:

.....

Ocena pracy (słownie): .....
(skala ocen: bardzo dobra, dobra i pół, dobra, dostateczna i pół, dostateczna)

....
(data) (podpis kierownika dziekanatu)

Dopuszczam pracę do obrony.

....
(data) (podpis dziekana)

OŚWIADCZENIE AUTORA PRACY DYPLOMOWEJ

Świadom(a) odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami prawa, w szczególności z ustawą z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. z 1994 r. Nr 24, poz. 83 – tekst pierwotny i Dz. U. z 2000 r. Nr 80, poz. 904 – tekst jednolity, z późniejszymi zmianami).

Oświadczam, że wszystkie narzędzia informatyczne zastosowane do wykonania niniejszej pracy wykorzystałem(am) zgodnie z obowiązującymi przepisami prawa w zakresie ochrony własności intelektualnej i przemysłowej.

Oświadczam, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w szkole wyższej.

Jednocześnie stwierdzam, że niniejszy tekst pracy dyplomowej jest identyczny z załączoną wersją elektroniczną.

.....	.....
Data	Podpis autora pracy

SPIS TREŚCI

WSTĘP **4**

1. WPROWADZENIE DO TEORII **6**

1.1. System informacyjny, tablica decyzyjna, funkcje boolowskie *6*
1.2. Minimalizacja funkcji boolowskich *10*

2. EKSPANSJA (UOGÓLNIANIE REGUŁY DECYZYJNEJ) **12**

2.1. Idea ekspansji *12*
2.2. Macierze F i R *13*
2.3. Macierz blokująca B *14*
2.4. Minimalne pokrycie kolumnowe macierzy blokującej *15*
2.5. Tablica implikantów prostych *16*

3. UZUPEŁNIANIE (REDUKCJA ZBIORU REGUŁ DECYZYJNYCH) **17**

3.1. Idea uzupełniania *17*
3.2. Rozkład Shannona *17*
3.3. Opis algorytmu *18*

4. JEDNOLITA METODA OBLICZANIA REDUKTÓW I REGUŁ DECYZYJNYCH **21**

5. PROGRAM INSTANTRS **26**

5.1. Wprowadzenie *26*
5.2. Budowa i przegląd funkcji programu *26*
5.3. Wyniki eksperymentów *32*

WNIOSKI KOŃCOWE **40**

WYKAZ TABEL I RYSUNKÓW **41**

BIBLIOGRAFIA **42**

STRESZCZENIE **44**

ABSTRACT **44**

DODATEK **45**

Wstęp

Uogólnianie reguł decyzyjnych jest procesem stosowanym w dwóch odrębnych dziedzinach wiedzy, a mianowicie w zagadnieniach związanych z klasyfikacją danych (maszynowe uczenie, eksploracja danych itp.) [1], [2], [12], [13], [17] oraz w zagadnieniach związanych z optymalizacją i syntezą logiczną układów cyfrowych [3], [9], [10]. W obu przypadkach jest to problem polegający na tworzeniu reguł (wyrażeń boolowskich) reprezentujących pierwotne obiekty zapisane w tablicach danych (tablicach prawdy układów logicznych), przy czym w przypadku układów logicznych procesy takie są określane mianem minimalizacji funkcji boolowskich. Uzyskiwane w wyniku takiego procesu reguły są wykorzystywane do klasyfikacji danych albo do kolejnych etapów optymalizacji logicznej (w przypadku układów cyfrowych). Oczywiście dane wejściowe algorytmów uogólniania reguł są w obu przypadkach zasadniczo różne. Dla tablic danych mamy do czynienia z wielowartościowymi atrybutami warunkowymi i wielowartościowymi atrybutami decyzyjnymi. Dla tablic prawdy wartości argumentów i wartości funkcji są binarne. Jednocześnie całkowicie inaczej są interpretowane tzw. wartości nieokreślone. W tablicy danych wartość nieokreślona atrybutu warunkowego oznacza, że wartość tego atrybutu nie została ustalona [6], [12], a w przypadku tablic funkcji logicznych nieokreśloność argumentu wektora wejściowego oznacza występowanie w specyfikacji wektorów o wszystkich możliwych wartościach danego argumentu [3], [10].

Na abstrakcyjnym poziomie algorytmów eksploracja danych polega m.in. na tzw. uogólnianiu/generacji reguł decyzyjnych, redukcji atrybutów, hierarchicznym podejmowaniu decyzji. Można wykazać, że algorytmy te są odpowiednikami algorytmów syntezy logicznej, a w szczególności tych które powstały w czasie ostatnich 30 lat. Na przykład generacja/uogólnianie reguł decyzyjnych – jest to typowa procedura stosowana w eksploracji danych i odpowiada minimalizacji funkcji boolowskiej, redukcja atrybutów odpowiada redukcji argumentów, natomiast hierarchiczne podejmowanie decyzji jest to dekompozycja funkcjonalna.

Można również przypuszczać, że stosowane w syntezie logicznej metody i algorytmy minimalizacji okażą się skuteczniejsze niż analogiczne algorytmy generacji reguł stosowane w analizie danych [2], [7].

Celem niniejszej pracy było stworzenie narzędzia komputerowego umożliwiającego analizę i porównanie skuteczności algorytmów syntezy logicznej z algorytmami eksploracji danych. Z tych powodów w opracowanym i zaimplementowanym narzędziu (InstantRS) zastosowano dwie główne procedury, a mianowicie procedurę ekspansji oraz procedurę redukcji atrybutów. Procedury te połączono w taki sposób, aby wyniki jednej mogły być odpowiednio stosowane w drugiej. Jednocześnie jako główny algorytm obliczeniowy zastosowano algorytm uzupełniania funkcji boolowskich, dla którego już w poprzednich pracach wykazano rewelacyjną skuteczność w analizie danych.

Ponadto, w celu uzyskania jednolitości porównań wyników dla danych z różnych dziedzin (układy logiczne vs. eksploracja danych) założono wykonywanie eksperymentów na binarnych tablicach danych. Takie dane są powszechnie stosowane w specyfikacjach funkcji boolowskich, a jednocześnie można je znaleźć w wielu zastosowaniach charakterystycznych dla eksploracji danych. Typowym przykładem są dane reprezentujące głosy kongresmenów w sprawie 16 kluczowych problemów zebrane w roku 1984 (binarne odpowiedzi na pytania: tak lub nie), powszechnie stosowane w programów eksploracji danych i Machine Learning [5].

1. Wprowadzenie do teorii

1.1. System informacyjny, tablica decyzyjna, funkcje boolowskie

Za punkt wyjścia do dalszych rozważań posłuży pojęcie systemu informacyjnego. System informacyjny [13] to uporządkowana czwórka (*U, A, V, f*), gdzie: *U* jest niepustym i skończonym zbiorem obiektów zwanym uniwersum, *A* jest niepustym i skończonym zbiorem atrybutów, $V = U_{a \in A} V_a$ (gdzie V_a jest dziedziną atrybutu $a \in A$) oraz $f : U \times A \to V$, przy czym $\forall a \in A, x \in U, f(a, x) \in V_a$, jest funkcją informacyjną odwzorowującą zbiór obiektów opisany przez dany zbiór atrybutów na konkretne wartości tych atrybutów. Jeżeli w systemie informacyjnym wyróżniamy rozłączne zbiory atrybutów warunkowych *C* i atrybutów decyzyjnych *D*, gdzie $A = C \cup D$ to system taki nazywamy tablicą decyzyjną [21]. Zaletą tablicy decyzyjnej jest przejrzysty sposób prezentacji zależności między zbiorem warunków, a zbiorem odpowiadających im decyzji.

Tab. 1.1. Przykładowa tablica reguł decyzyjnych

a	*b*	*c*	*d*	*e*
0	0	0	0	0
1	0	0	1	0
1	1	0	0	1
1	1	1	1	1
1	2	1	0	1
2	1	0	2	2
2	2	2	2	2

Atrybuty

Klasyfikacja (decyzja)

W tablicy reguł decyzyjnych (Tab. 1.1) każdy wiersz reprezentuje obiekt, który jest opisywany wartościami atrybutów *a, b, c, d* oraz jest klasyfikowany w kolumnie *e*. Reguły decyzyjne zapisane w pierwotnej tablicy mają określone wartości atrybutów, a ustalonemu zbiorowi wartości atrybutów odpowiada określona decyzja. Przykładowa reguła dla omawianej tabeli zdefiniowana jest poniższym wyrażeniem:

$(a,1)\ \&\ (b,0)\ \&\ (c,0)\ \&\ (d,1) \to (e,0)$

Tabela 1.2. przedstawia przykładową tablicę decyzyjną opisującą zbiór pacjentów, którzy starali się o uzyskanie recepty na szkła kontaktowe. Tablica składa się z dwudziestu

czterech obiektów oraz czterech atrybutów warunkowych - każda z badanych osób została scharakteryzowana za pomocą czterech cech jakościowych: wieku, wady wzroku, astygmatyzmu oraz łzawienia. Korzystając z wcześniej opisanych definicji można tablicę tę zapisać w postaci:

TD = (U, C, D, V, f), gdzie:

U = {1,2,3 …, 24};

C = {wiek, wada wzroku, astygmatyzm, łzawienie};

D = {szkła};

V_{wiek}= {M, P, S}, cecha porządkowa o 3 wartościach młodość (M), P przedstarczowzroczność (P), starczowzroczność (S);

$V_{wada\ wzroku}$={K, D}, cecha nominalna o 2 wartościach: krótkowzroczność (K), dalekowzroczność (D);

$V_{astygmatyzm}$ ={Tak, Nie}, cecha nominalna o 2 wartościach: występuje, nie występuje;

$V_{łzawienie}$ ={Normalne, Zmniejszone}, cecha nominalna o 2 wartościach: zmniejszone, normalne.

$$f: U \times A \rightarrow V \; np. f(pacjent\ nr\ 1, wiek) = M$$

Tab. 1.2. Dane o 24 pacjentach starających się o szkła kontaktowe [4]

Pacjent	Wiek	Wada	Astygmatyzm	Łzawienie	Szkła
1	M	K	Nie	Zmniejszone	Brak
2	M	K	Nie	Normalne	Miękkie
3	M	K	Tak	Zmniejszone	Brak
4	M	K	Tak	Normalne	Twarde
5	M	D	Nie	Zmniejszone	Brak
6	M	D	Nie	Normalne	Miękkie
7	M	D	Tak	Zmniejszone	Brak
8	M	D	Tak	Normalne	Twarde
9	P	K	Nie	Zmniejszone	Brak
10	P	K	Nie	Normalne	Miękkie
11	P	K	Tak	Zmniejszone	Brak
12	P	K	Tak	Normalne	Twarde
13	P	D	Nie	Zmniejszone	Brak
14	P	D	Nie	Normalne	Miękkie
15	P	D	Tak	Zmniejszone	Brak
16	P	D	Tak	Normalne	Brak
17	S	K	Nie	Zmniejszone	Brak
18	S	K	Nie	Normalne	Brak
19	S	K	Tak	Zmniejszone	Brak

20	S	K	Tak	Normalne	Twarde
21	S	D	Nie	Zmniejszone	Brak
22	S	D	Nie	Normalne	Miękkie
23	S	D	Tak	Zmniejszone	Brak
24	S	D	Tak	Normalne	Brak

Szczególnym przypadkiem tablic decyzyjnych są binarne tablice danych.

Na przykład w tablicy 1.3 zapisane są hipotetyczne dane zebrane przed wyborami prezydenckimi w pewnej republice. Dane te reprezentują odpowiedzi na pytania (tak, nie) uzyskane od 10 respondentów. W tablicy tej odpowiedziom *tak* przyporządkowano wartość 1, odp. *nie* – 0, przy jednoczesnym wyróżnieniu zwolenników ocenianego kandydata na prezydenta atrybutem TAK, a przeciwników, atrybutem NIE.

Tab. 1.3. Przykładowa ankieta

	x_1	x_2	x_3	x_4	x_5	y
1	0	0	0	0	1	tak
2	0	0	0	1	0	tak
3	0	1	0	1	0	tak
4	0	1	1	1	1	tak
5	1	1	0	1	1	tak
6	0	0	1	1	0	nie
7	0	1	0	0	0	nie
8	0	1	1	0	0	nie
9	1	0	0	0	1	nie
10	1	0	1	1	1	nie

Typowym zadaniem w eksploracji danych byłoby w tym przypadku obliczenie uogólnionej reguły decyzyjnej określającej zwolenników tego kandydata dla respondentów o dowolnych odpowiedziach na pytania sondażowe. Dla tablicy 1.3 wywieść można uogólnioną regułę decyzyjną w postaci: $(x_1,0)\&(x_5,1)$ lub $(x_3,0)\&(x_4,1) \rightarrow (y,tak)$

Analogiczny problem występuje w układach logicznych, w których dane zapisuje się w postaci tzw. tablic prawdy.

Tab. 1.4. Przykładowa tablica prawdy

i	x1	x2	X3	*f*
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Tablica prawdy to tablica o n+1 kolumnach oraz m wierszach, w których kolejno zapisywane są wartości sygnałów wejściowych (składowe wektora x) oraz wyjściowa wartość funkcji (y). Łatwo zauważyć, iż struktury tablicy prawdy i opisanej uprzednio tablicy decyzyjnej są izomorficzne. Tablicę prawdy łatwo zinterpretować można jako binarną tablicę danych, gdzie sygnały wejściowe to kolejne atrybuty, wartość funkcji to wynik decyzji, a poszczególne wiersze sygnałów wejściowych to obiekty (przykłady, instancje).

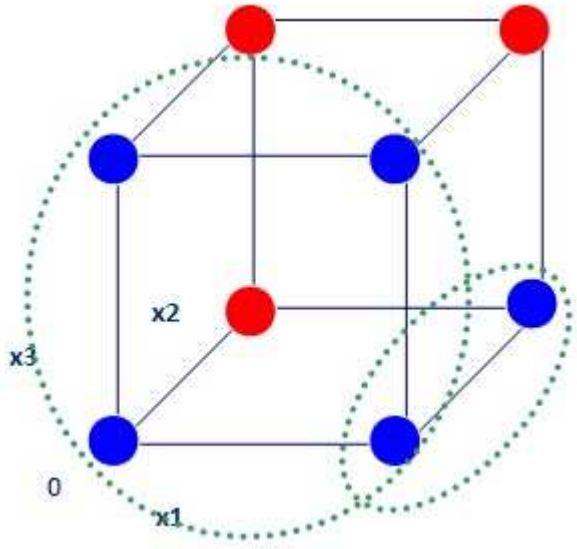
Funkcję opisaną tablicą prawdy można też wyrazić przy pomocy odpowiedniej formuły boolowskiej. Np. dla tablicy 1.4. funkcja przyjmuje następującą postać

$$f = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3$$

W układach logicznych szczególnie istotny jest proces upraszczania wyrażeń boolowskich – na jego potrzeby opracowano wiele zaawansowanych metod syntezy, zwanych metodami minimalizacji funkcji boolowskich. Algorytmy jednej z bardziej znanych metod z tego obszaru - metody Espresso – stanowią podstawę implementacji komputerowego algorytmu programu **InstantRS** (omówionego w dalszej części pracy), który umiejętnie wykorzystuje je w procesie uogólniania reguł decyzyjnych.

1.2. Minimalizacja funkcji boolowskich

Każdy wektor z tablicy prawdy można też zdefiniować jako n-wymiarową „kostkę” (n-krotkę o składowych 0,1,*), przy czym n określone jest długością wektora.



Rys. 1.1. Kostka - opracowanie własne

Zaprezentowana poprzednio przykładowa tablica (Tab.1.4) przekonwertowana do schematu kostek obrazuje za pomocą wielowymiarowego grafu (Rys.1.1) i oznaczonych na nim wierzchołków rozkład wartości funkcji boolowskiej w zależności od wartości sygnałów wejściowych (niebieskie – 1, czerwone – 0).

Minterm – to kostka, która ma wszystkie zmienne wejściowe i wyjściowe określone (zawiera wszystkie literały danej funkcji) oraz przyjmuje wartość 1 dla dokładnie jednej kombinacji wejść danej funkcji. Do każdej funkcji boolowskiej $f(x_{1,}x_{2,}...,x_n)$ z n literałami (zmiennymi boolowskimi) istnieje maksymalnie 2^n mintermów. Można zauważyć, iż wszystkie niebieskie wierzchołki z rysunku stanowią zbiór mintermów analizowanej funkcji. W procesie minimalizacji funkcji boolowskich istotną rolę odgrywają takie pojęcia, jak implikant, a w szczególności implikant prosty.

Implikant funkcji boolowskiej f - jest to iloczyn literałów (czyli zmiennych prostych lub zanegowanych), przy czym dla wszystkich kombinacji wartości zmiennych, dla których implikant przyjmuje wartość 1, również funkcja f jest równa 1. Implikanty opisać można jako zbiory mintermów funkcji boolowskiej (co często jest równoznaczne z brakiem danego literału w definicji implikantu – jego wartość jest wówczas nieokreślona).

Z kolei poszerzanie zbioru pokrywanych mintermów implikantu prowadzi ostatecznie do uzyskania ***implikantu prostego*** – to taki implikant, którego nie da się dalej poszerzać (nie można usunąć kolejnego literału), gdyż skutkowałoby to włączeniem do zakresu pokrywanych przez ten implikant mintermów, mintermy ze zbioru R (zbior mintermów, gdzie wartość funkcji wynosi 0), a tym samym przestałby on być implikantem.

Ideę minimalizacji funkcji boolowskich w uproszczony sposób przedstawiają przerywane elipsy z rysunku 1.1. Jako zasadnicze etapy tego procesu można wyróżnić:

- generowanie implikantów prostych
- generowanie kolejnych rozwiązań funkcji przy użyciu kombinacji implikantów prostych pokrywających zbiór wszystkich mintermów
- znajdowanie najefektywniejszych rozwiązań (tj. najmniej licznych, z implikantów prostych złożonych z minimalnej ilości literałów)

Po zastosowaniu minimalizacji, pierwotna funkcja boolowską z tablicy 1.4 upraszcza się do następującej formuły:

$$f = \bar{x}_2 + x_1\bar{x}_3$$

Łatwo zauważyć bezpośrednią analogię do procesu ugólniania reguł decyzyjnych – w kontekście tablicy decyzyjnej można stwierdzić, iż początkowe 5 reguł udało się uogólnić do 2 reguł, każda o zredukowanej liczbie atrybutów, pokrywających wszystkie obiekty decyzyjne o wartości 1.

2. Ekspansja (uogólnianie reguły decyzyjnej)

2.1. Idea ekspansji

Algorytm ekspansji jest jedną z podstawowych procedur metody Espresso, opracowanej na Uniwersytecie Kalifornijskim (Berkley) przez Roberta Braytona dla celów projektowania układów logicznych [3]. Realizacja procedury ekspansji kostki $k \in F$ dokonuje się zgodnie z zasadą, że składową (jednoznacznie określoną) $k_i \neq *$ kostki $k = (k_1,...,k_n) \in F$ można zastąpić przez $k_i = *$ wówczas, gdy w zbiorze R nie będzie istnieć kostka r taka, że $r \cdot k = (e_1,...,e_n)$, gdzie $e_j = \phi$ dla $j = i$ oraz $e_j \neq \phi$ dla $j \neq i$.

Powyższy zapis można zinterpretować w sposób następujący: celem jest tu to generowanie implikantów prostych pokrywających kostkę k, czyli opisane wyżej maksymalne powiększanie kostki k (maksymalizowanie liczby '*' w definicji kostki), w taki sposób, aby jednocześnie nie pokryć żadnej kostki (wektora) zbioru R (wartość funkcji 0). Jednocześnie, definicję „pokrywania" interpretuje się zgodnie z tablicą Tab.2.1.

Tab. 2.1. Pokrycia [3]

		a		
		0	**1**	*****
b	**0**	$\subseteq$	$\not\subset$	$\subset$
	1	$\not\subset$	$\subseteq$	$\subset$
	*****	$\not\subset$	$\not\subset$	$\subseteq$

Na tej podstawie mówimy, iż kostka *a* pokrywa kostkę *b*, jeśli $\underset{a_i \in a}{\forall} a_i \supseteq b_i$. Dodatkowo, kostka *a* ściśle pokrywa kostkę *b*, jeśli $a \supseteq b \wedge \underset{i}{\exists} a_i \supset b_i$.

W procesie uogólniania reguł decyzyjnych, ekspansja służyć ma możliwie maksymalnej redukcji atrybutów z reguły dotyczącej konkretnego obiektu decyzyjnego (uproszczenie reguły), przy założeniu, iż ich otrzymany zbiór nadal będzie poprawnie determinował wartość decyzji dla tego obiektu. Tak wygenerowane uogólnione reguły posłużą w dalszym etapie do tworzenia minimalnego zbioru reguł pokrywającego całe uniwersum (wszystkie przypadki decyzyjne).

2.2. Macierze F i R

Ekspansja jest procesem bazującym na wykorzystaniu macierzy wygenerowanych ze zbiorów F i R (F – wiersze z wartością funkcji równą 1, R – wiersze z wartością funkcji równą 0).

Rozważmy tablicę decyzyjną zdefiniowaną następująco:

```
0100101 1
1000110 1
1010000 1
1010110 1
1110101 1
1000101 0
1011110 0
1101110 0
1110111 0
```

W powyższym przypadku zbiory F i R opisane są następującymi macierzami:

$$F = \begin{bmatrix} 0100101 \\ 1000110 \\ 1010000 \\ 1010110 \\ 1110101 \end{bmatrix} \quad , \quad R = \begin{bmatrix} 1000101 \\ 1011110 \\ 1101110 \\ 1110111 \end{bmatrix}$$

Przy omawianiu ekspansji w kontekście boolowskich funkcji jednowyjściowych (będących przemiotem niniejszej pracy), wspomnieć należy, iż w przypadku funkcji wielowyjściowych procedurę ekspansji należy zastosować oddzielnie dla każdej kolumny wartości funkcji. Często zdarza się, że powstają wtedy powtarzające się identyczne wiersze, których nadmiar trzeba usunąć na samym początku.

2.3. Macierz blokująca B

Macierz blokującą (kostkę k) definiuje się następująco:

$$B(k,R) = \lfloor b_{ij} \rfloor \quad , \quad 1 \le i \le |R| \quad , \quad 1 \le j \le n$$

w której każdy element $b_{ij} \in \{0,1\}$ przyjmuje wartość:

$$\begin{cases} b_{ij} = 1, \text{ jeśli } k_j = 1 \text{ oraz } r_{ij} = 0 \text{ lub } k_j = 0 \text{ oraz } r_{ij} = 1 \\ b_{ij} = 0, \text{ w pozostałych przypadkach.} \end{cases}$$

Interpretując powyższy zapis można stwierdzić, iż macierz blokująca wyznacza maksymalny dopuszczalny zakres ekspansji kostki k ze zbioru F. Macierz blokująca jest generowana w wyniku przyrównania danej kostki k do każdej kostki zbioru R, i ustawieniu elementów tak utworzonej macierzy na 1 na pozycji, gdzie składowe kostki z F i R przyjmują przeciwne wartości.

Stosując wyżej podaną zasadę, dla kostki $k_1 = (0100101)$ powstaje następująca macierz blokująca:

$$B = B(k_1, R) = \begin{bmatrix} 1100000 \\ 1111011 \\ 1001011 \\ 1010010 \end{bmatrix}$$

Analogicznie, dla kostki $k_2 = (1000110)$:

$$B = B(k_2, R) = \begin{bmatrix} 0000011 \\ 0011000 \\ 0101000 \\ 0110001 \end{bmatrix}$$

W każdym wierszu macierzy B powinna znajdować się przynajmniej jedna ‘1’. Jeśli okazałoby się, iż wygenerowany został wiersz samych zer, oznacza to, iż funkcja będącą przedmiotem ekspansji jest sprzeczna – kostka równoważna jej mintermowi znajduje się także w zbiorze R.

Macierz blokująca jest w zasadzie punktem wyjścia do ekspansji kostki. Zawiera ona istotną informację o tym, które wartości argumentów kostki należy bezwzględnie

zachować (tj. nie rozszerzać ich, czyli nie zamieniać w ‘*’), tak aby podlegająca ekspansji kostka k ze zbioru F nie pokryła jakiejkolwiek kostki ze zbioru R. Pozycje tych „blokujących” argumentów są wyznaczone przez ‘1’ w macierzy B, a celem jest tu znalezienie takich kombinacji ‘1’ pokrywających wszystkie wiersze macierzy B, aby zawrzeć je w jak najmniejszej liczbie kolumn macierzy B, a tym samym maksymalizować ekspansję kostki *k*, co szerzej opisuje kolejny podrozdział.

2.4. Minimalne pokrycie kolumnowe macierzy blokującej

Pokrycie kolumnowe macierzy blokującej $B = \lfloor b_{ij} \rfloor$, $i \in \{1,...,w\}$, $j \in \{1,...,n\}$ to zbiór numerów kolumn $L \subseteq \{1,...,n\}$, taki że dla każdego i istnieje $j \in L$, dla którego $b_{ij} = 1$, czyli zagwarantowane zostaje, iż w obrębie tych kolumn w każdym wierszu macierzy B znajdzie sie przynajmniej jeden element o wartości ‘1’ (wspomniany wcześniej przypadek wiersza samych zer oznacza błąd w definicji funkcji).

Minimalne pokrycie kolumnowe macierzy blokującej B to taki podzbiór L', iż nie istnieje żaden mniejszy podzbiór L'' będący pokryciem kolumnowym macierzy B. Przykładowe pokrycie macierzy blokującej kostki k_1, $L = \{2,6,7\}$. Niemniej, nie jest to pokrycie minimalne – są nimi podzbiory $L' = \{2,6\}$ oraz $L' = \{1\}$.

Ekspansję kostki k_i definiujemy jako:

$$k_j^{+}(L,k) = \begin{cases} k_j, & gdy\ j \in L \\ *, & w\ pozostalych\quad przypadkach \end{cases}$$

Ekspansje odpowiadające wyznaczonym zbiorom pokrycia kolumnowego powstają z kostki *k* po zastąpieniu gwiazdkami (wartość nieokreślona) dotychczasowych wartości argumentów na pozycjach, których numery nie są zawarte w zbiorze minimalnego pokrycia kolumnowego.

Przykładowo, dla kostki k_1, zbiór $L = \{2,6\}$ wyznacza pokrycie kolumnowe B, a zatem: $k_1^{+}(L,k) = (*1***0*)$.

Zbiór implikantów prostych jest zbiorem ekspansji kostki zapisanym tak, iż w miejsce jedynek pojawiają się kolejne literały określone poprzez pozycję argumentu, przy czym

literały wpisywane w miejsce zer są negowane, a na pozycjach, gdzie pojawiają się '*', literały zostają usunięte z definicji implikantu.

W procesie ekspansji dla każdej kostki k ze zbioru F otrzymujemy zbiór pokrywających ją implikantów prostych. Analogicznie, w kontekście reguł decyzyjnych, możemy stwierdzić, iż w wyniku ekspansji pierwotna reguła opisująca dany obiekt decyzyjny zostaje uogólniona i przypisany jej zostaje zestaw reguł bazujących na zredukowanej liczbie atrybutów.

2.5. Tablica implikantów prostych

Proces wydrębnienia implikantów prostych (i analogicznie - uogólniania reguł) jest fundamentalną przesłanką do rozpoczęcia faktycznego procesu minimalizacji, jako że posłuży on do znajdowania w dalszej kolejności implikantów prostych wspólnych dla pewnych zbiorów kostek, a tym samym umożliwi zastępienie ich przez owe implikanty. W obszarze eksploracji danych oznaczać to będzie, iż uogólniona reguła znajdzie zastosowanie do więcej niż jednego przypadku decyzyjnego.

Ostatnim etapem minimalizacji jest umieszczenie informacji o wyznaczonych zbiorach implikantów prostych w tzw. tablicy implikantów prostych[1], co realizuje się w sposób następujący: każdy wiersz tablicy odpowiada kostce k_i z macierzy F, natomiast każda kolumna tablicy identyfikuje dany implikant z całego zbioru. Jeśli implikant I_j pokrywa kostkę k_i, to na przecięciu wiersza k_i z kolumną I_j pojawia się 1, w przeciwnym wypadku wpisywane jest 0. Minimalne zbiory implikantów prostych reprezentujących funkcję f otrzymuje się poprzez obliczenie minimalnych pokryć kolumnowych tablicy implikantów prostych.

Ponownie przekładając to na język reguł decyzyjnych – postępowanie to prowadzić będzie do wyznaczenia minimalnych zestawów reguł pokrywających wszystkie obiekty decyzyjne.

[1] Tablicę tę będziemy w kontekście eksploracji danych nazywać także „tablicą minimalnych reguł".

3. Uzupełnianie (redukcja zbioru reguł decyzyjnych)

3.1. Idea uzupełniania

Podobnie jak w przypadku ekspansji procedura uzupełniania (Complement) wchodzi w skład metody Espresso [3]. Jej zadaniem jest wyznaczenie zbioru *D*, czyli zbioru wszystkich nieokreśloności funkcji.

Punktem wyjścia jest tu przekształcenie danej macierzy *F* reprezentującej funkcję jednorodną w macierz porównań *M*. Transformacja ta jest opisana następująco:

$$\begin{cases} M[i,j]=0 \text{ , jeżeli } F[i,j]=* \\ M[i,j]=1 \text{ , jeżeli } F[i,j]=1 \text{ lub } F[i,j]=0 \end{cases}$$

Należy zauważyć, iż każdy wiersz *i* macierzy $\overline{M}$, stanowiącej uzupełnienie macierzy porównań *M* reprezentuje pokrycie kolumnowe *M*, gdzie $j \in M$ wtedy i tylko wtedy, gdy $\overline{M}_{ij}=1$.

Powyższe twierdzenie oznacza, iż problem generowania pokryć kolumnowych macierzy sprowadzić można do obliczania uzupełnienia jednorodnej funkcji boolowskiej. Procedura ta została nazwana *UNATE_COMPLEMENT* i właśnie w takim charakterze zostaje zaimplementowana w przedstawionym w kolejnym rozdziale programie komputerowym InstantRS (wykorzystana została zarówno do obliczania minimalnych pokryć kolumnowych macierzy blokujących, jak też przede wszystkich do obliczania minimalnych pokryć kolumnowych tablicy implikantów prostych).

Warto przy tym wspomnieć, iż poszukiwanie minimalnych pokryć kolumnowych macierzy zalicza się do problemów NP-zupełnych, i zmierzenie się z tym problemem w sposób systematyczny stanowi próbę badania granic oraz uwarunkowań efektywności takiego podejścia.

3.2. Rozkład Shannona

Procedura uzupełniania polega ona na iteracyjnym rozkładzie zbioru kostek z macierzy F na kofaktory (tj. kofaktor 0 i 1). Rozpinane jest drzewo, którego liście stanowią kofaktory, dla których spełnione są warunki zakończenia (opisane w dalszej części).

Następnie rozpoczyna się „zwijanie drzewa”, a wyniki cząstkowe są scalane, w wyniku czego uzyskana zostaje macierz $\overline{M}$ (uzupełnienie macierzy M).

Postępowanie takie opiera się na obserwacji, iż uzupełnienie funkcji reprezentowanej rozkładem Shannona:

$$f = x_j f_{x_j} + \bar{x}_j f_{\bar{x}_j}$$

można wyznaczyć obliczając najpierw kofaktory f_{x_j} oraz $f_{\bar{x}_j}$, a następnie ich uzupełnienie, czyli:

$$\bar{f} = x_j \bar{f}_{x_j} + \bar{x}_j \bar{f}_{\bar{x}_j}$$

Uzupełnienie $\bar{f}$ powstaje w wyniku „sklejenia” wyników cząstkowych, tj. $\bar{f}_{x_j}$ oraz $\bar{f}_{\bar{x}_j}$. Dla funkcji monotonicznych, uzupełnienie upraszcza się do następujących wzorów:

a) monotonicznie rosnącej w punkcie x_j,

$$\overline{F} = \bar{x}_j \overline{F}_{\bar{x}_j} + \overline{F}_{x_j}$$

b) monotonicznie malejącej w punkcie x_j,

$$\overline{F} = x_j \overline{F}_{x_j} + \overline{F}_{\bar{x}_j}$$

3.3. Opis algorytmu

Obliczenie kofaktorów otrzymanej macierzy *M* rozpoczyna się od wyboru zmiennej do rozkładu. Odpowiedni wybór zmiennej ma istotne znaczenie dla redukcji obliczeń. Wybór zmiennej przeprowadza się według następującego schematu:

1) Wybrana zostaje kostka (wiersz macierzy *M*) z największą liczbą zer.
2) W wybranej kostce zapamiętywane są zmienne zawierające jedynkę.
3) Spośród wybranych w punkcie 2) zmiennych wybrana zostaje ta, która ma najwięcej jedynek w swojej kolumnie.

Kofaktory macierzy *M* oblicza się w sposób następujący: Kofaktor jedynkowy macierzy *M* względem zmiennej x_j otrzymuje się przez ustawienie wszystkich pozycji *j*-tej kolumny macierzy *M* na zera. Kofaktor zerowy macierzy *M* względem zmiennej x_j powstaje przez

wypisanie z M tych kostek (wierszy), w których zmienna x_j przyjmuje wartość zero. Na przykład dla macierzy M:

$$M = \begin{bmatrix} 1100 \\ 1000 \\ 0110 \\ 0111 \end{bmatrix}$$

wybór zmiennej x_2 prowadzi do kofaktorów:

$$\begin{bmatrix} 1000 \\ 1000 \\ 0010 \\ 0011 \end{bmatrix} \quad \text{dla } x_2 = 1$$

oraz

$$\begin{bmatrix} 0110 \\ 0111 \end{bmatrix} \quad \text{dla } x_2 = 0.$$

Obliczanie uzupełnienia funkcji jednorodnej polega na rozkładzie macierzy M na podzbiory kostek, których uzupełnienie można łatwo obliczyć. Wyróżnia się trzy przypadki, w których uzupełnienie oblicza się bezpośrednio z postaci uzyskanego kofaktora (dochodzimy do liścia w drzewie kofaktorów):

a) Kofaktor jest zbiorem pustym (kofaktor 0), czyli funkcja jest funkcją stałą o wartości logicznej 0. Jej uzupełnienie jest funkcją stałą o wartości 1, czyli tautologią.
b) Kofaktor zawiera wiersz samych zer (kofaktor 1), czyli kostkę uniwersalną (tautologia). Jej uzupełnienie jest zbiorem pustym.
c) Kofaktor o jednym wierszu reprezentujący pojedynczą kostkę.

Uzupełnienie kofaktora o jednym wierszu oblicza się według następujących zasad:

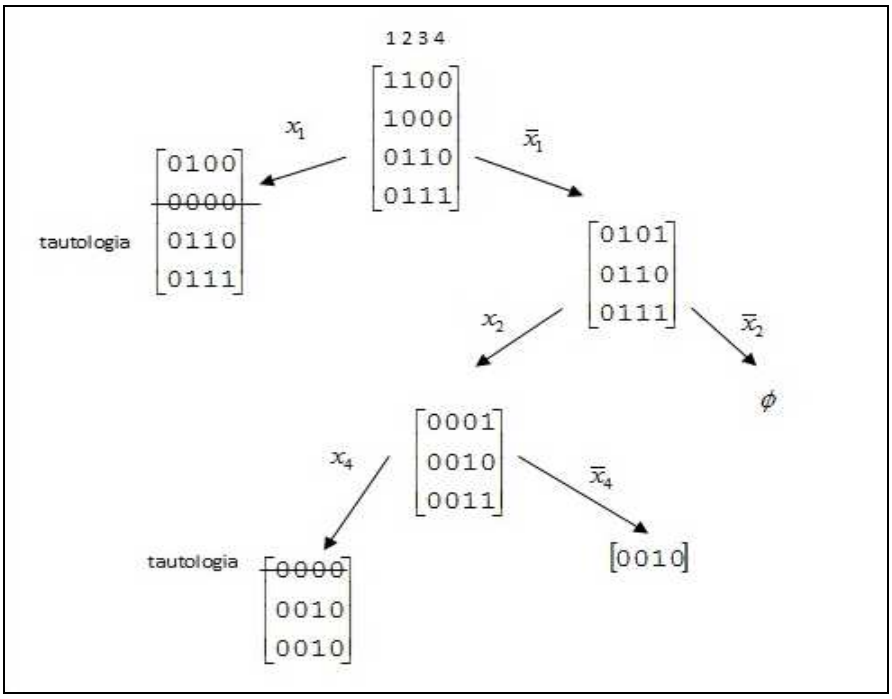
a) Jeśli kofaktor zawiera tylko jedną jedynkę, jego uzupełnienie jest identyczne jak kofaktor.
b) Jeśli kofaktor zawiera więcej niż jedną jedynkę, jego uzupełnienie zawiera tyle kostek (wierszy), ile jest jedynek w kofaktorze, przy czym wszystkie wiersze mają jedynkę na pozycjach odpowiadających kolejnym jedynkom kofaktora.

Przykładowo, dla kofaktora [0110001] uzupełnieniem będzie macierz: $\begin{bmatrix}0100000\\0010000\\0000001\end{bmatrix}$

Po obliczeniu uzupełnień na poszczególnych poziomach rozkładu należy scalić wyniki cząstkowe zgodnie ze wzorem:

$$F = x_j \cdot F_0 + F_1$$

Tym samym, jeżeli otrzymany kofaktor był zerowy (ozn. F_0), to mnożymy go przez odpowiednie x_j i dodajemy, a jeżeli był jedynkowy (ozn. F_1), to tylko dodajemy.



Rys. 3.1. Obliczanie uzupełnienia funkcji jednorodnej – opracowanie własne

W przedstawionym wyżej przykładzie (rys. 3.1.) , macierz dzieli się kolejno na kofaktory, dla których rozkład zatrzymuje się w momencie wystąpienia jednego z warunków zakończenia (tautologia, zbiór pusty, pojedynczy wiersz). W wyniku scalania otrzymujemy macierz $\begin{bmatrix}1011\\1100\end{bmatrix}$.

4. Jednolita metoda obliczania reduktów i reguł decyzyjnych

Celem niniejszej pracy jest było stworzenie narzędzia komputerowego umożliwiającego analizę i porównanie skuteczności algorytmów syntezy logicznej z algorytmami eksploracji danych. Dla jednolitości porównań założono analizę binarnych tablic danych. Przyjęcie takiego założenia ułatwiło przeprowadzenie eksperymentów uwypuklających różnice w jakości i skuteczności algorytmów. Potwierdzeniem tych założeń jest przykład funkcji boolowskiej KAZ, (Tab. 4.1.), która dla algorytmu redukcji atrybutów w systemie RSES liczy się ok. 45 min., natomiast algorytm systemu InstantRS oblicza wszystkie redukty w ciągu ok. 3 sek.

Tab. 4.1. Przykład KAZ

```
.type fr
.i 21
.o 1
.p 31
100110010110011111101 1
111011111011110111100 1
001010101000111100000 1
001001101100110110001 1
100110010011011001101 1
100101100100110110011 1
001100100111010011011 1
001101100011011011001 1
110110010011001001101 1
100110110011010010011 1
110011011011010001100 1
010001010000000100111 0
100110101011111110100 0
111001111011110011000 0
101101011100010111100 0
110110000000101010000 0
110110110111100010111 0
110000100011110010001 0
001001000101111101101 0
100100011111100110110 0
100011000110011011110 0
110101000110101100001 0
110110001101101100111 0
010000111001000000001 0
001001100101111110000 0
100100111111001110010 0
000010001110001101101 0
101000010100001110000 0
101000110101010011111 0
101010000000110001101 0
011100111110111101111 0
.end
```

Głównym pomysłem do przeprowadzenia eksperymentów dotyczących analizy skuteczności algorytmów eksploracji danych jest zastosowanie w procedurze uogólniania reguł algorytmu ekspansji i uzupełniania. Połączenie tych procedur w jednolitej metodzie uogólniania reguł polega na:

1) Utworzeniu macierzy blokującej (macierzy porównań) dla konkretnego obiektu danej klasy,
2) Obliczeniu minimalnych pokryć kolumnowych metodą uzupełniania funkcji boolowskiej reprezentującej binarną macierz blokującą,
3) Utworzeniu ze zbioru minimalnych reguł poszczególnych obiektów globalnego zbioru wszystkich reguł minimalnych,
4) Utworzenie globalnej tablicy reguł minimalnych (wyszukiwanie minimalnych pokryć),
5) Obliczenie minimalnego zbioru reguł reprezentującego wszystkie pierwotne obiekty; w obliczaniu tego zbioru stosowana jest również metoda uzupełniania.

Przykład

Tablica dla f=1,

```
0100101
1000110
1010000
1010110
1110101
```

Poszczególne wiersze tej tablicy reprezentują kostki funkcji; oznaczamy je C1,..., C5; w tablicach danych sa to obiekty. Składowe kostek oznaczamy x1,..., x7.

Tablica dla f=0

```
1000101
1011110
1101110
1110111
```

Macierz blokująca kostki C1 = (0100101)

```
1100000
1111011
1001011
1010010
```

Po obliczeniu minimalnego pokrycia kolumnowego uzyskujemy ekspansję kostki, (uogólnioną regułę decyzyjną)

```
0******
```

Macierz blokująca kostki C2 = (0100101)

```
0000011
0011000
0101000
0110001
```

i jej uogólnienie: ***0**0

Kolejne uogólnione reguły są następujące:

```
****0**
*1***0*
**1**0*
```

W rezultacie otrzymujemy globalny zbiór wszystkich reguł minimalnych:

```
0******  ==>  I1 = !x1
***0**0  ==>  I2 = !x4!x7
****0**  ==>  I3 = !x5
*1***0*  ==>  I4 = x2!x6
**1**0*  ==>  I5 = x3!x6
```

Globalna tablica reguł minimalnych (pokryć) jest przedstawiona poniżej; kolumny oznaczamy I1,..., I7:

```
10010
01000
01101
01000
00011
```

Skoro minimalne pokrycie kolumnowe jest I2, I5, to zbiór reguł reprezentujących wszystkie pierwotne obiekty, dla zwartości zapisu podany w formie wyrażenia boolowskiego jest następujący:

```
y1 = !x4!x7 + x2!x6
```

W powyższym przykładzie obliczenia (w szczególności minimalnych pokryć kolumnowych) były tak proste, ze nie trzeba było stosować procedury uzupełniania.

W kolejnym przykładzie pokażemy zastosowanie tej procedury w obliczaniu minimalnego zbioru reguł.

Przykład

```
.type fr              I1 = !x4!x5
.i    7               I2 = !x5x6
.o    1               I3 = !x1
.p    8               I4 = x2x7
1100010 1             I5 = !x5x7
1101000 0             I6 = x6x7
1001101 0             I7 = x4x6
0101100 1             I8 = x3
1100110 0             I9 = !x4!x6
1101011 1
1010101 1
1100110 0
.e
```

Globalna tablica reguł minimalnych:

```
 #|I1|I2|I3|I4|I5|I6|I7|I8|I9|
c1| 1| 1| 0| 0| 0| 0| 0| 0| 0|
c2| 0| 0| 1| 0| 0| 0| 0| 0| 0|
c3| 0| 1| 0| 1| 1| 1| 1| 0| 0|
c4| 0| 0| 0| 0| 0| 0| 0| 1| 1|
```

Tablicę tę reprezentujemy funkcją boolowską:

```
.type f
.i 9
.o 1
.p 4
11------- 1
--1------ 1
-1-1111-- 1
-------11 1
.e
```

Uzupełnienie tej funkcji obliczone procedurą Complement:

```
0-00----0
0-0-0---0
0-0--0--0
0-0---0-0
0-00---0-
0-0-0--0-
0-0--0-0-
0-0---00-
-00----0-
-00-----0
```

reprezentuje wszystkie reguły minimalne; oczywiście do rozwiązania wybieramy reguły minimalne o najmniejszej liczbie atrybutów. W tym przypadku są one reprezentowane dwoma ostatnimi wierszami (pozycje 0 wskazują numer kolumny, czyli regułę).

W rezultacie uzyskujemy dwa rozwiązania:

```
1] !x5x6 + !x1 + x3
2] !x5x6 + !x1 + !x4!x6
```

Warto podkreślić, że rozwiązanie dla tej funkcji uzyskane programem Espresso jest gorsze:

```
y1 = !x4x7 + x2x5!x6 + !x5x6
```

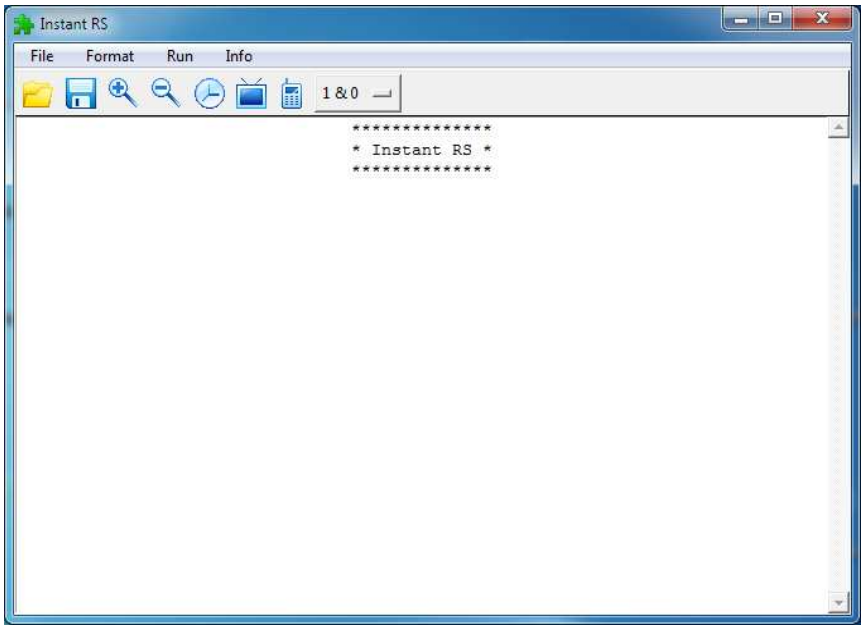
5. Program InstantRS

5.1. Wprowadzenie

Program instantRS, stanowiący próbę zmierzenia się z zadaniem ugólniania reguł decyzyjnych w podejściu systematycznym, i poszukiwania granic efektywności tego podejścia, to aplikacja zawierająca algorytmy, które można wykorzystać zarówno w zadaniach syntezy logicznej, jak i w obszarze eksploracji danych. Program może służyć do interdyscyplinarnych porównań, badania zależności między specyfiką i charakterem danych wejściowych (tablice danych oraz tablice prawdy), a efektywnością wykonywanych obliczeń. W programie tym umożliwiono prezentowanie wyników obliczeń zarówno w notacji syntezy logicznej (implikanty proste reprezentujące funkcję), jak też notacji typowej dla eksploracji danych (redukty, reguły decyzyjne).

5.2. Budowa i przegląd funkcji programu

Interfejs graficzny programu zaimplementowany został w języku Python - jest to język programowania wysokiego poziomu, o szerokim zastosowaniu, który też stosunkowo łatwo integruje się z C. W języku C z kolei napisany został silnik algorytmiczny aplikacji (m.in. zawiera on omówioną wcześniej procedurę Complement). Główny panel aplikacji przestawia rys. 5.1.



Rys. 5.1. Panel InstantRS

Program zawiera zarówno pasek menu, jak też pasek narzędziowy, który w zasadzie powtarza i wizualizuje opcje dostępne z poziomu paska menu. Poniżej zaprezentowany został przegląd poszczególnych pozycji menu.

Menu File

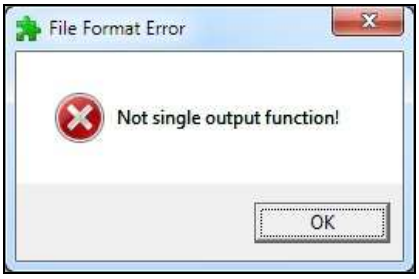
Menu File zawiera standardowe opcje obsługi pliku wejściowego dla programu. Program wspiera obecnie 2 popularne formaty danych – format .pla (format wprowadzony wraz z metodą Espresso) oraz format .tab (format danych programu Rough Set Exploration System – RSES).

Rys. 5.2. Menu File

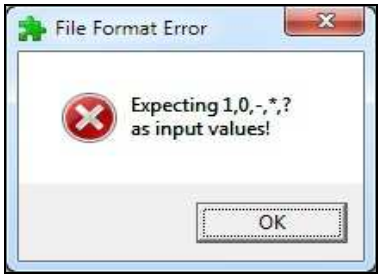
Tab. 5.1. Opcje Menu File

Opcja	Opis
Open...	Załadowanie pliku z danymi wejściowymi
Save...	Zapisanie do pliku zawartości okna instantRS
Quit	wyjście z programu

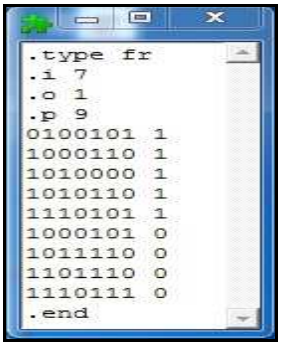
Warto przy tym zauważyć, iż program jest przeznaczony do przetwarzania binarnych tablic danych z jednym wyjściem (0 lub 1), w związku z czym próba załadowania pliku niespełniającego powyższych wymagań (np. dane wielowartościowe lub wiele wyjść) skutkować będzie pojawieniem się okna dialogowego sygnalizującego błąd.



Rys. 5.3. Błąd programu (1)



Rys. 5.4. Błąd programu (2)

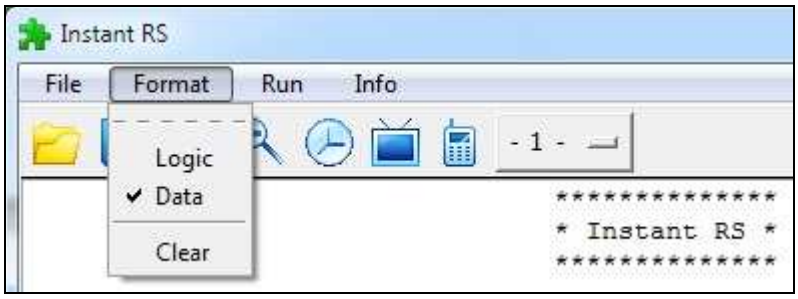


Rys. 5.5. Plik wejściowy

Po załadowaniu pliku, z boku głównego okna aplikacji, dostępny jest boczne okno z widokiem pliku danych wejściowych, które w razie potrzeby może zostać zamknięte.

Menu Format

W tym menu dokonać można wyboru notacji, w jakiej wyświetlane będą wyniki obliczeń minimalnych reguł. Dostępne są dwie perspektywy, tj. zarówno zapis w formacie spotykanym w syntezie logicznej, jak też zapis stosowany w obszarze eksploracji danych.



Rys. 5.6. Menu Format

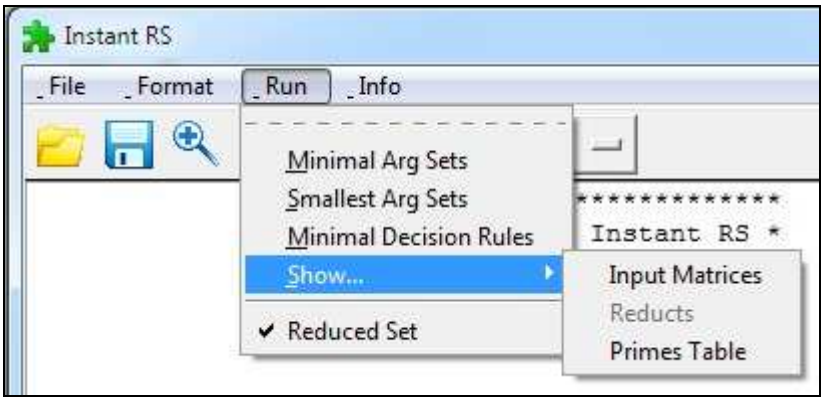
Tab. 5.2. Opcje Menu Format

Opcja	Opis
Logic	notacja z obszaru syntezy logicznej
Data	notacja z obszaru danych/reguł decyzyjnych
Clear	wyczyszczenie okna panelu

Menu Run

Menu Run zawiera podstawowe opcje obliczeniowe programu. Jak widać oprócz zasadniczej funkcjonalności ugólniania reguł decyzyjnych, program oferuje także możliwość wyliczenia reduktów – zarówno minimalnych zbiorów, jak i najmniej licznych minimalnych zbiorów. Ma to istotne znaczenie w przypadku większych zestawów danych, ze znaczną liczbą zmiennych wejściowych, dla których policzenie minimalnych reguł decyzyjnych natrafia na barierę w postaci ograniczeń pamięci komputera (dot. zwłaszcza

przechowywania wyników cząstkowych powstających w procesie wyliczania minimalnych pokryć kolumnowych w trakcie obliczeń dokonywanych zaadaptowaną na potrzeby programu metodą uzupełniania). W tym przypadku pomocne okazuje się wygenerowanie uogólnionych reguł na zbiorze danych wejściowych zredukowanym do wybranego zbioru reduktów.



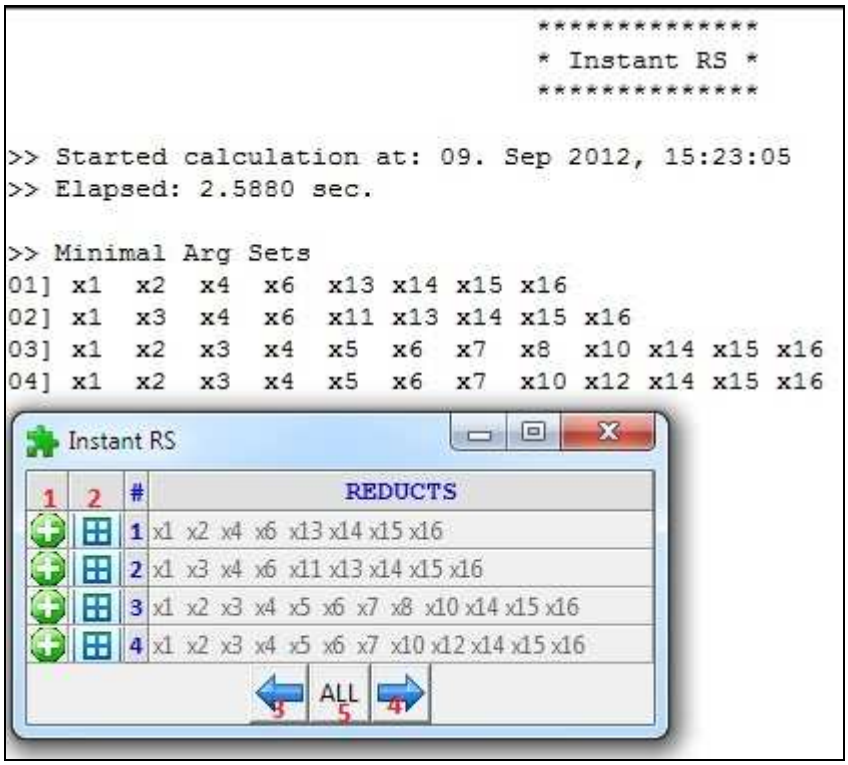
Rys. 5.7. Menu Run

Tab. 5.3. Opcje Menu Run

Opcja	Opis
Minimal Arg Sets	oblicz mimalne zbiory reduktów
Smallest Arg Sets	oblicz najmniej liczne minimalne zbiory reduktów
Minimal Decision Rules	oblicz minimalne uogólnione reguły decyzyjne
Show...	wyświetl wybrany typ tablicy/macierzy
Show... -> Input Matrices	pokaż macierze F i R wyodrębnione z danych wejściowych
Show... -> Reducts	pokaż tablicę reduktów
Show... -> Primes	wylistuj implikanty proste, jak też wyświetl tablicę implikantów prostych
Reduced Set	uwzględniaj tylko najmniej liczne implikanty proste

Na bliższą uwagę zasługuje wspomniana wyżej funkcjonalność związana z wyliczaniem reduktów. Po wybraniu opcji *Show...-> Reducts*, ukazuje się dodatkowe okno pokazujące poszczególne redukty (opcja ta staje się aktywna po uprzednim wyliczeniu reduktów; wyświetlany zbiór reduktów odpowiada wybranej przedtem kalkulacji - wszystkie bądź tylko najmniej liczne zbiory minimalnych reduktów). W prawej kolumnie wyświetlone zostają poszczególne redukty, natomiast dwie pierwsze z lewej kolumny oferują dodatkowe opcje pomocne przy obliczaniu oraz analizowaniu zbioru reguł. Przycisk *1* („+”) umożliwia obliczenie reguł decyzyjnych, a przycisk *2* („okno”) pozwala na wyświetlenie listy implikantów prostych (wraz z liczbą pokrywanych przez nie wektorów wejściowych), jak też tablicy implikantów prostych dla macierzy F i R

zredukowanych do zbioru kolumn wskazanego przez składniki reduktu. Dzięki temu InstantRS staje się wygodnym narzędziem do analizy większych zbiorów danych poprzez rozbicie ich na mniejsze podzbiory odpowiadające reduktom. Na dole okna reduktów znajdują się strzałki do przewijania zbioru (przycisk *3 i 4*), a także przycisk „*ALL*” (*5*), powodujące wyliczenie uogólnionych reguł decyzyjnych per każdy wyświetlony redukt.



Rys. 5.8. Redukty

Istotnym wzbogaceniem programu jest opcja wyboru *Reduced Set* (domyślnie aktywna) – jej zaznaczenie skutkuje obliczaniem reguł decyzyjnych w oparciu o najmniej liczne implikanty proste wyliczone per kostka macierzy *F*.

Menu Info

Ostatnie menu zawiera streszczony opis programu, jak też informacje o autorach programu.

Rys. 5.9. Menu Info

Pasek Narzędziowy

Wiele opcji zawartych w omówionym powyżej pasku menu jest jednocześnie dostępnych z poziomu paska narzędziowego.

Rys. 5.10. Pasek narzedziowy

Tab. 5.4. Opcje paska narzędziowego

Opcja	Opis
1	otwórz plik
2	zapisz plik
3	powiększ tekst w oknie panelu
4	pomniejsz tekst w oknie panelu
5	ustaw maksymalny czas obliczeń (potencjalna opcja do zaimplementowania)
6	wyświetl tablicę wybranego typu...
7	wykonaj obliczenia (generowanie reduktów / uogólnionych reguł)
8	określ typ decyzji, dla której liczone będą reguły (1, 0 lub 1 & 0)

Najistotniejszą z ikon paska narzędziowego jest przycisk wyboru decyzji (*8*). Przy jego pomocy definiuje się zakres generowanych minimalnych reguł decyzyjnych (dla f=0 lub dla f=1 bądź dla f=0 i f=1), przy czym wybranie preferowanej decyzji skutkować będzie ustawieniem odpowiedniego zakresu obliczeń dla najbliższej kalkulacji reguł.

5.3. Wyniki eksperymentów

Badając granice możliwości podejścia systematycznego w procesie ugólniania reguł decyzyjnych, analizie poddano kilka przykładów, uporządkowanych według rosnącej liczby atrybutów oraz obiektów decyzyjnych. Dla celów badania wyniki generowane przez program RSES v2.2.2 (Rough Set Exploration System) [14], [15] porównano z wynikami osiąganymi przy pomocy aplikacji InstantRS. Eksperymenty przeprowadzono na komputerze o następujących parametrach: procesor - 2.00GHz, pamięć RAM – 2GB. W programie RSES reguły obliczano wybierając opcję heurystycznego algorytmu *LEM2* - algorytmu zaczerpniętego z systemu danych LERS (Learning from Examples based on Rough Sets) często stosowanego w indukcji reguł decyzyjnych [5], ustawiając przy tym wartość *Cover parameter* na 1 (pokrycie całego zbioru obiektów decyzyjnych). Z kolei opisywana w niniejszej pracy aplikacja InstantRS wykorzystuje do obliczeń algorytmy systematyczne (opisane w poprzednim rozdziale), bazujące na procedurach ekspansji i uzupełniania wywodzących się z Espresso.

W przypadku programu RSES, generowany jest jeden zoptymalizowany zestaw reguł per decyzja (1 lub 0 w przypadku omawianych w niniejszej pracy tablic binarnych), natomiast program InstantRS jest zorientowany na generowanie najmniej licznego (pod względem liczby termów) zestawu reguł, przy czym możliwe jest więcej niż jedno rozwiązanie per decyzja, w przypadku rozwiązań równolicznych. Dla celów porównawczych, w przypadku, gdy InstantRS generował więcej niż jedno minimalne rozwiązanie, do porównania z rezultatem obliczeń RSES wybierano rozwiązanie najgorsze pod kątem maksymalnej liczby literałów. Dla każdego przykładu badano liczbę reguł, jak też maksymalną liczbę atrybutów[2] uzyskiwaną dla danej decyzji (zaprezentowane w kolumnach ‘licz. reg.’ oraz ‘max atr.’, w postacji liczba[*decyzja*], np. 8[1] oznacza 8 dla decyzji 1).

Uzyskane wyniki wskazują, iż w większości analizowanych przykładów, podejście systematyczne dostarcza wydajniejszych rozwiązań. We wszystkich badanych przypadkach maksymalna liczba atrybutów w rozwiązaniach generowanych przez InstantRS jest niższa niż w w tych dostarczanych przez RSES. Podobnie porównania rezultatów w zakresie liczby reguł w ponad połowie badanych przypadków wychodzą na korzyść InstantRS. Przykładowo, dla zbioru danych *TL27*, dla decyzji 1, RSES generuje

[2] W syntezie logicznej równoważne pojęcia to liczba termów (reguły) oraz liczba literałów (atrybuty) funkcji boolowskiej.

zestaw 7 reguł, podczas gdy InstantRS generuje wynik o 2 mniejszy. Z kolei RSES oblicza regułę złożoną maksymalnie aż z 10 atrybutów, podczas gdy InstantRS maksymalnie wykorzystuje 3 atrybuty. Mniejsza liczba reguł w połączeniu z mniejszą liczbą atrybutów przekłada się na mniej złożone, a tym samym wyższej jakości rozwiązania dostarczane przez program InstantRS.

Niemniej, algorytm systematyczny wykazuje też istotne ograniczenia. Dla znanego z literatury przykładu *House*, algorytm nie jest w stanie obliczyć rozwiązań dla decyzji 0 (program przerywa pracę z powodu braku pamięci). Niestety nawet niewielki wzrost liczby atrybutów skutkuje wygenerowaniem licznych reguł, których dobieranie w minimalne zbiory pokrywające całe uniwersum obiektów decyzyjnych jest procesem zasobochłonnym (zważywszy, iż zgodnie z podejściem systematycznym, eksplorowany jest zakres wszystkich możliwych rozwiązań, z których wybierane są te minimalne).

W tym przypadku, program InstantRS dostarcza instrumentu do uproszczenia badanej przestrzeni reguł – mianowicie pierwotna tablica danych może być dowolnie redukowana do kolumn odpowiadających reduktom tej tablicy. Po obliczeniu reduktów dla tablicy i transformacji tablicy w oparciu o konkretnie wskazany redukt, obliczyć można większe przykłady. Zgodnie z powyższym rozumowaniem, dla zbioru danych *House* wyliczamy najmniej liczny redukt (x1,x2,x4,x6,x13,x14,x15,x16). Po zredukowaniu tablicy w programie instantRS, otrzymujemy rozwiązanie dla decyzji 0, znacznie mniejsze, a tym samym wydajniejsze niż korespondujące rozwiązanie dostarczone przez RSES (maksymalna reguła zawiera kolejno: RSES - 16 atrybutów, InstantRS - 4 atrybuty).

Przykład	L. obj.	L. atr.	**RSES**			**InstantRS**		
			Wynik	licz. reg.	max atr.	Wynik	licz. reg.	max atr.
B11	16	6	(x3=1)*(x2=1)+(x2=0)*(x1=1)*(x5=0) +(x2=0)*(x3=0) => R=0 (x5=1)*(x1=0)*(x2=0)*(x3=1)+(x2=1)*(x3=0)+(x1=1) *(x6=0)*(x2=0)*(x3=1)*(x4=0)*(x5=1) => R=1	3[0] 3[1]	3[0] 6[1]	(x2=0)*(x3=0)+(x2=1)*(x3=1)+(x2=0)*(x5=0) => R=0 (x2=0)*(x3=1)*(x5=1)+(x2=1)*(x3=0) => R=1	3[0] 2[1]	2[0] 3[1]
A20	20	8	(x5=1)*(x4=1)*(x1=1)*(x2=0)*(x3=0)+(x3=1)*(x5=1) *(x4=1)*(x6=0)+(x3=1)*(x1=0)*(x2=1)*(x6=1)*(x7= 0)*(x4=0)+(x1=1)*(x2=0)*(x4=0)*(x5=0)+(x1=0)*(x2 =1)*(x3=1)*(x4=1)*(x5=1) => R=0 (x1=0)*(x2=1)*(x3=1)*(x4=1)*(x5=0)+(x5=1)*(x1=0) *(x2=1)*(x4=0)*(x7=1)+(x5=1)*(x2=0)*(x1=1)*(x3= 1)*(x7=0)*(x4=0)+(x8=1)*(x2=0)*(x5=1)*(x1=1)*(x3 =0)*(x4=0)+(x4=1)*(x8=1)*(x1=0)*(x2=0)+(x4=1)*(x 8=1)*(x1=0)*(x2=1)*(x3=0)+(x1=1)*(x2=0)*(x3=1)* (x4=1)*(x5=1)*(x6=1) => R=1	5[0] 7[1]	6[0] 9[1]	(x4=1)*(x6=0)*(x8=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x3=0)*(x4=1)*(x5=1) => R=0 (x2=0)*(x6=1)*(x8=1)+(x3=0)*(x4=0)*(x5=1)+(x4=1)*(x5=0)+(x4=0)*(x6=0)*(x8=0) => R=1	4[0] 4[1]	3[0] 3[1]
TL27	25	10	(x4=0)*(x6=1)*(x10=0)*(x1=0)*(x2=1)*(x3=0)*(x5= 0)+(x4=0)*(x6=1)*(x7=0)*(x1=1)+(x9=1)*(x10=0)*(x 1=0)*(x8=0)+(x5=1)*(x3=1)*(x9=1)*(x1=1)*(x6=0)* (x2=0)+(x10=0)*(x1=0)*(x8=0)*(x2=1)*(x3=0)*(x7= 0)*(x4=0)*(x5=1)*(x6=1)+(x1=1)*(x2=1)*(x3=1)*(x4 =0) => R=0 (x2=1)*(x5=1)*(x4=1)+(x4=0)*(x1=0)*(x5=0)*(x8=1) *(x9=1)*(x3=1)+(x9=0)*(x4=0)*(x1=0)*(x10=0)*(x2 =0)+(x6=1)*(x9=0)*(x1=1)*(x3=1)*(x10=1)+(x4=0)* (x1=0)*(x3=0)*(x2=1)*(x5=1)*(x8=0)*(x9=0)*(x10= 0)*(x6=0)+(x4=0)*(x1=0)*(x3=0)*(x6=1)*(x2=1)*(x7 =1)*(x5=0)*(x8=1)*(x9=1)*(x10=1)+(x1=0)*(x2=1)* (x3=0)*(x4=0)*(x5=1)*(x6=1)*(x7=1) => R=1	5[0] 7[1]	9[0] 10[1]	(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)*(x8=0) => R=0 (x1=0)*(x10=1)+(x1=0)*(x2=0)*(x7=0)+(x1=1)*(x2= 1)*(x4=1)+(x1=0)*(x4=0)*(x6=0)+(x7=1)*(x9=0) => R=1	4[0] 5[1]	4[0] 3[1]

G42	42	12	(x4=1)*(x7=1)*(x10=0)*(x1=1)*(x2=0)*(x8=1)*(x9=1)*(x12=0)+(x1=0)*(x4=1)*(x2=1)*(x7=1)*(x8=0)*(x9=0)*(x12=1)+(x2=0)*(x3=1)*(x5=1)*(x6=0)*(x8=1) => R=0 (x1=1)*(x4=1)*(x12=1)+(x8=0)*(x12=0)+(x3=0)*(x1=0)*(x8=1)+(x2=1)*(x4=0)+(x4=0)*(x1=1)*(x3=1)*(x7=0) => R=1	3[0] 5[1]	8[0] 4[1]	(x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x7=1)*(x8=1)*(x12=0) => R=0 (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1	3[0] 5[1]	4[0] 2[1]
House	232	16	(x13=1)*(x3=1)*(x12=1)*(x1=1)*(x6=0) +(x13=1)*(x3=1)*(x12=1)*(x11=1)*(x8=0)*(x9=0)*(x14=0)*(x2=0)*(x5=1)*(x4=1)*(x10=0)*(x16=0)*(x6=0)+(x13=1)*(x3=1)*(x12=1)*(x11=1)*(x8=0)*(x9=0)*(x14=0)*(x2=0)*(x5=1)*(x4=1)*(x15=1)+(x13=1)*(x3=1)*(x15=0)*(x5=1)*(x8=0)*(x12=1)*(x9=0)*(x10=0)*(x14=0)*(x1=0)*(x4=1)*(x6=0)+(x13=1)*(x3=1)*(x1=1)*(x5=0)*(x7=1)+(x14=0)*(x2=0)*(x8=0)*(x9=0)*(x10=0)*(x12=1)*(x13=1)*(x4=1)*(x6=1)*(x11=1)*(x15=1)+(x13=1)*(x3=1)*(x1=1)*(x6=1)*(x8=0)*(x11=1)*(x12=1)*(x5=1)*(x15=0)+(x3=1)*(x4=1)*(x5=1)*(x6=1)*(x7=0)*(x8=0)*(x11=1)*(x12=1)*(x13=1)*(x14=0)*(x1=1)+(x3=1)*(x14=0)*(x2=0)*(x15=0)*(x1=0)*(x5=1)*(x7=0)*(x8=0)*(x9=0)*(x10=0)*(x12=1)*(x13=1)*(x4=0)+(x3=1)*(x14=0)*(x1=1)*(x9=1)*(x10=1)*(x2=0)*(x4=0)*(x5=0)*(x6=0)+(x1=0)*(x2=0)*(x3=1)*(x4=1)*(x5=1)*(x6=1)*(x7=0)*(x8=0)*(x9=0)*(x10=0)*(x11=1)*(x12=1)*(x13=1)*(x14=0)*(x15=0)*(x16=1) => 0 (x13=0)*(x1=1)*(x5=0)*(x14=1)+(x1=1)*(x13=0)*(x10=1)*(x11=1)+(x14=0)*(x1=1)*(x13=0)*(x2=1)+(x11=1)*(x3=1)*(x6=1)*(x12=1)*(x2=0)*(x8=0)*(x9=0)*(x10=0)*(x4=1)*(x16=0)*(x7=1)*(x13=0)+(x1=0)*(x6=1)*(x7=0)*(x10=0)*(x14=0)*(x2=1)+(x3=1)*(x4=1)*(x5=1)*(x10=0)*(x11=1)*(x6=1)*(x7=0)*(x1=0)*(x14=0)*(x2=1)+(x1=1)*(x3=1)*(x5=1)*(x4=1)*(x10	11[0] 11[1]	16[0] 16[1]	[BRAK PAMIĘCI] ---------- na bazie najmniej licznego reduktu: x1,x2,x4,x6,x13,x14,x15,x16: (x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x2=0)*(x6=0)*(x14=0)*(x16=1)+(x4=0)*(x13=1)*(x14=1)+(x1=0)*(x15=0)*(x16=1) => R=0 (x2=1)*(x13=0)+(x8=0)*(x13=0)+(x13=0)*(x14=1)+(x4=0)*(x7=0)*(x15=1)+(x6=1)*(x10=0)*(x14=1)+(x1=0)*(x6=1)*(x15=0)*(x16=0) => R=1	n/a 6[0] 6[1]	n/a 4[0] 4[1]

			=0)*(x11=1)*(x12=1)*(x15=1)*(x2=0)*(x6=1)*(x13=1)*(x14=1)+(x6=1)*(x2=0)*(x3=1)*(x7=0)*(x14=0)*(x1=1)*(x4=0)+(x11=1)*(x3=1)*(x6=1)*(x12=1)*(x2=0)*(x8=0)*(x9=0)*(x5=1)*(x7=0)*(x10=0)*(x13=1)*(x1=0)*(x14=0)*(x16=0)*(x4=0)+(x1=0)*(x6=1)*(x10=0)*(x2=0)*(x3=1)*(x4=1)*(x8=0)*(x9=0)*(x11=1)*(x12=1)*(x13=1)*(x16=0)*(x5=0)+(x1=0)*(x2=0)*(x3=1)*(x4=1)*(x5=1)*(x6=1)*(x7=0)*(x8=0)*(x9=0)*(x10=0)*(x11=1)*(x12=1)*(x13=1)*(x14=0)*(x15=0)*(x16=0) => 1					

Zalety jednolitej metody zastosowanej w programie InstantRS szczególnie uwidaczniają się na przykładzie zasygnalizowanej w rozdz. 4 funkcji KAZ (Tab. 4.1). Dla celów porównawczych obliczono reguły decyzyjne w dwóch eksperymentach. W pierwszym ponownie zastosowano algorytm LEM2 systemu RSES (przy konfiguracji pokrycia wszystkich obiektów – *cover parameter* = 1), a w drugim do obliczeń zastosowano dwie procedury systemu InstantRS, a mianowicie algorytm redukcji atrybutów oraz algorytm generacji reguł.

Algorytm LEM2 bezpośrednio obliczył reguły dla obu klas decyzyjnych, odpowiadających dwóm wartościom funkcji KAZ:

```
RULES 10
(x3=0)&(x2=1)&(x18=0)=>(x22=0[7]) 7
(x14=1)&(x2=0)&(x21=1)&(x4=1)=>(x22=1[6]) 6
(x10=1)&(x15=1)&(x16=1)&(x5=0)=>(x22=0[6]) 6
(x20=0)&(x4=0)&(x9=1)&(x14=1)&(x3=1)&(x7=1)&(x13=1)&(x16=1)=>(x22=1[
3]) 3
(x1=1)&(x17=1)&(x2=0)&(x6=0)&(x12=1)&(x3=0)&(x4=1)&(x9=1)=>(x22=0[3]
) 3
(x1=1)&(x9=0)&(x6=0)&(x12=1)&(x15=0)&(x16=0)&(x17=1)&(x21=1)&(x4=0)=
>(x22=0[3]) 3
(x1=1)&(x2=1)&(x3=0)&(x5=1)&(x7=0)&(x8=1)=>(x22=1[2]) 2
(x18=1)&(x1=1)&(x6=1)&(x14=1)&(x17=1)&(x21=0)&(x2=0)=>(x22=0[2]) 2
(x4=0)&(x9=1)&(x11=1)&(x18=1)&(x20=0)&(x1=1)&(x2=1)&(x3=1)&(x5=0)=>(
x22=0[1]) 1
(x1=0)&(x2=0)&(x3=0)=>(x22=0[1]) 1
```

Dla ułatwienia porównania reguły te można zapisać w formie wyrażeń boolowskich, co uczyniono poniżej:

Dla f=0:

```
!x3x2!x18 + x10x15x16!x5 + x1x17!x2!x6x12!x3x4x9 +
x1!x9!x6x12!x15!x16x17x21!x4 + x18x1x6x14x17!x21!x2 +
!x4x9x11x18!x20x1x2x3!x5 + !x1!x2!x3
```

Dla f=1:

```
x14!x2x21x4 + !x20!x4x9x14x3x7x13x16 + x1x2!x3x5!x7x8
```

Z kolei przy zastosowaniu programu InstantRS najpierw policzono wszystkie redukty o najmniejszej liczności. W rezultacie uzyskano 35 reduktów:

```
**************
* Instant RS *
**************

>> Started calculation at: 10. Sep 2012, 22:38:48
>> Elapsed: 0.3130 sec.
```

```
>> Smallest Arg Sets
01] x1  x5  x8  x10 x15
02] x3  x5  x8  x10 x15
03] x4  x5  x8  x10 x15
04] x5  x8  x10 x11 x15
05] x5  x8  x10 x12 x15
06] x5  x8  x10 x15 x17
07] x5  x8  x10 x15 x19
08] x2  x4  x5  x9  x18
09] x2  x5  x8  x17 x21
10] x2  x4  x5  x18 x21
11] x5  x8  x12 x13 x14
12] x2  x4  x16 x17 x21
13] x4  x16 x17 x19 x21
14] x2  x11 x16 x17 x21
15] x4  x9  x11 x13 x16
16] x4  x9  x14 x16 x17
17] x2  x4  x7  x9  x16
18] x4  x7  x9  x16 x19
19] x4  x9  x16 x17 x19
20] x4  x11 x12 x13 x16
21] x2  x4  x10 x19 x21
22] x4  x7  x9  x12 x19
23] x4  x9  x12 x17 x19
24] x2  x4  x7  x9  x19
25] x2  x4  x9  x10 x19
26] x2  x4  x9  x13 x19
27] x2  x4  x9  x15 x19
28] x2  x4  x9  x17 x19
29] x2  x4  x9  x18 x19
30] x2  x4  x9  x19 x20
31] x4  x7  x8  x9  x19
32] x2  x9  x11 x19 x20
33] x8  x10 x15 x19 x20
34] x10 x15 x17 x19 x20
35] x4  x10 x15 x17 x19
```

W drugim etapie obliczeń wygenerowano reguły dla jednego z obliczonych reduktów i uzyskano następujące reguły:

```
Started calculation at: 10. Sep 2012, 22:39:46
Calculcating Rule Set for Reduct: x2  x4  x5  x9  x18
Elapsed: 0.1250 sec.
Min Decision Rules (Reduced Set)
```

Dla f = 1:

```
!x2x4!x9 + x2x5x18 + !x2!x4x9!x18
```

Dla f = 0:

```
1] x2!x5 + x2!x18 + !x4!x9 + x4x9 + !x2x9x18
2] x2!x5 + x2!x18 + !x4!x9 + x4x9 + !x2!x4x18
```

Porównując oba rozwiązania należy zauważyć, że RSES dla decyzji f = 1 generuje 3 reguły o łącznej liczbie atrybutów równej 13, a dla decyzji f = 0

odpowiednio: 7 reguł o łącznej liczbie atrybutów 17. Znacząco prostsze są reguły obliczone programem InstantRS: z założenia mają one 5 atrybutów, a liczby reguł wynoszą odpowiednio: 3 dla f = 1, 5 dla f = 0.

Należy przy tym podkreślić, że uzyskiwana w programie InstantRS jakość obliczeń nie jest okupiona nieakceptowanym czasem obliczeń, gdyż zarówno redukty, jak też reguły są w przypadku funkcji KAZ liczone w ułamkach sekund. Na szybkość obliczeń w programie InstantRS największy wpływ ma błyskawicznie działająca procedura liczenia reduktów. W przypadku funkcji KAZ czas obliczenia wszystkich reduktów wynosi, dla RSES 45 min., natomiast InstantRS liczy wszystkie redukty w ciągu ok. 3 sekund.

```
>> Started calculation at: 12. Sep 2012, 20:01:33
>> Elapsed: 3.2760 sec.

>> Minimal Arg Sets
01] x1  x5  x8  x10 x15
02] x3  x5  x8  x10 x15
03] x4  x5  x8  x10 x15
04] x5  x8  x10 x11 x15
05] x5  x8  x10 x12 x15
06] x5  x8  x10 x15 x17
07] x5  x8  x10 x15 x19
08] x2  x3  x5  x8  x13 x15
09] x3  x4  x5  x8  x13 x15
10] x3  x5  x6  x8  x13 x15
...
...
...
5569] x3  x8  x11 x14 x15 x17 x19
5570] x3  x8  x11 x15 x17 x18 x19
5571] x1  x8  x11 x15 x17 x18 x19
5572] x7  x8  x11 x15 x17 x18 x19
5573] x8  x11 x13 x15 x17 x18 x19
5574] x8  x11 x13 x14 x15 x17 x19
```

Wnioski Końcowe

Przeprowadzone eksperymenty potwierdzają, że istniejące komputerowe narzędzia eksploracji danych nie w pełni wykorzystują możliwości algorytmów wypracowanych dla potrzeb syntezy logicznej. Wykazano, że zastosowanie w typowych procedurach eksploracji danych algorytmu uzupełniania funkcji boolowskiej przyśpiesza proces obliczania reduktów w tak znacznym stopniu, że realne staje się systematyczne obliczanie minimalnych reguł decyzyjnych. Oczywiście bariera złożoności obliczeniowej systematycznego obliczania reguł nie znika, przesuwa się tylko „punkt ciężkości" tych obliczeń. Przyczyną tego przesunięcia jest zmniejszenie wymiaru globalnej tablicy minimalnych reguł, czyli zmniejszenie liczności zbioru wszystkich reguł minimalnych potrzebnych do reprezentacji danej klasy decyzyjnej. Skalę tego zjawiska można zaobserwować na przykładzie funkcji KAZ, w której bezpośrednio obliczany zbiór minimalnych reguł ma liczność 5574. Barierę tę można jednak pokonać błyskawicznych algorytmem obliczania reduktów. Na przykład liczba minimalnych reguł dla reduktu x2 x4 x5 x9 x18 wynosi zaledwie 3.

Istotna zaletą programu InstantRS jest jego niezwykła przydatność do przeprowadzania eksperymentów. W szczególności program umożliwia:

a) obliczanie reduktów z najmniejszą liczbą atrybutów (opcja niedostępna w RSES);

b) obliczenie metodą systematyczną dla wybranego reduktu zbioru reguł minimalnych.

Ponadto program udostępnia podgląd – istotnych z punktu widzenia dalszych eksperymentów – wyników pośrednich. Z tych samych powodów istotną zaletą programu InstantRS jest umożliwienie obliczeń dla danych zapisanych zarówno w standardzie RSES jak też w standardzie Espresso. Niezmiernie ciekawe (i być może inspirujące do nowych i lepszych algorytmów) będzie porównanie wyników uzyskiwanych za pomocą InstantRS (również RSES) z wynikami programu espresso.

Wykaz tabel i rysunków

Tab. 1.1. Przykładowa tablica reguł decyzyjnych....6
Tab. 1.2. Dane o 24 pacjentach starających się o szkła kontaktowe [4]....7
Tab. 1.3. Przykładowa ankieta....8
Tab. 1.4. Przykładowa tablica prawdy....8
Tab. 2.1. Pokrycia [3]....12
Tab. 4.1. Przykład KAZ....21
Tab. 5.1. Opcje Menu File....27
Tab. 5.2. Opcje Menu Format....28
Tab. 5.3. Opcje Menu Run....29
Tab. 5.4. Opcje paska narzędziowego....31

Rys. 1.1. Kostka - opracowanie własne....10
Rys. 3.1. Obliczanie uzupełnienia funkcji jednorodnej – opracowanie własne....20
Rys. 5.1. Panel InstantRS....26
Rys. 5.2. Menu File....27
Rys. 5.3. Błąd programu (1)....27
Rys. 5.4. Błąd programu (2)....27
Rys. 5.5. Plik wejściowy....28
Rys. 5.6. Menu Format....28
Rys. 5.7. Menu Run....29
Rys. 5.8. Redukty....30
Rys. 5.9. Menu Info....31
Rys. 5.10. Pasek narzedziowy....31

Bibliografia

[1] Bazan J., Nguyen Son H., Nguyen Sinh H., Synak P., and Wróblewski J.: *Rough Set Algorithms in Classification Problems*. In: Lech Polkowski, Tsau Young Lin, and Shusaku Tsumoto (eds.): Rough Set Methods and Applications: New Developments in Knowledge Discovery in Information Systems, volume 56 of Studies in Fuzziness and Soft Computing, pp. 49–88. Physica-Verlag, Heidelberg, Germany, 2000.

[2] Borowik, G., Luba, T., Zydek, D.: Features Reduction using logic minimization techniques. In: Intl. Journal of Electronics and Telecommunications, vol. 58, No.1, pp. 71-76, (2012)

[3] Brayton R.K., Hachtel G.D., McMullen C.T., Sangiovanni-Vincentelli A.: *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, 1984.

[4] Gatnar E.: *Symboliczne metody klasyfikacji danych*. Wydawnictwo Naukowe PWN, Warszawa, 1998.

[5] Grzymała-Busse J.W.: *LERS – A System to Learning from Examples Based on Rough Sets*. In: Słowiński R. (ed.): Intelligent Decision Support – Handbook of Application and Advanced of the Rough Sets Theory, Kluwer Academic Publishers, 1992.

[6] Grzymala-Busse J. W.: *Data with Missing Attribute Values: Generalization of Indiscernibility Relation and Rule Induction*. In: Peters J.F. et al. (eds.): Transactions on Rough Sets I, LNCS 3100, pp. 78–95, Springer-Verlag, Berlin, 2004.

[7] Królikowski K.: *Implementacja algorytmu obliczania reduktów metodą uzupełniania funkcji boolowskich*, praca dyplomowa, WIT, Warszawa 2012.

[8] Kryszkiewicz M., Lasek P.: *FUN: Fast Discovery of Minimal Sets of Attributes Functionally Determining a Decision Attribute*. In: Peters J.F. et al. (eds.): Transactions on Rough Sets IX, LNCS 5390, pp. 76–95, Springer-Verlag, Berlin, 2008.

[9] Łuba T., Rybnik J.: *Rough Sets and Some Aspects in Logic synthesis*. In: Słowiński R. (ed.): Intelligent Decision Support – Handbook of Application and Advances of the Rough Sets Theory, Kluwer Academic Publishers, 1992.

[10] Łuba T.: *Synteza układów logicznych*, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2005.

[11] Nguyen D.T., Nguyen X.H.: *A New Method to Attribute Reduction of Decision Systems with Covering Rough Sets*. In: Georgian Electronic Scientific Journal: Computer Science and Telecommunications, vol. 1(24), pp. 24–31, 2010.

[12] Nowicki R. K.: *Rozmyte systemy decyzyjne w zadaniach z ograniczoną wiedzą*, Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2009.

[13] Pawlak Z.: *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, 1991.

[14] ROSETTA – A Rough Set Toolkit for Analysis of Data, http://www.lcb.uu.se/tools/rosetta/

[15] RSES – Rough Set Exploration System, http://logic.mimuw.edu.pl/~rses/

[16] Skowron A., Rauszer C.: *The Discernibility Matrices and Functions in Information Systems*. In: Słowiński R. (ed.): Intelligent Decision Support - Handbook of Application and Advances of the Rough Sets Theory, Kluwer Academic Publishers, 1992.

[17] Stefanowski J.: *Algorytmy indukcji reguł decyzyjnych w odkrywaniu wiedzy*. Rozprawa habilitacyjna, wersja z 8 lutego 2001, Wydawnictwo Politechniki Poznańskiej, Seria Rozprawy nr 361, 2001.

Streszczenie

Celem niniejszej pracy było stworzenie programu InstantRS, który znajduje szczególne zastosowanie przy przeprowadzaniu eksperymentów w obszarze minimalizacji reguł decyzyjnych. Główny algorytm obliczeniowy programu stanowi procedura uzupełniania funkcji boolowskich. W pracy zaprezentowano metodologię przeprowadzania doświadczeń i porównań służących poszukiwaniu jak najefektywniejszych rozwiązań uogólniania i redukcji zbioru reguł decyzyjnych.

Abstract

The main objective of this thesis was to provide a tool (InstantRS application) which would be particularly useful for conducting experiments in the area of decision rule minimization. The algorithm engine of the application is based on the complement of boolean functions. The thesis presents a methodology to carry out experiments and comparisons that would facilitate search for most effective solutions relating to decision rule set reduction and simplification.

Dodatek

Poniżej zaprezentowane zostały pełne wyniki otrzymywane z programu instantRS dla przykładów omawianych w podrozdziale 5.3.

Przykład House

```
>> Calculcating Rule Set for Reduct: x1  x2  x4  x6  x13 x14 x15 x16
>> Elapsed: 0.0630 sec.

>> Min Decision Rules (Reduced Set)
01]
(x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x13=1)*(x15=0)*
(x16=1)+(x2=0)*(x6=0)*(x14=0)*(x16=1)+(x4=0)*(x13=1)*(x14=1) => R=0
02]
(x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x4=1)*(x13=1)*(
x14=0)*(x16=1)+(x2=0)*(x6=0)*(x14=0)*(x16=1)+(x4=0)*(x13=1)*(x14=1) => R=0
03]
(x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x2=0)*(x4=1)*(x
14=0)*(x16=1)+(x2=0)*(x6=0)*(x14=0)*(x16=1)+(x4=0)*(x13=1)*(x14=1) => R=0
04]
(x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x2=0)*(x6=0)*(x
14=0)*(x16=1)+(x1=0)*(x13=1)*(x16=1)+(x4=0)*(x13=1)*(x14=1) => R=0
05]
(x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x2=0)*(x6=0)*(x
14=0)*(x16=1)+(x1=0)*(x2=0)*(x16=1)+(x4=0)*(x13=1)*(x14=1) => R=0
06]
(x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x2=0)*(x6=0)*(x
14=0)*(x16=1)+(x4=0)*(x13=1)*(x14=1)+(x1=0)*(x15=0)*(x16=1) => R=0
07]
(x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x13=1)*(x15=0)*
(x16=1)+(x4=0)*(x13=1)*(x14=1)+(x2=0)*(x4=0)*(x6=0)*(x14=0) => R=0
08]
(x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x4=1)*(x13=1)*(
x14=0)*(x16=1)+(x4=0)*(x13=1)*(x14=1)+(x2=0)*(x4=0)*(x6=0)*(x14=0) => R=0
09]
(x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x2=0)*(x4=1)*(x
14=0)*(x16=1)+(x4=0)*(x13=1)*(x14=1)+(x2=0)*(x4=0)*(x6=0)*(x14=0) => R=0
10]
(x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x1=0)*(x13=1)*(
x16=1)+(x4=0)*(x13=1)*(x14=1)+(x2=0)*(x4=0)*(x6=0)*(x14=0) => R=0
11]
(x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x1=0)*(x2=0)*(x
16=1)+(x4=0)*(x13=1)*(x14=1)+(x2=0)*(x4=0)*(x6=0)*(x14=0) => R=0
12]
(x6=0)*(x13=1)+(x4=1)*(x13=1)*(x14=0)*(x15=1)+(x1=1)*(x13=1)*(x15=0)+(x4=0)*(x13=1)*(
x14=1)+(x2=0)*(x4=0)*(x6=0)*(x14=0)+(x1=0)*(x15=0)*(x16=1) => R=0
```

Przykład B11

```
>> Min Decision Rules (Reduced Set)
1] (x2=0)*(x3=0)+(x3=1)*(x5=0)+(x2=1)*(x3=1) => R=0
2] (x2=0)*(x3=0)+(x2=1)*(x3=1)+(x2=0)*(x5=0) => R=0
3] (x2=0)*(x3=1)*(x5=1)+(x2=1)*(x3=0) => R=1
```

Przykład A20

```
>> Min Decision Rules (Reduced Set)
01] (x4=1)*(x5=1)*(x8=0)+(x2=0)*(x4=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0) =>
R=0
02] (x4=1)*(x5=1)*(x8=0)+(x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0) =>
R=0
03] (x4=1)*(x5=1)*(x8=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x3=0)*(x7=0) => R=0
04] (x4=1)*(x5=1)*(x8=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x3=0)*(x4=1) =>
R=0
```

05] (x4=1)*(x5=1)*(x8=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x2=0)*(x3=0)*(x4=1) =>
R=0
06] (x4=1)*(x5=1)*(x8=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x3=0)*(x4=1)*(x5=1) =>
R=0
07] (x4=1)*(x5=1)*(x8=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x4=1)*(x6=0) =>
R=0
08] (x2=0)*(x4=1)*(x8=0)+(x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0) =>
R=0
09] (x2=0)*(x4=1)*(x8=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x4=1)*(x6=0) =>
R=0
10] (x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x3=0)*(x6=1) =>
R=0
11] (x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x6=1)*(x8=0) =>
R=0
12] (x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x3=0)*(x8=0) =>
R=0
13] (x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x2=0)*(x3=0)*(x8=0) =>
R=0
14] (x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x2=0)*(x6=1)*(x8=0) =>
R=0
15] (x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x2=0)*(x3=0)*(x6=1) =>
R=0
16] (x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x3=0)*(x4=1) =>
R=0
17] (x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x2=0)*(x3=0)*(x4=1) =>
R=0
18] (x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x3=0)*(x4=1)*(x5=1) =>
R=0
19] (x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x3=0)*(x4=1)*(x6=1) =>
R=0
20] (x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x3=0)*(x4=1)*(x8=0) =>
R=0
21] (x4=1)*(x5=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x4=1)*(x8=0) =>
R=0
22] (x2=0)*(x4=1)*(x6=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x4=1)*(x8=0) =>
R=0
23] (x4=1)*(x6=0)*(x8=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x3=0)*(x4=1) =>
R=0
24] (x4=1)*(x6=0)*(x8=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x2=0)*(x3=0)*(x4=1) =>
R=0
25] (x4=1)*(x6=0)*(x8=0)+(x4=0)*(x5=0)+(x2=1)*(x5=1)*(x7=0)+(x3=0)*(x4=1)*(x5=1) =>
R=0
26]
(x4=1)*(x5=1)*(x6=0)+(x3=1)*(x4=0)*(x8=1)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x3=0)*(x6=1)
=> R=0
27]
(x4=1)*(x5=1)*(x6=0)+(x3=1)*(x4=0)*(x8=1)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x6=1)*(x8=0)
=> R=0
28]
(x4=1)*(x5=1)*(x6=0)+(x3=1)*(x4=0)*(x8=1)+(x2=1)*(x5=1)*(x7=0)+(x1=1)*(x3=0)*(x8=0)
=> R=0
29]
(x4=1)*(x5=1)*(x6=0)+(x3=1)*(x4=0)*(x8=1)+(x2=1)*(x5=1)*(x7=0)+(x2=0)*(x3=0)*(x8=0)
=> R=0
30]
(x4=1)*(x5=1)*(x6=0)+(x3=1)*(x4=0)*(x8=1)+(x2=1)*(x5=1)*(x7=0)+(x2=0)*(x6=1)*(x8=0)
=> R=0
31]
(x4=1)*(x5=1)*(x6=0)+(x3=1)*(x4=0)*(x8=1)+(x2=1)*(x5=1)*(x7=0)+(x2=0)*(x3=0)*(x6=1)
=> R=0
32] (x2=0)*(x6=1)*(x8=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x4=0)*(x5=1)*(x6=0) =>
R=1
33] (x2=0)*(x6=1)*(x8=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x4=0)*(x6=0)*(x8=0) =>
R=1
34] (x2=0)*(x6=1)*(x8=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x1=1)*(x4=0)*(x5=1) =>
R=1
35] (x2=0)*(x6=1)*(x8=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x2=0)*(x4=0)*(x5=1) =>
R=1
36] (x2=0)*(x6=1)*(x8=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x1=1)*(x4=0)*(x7=0) =>
R=1
37] (x2=0)*(x6=1)*(x8=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x2=0)*(x4=0)*(x7=0) =>
R=1
38] (x2=0)*(x6=1)*(x8=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x4=0)*(x6=0)*(x7=0) =>
R=1
39] (x2=0)*(x6=1)*(x8=1)+(x1=0)*(x3=0)+(x4=1)*(x5=0)+(x4=0)*(x5=1)*(x6=0) => R=1
40] (x2=0)*(x6=1)*(x8=1)+(x2=1)*(x3=0)+(x4=1)*(x5=0)+(x4=0)*(x5=1)*(x6=0) => R=1

```
41] (x2=0)*(x6=1)*(x8=1)+(x3=0)*(x4=0)*(x5=1)+(x4=1)*(x5=0)+(x4=0)*(x5=1)*(x6=0) =>
R=1
42] (x2=0)*(x6=1)*(x8=1)+(x1=0)*(x7=1)+(x4=1)*(x5=0)+(x4=0)*(x5=1)*(x6=0) => R=1
43] (x2=0)*(x6=1)*(x8=1)+(x2=1)*(x4=0)*(x7=1)+(x4=1)*(x5=0)+(x4=0)*(x5=1)*(x6=0) =>
R=1
44] (x2=0)*(x6=1)*(x8=1)+(x1=0)*(x7=1)+(x4=1)*(x5=0)+(x1=1)*(x4=0)*(x5=1) => R=1
45] (x2=0)*(x6=1)*(x8=1)+(x1=0)*(x7=1)+(x4=1)*(x5=0)+(x2=0)*(x4=0)*(x5=1) => R=1
46] (x2=0)*(x6=1)*(x8=1)+(x2=1)*(x4=0)*(x7=1)+(x4=1)*(x5=0)+(x1=1)*(x4=0)*(x5=1) =>
R=1
47] (x2=0)*(x6=1)*(x8=1)+(x2=1)*(x4=0)*(x7=1)+(x4=1)*(x5=0)+(x2=0)*(x4=0)*(x5=1) =>
R=1
48] (x2=0)*(x6=1)*(x8=1)+(x3=0)*(x4=0)*(x5=1)+(x4=1)*(x5=0)+(x4=0)*(x6=0)*(x8=0) =>
R=1
49] (x2=0)*(x3=1)*(x6=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x4=0)*(x5=1)*(x6=0) =>
R=1
50] (x2=0)*(x3=1)*(x6=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x4=0)*(x6=0)*(x8=0) =>
R=1
51] (x2=0)*(x3=1)*(x6=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x1=1)*(x4=0)*(x5=1) =>
R=1
52] (x2=0)*(x3=1)*(x6=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x2=0)*(x4=0)*(x5=1) =>
R=1
53] (x2=0)*(x3=1)*(x6=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x1=1)*(x4=0)*(x7=0) =>
R=1
54] (x2=0)*(x3=1)*(x6=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x2=0)*(x4=0)*(x7=0) =>
R=1
55] (x2=0)*(x3=1)*(x6=1)+(x4=0)*(x5=1)*(x7=1)+(x4=1)*(x5=0)+(x4=0)*(x6=0)*(x7=0) =>
R=1
56] (x2=0)*(x3=1)*(x6=1)+(x1=0)*(x3=0)+(x4=1)*(x5=0)+(x4=0)*(x5=1)*(x6=0) => R=1
57] (x2=0)*(x3=1)*(x6=1)+(x2=1)*(x3=0)+(x4=1)*(x5=0)+(x4=0)*(x5=1)*(x6=0) => R=1
58] (x2=0)*(x3=1)*(x6=1)+(x3=0)*(x4=0)*(x5=1)+(x4=1)*(x5=0)+(x4=0)*(x5=1)*(x6=0) =>
R=1
59] (x2=0)*(x3=1)*(x6=1)+(x1=0)*(x7=1)+(x4=1)*(x5=0)+(x4=0)*(x5=1)*(x6=0) => R=1
60] (x2=0)*(x3=1)*(x6=1)+(x2=1)*(x4=0)*(x7=1)+(x4=1)*(x5=0)+(x4=0)*(x5=1)*(x6=0) =>
R=1
61] (x2=0)*(x3=1)*(x6=1)+(x1=0)*(x7=1)+(x4=1)*(x5=0)+(x1=1)*(x4=0)*(x5=1) => R=1
62] (x2=0)*(x3=1)*(x6=1)+(x1=0)*(x7=1)+(x4=1)*(x5=0)+(x2=0)*(x4=0)*(x5=1) => R=1
63] (x2=0)*(x3=1)*(x6=1)+(x2=1)*(x4=0)*(x7=1)+(x4=1)*(x5=0)+(x1=1)*(x4=0)*(x5=1) =>
R=1
64] (x2=0)*(x3=1)*(x6=1)+(x2=1)*(x4=0)*(x7=1)+(x4=1)*(x5=0)+(x2=0)*(x4=0)*(x5=1) =>
R=1
65] (x2=0)*(x3=1)*(x6=1)+(x3=0)*(x4=0)*(x5=1)+(x4=1)*(x5=0)+(x4=0)*(x6=0)*(x8=0) =>
R=1
```

Przykład T27

```
>> Min Decision Rules (Reduced Set)
01]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)*
(x8=0) => R=0

02]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x4=0)*(x6=1)*
(x7=0) => R=0
03]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x6=1)*(x7=0)*
(x10=0) => R=0
04]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x1=0)*(x2=1)*(x6=1)*
(x7=0) => R=0
05]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x1=0)*(x5=1)*(x6=1)*
(x7=0) => R=0
06]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x5=1)*(x6=1)*
(x7=0) => R=0
07]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x5=1)*(x6=1)*(x7=0)*
(x9=0) => R=0
08]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x5=1)*(x6=1)*(x7=0)*
(x10=0) => R=0
09]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x1=0)*(x6=1)*(x7=0)*
(x8=0) => R=0
```

```
10]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x6=1)*(x7=0)*(x8=0)*
(x10=0) => R=0
11]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x4=0)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)
*(x8=0) => R=0
12]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x4=0)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x4=0)*(x6=1)
*(x7=0) => R=0
13]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x8=1)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)
*(x8=0) => R=0
14]
(x2=0)*(x5=1)*(x8=0)+(x1=1)*(x8=1)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x4=0)*(x6=1)
*(x7=0) => R=0
15]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)*
(x8=0) => R=0
16]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x4=0)*(x6=1)*
(x7=0) => R=0
17]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x6=1)*(x7=0)*
(x10=0) => R=0
18]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x1=0)*(x2=1)*(x6=1)*
(x7=0) => R=0
19]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x1=0)*(x5=1)*(x6=1)*
(x7=0) => R=0
20]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x5=1)*(x6=1)*
(x7=0) => R=0
21]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x5=1)*(x6=1)*(x7=0)*
(x9=0) => R=0
22]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x5=1)*(x6=1)*(x7=0)*
(x10=0) => R=0
23]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x1=0)*(x6=1)*(x7=0)*
(x8=0) => R=0
24]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x6=1)*(x7=0)*(x8=0)*
(x10=0) => R=0
25]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x4=0)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)
*(x8=0) => R=0
26]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x4=0)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x4=0)*(x6=1)
*(x7=0) => R=0
27]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x8=1)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)
*(x8=0) => R=0
28]
(x2=0)*(x5=1)*(x9=1)+(x1=1)*(x8=1)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x4=0)*(x6=1)
*(x7=0) => R=0
29]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)*
(x8=0) => R=0
30]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x4=0)*(x6=1)*
(x7=0) => R=0
31]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x6=1)*(x7=0)*
(x10=0) => R=0
32]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x1=0)*(x2=1)*(x6=1)*
(x7=0) => R=0
33]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x1=0)*(x5=1)*(x6=1)*
(x7=0) => R=0
34]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x5=1)*(x6=1)*
(x7=0) => R=0
```

```
35]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x5=1)*(x6=1)*(x7=0)*
(x9=0) => R=0
36]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x5=1)*(x6=1)*(x7=0)*
(x10=0) => R=0
37]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x1=0)*(x6=1)*(x7=0)*
(x8=0) => R=0
38]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x6=1)*(x7=0)*(x8=0)*
(x10=0) => R=0
39]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)
*(x8=0) => R=0
40]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x4=0)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x4=0)*(x6=1)
*(x7=0) => R=0
41]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x8=1)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)
*(x8=0) => R=0
42]
(x2=0)*(x7=1)*(x8=0)+(x1=1)*(x8=1)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x4=0)*(x6=1)
*(x7=0) => R=0
43]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)*
(x8=0) => R=0
44]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x4=0)*(x6=1)*
(x7=0) => R=0
45]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x6=1)*(x7=0)*
(x10=0) => R=0
46]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x1=0)*(x2=1)*(x6=1)*
(x7=0) => R=0
47]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x1=0)*(x5=1)*(x6=1)*
(x7=0) => R=0
48]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x5=1)*(x6=1)*
(x7=0) => R=0
49]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x5=1)*(x6=1)*(x7=0)*
(x9=0) => R=0
50]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x5=1)*(x6=1)*(x7=0)*
(x10=0) => R=0
51]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x1=0)*(x6=1)*(x7=0)*
(x8=0) => R=0
52]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x4=0)*(x7=0)+(x3=0)*(x9=1)*(x10=0)+(x6=1)*(x7=0)*(x8=0)*
(x10=0) => R=0
53]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x4=0)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)
*(x8=0) => R=0
54]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x4=0)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x4=0)*(x6=1)
*(x7=0) => R=0
55]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x8=1)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x4=0)*(x6=1)*(x7=0)
*(x8=0) => R=0
56]
(x2=0)*(x8=0)*(x9=1)+(x1=1)*(x8=1)*(x10=0)+(x3=0)*(x9=1)*(x10=0)+(x2=1)*(x4=0)*(x6=1)
*(x7=0) => R=0
57]
(x1=0)*(x10=1)+(x3=0)*(x5=0)*(x9=0)+(x1=1)*(x2=1)*(x4=1)+(x1=0)*(x4=0)*(x6=0)+(x7=1)*
(x9=0) => R=1
58]
(x1=0)*(x10=1)+(x2=0)*(x3=0)+(x1=1)*(x2=1)*(x4=1)+(x1=0)*(x4=0)*(x6=0)+(x7=1)*(x9=0)
=> R=1
59]
(x1=0)*(x10=1)+(x1=0)*(x2=0)*(x5=0)+(x1=1)*(x2=1)*(x4=1)+(x1=0)*(x4=0)*(x6=0)+(x7=1)*
(x9=0) => R=1
```

```
60]
(x1=0)*(x10=1)+(x1=0)*(x2=0)*(x7=0)+(x1=1)*(x2=1)*(x4=1)+(x1=0)*(x4=0)*(x6=0)+(x7=1)*
(x9=0) => R=1
61]
(x1=0)*(x10=1)+(x1=0)*(x2=0)*(x8=1)+(x1=1)*(x2=1)*(x4=1)+(x1=0)*(x4=0)*(x6=0)+(x7=1)*
(x9=0) => R=1
```

Przykład G42

```
001] (x2=0)*(x3=1)*(x8=1)+(x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x2=1)*(x12=1) => R=0
002] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x3=0)*(x9=1)*(x11=1) => R=0
003] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x3=0)*(x9=1)*(x11=1) => R=0
004] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x3=0)*(x4=1)*(x9=1)*(x11=1) => R=0
005] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x3=0)*(x6=1)*(x9=1)*(x11=1) => R=0
006] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x3=0)*(x7=1)*(x9=1)*(x11=1) => R=0
007] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x3=0)*(x9=1)*(x10=0)*(x11=1) => R=0
008] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x8=1)*(x9=1)*(x11=1) => R=0
009] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x8=1)*(x9=1)*(x11=1) => R=0
010] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x4=1)*(x8=1)*(x9=1)*(x11=1) => R=0
011] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x6=1)*(x8=1)*(x9=1)*(x11=1) => R=0
012] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x7=1)*(x8=1)*(x9=1)*(x11=1) => R=0
013] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x8=1)*(x9=1)*(x10=0)*(x11=1) => R=0
014] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x6=1)*(x9=1)*(x11=1) => R=0
015] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x9=1)*(x11=1)*(x12=0) => R=0
016] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x4=1)*(x5=1)*(x9=1)*(x11=1) => R=0
017] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x4=1)*(x9=1)*(x10=0)*(x11=1) => R=0
018] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x5=1)*(x6=1)*(x9=1)*(x11=1) => R=0
019] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x5=1)*(x7=1)*(x9=1)*(x11=1) => R=0
020] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x8=1)*(x11=1)*(x12=0) => R=0
021] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x8=1)*(x11=1)*(x12=0) => R=0
022] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x4=1)*(x8=1)*(x11=1)*(x12=0) => R=0
023] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x6=1)*(x8=1)*(x11=1)*(x12=0) => R=0
024] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x7=1)*(x8=1)*(x11=1)*(x12=0) => R=0
025] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x8=1)*(x10=0)*(x11=1)*(x12=0) => R=0
026] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x3=0)*(x11=1)*(x12=0) => R=0
027] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x3=0)*(x4=1)*(x11=1)*(x12=0) => R=0
028] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x3=0)*(x6=1)*(x11=1)*(x12=0) => R=0
029] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x3=0)*(x7=1)*(x11=1)*(x12=0) => R=0
030] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x3=0)*(x10=0)*(x11=1)*(x12=0) => R=0
031] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x5=1)*(x11=1)*(x12=0) => R=0
032] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x4=1)*(x5=1)*(x11=1)*(x12=0) => R=0
033] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x5=1)*(x6=1)*(x11=1)*(x12=0) => R=0
034] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x5=1)*(x7=1)*(x11=1)*(x12=0) => R=0
035] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x5=1)*(x10=0)*(x11=1)*(x12=0) => R=0
036] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x6=1)*(x11=1)*(x12=0) => R=0
037] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x5=1)*(x6=1)*(x11=1) => R=0
038] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x6=1)*(x7=1)*(x11=1) => R=0
039] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x8=1)*(x10=0)*(x11=1) => R=0
040] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x6=1)*(x7=1)*(x8=1)*(x11=1) => R=0
041] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x2=0)*(x8=1)*(x12=0) => R=0
042] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x4=1)*(x8=1)*(x12=0) => R=0
043] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x5=1)*(x8=1)*(x12=0) => R=0
044] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x8=1)*(x9=1)*(x12=0) => R=0
045] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x8=1)*(x10=0)*(x12=0) => R=0
046] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x7=1)*(x8=1)*(x12=0) => R=0
047] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x4=1)*(x7=1)*(x8=1)*(x12=0) => R=0
048] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x5=1)*(x7=1)*(x8=1)*(x12=0) => R=0
049] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x7=1)*(x8=1)*(x9=1)*(x12=0) => R=0
050] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x7=1)*(x8=1)*(x10=0)*(x12=0) => R=0
051] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x5=1)*(x6=1)*(x12=0) => R=0
052] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x5=1)*(x6=1)*(x12=0) => R=0
053] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x2=1)*(x12=1)+(x5=1)*(x6=1)*(x7=1)*(x12=0) => R=0
054] (x2=0)*(x3=1)*(x8=1)+(x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x8=0)*(x12=1) => R=0
055] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x3=0)*(x9=1)*(x11=1) => R=0
056] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x3=0)*(x9=1)*(x11=1) => R=0
057] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x3=0)*(x4=1)*(x9=1)*(x11=1) => R=0
058] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x3=0)*(x6=1)*(x9=1)*(x11=1) => R=0
059] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x3=0)*(x7=1)*(x9=1)*(x11=1) => R=0
060] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x3=0)*(x9=1)*(x10=0)*(x11=1) => R=0
061] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x8=1)*(x9=1)*(x11=1) => R=0
062] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x8=1)*(x9=1)*(x11=1) => R=0
063] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x4=1)*(x8=1)*(x9=1)*(x11=1) => R=0
064] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x6=1)*(x8=1)*(x9=1)*(x11=1) => R=0
065] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x7=1)*(x8=1)*(x9=1)*(x11=1) => R=0
066] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x8=1)*(x9=1)*(x10=0)*(x11=1) => R=0
```

```
067] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x6=1)*(x9=1)*(x11=1) => R=0
068] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x9=1)*(x11=1)*(x12=0) => R=0
069] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x4=1)*(x5=1)*(x9=1)*(x11=1) => R=0
070] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x4=1)*(x9=1)*(x10=0)*(x11=1) => R=0
071] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x5=1)*(x6=1)*(x9=1)*(x11=1) => R=0
072] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x5=1)*(x7=1)*(x9=1)*(x11=1) => R=0
073] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x8=1)*(x11=1)*(x12=0) => R=0
074] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x8=1)*(x11=1)*(x12=0) => R=0
075] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x4=1)*(x8=1)*(x11=1)*(x12=0) => R=0
076] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x6=1)*(x8=1)*(x11=1)*(x12=0) => R=0
077] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x7=1)*(x8=1)*(x11=1)*(x12=0) => R=0
078] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x8=1)*(x10=0)*(x11=1)*(x12=0) => R=0
079] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x3=0)*(x11=1)*(x12=0) => R=0
080] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x3=0)*(x4=1)*(x11=1)*(x12=0) => R=0
081] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x3=0)*(x6=1)*(x11=1)*(x12=0) => R=0
082] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x3=0)*(x7=1)*(x11=1)*(x12=0) => R=0
083] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x3=0)*(x10=0)*(x11=1)*(x12=0) => R=0
084] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x5=1)*(x11=1)*(x12=0) => R=0
085] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x4=1)*(x5=1)*(x11=1)*(x12=0) => R=0
086] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x5=1)*(x6=1)*(x11=1)*(x12=0) => R=0
087] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x5=1)*(x7=1)*(x11=1)*(x12=0) => R=0
088] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x5=1)*(x10=0)*(x11=1)*(x12=0) => R=0
089] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x6=1)*(x11=1)*(x12=0) => R=0
090] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x5=1)*(x6=1)*(x11=1) => R=0
091] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x6=1)*(x7=1)*(x11=1) => R=0
092] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x8=1)*(x10=0)*(x11=1) => R=0
093] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x6=1)*(x7=1)*(x8=1)*(x11=1) => R=0
094] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x2=0)*(x8=1)*(x12=0) => R=0
095] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x4=1)*(x8=1)*(x12=0) => R=0
096] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x5=1)*(x8=1)*(x12=0) => R=0
097] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x8=1)*(x9=1)*(x12=0) => R=0
098] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x8=1)*(x10=0)*(x12=0) => R=0
099] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x7=1)*(x8=1)*(x12=0) => R=0
100] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x4=1)*(x7=1)*(x8=1)*(x12=0) => R=0
101] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x5=1)*(x7=1)*(x8=1)*(x12=0) => R=0
102] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x7=1)*(x8=1)*(x9=1)*(x12=0) => R=0
103] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x7=1)*(x8=1)*(x10=0)*(x12=0) => R=0
104] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x5=1)*(x6=1)*(x12=0) => R=0
105] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x5=1)*(x6=1)*(x12=0) => R=0
106] (x2=0)*(x3=1)*(x8=1)+(x1=0)*(x8=0)*(x12=1)+(x5=1)*(x6=1)*(x7=1)*(x12=0) => R=0
107] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x2=0)*(x8=1)*(x12=0)
=> R=0
108] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x4=1)*(x8=1)*(x12=0)
=> R=0
109] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x8=1)*(x9=1)*(x12=0)
=> R=0
110] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x8=1)*(x10=0)*(x12=0)
=> R=0
111] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x7=1)*(x8=1)*(x12=0)
=> R=0
112] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x4=1)*(x7=1)*(x8=1)*(x12=0)
=> R=0
113] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x7=1)*(x8=1)*(x9=1)*(x12=0)
=> R=0
114] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x7=1)*(x8=1)*(x10=0)*(x12=0)
=> R=0
115] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x3=1)*(x5=0)*(x8=1) => R=0
116] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x2=0)*(x8=1)*(x12=0)
=> R=0
117] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x4=1)*(x8=1)*(x12=0)
=> R=0
118] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x8=1)*(x9=1)*(x12=0)
=> R=0
119] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x8=1)*(x10=0)*(x12=0)
=> R=0
120] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x7=1)*(x8=1)*(x12=0)
=> R=0
121] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x4=1)*(x7=1)*(x8=1)*(x12=0)
=> R=0
122] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x7=1)*(x8=1)*(x9=1)*(x12=0)
=> R=0
123] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x7=1)*(x8=1)*(x10=0)*(x12=0)
=> R=0
124] (x2=0)*(x5=1)*(x8=1)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x3=1)*(x5=0)*(x8=1) => R=0
125]
(x3=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x2=0)*(x8=1)*(x12=0) =>
R=0
```

126]
(x3=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x8=1)*(x9=1)*(x12=0) =>
R=0
127]
(x3=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x8=1)*(x10=0)*(x12=0) =>
R=0
128]
(x3=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x7=1)*(x8=1)*(x12=0) =>
R=0
129]
(x3=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x7=1)*(x8=1)*(x9=1)*(x12=0) =>
R=0
130]
(x3=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x7=1)*(x8=1)*(x10=0)*(x12=0) =>
R=0
131]
(x3=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x2=0)*(x8=1)*(x12=0) =>
R=0
132]
(x3=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x8=1)*(x9=1)*(x12=0) =>
R=0
133]
(x3=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x8=1)*(x10=0)*(x12=0) =>
R=0
134]
(x3=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x7=1)*(x8=1)*(x12=0) =>
R=0
135]
(x3=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x7=1)*(x8=1)*(x9=1)*(x12=0) =>
R=0
136]
(x3=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x7=1)*(x8=1)*(x10=0)*(x12=0) =>
R=0
137]
(x5=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x2=0)*(x8=1)*(x12=0) =>
R=0
138]
(x5=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x8=1)*(x9=1)*(x12=0) =>
R=0
139]
(x5=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x1=1)*(x8=1)*(x10=0)*(x12=0) =>
R=0
140]
(x5=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x2=0)*(x7=1)*(x8=1)*(x12=0) =>
R=0
141]
(x5=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x7=1)*(x8=1)*(x9=1)*(x12=0) =>
R=0
142]
(x5=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x2=1)*(x12=1)+(x7=1)*(x8=1)*(x10=0)*(x12=0) =>
R=0
143]
(x5=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x2=0)*(x8=1)*(x12=0) =>
R=0
144]
(x5=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x8=1)*(x9=1)*(x12=0) =>
R=0
145]
(x5=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x1=1)*(x8=1)*(x10=0)*(x12=0) =>
R=0
146]
(x5=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x2=0)*(x7=1)*(x8=1)*(x12=0) =>
R=0
147]
(x5=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x7=1)*(x8=1)*(x9=1)*(x12=0) =>
R=0
148]
(x5=1)*(x10=1)*(x11=0)*(x12=0)+(x1=0)*(x8=0)*(x12=1)+(x7=1)*(x8=1)*(x10=0)*(x12=0) =>
R=0
149] (x3=0)*(x6=0)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x12=1) => R=1
150] (x3=0)*(x6=0)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x12=1) => R=1
151] (x3=0)*(x6=0)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
152] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x12=1) => R=1
153] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x12=1) => R=1
154] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
155] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x12=1) => R=1
156] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x12=1) => R=1

157] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
158] (x1=0)*(x4=0)+(x3=0)*(x7=0)+(x8=1)*(x9=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) => R=1
159] (x3=0)*(x4=0)+(x3=0)*(x6=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) => R=1
160] (x4=0)*(x5=0)+(x3=0)*(x6=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) => R=1
161] (x3=0)*(x6=0)+(x4=0)*(x10=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) => R=1
162] (x3=0)*(x6=0)+(x4=0)*(x9=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) => R=1
163] (x3=0)*(x6=0)+(x8=1)*(x9=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) => R=1
164] (x3=0)*(x6=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x12=1) => R=1
165] (x3=0)*(x6=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x12=1) => R=1
166] (x3=0)*(x6=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
167] (x3=0)*(x6=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x9=0)*(x12=0)+(x1=1)*(x12=1) => R=1
168] (x3=0)*(x6=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x4=0)*(x6=1)+(x1=1)*(x12=1) => R=1
169] (x3=0)*(x6=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x10=1)+(x1=1)*(x12=1) => R=1
170] (x3=0)*(x6=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x9=0)+(x1=1)*(x12=1) => R=1
171] (x3=0)*(x6=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1)+(x1=1)*(x2=1) => R=1
172] (x3=0)*(x4=0)+(x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) => R=1
173] (x2=0)*(x12=1)+(x4=0)*(x10=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) =>
R=1
174] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x12=1) => R=1
175] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x12=1) => R=1
176] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) =>
R=1
177] (x3=0)*(x4=0)+(x8=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) => R=1
178] (x8=1)*(x12=1)+(x4=0)*(x10=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) =>
R=1
179] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x12=1) => R=1
180] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x12=1) => R=1
181] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) =>
R=1
182] (x1=0)*(x4=0)+(x8=1)*(x9=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) => R=1
183] (x3=0)*(x4=0)+(x8=1)*(x9=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) => R=1
184] (x3=0)*(x4=0)+(x2=0)*(x9=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) => R=1
185] (x3=0)*(x6=0)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x12=1) => R=1
186] (x3=0)*(x6=0)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x12=1) => R=1
187] (x3=0)*(x6=0)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
188] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x12=1) => R=1
189] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x12=1) => R=1
190] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
191] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x12=1) => R=1
192] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x12=1) => R=1
193] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
194] (x1=0)*(x4=0)+(x8=1)*(x9=0)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x1=1)*(x12=1) => R=1
195] (x3=0)*(x6=0)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x12=1) => R=1
196] (x2=0)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x12=1) => R=1
197] (x8=1)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x12=1) => R=1
198] (x3=0)*(x6=0)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
199] (x2=0)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
200] (x8=1)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
201] (x3=0)*(x6=0)+(x7=0)*(x10=0)+(x8=0)*(x9=1)+(x9=0)*(x12=0)+(x1=1)*(x12=1) => R=1
202] (x3=0)*(x6=0)+(x7=0)*(x10=0)+(x8=0)*(x9=1)+(x1=1)*(x9=0)+(x1=1)*(x12=1) => R=1
203] (x3=0)*(x6=0)+(x7=0)*(x10=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
204] (x3=0)*(x6=0)+(x6=1)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
205] (x3=0)*(x6=0)+(x3=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
206] (x3=0)*(x6=0)+(x5=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
207] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
208] (x2=0)*(x12=1)+(x5=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
209] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
210] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
211] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
212] (x8=1)*(x12=1)+(x5=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
213] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
214] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x12=1) => R=1
215] (x1=0)*(x4=0)+(x2=0)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) => R=1
216] (x2=0)*(x12=1)+(x4=0)*(x7=0)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) => R=1
217] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) => R=1
218] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x2=1) => R=1
219] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x7=0)+(x1=1)*(x2=1) => R=1
220] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x2=1) => R=1
221] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x9=1)+(x1=1)*(x2=1) => R=1
222] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x2=1) => R=1
223] (x1=0)*(x4=0)+(x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) => R=1
224] (x3=0)*(x4=0)+(x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) => R=1
225] (x3=0)*(x6=0)+(x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) => R=1
226] (x2=0)*(x12=1)+(x4=0)*(x10=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) => R=1
227] (x2=0)*(x12=1)+(x4=0)*(x7=0)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) => R=1
228] (x2=0)*(x12=1)+(x4=0)*(x11=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) => R=1
229] (x2=0)*(x12=1)+(x7=0)*(x11=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) => R=1

```
230] (x2=0)*(x12=1)+(x10=1)*(x11=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) =>
R=1
231] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x2=1) => R=1
232] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x7=0)+(x1=1)*(x2=1) => R=1
233] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x2=1) => R=1
234] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x9=1)+(x1=1)*(x2=1) => R=1
235] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x2=1) => R=1
236] (x1=0)*(x4=0)+(x2=0)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) => R=1
237] (x2=0)*(x12=1)+(x4=0)*(x7=0)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x1=1)*(x2=1) => R=1
238] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x2=1) => R=1
239] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x7=0)+(x1=1)*(x2=1) => R=1
240] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x2=1) => R=1
241] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x9=1)+(x1=1)*(x2=1) => R=1
242] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x2=1) => R=1
243] (x2=0)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x7=0)+(x1=1)*(x2=1) => R=1
244] (x2=0)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x2=1) => R=1
245] (x2=0)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x9=1)+(x1=1)*(x2=1) => R=1
246] (x2=0)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x2=1) => R=1
247] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x2=1) => R=1
248] (x2=0)*(x12=1)+(x5=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x2=1) => R=1
249] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x2=1) => R=1
250] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x2=1) => R=1
251] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x8=0) => R=1
252] (x2=0)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x8=0) => R=1
253] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x8=0) => R=1
254] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x8=0) => R=1
255] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x9=0) => R=1
256] (x2=0)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x9=1)+(x1=1)*(x9=0) => R=1
257] (x3=0)*(x6=0)+(x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x9=0) => R=1
258] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x9=0) => R=1
259] (x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x9=1)+(x1=1)*(x9=0) => R=1
260] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x9=0) => R=1
261] (x2=0)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x9=1)+(x1=1)*(x9=0) => R=1
262] (x2=0)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x9=0) => R=1
263] (x2=0)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x9=1)+(x1=1)*(x9=0) => R=1
264] (x3=0)*(x6=0)+(x2=0)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x9=1)+(x1=1)*(x9=0) => R=1
265] (x1=0)*(x4=0)+(x8=1)*(x12=1)+(x3=0)*(x7=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
266] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
267] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x4=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
268] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x9=1)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
269] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
270] (x1=0)*(x4=0)+(x8=1)*(x12=1)+(x5=0)*(x7=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
271] (x8=1)*(x12=1)+(x5=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
272] (x8=1)*(x12=1)+(x5=0)*(x7=0)+(x4=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
273] (x8=1)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x9=1)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
274] (x8=1)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
275] (x1=0)*(x4=0)+(x8=1)*(x12=1)+(x6=1)*(x7=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
276] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
277] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x4=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
278] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x9=1)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
279] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
280] (x1=0)*(x4=0)+(x8=1)*(x12=1)+(x7=0)*(x10=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
281] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
282] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x4=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
283] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x9=1)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
284] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x12=0)+(x1=1)*(x8=0) => R=1
285] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x8=0) => R=1
286] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x8=0) => R=1
287] (x3=0)*(x4=0)+(x8=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x8=0) => R=1
288] (x8=1)*(x12=1)+(x4=0)*(x10=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x8=0) => R=1
289] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x8=0) => R=1
290] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x8=0) => R=1
291] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x4=0)+(x1=1)*(x8=0) => R=1
292] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x8=0) => R=1
293] (x8=1)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x12=0)+(x2=1)*(x8=1)+(x1=1)*(x8=0) => R=1
294] (x1=0)*(x4=0)+(x8=1)*(x12=1)+(x6=1)*(x7=0)+(x9=0)*(x12=0)+(x1=1)*(x8=0) => R=1
295] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x8=0)*(x9=1)+(x2=1)*(x12=0)+(x1=1)*(x9=0) => R=1
296] (x8=1)*(x12=1)+(x5=0)*(x7=0)+(x8=0)*(x9=1)+(x2=1)*(x12=0)+(x1=1)*(x9=0) => R=1
297] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x8=0)*(x9=1)+(x2=1)*(x12=0)+(x1=1)*(x9=0) => R=1
298] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x9=1)+(x2=1)*(x12=0)+(x1=1)*(x9=0) => R=1
299] (x3=0)*(x6=0)+(x8=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x9=1)+(x1=1)*(x9=0) => R=1
300] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x9=0) => R=1
301] (x8=1)*(x12=1)+(x5=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x9=0) => R=1
302] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x9=0) => R=1
303] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x9=0) => R=1
304] (x8=1)*(x12=1)+(x3=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x2=1) => R=1
305] (x8=1)*(x12=1)+(x5=0)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x2=1) => R=1
```

```
306] (x8=1)*(x12=1)+(x6=1)*(x7=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x2=1) => R=1
307] (x8=1)*(x12=1)+(x7=0)*(x10=0)+(x2=0)*(x8=0)+(x2=1)*(x12=0)+(x1=1)*(x2=1) => R=1
308] (x3=0)*(x6=0)+(x9=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x9=1)+(x1=1)*(x9=0) => R=1
309] (x3=0)*(x6=0)+(x9=1)*(x12=1)+(x7=0)*(x10=0)+(x8=0)*(x12=0)+(x1=1)*(x9=0) => R=1
```