УДК 519.714.71

ПОСТРОЕНИЕ РЕЛЕЙНО КОНТАКТНЫХ СХЕМ НА ОСНОВЕ МИНИМИЗАЦИИ БУЛЕВЫХ ФУНКЦИЙ

Тимофеев В.Д.
Научный руководитель – Юринок В.И., к. т.н., доцент

В докладе описаны алгоритмы минимизации булевых функций (БФ) методом Квайна и построение релейно контактной схемы. Алгоритм позволяет минимизировать данную функцию и построить ее изначальный вид, сокращенные версии Совершенной Дизъюнктивной Нормальной Формы (СДНФ), Совершенной Конъюктивной Нормальной Формы (СКНФ) а так же Минимизированную Дизъюнктивную Нормальную Форму (МДНФ), Минимизированную Конъюктивную Нормальную Форму (МКНФ).

Метод Квайна по минимизации БФ основывается на двух операциях – сжатие и поглощение. Этап сжатия основан на следующих двух формулах, для СДНФ (1) и СКНФ (2) соответственно :

$$w \cdot x \vee w \cdot \overline{x} = w \cdot (x \vee \overline{x}) = w \quad (1)$$

$$w \vee x \wedge w \vee \overline{x} = w \vee (x \wedge \overline{x}) = w \quad (2)$$

После получения формулы в максимально сокращенном виде наступает этап поглощения – получение МДНФ и МКНФ. Этот этап подразумевает построение импликантной таблицы, составление ядра функции, что показано на Рис. 1.

Простая импликанта	$\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4}$	$\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4$	$\overline{x_1} \cdot \overline{x_2} \cdot x_3 \cdot \overline{x_4}$	$\overline{x_1} \cdot x_2 \cdot x_3 \cdot \overline{x_4}$	$x_1 \cdot x_2 \cdot x_3 \cdot \overline{x_4}$	$x_1 \cdot x_2 \cdot x_3 \cdot x_4$
$\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3}$	×	×				
$\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_4}$	×		×			
$\overline{x_1} \cdot x_3 \cdot \overline{x_4}$			×	×		
$x_2 \cdot x_3 \cdot \overline{x_4}$				×	×	
$x_1 \cdot x_2 \cdot x_3$					×	×

Рис.1. Импликантная таблица

Из приведенной выше таблицы следует, что в ядро входят импликанты первой и последней строки. Изначальные импликанты, не поглощенные ядром, находятся в 3 и 4 столбцах. Они могут быть поглощены одним из двух вариантов: строками 2, 4 или строкой 3. Разработанный алгоритм подразумевает нахождение самого выгодного варианта, т. е. минимального набора импликант в финальном виде, а значит, будет выбран вариант поглощения при помощи импликанты в строке 3.

Построение релейно контактной схемы исходя из логического выражения, содержащего операции логического сложения и умножения, сводится следующему. Логическое умножение и сложение на схеме выглядит следующим образом (Рис. 2.):

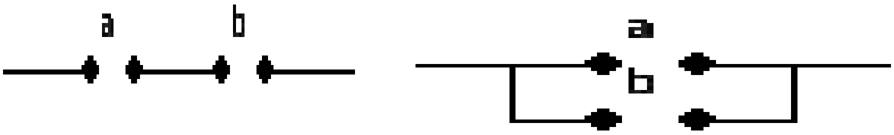

Рис. 2. Логическое умножение (последовательное соединение) и сложение (параллельное соединение)

Алгоритм отображения реализован на языке программирования Python с использованием библиотеки Pygame. Для работы с выражениями имеющими в своем составе такие же выражения была разработана главная рекурсивная функция. И пока функцией не было получено базовое выражение, такое, что не поддается разбиению на другие выражения, функция будет разбивать полученное (примеры базовых выражений : (a ^ b), (a∨b)). Алгоритм минимизации реализован на той же технологии. Сперва функция, принимающая СДНФ, разбивает формулу на импликанты, так формула ((a ^ b) ∨(c ^ d)), будет разбита на следующие части– (a^b), (c^d). После следует склейка, все элементы перебираются попарно и склеиваются по вышеописанным правилам. После чего в случае если полученная формула ничем не отличается от изначальной – функция выдает результат. Иначе алгоритм будет повторятся рекурсивно.

Возьмем следующую функцию: (f ^ ((((((a ^ b)∨f) ^ b) ^ (b∨a))∨f)^(a∨(b^f)))). Она имеет в своем составе 4 разные переменные. Ее длина 11 элементов . Применим к ней вышеописанные алгоритмы (Рис. 3.).

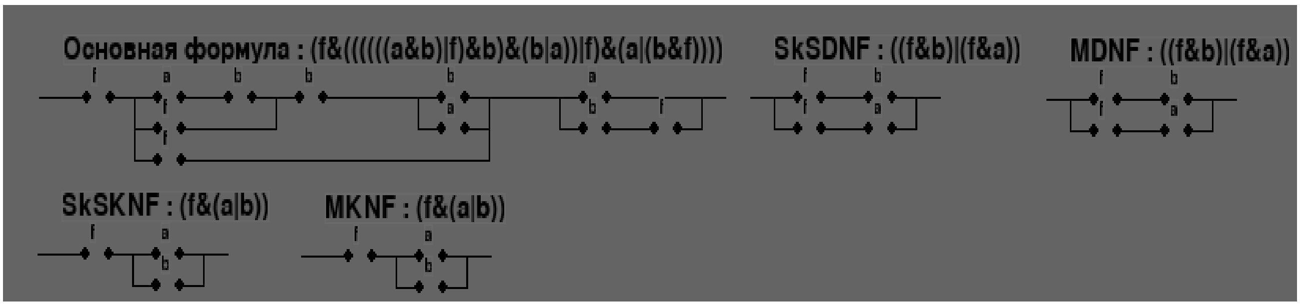

Рис. 3. Результат упрощения формулы (f ^ ((((((a ^ b)∨f) ^ b) ^ (b∨a))∨f)^(a∨(b^f))))

Как видно, формулу длиной в 11 элементов сократилась сперва до 4х элементов, после составления СДНФ и МДНФ, а после и до 3х после составления СКНФ и МКНФ. Значительное сокращение обусловлено положением переменной f. Если она имеет значение "Ложь" – не будет выполнено все выражение. Убедимся в этом путем удаления элемента f Рис. 4.).

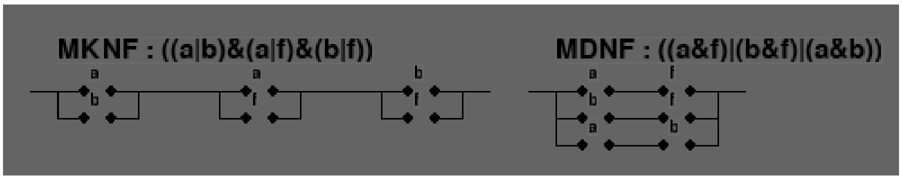

Рис. 4. Результат упрощения формулы $(f \wedge ((((((a \wedge b)\vee f) \wedge b) \wedge (b\vee a))\vee f)\wedge(a\vee(b\wedge f))))$

Как видно из примера выше, в итоге формула сократилась до 6 элементов.

Подводя итог, можно сказать, что разработанный алгоритм подходит для работы с выражениями, имеющими в своем составе до 7 различных элементов. При превышении данного ограничения результат выполнения программы требует значительного времени. Алгоритм отображения в свою очередь является универсальным для любого количества переменных и их вариативности.

Литература

1. Савельев А.Я. Прикладная теория цифровых автоматов. – М.: Высшая школа, 1987.
2. Новиков Ф. А. Дискретная математика для программистов. – СПб.: Питер, 2001.

УДК 519.10

ТЕРНАРНЫЕ ОПЕРАЦИИ В ЗАДАЧАХ ПОИСКА КРИТИЧЕСКИХ ПУТЕЙ В ГРАФЕ

Халецкий Е.С.
Научный руководитель- Корзников А.Д., к.ф.-м.н., доцент

В задачах теории расписаний, сетевого планирования, возникает проблема поиска путей между вершинами графа, длина которых максимальна. Известными методами решения задач такого типа, в основном являются графическими, что значительно затрудняет их решение задачи при большом количестве вершин и дуг[1,2].