Санкт-Петербургский государственный университет

Кафедра системного программирования
Программная инженерия

Кузиванов Сергей Юрьевич

Реализация алгоритма построения представления группы по машине Тьюринга

Отчёт по производственной практике

Научный руководитель:
доцент кафедры информатики, к. ф.-м. н. С. В. Григорьев

Санкт-Петербург
2021

Оглавление

Введение

В наши дни такие отрасли системного программирования как разработка трансляторов, статических анализаторов, парсеров, графовых баз данных не обходится без теории формальных языков.

Данная теория изначально была призвана обеспечить математическое обоснование естественных языков. Сегодня результаты в этой области граничат с такими областями как теория автоматов и алгебраическая теория групп. За время развития теории формальных языков появилось множество различных конструкций, служащих для описания языков, а также возникло множество пока нерешённых проблем, связанных с этими конструкциями. В частности, такие классы формальных языков как конъюнктивные и булевы, пока слабо изучены. Вследствие чего при работе с данными классами языков возникают различного рода трудности. Например, проблема ограничения выразительной силы данных грамматик: в то время, как выразительная сила контекстно-свободных грамматик изучена к настоящему времени довольно хорошо, выразительная сила конъюнктивных грамматик, расширения контекстно-свободных, изучена крайне слабо.

В связи с этим одно из направлений исследований в этой области является применение результатов, связанных с другими областями знаний, в развитии теории формальных языков. Так, например, одно из направлений изучения формальных языков является преобразование языка, записанного в виде грамматики, в представление группы, после чего полученное представление изучается посредством методов алгебраической теории групп.

Данный способ изучения свойств грамматик имеет преимущество по сравнению с другими способами в том плане, что уже существуют такие пакеты вычислительной алгебры, как Maple и GAP, с помощью которых можно изучать свойства полученного представления группы и утверждать о свойствах исходной грамматики. Стоит отметить, что речь идёт о представлении группы, изоморфной исходной грамматике. В то время как по представлению группы легко строится изоморфная

ей грамматика, обратное преобразование является достаточно сложной задачей.

Следовательно, задача алгоритмизации построения представления группы по конъюнктивной или булевой грамматике – задача нетривиальная и нужная для возможности изучения свойств пока слабо изученных видов грамматик.

В прошлом году в рамках дипломной работы [1] выпускником кафедры системного программирования СПбГУ Максимом Борисовичем Шамраем был реализован инструмент, позволяющий по контекстно-свободной грамматике строить представление изоморфной ей группы, а также приводящий полученной представление в вид, пригодный для дальнейшей работы с ним в таких пакетах компьютерной алгебры как Maple и GAP. За основу работы данного инструмента была взята статья [2].

Наша команда занимается расширением возможностей и улучшением данного инструмента, а именно перед нами поставлены 2 основные задачи:

1. Добавить поддержку работы с конъюнктивными и булевыми грамматиками

2. Минимизировать представление получающейся группы

Необходимо отметить, что размер представления, получаемого с помощью данного инструмента, на текущий момент не является удовлетворительным, поскольку система GAP, с помощью которой планировалось изучать свойства контекстно-свободных грамматик, не может обработать получающееся представление группы. Однако преобразование, реализованное в текущем инструменте, сохраняет многие свойства изначальной грамматики. Поэтому передо мной была поставлена задача минимизации получающегося представления группы возможно за счёт несохранения многих свойств исходной грамматики.

Постановка задачи

Целью данной курсовой является расширение и улучшение возможностей существующего инструмента для изучения свойств грамматик через рассмотрение представления изоморфной группы.

Для достижения данной цели были поставлены следующие задачи:

1. Реализовать описанный в статье [3] алгоритм преобразования машины Тьюринга в представление группы
2. Внедрить реализованный алгоритм в инструмент
3. Поставить эксперименты, нацеленные на выяснение свойств заданных конъюнктивных и булевых грамматик, используя данный инструмент и систему GAP

1. Обзор

В данном разделе будут представлены основные определения, используемые в данной работе, рассмотрен алгоритм получения представления группы по машине Тьюринга, описанный в статье [3], а также представлена архитектура существующего инструмента, на улучшение которого и была направлена данная работа.

1.1. Формальная грамматика

Определение 1.1.1 *Алфавит — это конечное непустое множество, элементы которого называются символами. Обычно обозначается как* Σ.

Определение 1.1.2 *Слово — это конечный упорядоченный кортеж (так называемый список), элементы которого являются символами из произволного алфавита* Σ. *Обычно обозначается как* w. *Часто про такое слово говорят "слово над алфавитом* Σ".

Определение 1.1.3 *Бинарная операция* $(:)$ *— это операции конкатенации строки и символа, при этом запись "<строка> : <символ>" означает прибавление символа в конец строки, а "<символ> : <строка>" — в начало.*

Определение 1.1.4 *Язык — это множество (не обязательно конечное) слов над произвольным алфавитом.*

Определение 1.1.5 *Формальная грамматика — это четвёрка* $\langle N, \Sigma, P, S \rangle$, *где:*

- N *— алфавит, символы которого называются нетерминалами*
- Σ *— алфавит, символы которого называются терминалами, при этом* $N \cap \Sigma = \emptyset$
- P *— конечное подмножество множества*

$$(N \cup \Sigma)^* \times N \times (N \cup \Sigma)^* \times (N \cup \Sigma)^*$$

Каждый элемент множества P называется правилом и записывается как

$$u\,A\,v \to w$$

где $u, v, w \in (N \cup \Sigma)^*$, а $A \in N$

- $S \in N$ *— нетерминал, называемый стартовым*

В определении 1.1.5 в случае, когда w является пустым, пишут символ ε. Также следует заметить, что правила обычно для удобства пронумеровывают натуральными числами.

Определение 1.1.6 *Будем говорить, что слово w' над алфавитом $N \cup \Sigma$* ***получено*** *из слова w над тем же алфавитом в грамматике $G = \langle N, \Sigma, P, S\rangle$ путём применения правила $p_i \in P$ если*

1. $p_i = (u_i\,A_i\,v_i \to w_i)$
2. $w = u\,u_i\,A_i\,v_i\,v$
3. $w' = u\,w_i\,v$

Данный факт будем записывать как $w \overset{p_i}{\Rightarrow}_G w'$ или $w \overset{i}{\Rightarrow}_G w'$.

Определение 1.1.7 *Будем говорить, что слово w' над алфавитом $N \cup \Sigma$* ***выводимо*** *из слова w над тем же алфавитом в грамматике $G = \langle N, \Sigma, P, S\rangle$ если $\exists p_1, p_2, \ldots, p_n \in P \forall i \in 1 \ldots n : w_{i-1} \overset{p_i}{\Rightarrow} w_i$, где $w = w_0$, а $w' = w_n$.*

Данный факт будем записывать как $w \overset{*}{\Rightarrow}_G w'$.

Определение 1.1.8 *Будем говорить, что грамматика $G = \langle N, \Sigma, P, S\rangle$* ***порождает*** *язык $L(G)$ если*

$$\forall w \in L(G) : S \overset{*}{\Rightarrow}_G w\ u$$

$$\forall w \notin L(G) : S \overset{*}{\not\Rightarrow}_G w$$

Дополнительно с данными определениями можно ознакомиться в книге [4].

1.2. Конъюнктивная и булева грамматики

Определение 1.2.1 *Контекстно-свободная грамматика — это грамматика, для которой слова u и v во всех правилах являются пустыми (то есть левые части всех правил состоят из единственного нетерминала).*

Обычно если в контекстно-свободной грамматике несколько правил имеют один и тот же нетерминал A в левой части, то правые части этих правил записывают в одном правиле через символ "|". Например, набор правил

$$S \to \varepsilon$$

$$S \to a$$

$$S \to b\,a$$

записывают как правило

$$S \to \varepsilon \mid a \mid b\,a$$

Таким образом можно сказать, что

$$w \xRightarrow{A \to r_1 \mid r_2}_G w' \Leftrightarrow w \xRightarrow{A \to r_1}_G w' \vee w \xRightarrow{A \to r_2}_G w'$$

Определение 1.2.2 *Конъюнктивная грамматика — это ресширение контекстно-свободной грамматики специальным знаком "&", обладающим следующей семантикой:*

$$w \xRightarrow{A \to r_1 \ \& \ r_2}_G w' \Leftrightarrow w \xRightarrow{A \to r_1}_G w' \wedge w \xRightarrow{A \to r_2}_G w'$$

Совершенно аналогично определяется семантика записи с произвольным числом r_i.

Определение 1.2.3 *Булева грамматика — это ресширение конъюнктивной грамматики специальным знаком "¬", обладающим следующей семантикой:*

$$w \xRightarrow{A \to \neg r}_G w' \Leftrightarrow w \not\xRightarrow{A \to r}_G w'$$

Более подробно с конъюнктивной и булевой грамматиками можно ознакомиться в статьях А. Охотина [5, 6, 7].

1.3. Машина Тьюринга

Классическое определение машины Тьюринга выглядит следующим образом:

Определение 1.3.1 *Машина Тьюринга — это семёрка* $\langle Q, \Sigma, \Gamma, \delta, q_0, B, F \rangle$, *компоненты которой имеют следующий смысл:*

- Q — *конечное множество состояний*
- Σ — *алфавит входных символов*
- $\Gamma \supseteq \Sigma$ — *алфавит ленточных символов*
- $\delta : Q \times \Gamma \to \{\emptyset\} \cup (Q \times \Gamma \times \{L, R\})$ — *функция переходов*
- $q_0 \in Q$ — *начальное состояние*
- $B \in \Gamma$ — *пустой символ*
- $F \subseteq Q$ — *множество допускающих состояний*

Для полноты определения также введём мгновенное описание и шаг машины Тьюринга:

Определение 1.3.2 *Мгновенное описание машины Тьюринга — это четвёрка* $\langle q, T_L, \gamma, T_R \rangle$, *компоненты которой имеют следующий смысл:*

- $q \in Q$ — *текущее состояние*
- $\gamma \in \Gamma$ — *текущий символ*
- T_L *и* T_R — *упорядоченные списки из элементов множества* Γ, *называемые левой и правой частью ленты соответственно*

Определение 1.3.3 *Шаг машины Тьюринга — это преобразование мгновенного описания машины Тьюринга* $\langle q, T_L, \gamma, T_R\rangle$ *в другое мгновенное описание* $\langle p, U_L, \zeta, U_R\rangle$*, где* $(p, \eta, D) = \delta(q, \gamma)$*, при этом:*

- *если* $D = L$ *и* $T_L = V : \zeta$*, то* $U_L = V$ *и* $U_R = \eta : T_R$
- *если* $D = L$ *и* $T_L = \varepsilon$ *(пустой кортеж), то* $\zeta = B$*,* $U_L = \varepsilon$ *и* $U_R = \eta : T_R$
- *если* $D = R$ *и* $T_R = \zeta : V$*, то* $U_L = T_L : \eta$ *и* $U_R = V$
- *если* $D = R$ *и* $T_R = \varepsilon$*, то* $\zeta = B$*,* $U_L = T_L : \eta$ *и* $U_R = \varepsilon$

Заметим, что шаг машины Тьюринга определён только если $\delta(q, \gamma) \neq \emptyset$, иначе машина Тьюринга останавливается.

Введём определения допускания машиной Тьюринга слова из символов алфавита Σ и порождения языка машиной Тьюринга:

Определение 1.3.4 *Пусть даны машина Тьюринга* T*, слово* $w = w_{head} : w_{tail}$ *из входных символов алфавита* Σ *машины* T*, мгновенное описание* $\langle q_0, \varepsilon, w_{head}, w_{tail}\rangle$*, которое мы назовём стартовым, и мгновенное описание* $\langle p, T'_L, \eta, T'_R\rangle$*, получаемое из стартового путём применения шагов машины Тьюринга* T *до остановки (то есть* $\delta(p, \eta) = \emptyset$*). Если* $p \in F$*, то мы говорим, что слово* w ***допускается*** *машиной Тьюринга* T*, иначе не допускается.*

Определение 1.3.5 *Машина Тьюринга* T ***задаёт*** *язык* $L(T)$*, все слова которого и только они допускаются машиной* T*. Иными словами,* $\forall w \in L(T) : w$ *допускается* T *и* $\forall w \notin L(T) : w$ *не допускается* T*.*

Однако следует отметить, что задача определения того, допустит ли машина Тьюринга T слово w является в общем случае неразрешимой. Иными словами, не существует единого способа определить для произвольных машины Тьюринга и слова, остановится ли машина Тьюринга на данном слове или нет.

На практике обычно применяются различные модификации машины Тьюринга, такие как:

- Недетерминированная машина Тьюринга (δ возвращает не одну тройку, а множество троек)
- N-ленточная машина Тьюринга (δ принимает кортеж из N двоек и возвращает N троек)
- Машина Тьюринга с двойной кареткой (δ принимает кортеж из множества $Q \times \Gamma \times \Gamma$)
- ...

Данное разнообразие модификаций позволяет использовать ту нотацию машины Тьюринга, которая наиболее хорошо подходит для применения в конкретной задаче. Также доказано, что эти и многие другие модификации машины Тьюринга равносильны, то есть для любой машины Тьюринга в любой нотации существует такая машина Тьюринга в любой другой нотации, которая задаёт тот же язык, что и первая. Преобразование из одной нотации машин Тьюринга в другую обычно достаточно тривиально. Однако в контексте задачи построения функции преобразования машины Тьюринга из одной нотации в другую с минимизацией количества продукций (ненулевых троек в функции δ) построение бывает достаточно сложным.

Дополнительные сведения о машинах Тьюринга можно прочесть в книге [4].

1.4. Представление группы

Определение 1.4.1 *Алгебраическая структура — это пара $\langle A, \cdot \rangle$, компоненты которой имеют следующий смысл:*

- *A — произвольное множество*
- *$\cdot$ — операция вида $A \times A \to A$*

Определение 1.4.2 *Полугруппа — это такая алгебраическая структура, что*

$$\forall a, b, c \in A : (a \cdot b) \cdot c = a \cdot (b \cdot c)$$

Определение 1.4.3 *Моноид — это такая полугруппа, что*

$$\exists e \in A : \forall a \in A : a \cdot e = e \cdot a = a$$

Определение 1.4.4 *Группа — это такой моноид, что*

$$\forall a \in A : \exists b \in A : a \cdot b = b \cdot a = e$$

при этом вводится унарный оператор $^{-1}$ *такой, что* $a \cdot a^{-1} = a^{-1} \cdot a = e \forall a \in A$.

Определение 1.4.5 *Конечное представление группы — это пара* $\langle S, R \rangle$, *компоненты которой имеют следующий смысл:*

- S *— конечное подмножество* A, *называемое порождающим множеством*
- R *— конечное множество пар* $\langle w, v \rangle$ *слов над алфавитом* A, *называемых соотношениями, таких, что*

  $$w_1 \cdot w_2 \cdot \ldots \cdot w_n = v_1 \cdot v_2 \cdot \ldots \cdot v_m$$

  где $w = \langle w_1, w_2, \ldots, w_n \rangle$, $v = \langle v_1, v_2, \ldots, v_m \rangle$

при этом $\forall a \in A : \exists w \in A^* : a = w_1 \cdot w_2 \cdot \ldots \cdot w_n$.

Далее для краткости мы будем называть конечное представление группы просто *представлением группы.*

Аналогичным образом вводится представление полугруппы.

1.5. Преобразование машины Тьюринга в представление полугруппы

Данный раздел является описанием алгоритма, предложенного в статье [3].

Как уже было сказано ранее, для удобства изложения многие авторы статей используют свою нотацию машины Тьюринга. И машина Тьюринга, описанная в статье [3], имеет следующую нотацию:

Определение 1.5.1 *Машина Тьюринга – это шестёрка* $\langle Q, S, \delta, q_1, q_0, B\rangle$, *компоненты которой имеют следующий смысл:*

- Q – *конечное множество состояний*
- S – *алфавит ленточных символов*
- $\delta : (Q \setminus \{q_0\}) \times S \to Q \times (\{L, R\} \cup S)$ – *функция переходов*
- q_1 *и* $q_0 \in Q$ – *начальное и конечное состояния соответственно*
- $B \in S$ – *пустой символ*

Остановка происходит при достижении состояния q_0, при этом все слова допускаются.

Для построения представления полугруппы по данной машине Тьюринга необходимо сначала занумеровать все символы и состояния этой машины, начиная с 0, при этом полагаем, что $s_0 = B$. Также для удобства обозначим $N = |Q| - 1$, а $M = |S| - 1$. То есть $\{q_0, q_1, q_2, \ldots, q_N\} = Q$, а $\{s_0, s_1, s_2, \ldots, s_M\} = S$.

Теперь представление полугруппы по машине Тьюринга T строится следующим образом:

$\Gamma(T) = \langle \{q, h, s_0, s_1, \ldots, s_M, q_0, q_1, \ldots, q_N\}, R\rangle$, где R состоит из

$$
\begin{aligned}
q_i \cdot s_j &= q_l \cdot s_k && \delta(q_i, s_j) = (q_l, s_k)\\
q_i \cdot s_j \cdot s_\beta &= s_j \cdot q_l \cdot s_\beta &&\\
q_i \cdot s_j \cdot h &= s_j \cdot q_l \cdot s_0 \cdot h && \delta(q_i, s_j) = (q_l, R)\\
s_\beta \cdot q_i \cdot s_j &= q_l \cdot s_\beta \cdot s_j &&\\
h \cdot q_i \cdot s_j &= h \cdot q_l \cdot s_0 \cdot s_j && \delta(q_i, s_j) = (q_l, L)\\
q_0 \cdot s_\beta &= q_0 &&\\
s_\beta \cdot q_0 \cdot h &= q_0 \cdot h &&\\
h \cdot q_0 \cdot h &= q &&
\end{aligned}
$$

$\forall \beta \in 0 \ldots M$

Для представления полугруппы $\Gamma(T)$ выполняется следующее утверждение:

Утверждение 1.5.1

$$w \in L(T) \Leftrightarrow h \cdot q_1 \cdot w \cdot h =_{\Gamma(T)} q$$

Доказательство этого утверждения можно найти в статье [3].

Для удобства последующих рассуждений заменим символ h на s_{M+1}, то есть добавим новый символ в множество S исходной машины Тьюринга.

1.6. Преобразование представления полугруппы в представление группы

Данный раздел также является описанием алгоритма, предложенного в статье [3].

Для начала необходимо занумеровать соотношения исходного представления полугруппы следующим образом:

$$F_i q_i G_i = H_i p_i K_i - i\text{-ое соотношение из } R \ (i \in 1 \ldots K, \text{ где } K = |R|)$$

где F_i, G_i, H_i и K_i – слово над S

Введём унарную операцию $^{\#}$ следующим образом:

$$\forall X = x_1 \cdot x_2 \cdot \ldots \cdot x_n : X^{\#} = x_1^{-1} \cdot x_2^{-1} \cdot \ldots \cdot x_n^{-1}$$

Теперь построим представление группы по машине Тьюринга T следующим образом:

$B(T) = \langle \{q, q_0, \ldots, q_N, s_0, \ldots, s_{M+1}, r_1, \ldots, r_K, x, t, k\}, R \rangle$, где R –

$$
\begin{aligned}
x \cdot s_\beta &= s_\beta \cdot x \cdot x \\
r_i \cdot s_\beta &= s_\beta \cdot x \cdot r_i \cdot x \\
r_i^{-1} \cdot F_i^{\#} \cdot q_i \cdot G_i \cdot r_i &= H_i^{\#} \cdot p_i \cdot K_i \\
t \cdot r_i &= r_i \cdot t \\
t \cdot x &= x \cdot t \\
k \cdot r_i &= r_i \cdot k \\
k \cdot x &= x \cdot k \\
k \cdot q^{-1} \cdot t \cdot q &= q^{-1} \cdot t \cdot q \cdot k
\end{aligned}
$$

$\forall \beta \in 0 \ldots M+1, i \in 1 \ldots K$

Определение 1.6.1 *Слово ω называется **специальным** если оно представимо в виде $\omega = X \cdot q_i \cdot Y$, где X и Y – слова над S.*

Для специального слова $\omega = X \cdot q_i \cdot Y$ введём унарную операцию * как

$$\omega^* = X^{\#} \cdot q_i \cdot Y$$

Также, как и для представления полугруппы $\Gamma(T)$, для представления группы $B(T)$ выполняется следующее утверждение:

Утверждение 1.6.1

$$\omega^* =_{\Gamma(T)} q \Leftrightarrow k \cdot \omega^{-1} \cdot t \cdot \omega =_{B(T)} \omega^{-1} \cdot t \cdot \omega \cdot k$$

Доказательство данного факта также можно найти в статье [3].

В итоге получаем следующее утверждение, которое является обединением утверждений 1.5.1 и 1.6.1 (напомню, что $h = s_{M+1}$):

Утверждение 1.6.2

$$w \in L(T) \Leftrightarrow k\, h^{-1}\, w^{-1}\, q_1^{-1}\, h\, t\, h^{-1}\, q_1\, w\, h\, k^{-1}\, h^{-1}\, w^{-1}\, q_1^{-1}\, h\, t^{-1}\, h^{-1}\, q_1\, w\, h =_{B(T)} 1$$

1.7. Существующий инструмент

В прошлом году в рамках дипломной работы студентом СПбГУ Максимом Борисовичем Шамраем был разработан инструмент на языке программирования Haskell для преобразования контекстно-свободной грамматики в представление группы в виде, понятном таким пакетам компьютерной алгебры, как Maple и GAP. Архитектура разработанного инструмента схематично представлена на рис. 1.7.1.

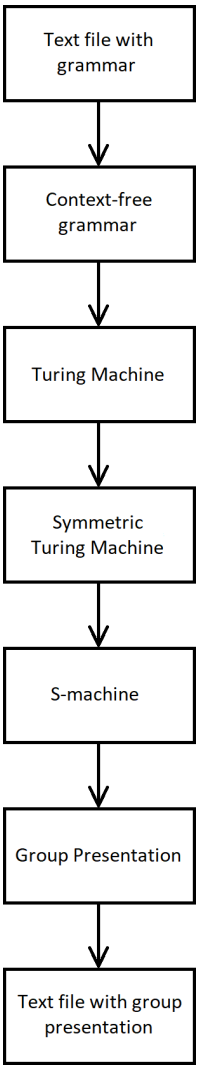

Рис. 1.7.1: Архитектура существующего инструмента

Более подробно с архитектурой реализованного в прошлом году инструмента можно ознакомиться в дипломной работе [1].

2. Описание реализации

В данном разделе будет представлена новая архитектура с включенным новым алгоритмом из статьи [3], а также будет в общих чертах рассмотрен алгоритм преобразования машины Тьюринга из нотации, используемой в изначальном инструменте, в нотацию, используемую в вышеупомянутом алгоритме.

2.1. Архитектура

В рамках данной курсовой работы была изменена архитектура данного инструмента путём добавления вышеупомянутого алгоритма преобразования машины Тьюринга в представление группы. Таким образом, новая архитектура данного инструмента выглядит так, как представлено на рис. 2.1.1. На рисунке красным цветом отмечено то, что было сделано мной в рамках данной курсовой работы.

2.2. Реализация алгоритма

Алгоритм преобразования машины Тьюринга в представление группы был реализован на языке программирования Haskell. Выбор данного языка был обусловлен тем, что исходный инструмент был написан на данном языке.

Основные шаги вышеописанного преобразования описаны в следующих модулях:

- TM2SP – преобразование машины Тьюринга в представление полугруппы (пункт 1.4 обзора)
- SP2GP – преобразование представления полугруппы в представление группы (пункт 1.5 обзора)

Сам алгоритм в виде функции groupBeta согласно описанному в статье представлению группы $B(T)$, построенному по машине Тьюринга T. Данная функция находится в модуле SP2GP.

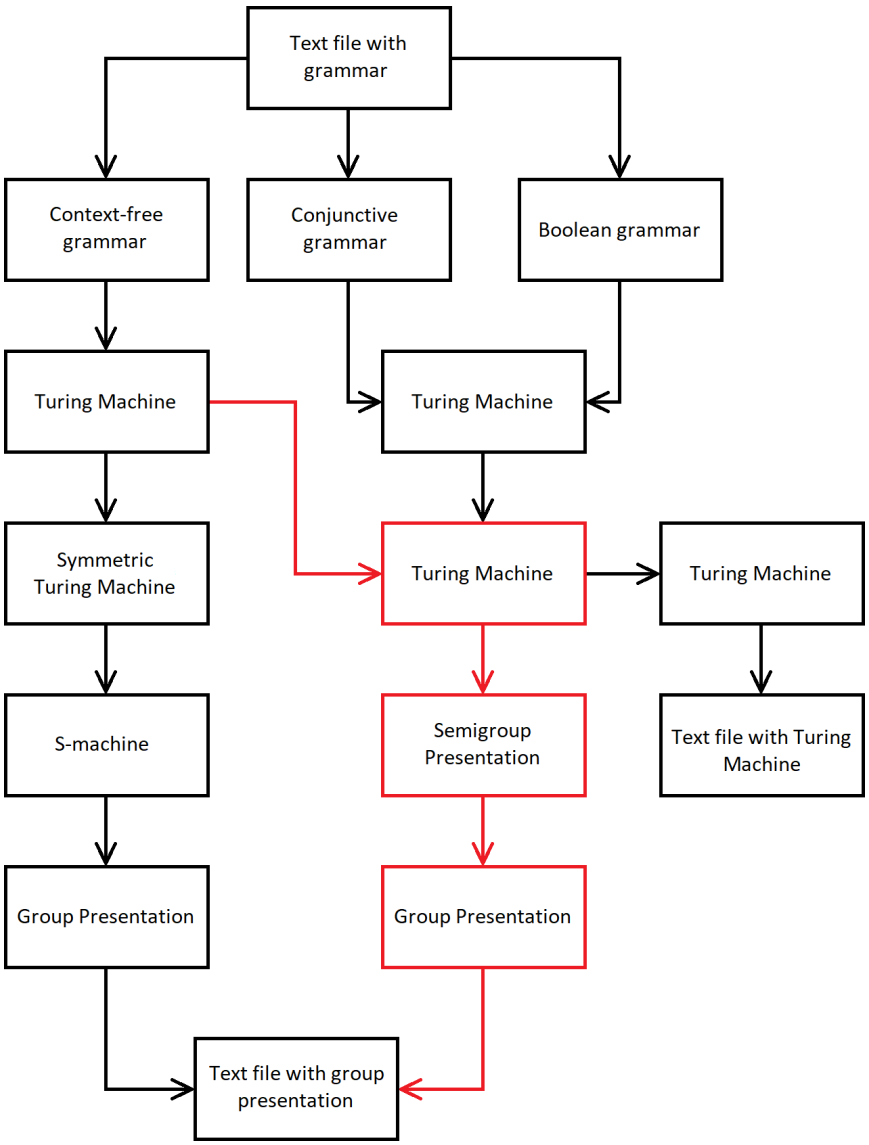

Рис. 2.1.1: Архитектура инструмента с добавлением нового алгоритма

2.3. Преобразование машины Тьюринга

Так как исходное приложение работает с машинами Тьюринга в другой нотации, для применения реализованного алгоритма необходимо сначала преобразовать машину Тьюринга в нужную нотацию.

Основная проблема преобразования состоит в том, что эти две нотации машин Тьюринга достаточно разные. А именно:

1. Первая нотация машины Тьюринга (в изначальном инструменте):

    - Недетерминированность

- Многоленточность (точнее 2 ленты, на которых каретки двигаются независимо)
- На ленту можно добавлять и удалять ячейки, а также менять символы в каждой ячейке
- Каждая каретка смотрит на 2 соседние ячейки
- При каждом шаге машина может и удалить одну из ячеек, на которую смотрит каретка, и добавить новую ячейку, и изменить символ в ячейке (движение осуществляется путём удаления одной ячейки и вставкой другой)
- При остановке машина может как допустить исходное слово, так и не допустить

2. Вторая нотация машины Тьюринга (в статье [3]):

   - Детерминированность
   - Одноленточность
   - На ленте можно только менять символы в ячейках
   - Единственная каретка смотрит только на одну ячейку
   - При каждом шаге машина может либо сдвинуться на одну ячейку вправо/влево, либо заменить символ в текущей ячейке
   - При остановке машины считает, что исходное слово допускается

В связи с этим преобразование машины из первой нотации во вторую – довольно нетривиальная задача.

Основные идеи данного преобразования следующие:

- Преобразование машины – это просто создание машины во второй нотации, полностью эмулирующей работу изначальной машины в первой нотации

- Для простых действий, таких как замена символа, передвижение каретки и т. д., можно достаточно просто построить машины во второй нотации, совершающие данные действия

- Для более сложных действий, в том числе и для полной эмуляции машины в первой нотации, можно построить машины во второй нотации, выполняющие данные действия, путём применения к уже реализованным машинам 4 видов бинарных операций. А именно, если машину во второй нотации схематично представить в виде, описанном на рис. 2.3.1, то данные операции схематично можно представить в виде, описанном на рис. 2.3.2 – 2.3.5.

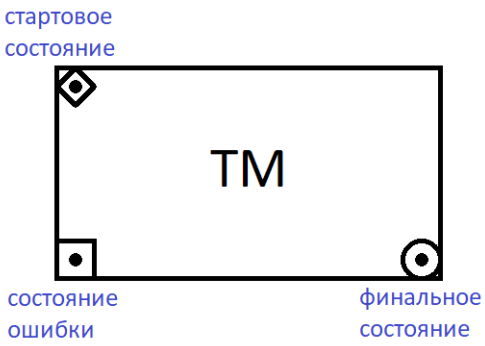

Рис. 2.3.1: Схематичное представление машины Тьюринга второй нотации

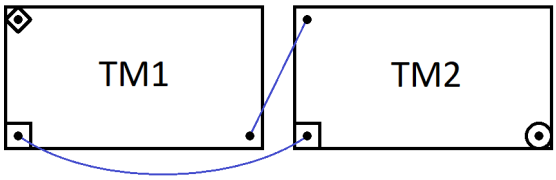

Рис. 2.3.2: Конкатенация

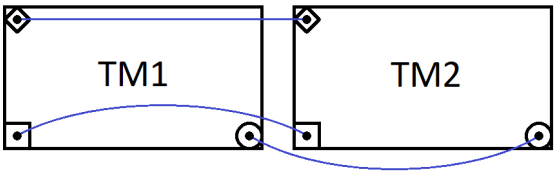

Рис. 2.3.3: Альтернатива

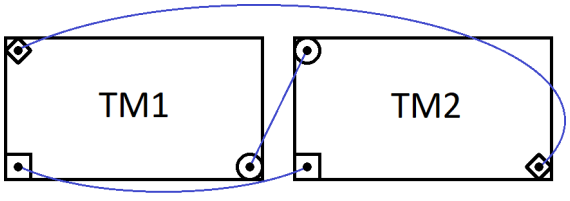

Рис. 2.3.4: Зацикливание

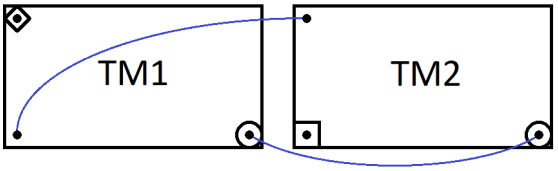

Рис. 2.3.5: Обработка ошибок

В рис. 2.3.2 – 2.3.5 использовались следующие обозначения:

- Левая верхняя точка прямоугольника – начальное состояние машины-операнда
- Правая нижняя точка прямоугольника – конечное состояние машины-операнда
- Левая нижняя точка прямоугольника – состояние ошибки машины-операнда (в машине Тьюринга второй нотации данное состояние не предусмотрено и введено мною для упрощения работы с машинами)
- Ромб (квадрат под углом 45°) – начальное состояние получаемой машины
- Круг – конечное состояние получаемой машины
- Квадрат – состояние ошибки получаемой машины
- Синяя кривая, соединяющая две точки – преобразование двух соответствующих состояний машин-операндов в одно состояние получаемой машины

2.4. Тестовый набор машин Тьюринга

Для проверки реализованного алгоритма было предложено использовать следующий набор грамматик (таблица 2.4.1) и машин Тьюринга, задающий их (таблица 2.4.2):

Суффикс "v.<число>" указывает, что язык, порождённый указанной грамматикой, задаётся несколькими различными машинами Тьюринга.

Необходимо заметить, что в таблице 2.4.2 мощность множества символов грамматики не совпадает с количеством символов в машине Тьюринга. Дело в том, что множество символов грамматики – это множество символов, слово только из которых может подаваться на ленту машины Тьюринга перед началом её работы, а количество символов

Название	Описание
a^*	Все возможные слова над алфавитом $\{a\}$
a^+	Все слова кроме пустого над алфавитом $\{a\}$
$a?$	Пустое слово или один символ над алфавитом $\{a\}$
$a\|b$	Один символ над алфавитом $\{a, b\}$
abc	Единственная строка над алфавитом $\{a, b, c\}$
$ababa$	Единственная строка над алфавитом $\{a, b\}$
$a(bc)^*ba$	Регулярный язык над алфавитом $\{a, b, c\}$
$Dyck$	Язык Дика над алфавитом $\{a, b\}$
$wwR\ 2s$	Язык чётных палиндромов над алфавитом $\{a, b\}$
$wwR\ 4s$	Язык чётных палиндромов над алфавитом $\{a - d\}$
$wwR\ 8s$	Язык чётных палиндромов над алфавитом $\{a - h\}$
$wwR\ 16s$	Язык чётных палиндромов над алфавитом $\{a - p\}$

Таблица 2.4.1: Тестовый набор грамматик

Название	Свойства		
	Количество состояний	Количество символов	Кол. правил перехода
a^*	1	1	2
a^+	2	1	4
$a?$	2	1	4
$a\|b$	2	2	6
abc	4	3	16
$ababa$	6	2	19
$a(bc)^*ba$	4	3	16
$Dyck\ v.1$	8	3	23
$Dyck\ v.2$	10	4	24
$Dyck\ v.3$	9	3	23
$Dyck\ v.4$	11	4	25
$Dyck\ v.5$	50	3	112
$wwR\ 2s\ v.1$	11	2	25
$wwR\ 2s\ v.2$	9	2	21
$wwR\ 4s$	15	4	55
$wwR\ 8s$	27	8	171
$wwR\ 16s$	51	16	595

Таблица 2.4.2: Машины Тьюринга, соответствующие тестовым грамматикам

машины Тьюринга — это количество различных символов, которые мо-

гут быть записаны на ленте в процессе её работы (пустой символ не считается).

2.5. Изменение построения представления полугруппы по машине Тьюринга

Мною было предложено изменение построения представления полугруппы по машине Тьюринга с целью уменьшения количества отношений в итоговом представлении группы.

Стоит напомнить оригинальное построение представления полугруппы по машине Тьюринга, описанное в статье [3] и выглядит следующим образом:

$\langle \{q, h, s_0, s_1, \ldots, s_M, q_0, q_1, \ldots, q_N\}, R \rangle$, где R состоит из

$$
\begin{aligned}
q_i \cdot s_j &= q_l \cdot s_k && \delta(q_i, s_j) = (q_l, s_k) \\
q_i \cdot s_j \cdot s_\beta &= s_j \cdot q_l \cdot s_\beta && && (1) \\
q_i \cdot s_j \cdot h &= s_j \cdot q_l \cdot s_0 \cdot h && \delta(q_i, s_j) = (q_l, R) && (2) \\
s_\beta \cdot q_i \cdot s_j &= q_l \cdot s_\beta \cdot s_j \\
h \cdot q_i \cdot s_j &= h \cdot q_l \cdot s_0 \cdot s_j && \delta(q_i, s_j) = (q_l, L) \\
q_0 \cdot s_\beta &= q_0 \\
s_\beta \cdot q_0 \cdot h &= q_0 \cdot h \\
h \cdot q_0 \cdot h &= q
\end{aligned}
$$

$\forall \beta \in 0 \ldots M$

Во-первых заметим, что в отношении (1) символ s_β не меняет своей позиции, поэтому можно попробовать заменить это равенство на

$$q_i \cdot s_j = s_j \cdot q_l$$

Однако в этом случае в ходе преобразования выражения в полугруппе может возникнуть ситуация, которая была невозможна в оригинальном построении — это появление подвыражения $q_i \cdot h$. Более подробно, если в выражении в полугруппе, которое надо проверить на равенство с

q, есть подвыражение вида $q_i \cdot s_j \cdot h$, то данное подвыражение в исходном преобразовании заменится отношением (2) на подвыражение $s_j \cdot q_i \cdot s_0 \cdot h$. Однако, поскольку изменённое равенство (1) также может быть применено в данном случае, мы можем получить подвыражение $s_j \cdot q_i \cdot h$. Возникновение данного подвыражения было невозможно в исходном преобразовании, поэтому в новое преобразование необходимо включить обработку данной ситуации. Эта ситуация означает, что каретка в данный момент расположена над пустым символом (то есть над символом s_0), но который сейчас может быть записан как символ h.

Поэтому для обработки данного подвыражения построим следующее преобразование машины Тьюринга в представление полугруппы:

$\langle \{q, h, s_0, s_1, \ldots, s_M, q_0, q_1, \ldots, q_N\}, R \rangle$, где R состоит из

$$
\begin{aligned}
q_i \cdot s_j &= q_l \cdot s_k & \delta(q_i, s_j) &= (q_l, s_k) \\
q_i \cdot h &= q_l \cdot s_k \cdot h & \delta(q_i, s_0) &= (q_l, s_k) \\
q_i \cdot s_j &= s_j \cdot q_l & \delta(q_i, s_j) &= (q_l, R) \\
q_i \cdot h &= s_0 \cdot q_l \cdot h & \delta(q_i, s_0) &= (q_l, R) \\
s_\beta \cdot q_i \cdot s_j &= q_l \cdot s_\beta \cdot s_j & \delta(q_i, s_j) &= (q_l, L) \\
s_\beta \cdot q_i \cdot h &= q_l \cdot s_\beta \cdot h & \delta(q_i, s_0) &= (q_l, L) \\
q_0 \cdot s_\beta &= q_0 \\
s_\beta \cdot q_0 \cdot h &= q_0 \cdot h \\
h \cdot q_0 \cdot h &= q
\end{aligned}
\qquad (3)
$$

$\forall \beta \in 0 \ldots M$

Во-вторых заметим, что аналогичное действие изменения отношения (1) можно проделать с отношением (3) если записывать символ q_i не перед тем символом, на который в данный момент смотрит каретка машины Тьюринга, а после. Для этого можно ввести дополнительные генераторы p_i в представление полугруппы, которые были бы аналогичны генераторам q_i, но записывались бы после, а не перед текущим символом.

Эта идея была выражена в построении следующего преобразования:

$\langle \{q, h, s_0, s_1, \ldots, s_M, q_0, q_1, \ldots, q_N, p_0, p_1, \ldots, p_N\}, R \rangle$, где R состоит из

$$
\begin{aligned}
q_i \cdot s_j &= q_l \cdot s_k & \delta(q_i, s_j) &= (q_l, s_k) \\
q_i \cdot h &= q_l \cdot s_k \cdot h & \delta(q_i, s_0) &= (q_l, s_k) \\
q_i \cdot s_j &= s_j \cdot q_l & \delta(q_i, s_j) &= (q_l, R) \\
q_i \cdot h &= s_0 \cdot q_l \cdot h & \delta(q_i, s_0) &= (q_l, R) \\
s_j \cdot p_i &= p_l \cdot s_j & \delta(q_i, s_j) &= (q_l, L) \\
h \cdot p_i &= h \cdot p_l \cdot s_0 & \delta(q_i, s_0) &= (q_l, L) \\
q_\alpha \cdot s_\beta &= s_\beta \cdot p_\alpha \\
q_\alpha \cdot h &= s_0 \cdot p_\alpha \cdot h \\
h \cdot p_\alpha &= h \cdot q_\alpha \cdot s_0 \\
q_0 \cdot s_\beta &= q_0 \\
s_\beta \cdot q_0 \cdot h &= q_0 \cdot h \\
h \cdot q_0 \cdot h &= q
\end{aligned}
$$

$\forall \alpha \in 0 \ldots N, \beta \in 0 \ldots M$

Для полученных 3 преобразований машины Тьюринга в представление полугруппы были проведены замеры размера получаемых представлений полугрупп и представлений групп. Результаты этих измерений представлены в таблицах 2.5.1 и 2.5.2 соответственно.

Машина Тьюринга	Исходное представление полугруппы		Новое представление полугруппы 1		Новое представление полугруппы 2	
	Кол. генераторов	Кол. отношений	Кол. генераторов	Кол. отношений	Кол. генераторов	Кол. отношений
a^*	6	9	5	7	5	7
a^+	7	13	6	8	6	8
$a?$	7	11	6	9	6	9
$a\|b$	8	19	7	10	7	10
abc	11	37	10	13	10	13
$a(bc)^*ba$	11	41	10	14	10	14
$ababa$	12	40	11	13	11	13
$Dyck\ v.1$	15	88	14	55	18	52
$Dyck\ v.2$	18	85	17	43	18	42
$Dyck\ v.3$	16	76	15	39	16	39
$Dyck\ v.4$	19	91	18	45	19	44
$Dyck\ v.5$	57	329	56	137	57	137
$wwR\ 2s\ v.1$	17	80	16	60	24	68
$wwR\ 2s\ v.2$	15	64	14	46	20	54
$wwR\ 4s$	23	216	22	116	32	110
$wwR\ 8s$	39	1000	38	352	56	270
$wwR\ 16s$	71	5832	70	1208	104	782

Таблица 2.5.1: Сравнение получаемых представлений полугрупп

Машина Тьюринга	Исходное представление группы		Новое представление группы 1		Новое представление группы 2	
	Кол. генераторов	Кол. отношений	Кол. генераторов	Кол. отношений	Кол. генераторов	Кол. отношений
a^*	18	60	15	48	15	48
a^+	23	84	17	54	17	54
$a?$	21	72	18	60	18	60
$a\|b$	30	140	20	77	20	77
abc	51	304	26	112	26	112
$a(bc)^*ba$	55	336	27	120	27	120
$ababa$	55	287	27	98	27	98
$Dyck\ v.1$	106	712	72	448	73	424
$Dyck\ v.2$	106	774	63	396	63	387
$Dyck\ v.3$	95	616	57	320	58	320
$Dyck\ v.4$	113	828	66	414	66	405
$Dyck\ v.5$	389	2640	196	1104	197	1104
$wwR\ 2s\ v.1$	100	567	79	427	95	483
$wwR\ 2s\ v.2$	82	455	63	329	77	385
$wwR\ 4s$	242	1953	141	1053	145	999
$wwR\ 8s$	1042	13013	393	4589	329	3523
$wwR\ 16s$	5906	122493	1281	25389	889	16443

Таблица 2.5.2: Сравнение получаемых представлений групп

Стоит заметить, что данные замеры были проведены с дополнительными улучшениями, не описанными в данной статье, поскольку данные улучшения незначительны или достаточно очевидны.

3. Экспериментальное исследование

В данном разделе будут описаны шаги, проделанные мною для применения данного теоретического алгоритма на практике.

3.1. Применение системы GAP

В модуле ToGAP описан алгоритм преобразования внутреннего представления представления группы в синтаксисе, понятном системе компьютерной алгебры GAP. Были проведены эксперименты по решению равенства в группе для всех представлений групп, соответствующих тестовым грамматикам.

Однако в ходе экспериментов выяснилось, что система GAP не может проверить за разумное время равенство одного из отношений представления группы для представления группы из 11 отношений и 12 генераторов.

Отношения данного представления группы (здесь все отношения равны 1):

$$c^{-1} \cdot a^{-1} \cdot c \cdot a$$

$$c^{-1} \cdot b^{-1} \cdot c \cdot b$$

$$d^{-1} \cdot a^{-1} \cdot d \cdot a$$

$$e^{-1} \cdot a^{-1} \cdot e \cdot a$$

$$f^{-1} \cdot a^{-1} \cdot f \cdot a$$

$$g^{-1} \cdot a^{-1} \cdot g \cdot a$$

$$h^{-1} \cdot a^{-1} \cdot h \cdot a$$

$$i^{-1} \cdot a^{-1} \cdot i \cdot a$$

$$j^{-1} \cdot a^{-1} \cdot j \cdot a$$

$$k^{-1} \cdot a^{-1} \cdot k \cdot a$$

$$l^{-1} \cdot b^{-1} \cdot l \cdot a^{-1} \cdot l^{-1} \cdot b \cdot l \cdot a$$

Проверяемое равенство (одно из отношений представления группы):

$$d^{-1} \cdot a^{-1} \cdot d \cdot a = 1$$

Данное представление группы является малой частью реального представления группы, построенного для машины Тьюринга a^*. Более того, представленные отношения являются теми отношениями, что присутствуют во всех получаемых представлениях групп по построению. Следовательно, был сделан вывод, что система GAP не подходит для проверки равенств в получаемых представления групп.

3.2. Применение алгоритма Кнута-Бендикса

Одним из способов получить систему, полезную с практической точки зрения, из полученного представления группы — это пполучение системы переписывания строк из данного представления группы с помощью алгоритма Кнута-Бендикса [8]:

Определение 3.2.1 *Система переписывания строк — это пара* $\langle \Sigma, R \rangle$, *где:*

- Σ — *конечный алфавит*
- R — *множество пар слов над алфавитом* Σ

Определение 3.2.2 *Алгоритм Кнута-Бендикса — это полуразрешимый алгоритм преобразования набора уравнений в систему переписывания строк.*

Таким образом если алгоритм Кнута-Бендикса завершается, то он возвращает систему переписывания строк, с помощью которой можно эффективно решать равенства слов в исходной группе.

Были проведены тесты на разрешимость алгоритма Кнута-Бендикса для представлений групп, полученных из тестовых грамматик, описанных в таблице 2.4.1. Для тестирования использовалась реализация алгоритма Кнута-Бендикса на языке Haskell из пакета HaskellForMaths версии 0.4.9.

Результаты применения алгоритма Кнута-Бендикса к представлениям полугрупп, полученных из тестовых машин Тьюринга, представлены в таблице 3.2.1.

Машина Тьюринга	Представление полугруппы		Мощность R системы переписывания строк
	Кол-во ген.	Кол-во отн.	
a^+	7	13	24
$a?$	7	11	25
$a\|b$	8	19	33
abc	11	37	104

Таблица 3.2.1: Результаты применения алгоритма Кнута-Бендикса

Для тестовых грамматик, не встречающихся в таблице 3.2.1, не удалось построить соответствующие им системы переписывания строк по представлениям полугрупп.

Далее мною была выбрана грамматика a^+, поскольку для её представления полугруппы удалось построить систему переписывания строк и её размер является наименьшим среди всех построеных.

Для дальнейшего изложения необходимо ввести следующее определение:

Определение 3.2.3 *Линейный (полный) порядок — это такое бинарное отношение $\leq$ на произвольном множестве A, для которого выполняются свойства*

Рефлексивности: $\forall a \in A : a \leq a$,

Антисимметричности: $\forall a, b \in A : a \leq b \wedge b \leq a \rightarrow a = b$,

Транзитивности: $\forall a, b, c \in A : a \leq b \wedge b \leq c \rightarrow a \leq c$ *и*

Полноты: $\forall a, b \in A : a \leq b \vee b \leq a$

Поскольку на разрешимость алгоритма Кнута-Бендикса влияет линейный порядок, заданный на множестве генераторов представления группы, было принято решение построить несколько различных линейных порядков и попробовать применить алгоритм Кнута-Бендикса к одному и тому же представлению группы, но с разными порядками.

Построение всех линейных порядков, которые были применены для тестирования, можно описать с помощью следующего алгоритма, описанного в модуле GPGenOrds:

1. Возьмём множество генераторов G представления группы и разобьём его на 7 попарно непересекающихся множеств $G_1, G_2, \ldots, G_7$:
   - $\{q\}$
   - $qs = \{q_0, q_1, \ldots, q_N\}$
   - $ss = \{s_0, s_1, \ldots, s_{M+1}\}$
   - $rs = \{r_1, r_2, \ldots, r_K\}$
   - $\{x\}$
   - $\{t\}$
   - $\{k\}$
2. Для множества qs определим линейный порядок либо как $q_i \leq q_j \Leftrightarrow i \leq j$, либо как $q_i \leq q_j \Leftrightarrow i \geq j$
3. Аналогичным образом определим линейный порядок на множестве ss ...
4. ...и rs
5. Зададим линейный порядок на 7 множествах, определённых нами выше, и скажем, что $\forall a \in G_i, b \in G_j, i \neq j : a \leq b \Leftrightarrow G_i \leq G_j$
6. Определим отношение $\leq$ между g и g^{-1} $\forall g \in G$
7. $\forall g_i, g_j \in G : g_i^{-1} \prec g_j^{-1} \Leftrightarrow g_i \prec g_j$

8. Осталось доопределить линейный порядок следующим образом:

   $\forall g_i, g_j \in G : g_i \prec g_j^{-1}$, где отношение $\prec$ определяется как $g_i \prec g_i^{-1}$ из предыдущего шага

   $\forall g_i, g_j \in G : g_i \prec g_j^{-1}$, где отношение $\prec$ определяется как $g_i \prec g_j$

В итоге было сгенерировано 161280 различных линейных порядков и для каждого из них был применён алгоритм Кнута-Бендикса к представлению группы для грамматики a^+. Однако ни для одного линейного порядка алгоритм не смог завершиться за разумное время.

3.3. Построение диаграммы Ван Кампена

Определение 3.3.1 *Диаграмма Ван Кампена — это ориентированный граф с метками на рёбрах, визуально представляющий тот факт, что какое-либо выражение из генераторов и обратных к ним элементов равно нейтральному элементу в группе, по которой строится данная диаграмма.*

Применение диаграммы Ван Кампена выглядит следующим образом:

1. Выбираем на диаграмме вершину v_0
2. Выбираем веришну v_1, которая соединена с v_0 прямым или обратным ребром с меткой l_1, то есть $v_0 \xrightarrow{l_1} v_1$ или $v_1 \xrightarrow{l_1} v_0$
3. Если $v_0 \xrightarrow{l_1} v_1$, то полагаем $x_1 = l_1$, иначе $x_1 = l_1^{-1}$
4. Выбираем вершину v_2, которая соединена с v_1 прямым или обратным ребром с меткой l_2
5. Аналогично если $v_1 \xrightarrow{l_2} v_2$, то полагаем $x_2 = l_2$, иначе $x_2 = l_2^{-1}$
6. ...

7. Если в какой-то момент $v_n = v_0$, то $x_1 \cdot x_2 \cdot \ldots \cdot x_n = 1$ в исходной группе

С помощью этого алгоритма можно убедиться, например, в выполнении равенства $k \cdot q^{-1} \cdot t \cdot q \cdot k^{-1} \cdot q^{-1} \cdot t^{-1} \cdot q = 1$ на диаграмме, изображённой на рис. 3.3.1.

Структура диаграммы Ван Кампена для представления группы, построенному в рассматриваемой статье [3], выглядит следующим образом:

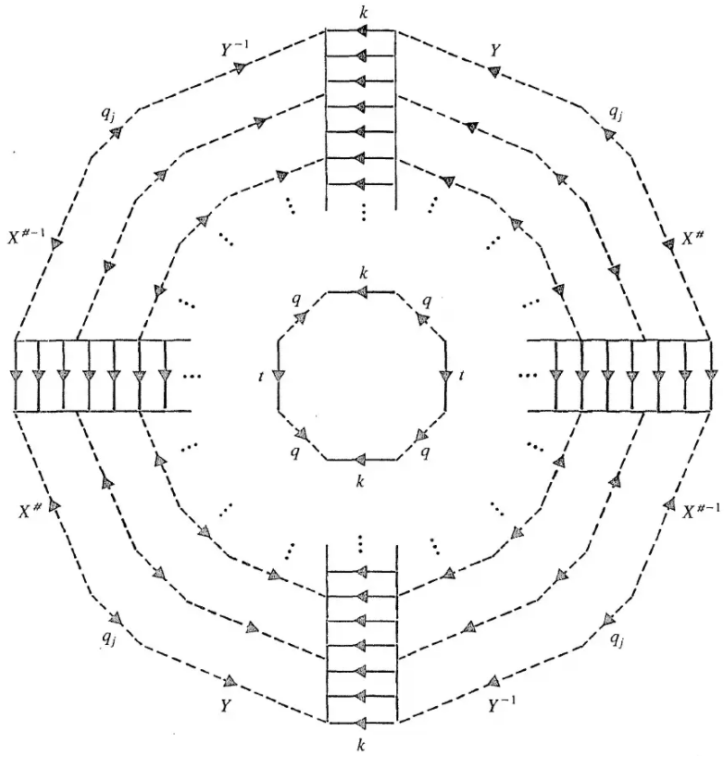

Рис. 3.3.1: Структура диаграммы Ван Кампена из статьи [3]

При этом полагается, что $X \cdot q_i \cdot Y = q$ в **полу**группе, по которой строится группа.

Для построения диаграммы Ван Кампена для конкретной группы и конкретных X и Y было решено использовать программу построения диаграмм Ван Кампена, рассположенную в репозитории на GitHub. Однако данная программа не смогла нарисовать диаграмму так, чтобы её структура совпадала со структурой диаграммы из статьи [3]. Это, скорее всего, возвано тем, что структура правильной диаграммы достаточно сложна.

Заключение

В ходе выполнения данной работы были достигнуты следующие результаты:

1. Реализован алгоритм, предложенный в статье [3], на языке программирования Haskell, программный код выложен в соответствующий репозиторий на GitHub
2. Преобразование внедрено в исходный инструмент:
   - Реализовано преобразование машины Тьюринга из изначального формата в формат, аналогичный описанному в статье
   - Реализовано преобразование представления группы в формат, понятный GAP
3. Предприняты попытки применить данный алгоритм на практике:
   - Проведено тестирование реализованного построения представления группы с помощью GAP
   - Проведена попытка применения алгоритма Кнута-Бендикса преобразования представления группы в систему переписывания строк
   - Проведено построение для представлений групп диаграммы Ван-Кампена

На данный момент есть понимание, что необходим глубокий анализ свойств получаемого представления группы и исследование зависимости этих свойств от свойств исходной грамматики.

Список литературы

[1] Shamrai Maksim. Implementation of the algorithm for constructing a finitely presented group by a Turing machine. 1995. URL: https://oops.math.spbu.ru/SE/diploma/2020/bmo/Shamrai-report.pdf.

[2] Sapir Mark V., Birget Jean-Camille, Rips Eliyahu. Isoperimetric and Isodiametric Functions of Groups // Annals of Mathematics. 2002. Т. 156, № 2. С. 345–466. URL: http://www.jstor.org/stable/3597195.

[3] Rotman J. An Introduction to the Theory of Groups. 1965.

[4] Hopcroft J., Ullman J. Introduction to Automata Theory, Languages and Computation. 1979.

[5] Okhotin A. Conjunctive Grammars // J. Autom. Lang. Comb. 2001. Т. 6. С. 519–535.

[6] Okhotin A. Boolean grammars // Information & Computation. 2004. Т. 194. С. 19–48.

[7] Okhotin A. Conjunctive and Boolean grammars: The true general case of the context-free grammars // Comput. Sci. Rev. 2013. Т. 9. С. 27–59.

[8] Knuth D., Bendix P. Simple Word Problems in Universal Algebras. 1983.