**Основные компоненты**

**1. Правило rule(X, Y)**

Это основа для описания преобразований выражений. Оно принимает два аргумента:

* X — исходное выражение.
* Y — результат после упрощения.

**Примеры использования:**

Если в выражении X обнаруживается структура, которая соответствует заданным в коде шаблонам, то это выражение заменяется на Y.

**2. Функция expression(X, A)**

Эта функция применяет правило rule(X, A) для поиска упрощённой формы выражения X.

**Логика:**

Она сопоставляет входное выражение с известными шаблонами в правилах (rule/2) и находит первый подходящий результат.

**3. Функция simplify(X, Y)**

Упрощает выражение рекурсивно, пока выражение не перестанет изменяться.

**Логика:**

1. Сначала expression(X, Z) находит упрощённую форму выражения X.
2. Если Z отличается от X, вызывается simplify(Z, Y).
3. Если изменений больше нет, результат записывается в Y.

**Подробный разбор правил**

**1. Упрощение до чисел**

**Правило rule(X, 0):**

Описывает ситуации, когда выражение превращается в 0. Например:

* X = 0 \* \_ — любое выражение, умноженное на 0, равно 0.
* X = (-A) \* A или X = A \* (-A) — произведение числа на его противоположное равно 0.

**Правило rule(X, 1):**

Описывает ситуации, когда выражение превращается в 1. Например:

* X = 1 + \_ или X = \_ + 1 — добавление единицы.
* X = (-A) + A — сумма числа и его противоположного.

**2. Упрощение до переменной A**

Правило rule(X, A): описывает преобразования, в которых выражение сводится к переменной A. Например:

* X = 1 \* A или X = A \* 1 — перемножение на 1.
* X = 0 + A или X = A + 0 — добавление 0.
* X = -(-A) — двойное отрицание.
* X = A \* A или X = A + A — квадраты и суммы переменных.

Для более сложных выражений:

* X = A \* B + A \* (-B) — если выражение имеет вид суммы произведений с противоположными знаками, оно упрощается до A.

**3. Законы Де Моргана**

Используются законы логики для преобразования выражений:

* rule(-(A \* B), (-A) + (-B)) — отрицание произведения равно сумме отрицаний.
* rule(-(A + B), (-A) \* (-B)) — отрицание суммы равно произведению отрицаний.

**4. Дистрибутивность**

Эти правила используют распределительное свойство умножения:

* rule(A \* (B + C), A \* B + A \* C) — раскрытие скобок.
* rule((A + B) \* (C + D), A \* C + A \* D + B \* C + B \* D) — раскрытие двойных скобок.
* rule((A + B) \* C, A \* C + B \* C) — раскрытие скобок для одного множителя.

**5. Свёртка выражений**

Оптимизирует выражения путём удаления лишних компонентов:

* rule(A + (-A) \* B, A + B) — убирает произведение, если есть противоположные компоненты.
* rule((-A) + A \* B, (-A) + B) — аналогично.

**6. Расширение выражений**

Это более сложное правило:

* rule(A \* B + (-A) \* C + B \* C, A \* B + (-A) \* C) — если в выражении есть сумма произведений с общими элементами, оно упрощается.