Информационно-графовая модель динамических баз данных и ее применение

А. А. Плетнев

В данной работе рассматривается математическая модель динамических баз данных, которая обрабатывает три типа запросов: поиск, вставка и удаление. Она построена на взаимодействие информационного графа [1] и конечного детерминированного автомата [2]. Модель позволяет решать динамические задачи поиска и оценивать сложность их решения. В качестве примера, иллюстрирующего работу модели, рассмотрена и решена известная динамическая задача поиска идентичных объектов.

Ключевые слова: динамические базы данных, математическое моделирование, информационный граф.

Введение

Функционирование базы данных — это обработка потока запросов типа поиск, вставка и удаление. При этом в результате выполнения запросов типа вставка и удаление база данных изменяется, а на запросы типа поиск выдается ответ. Если поток запросов на поиск существенно преобладает над запросами на изменение базы данных, то такие базы данных называются статическими. Для исследования таких баз данных предназначены информационные графы (ИГ) [1]. Если же поток запросов на изменение базы данных сравним с потоком запросов на поиск, то такие базы данных называются динамическими. Моделированию таких баз данных посвящена данная работа.

Предлагаемая модель динамических баз данных построена на взаимодействии конечного детерминированного автомата и информационного графа. Задача автомата заключается в перестраивании ин-

формационного графа при изменении базы данных, обрабатывая динамические запросы пользователя. Эту структуру будем называть динамическим информационным графом (ДИГ).

Автор благодарит профессора Э. Э. Гасанова за постановку задачи и помощь в работе.

1. Основные понятия

1.1. Информационный граф

Пусть X — множество запросов, Y — множество записей (объектов поиска), ρ — бинарное отношение на $X \times Y$, называемое отношением поиска. Тройку $I = \langle X, V, \rho \rangle$, где V — некоторое подмножество множества Y, в дальнейшем называемой библиотекой, будем называть задачей информационного поиска (ЗИП). Будем считать, что ЗИП $I = \langle X, V, \rho \rangle$ содержательно состоит в перечислении для произвольного взятого запроса $x \in X$ всех тех и только тех записей $y \in V$, что $x \rho y$.

В формальном определении понятия ИГ используются 4 множества: множество запросов X, множество записей Y, множество F одноместных предикатов (заданных на множестве X), множество G одноместных переключателей (заданных на множестве X). Предикат — это функция, множество значений которой есть $\{0, 1\}$. Переключатель — это функция, множество значений которых является начальным отрезком натурального ряда.

Понятие ИГ определяется следующим образом. Берется конечная многополюсная ориентированная сеть. В ней выбирается некоторый полюс, который называется корнем. Остальные полюса называются листьями и им приписываются записи из Y, причем разным листьям могут быть приписаны одинаковые записи. Некоторые вершины сети (в том числе могут быть и полюса) называются переключательными и им приписываются переключатели из G. Ребра, исходящие из каждой из переключательной вершин, нумеруются подряд, начиная с 1, и называются переключательными ребрами. Ребра, не являющиеся переключательными, называются предикатными и им приписываются предикаты из множества F. Таким образом нагруженную многополюсную ориентированную сеть называем ИГ над базовым множе-

ством $\mathcal{F} = \langle F, G \rangle$, где $F = \{f_j, j \in J\}$, $G = \{g_k, k \in K\}$, J и K — множества индексов.

Более подробное описание ИГ приведено в [1].

1.2. Функции над индексами

Введем множество функций преобразования индексов $\mathcal{R}$:

- $\mathcal{R} = \{r_m : J^{d_{1,m}} \times K^{d_{2,m}} \times Y^{d_{3,m}} \to J^{a_m} \times K^{b_m} \times Y^{c_m},\ m \in M,\ M \subseteq \mathbb{N};\ d_{1,m},\ d_{2,m},\ d_{3,m} \in \mathbb{N};\ a_m, b_m, c_m \in \{0,1\}$ и $a_m + b_m + c_m = 1\}$.

1.3. Преобразования ИГ

1.3.1. Определения

Введем три счетных множества переменных $\mathcal{P}_J, \mathcal{P}_K, \mathcal{P}_L$:

- $\mathcal{P}_J = \{p_J^j\}$, $j \in \mathbb{N}$, p_J^j принимают значения из J;
- $\mathcal{P}_K = \{p_K^k\}$, $k \in \mathbb{N}$, p_K^k принимают значения из K;
- $\mathcal{P}_L = \{p_L^l\}$, $l \in \mathbb{N}$, p_L^l принимают значения из Y.

Рассмотрим произвольный ИГ U. Он может содержать предикатные, переключательные и листовые вершины. Обозначим $F(U)$ — множество индексов предикатов, $G(U)$ — множество индексов переключателей и $Y(U)$ — множество записей, входящих в ИГ U. Заменим нагрузку всех входящих в U вершин и ребер по следующему правилу.

- Каждый предикат с индексом $j \in F(U)$ заменим на некоторую переменную из $\mathcal{P}_J$. Причем эта замена задается инъективной функцией $\Omega_F : F(U) \to \mathcal{P}_J$. Это означает, что предикаты с различными индексами $j \in F(U)$ заменим на различные переменные из $\mathcal{P}_J$, а с одинаковыми индексами $j \in F(U)$ заменим на одинаковые переменные из $\mathcal{P}_J$.
- Каждый переключатель с индексом $k \in G(U)$ заменим на переменную из $\mathcal{P}_K$. Причем эта замена задается инъективной функцией $\Omega_G : G(U) \to \mathcal{P}_K$.
- Каждую запись (приписанную листовой вершине) $y \in Y$ заменим на переменную из $\mathcal{P}_L$. Причем эта замена задается инъективной функцией $\Omega_Y : Y(U) \to \mathcal{P}_L$.

Рассмотрим множество $\mathcal{P}(U) = \Omega_F(F(U)) \cup \Omega_G(G(U)) \cup \Omega_Y(Y(U))$, то есть $\mathcal{P}(U)$ — множество переменных, которые получились в результате замены нагрузки всех вершин и ребер из U.

Рассмотрим функцию

$$\Omega : F(U) \cup G(U) \cup Y(U) \to \mathcal{P}(U),$$

где $\Omega = \Omega_F$ на множестве $F(U)$ ($\Omega \mid_{F(U)} = \Omega_F$), $\Omega \mid_{G(U)} = \Omega_G$ и $\Omega \mid_{Y(U)} = \Omega_Y$.

Очевидно из определения, что Ω — биективная функция, поэтому существует функция Ω^{-1}. Обозначим $I = \Omega^{-1}$ и будем называть **интерпретацией**, возникшей в результате замены индексов на переменные.

После этого сопоставления (замены нагрузки вершин и ребер на переменные в ИГ U) получим нагруженный граф. Выделим в нем множество вершин $V_{\mathcal{T}}$, которые назовем вершинами прикрепления, и занумеруем их числами от 1 до $|V_{\mathcal{T}}|$. Полученный граф назовем **простым шаблоном** $\mathcal{T}$. При этом будем писать $\mathcal{T} = \mathcal{T}(U, I, V_{\mathcal{T}})$, когда хотим подчеркнуть, что $\mathcal{T}$ получен из ИГ U и I, где I — интерпретация, возникшая в результате замены индексов на переменные, $V_{\mathcal{T}}$ — упорядоченное множество вершин прикрепления.

Пример простого шаблона смотри на рис. 1 (жирным «•» выделена единственная вершина прикрепления).

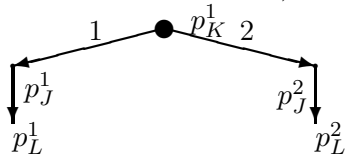

Рис. 1. Пример простого шаблона.

Рассмотрим ИГ U', выделим в нем множество вершин $V_{U'}$, которые назовем вершинами прикрепления, и занумеруем их числами от 1 до $|V_{U'}|$ (вершины прикрепления упорядоченное множество). Будем говорить, что ИГ U' и простой шаблон $\mathcal{T}$ **согласованы**, если выполнены следующие условия:

- U' и $\mathcal{T}$ совпадают как графы, и если в ИГ U' встречаются одинаковые индексы предикатов, индексы переключателей и записи, то в соответствующих местах шаблона $\mathcal{T}$ находятся одинаковые переменные из $\mathcal{P}_J$, $\mathcal{P}_K$ и $\mathcal{P}_L$;

- i-ая вершина прикрепления ИГ U' совпадает с i-ой вершиной прикрепления простого шаблона $\mathcal{T}$, $i = \overline{1, \ldots, |V_{U'}|}$ и $|V_{U'}| = |V_{\mathcal{T}}|$.

То есть $\mathcal{T} = \mathcal{T}(U', I, V_{\mathcal{T}})$ для некоторой интерпретации I, и будем писать $\mathcal{T} = \mathcal{T}(U', I, V_{U'})$, когда хотим подчеркнуть, что ИГ U' и простой шаблон $\mathcal{T}$ согласованы.

Рассмотрим простой шаблон $\mathcal{T}_1$ и заменим в нем каждую переменную, на формулу над множество операций из $\mathcal{R}$ и каким-либо множеством переменных $M \subseteq \mathcal{P}_J \cup \mathcal{P}_K \cup \mathcal{P}_L$. В результате, полученный граф назовем **шаблоном** $\mathcal{T}_2$. При этом будем писать $\mathcal{T}_2 = \mathcal{T}_2(\mathcal{T}_1, M, V_{\mathcal{T}_1})$, когда хотим подчеркнуть, что $\mathcal{T}_2$ получен из простого шаблона $\mathcal{T}_1$, множества переменных M и $V_{\mathcal{T}_1}$ упорядоченное множество вершин прикрепления.

На рисунках запись $r_i(M)$ будет означать формулу над множеством переменных из M, которая реализует функцию из $\mathcal{R}$.

Пример шаблона приведен на рис. 2 (жирным «•» выделена единственная вершина прикрепления).

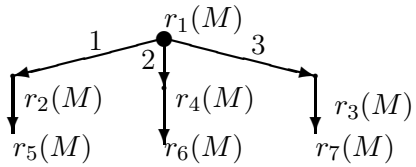

Рис. 2. Пример шаблона $\mathcal{T}_2$.

Рассмотрим простой шаблон $\mathcal{T}_1 = \mathcal{T}_1(U, I, V_{\mathcal{T}_1})$. Обозначим множество входящих в него переменных $M(\mathcal{T}_1)$. Пусть $p_L^0 \notin M(\mathcal{T}_1)$.

Преобразованием C назовем тройку

$$C = (\mathcal{T}_1(U, I, V_{\mathcal{T}_1}), \mathcal{T}_2(\mathcal{T}, M(\mathcal{T}_1) \cup p_L^0, V_{\mathcal{T}}), \xi),$$

где $\mathcal{T}_1$ — простой шаблон, $\mathcal{T}_2$ — шаблон (полученный из простого шаблона $\mathcal{T}$) в формулах которого встречаются только переменные из множества $M(\mathcal{T}_1) \cup p_L^0$, $V_{\mathcal{T}}$ упорядоченное множество вершин прикрепления простого шаблона $\mathcal{T}$, ξ — биективная функция сопоставления вершин прикрепления ($\xi : V_{\mathcal{T}_1} \to V_{\mathcal{T}}$). Левой частью преобразования C будем называть простой шаблон $\mathcal{T}_1$.

Пусть ИГ U_1 и простой шаблон $\mathcal{T}_1$ согласованы, то есть $\mathcal{T}_1 = \mathcal{T}_1(U_1, I, V_{U_1})$ для некоторой интерпретации I, и пусть I' — интерпретация множества переменных $M(\mathcal{T}_1) \cup p_L^0$, причем $I' = I$ на множестве переменных $M(\mathcal{T}_1)$, тогда применением преобразования $C = (\mathcal{T}_1(U, I, V_{U_1}), \mathcal{T}_2(\mathcal{T}, M(\mathcal{T}_1) \cup p_L^0, V_{\mathcal{T}}), \xi)$ к ИГ U_1 назовем ИГ U_2, получающийся из шаблона $\mathcal{T}_2(\mathcal{T}, M(\mathcal{T}_1) \cup p_L^0, V_{\mathcal{T}})$ подстановкой вместо каждой формулы значения данной формулы в интерпретации I' и упорядоченное множество вершин прикрепления V_{U_2} в ИГ U_2 однозначно соответствует упорядоченному множеству вершин прикрепления V_{U_1} в ИГ U_1 (так как ξ — биективная функция и простой шаблон $\mathcal{T}_1$ согласован с ИГ U_1).

Результат применения преобразования C к ИГ U_1 будем обозначать $C(U_1) = U_2$ (см. рис. 3).

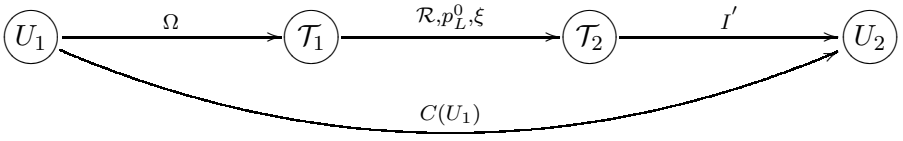

Рис. 3. Преобразование ИГ U_1 в ИГ U_2.

1.3.2. Описание алгоритма 2–3 дерева

Известно множество структур данных позволяющих за разумное время решать динамическую задачу поиска идентичных объектов, в частности структура 2–3 дерева, которая позволяет решать эту задачу за логарифмическое время и линейную память (от объема библиотеки). Описание 2–3 дерева возьмем из [3].

2–3 деревом называется дерево, в котором каждая вершина, не являющаяся листом, имеет двух или трех сыновей, а длины всех путей из корня в листья одинаковы. Заметим, что дерево, состоящее из единственного узла является 2–3 деревом.

Рассмотрим линейно упорядоченное множество. Его можно представить с помощью 2–3 дерева следующим образом: элементы множества приписываются листьям слева направо по порядку. В каждой внутренней вершине v хранятся две величины: $L[v]$ — максимальный элемент в левом поддереве узла v и $M[v]$ — максимальный элемент в среднем (если две дуги, тогда в правом) поддереве v.

Пример 2–3 дерева для множества $\{1, 2, 3, 4, 5, 7\}$ (см. рис. 4).

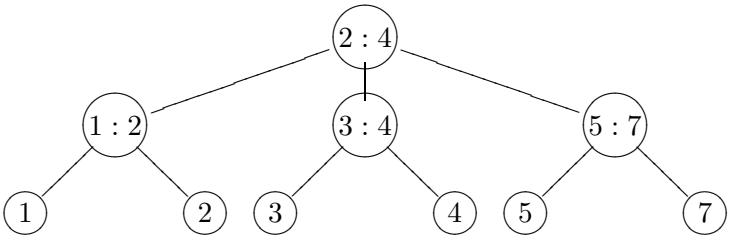

Рис. 4. Пример 2–3 дерева.

Опишем алгоритм **A** функционирования 2–3 дерева:

A_0**: Поиск** y**.**

Поиск элемента y в 2–3 дереве происходит путем последовательного сравнения по следующему алгоритму: если $y \leqslant L[v]$, то переходим к левому сыну v, иначе если у v два сына, или $v \leqslant M[v]$, то переходим к среднему сыну v, иначе переходим к правому сыну v.

A_1**: Вставка** y**.**

$A_{1.0}$: если дерево состоит из единственного узла l с меткой y_0, образуем новый корень l'. Образуем новый узел v с меткой y. $L(v) = \min\{y_0, y\}, M(v) = \max\{y_0, y\}$ и делаем y_0, y сыновьями корня l', причем l будет левым сыном, если $y_0 < y$, и правым в противном случае.

$A_{1.1}$: если дерево содержит более одного узла, то нужно найти место для нового листа l, который будет содержать y. Для этого ищут элемент y в дереве (A_0). Если дерево содержит более одного элемента, то поиск y окончится в узле f, имеющим двух или трех сыновей, которые являются листами. Образуем новый лист l с меткой y.

$A_{1.2}$: если у f два сына с метками y_1 и y_2, то делаем l сыном узла f таким образом, чтобы сохранилась упорядоченность меток слева направо.

$A_{1.3}$: если у f три сына, делаем l сыном узла f, сохраняя упорядоченность меток. Затем, чтобы включить узел f и его четырех сыновей в дерево, выполняем шаг $A_{1.4}$ (рекурсивный).

$A_{1.4}$: образуем новый узел g. Два левых сына оставим сыновьями f, а два правых переделаем в сыновей узла g. Затем сделаем g братом узла f, сделав его сыном отца узла f. Если отец узла f уже имел трех сыновей, то надо рекурсивно повторять $A_{1.4}$ до тех пор,

пока у всех узлов в дереве останется не более трех сыновей. Если у корня окажется четыре сына, образуем новый корень с двумя новыми сыновьями, каждый из которых будет иметь в качестве двух своих сыновей двух из четырех сыновей старого корня.

A_2: Удаление y.

Элемент y можно удалить из 2–3 дерева, по существу способом обратным к вставке. Пусть y — метка листа l.

$A_{2.0}$: если l — корень, удаляем его (y был единственным элементов в дереве).

$A_{2.1}$: если l — сын узла, имеющего трех сыновей, удаляем его.

$A_{2.2}$: если l — сын узла f, имеющего двух сыновей (s брат l), то возможно 2 случая ($A_{2.2.1}, A_{2.2.2}$).

$A_{2.2.1}$: если f — корень, то удаляем l и f и делаем корнем второго сына s.

$A_{2.2.2}$: если f — не корень и допустим f имеет брата g слева от себя (справа аналогично).

$A_{2.3}$: если у g два сына, делаем узел s самым правым сыном узла g, удаляем l и рекурсивно вызываем A_2, чтобы удалить f.

$A_{2.4}$: если у g три сына, то самого правого сына делаем левым сыном узла f и удаляем l.

Пусть база данных из примера приведенного выше и мы хотим добавить к ней число 6. В результате должно получиться следующее дерево (см. рис. 5).

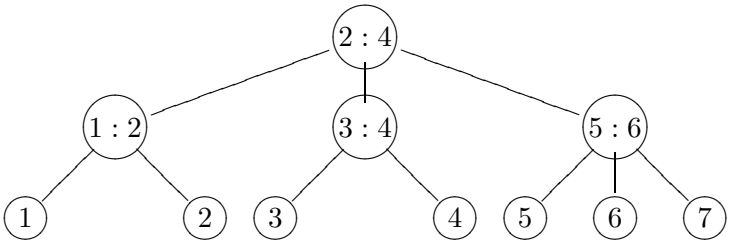

Рис. 5. К 2–3 дереву добавили 6.

Как видно мы применили к части дерева некоторое преобразование.

Такое преобразование можно описать формально, используя в качестве структуры ИГ и шаблонный язык. Для этого опишем 2–3 дерево на языке ИГ.

Пусть $J = \{(a) : a \in \mathbb{R}\}$, $K = \{(a,b) : a,b \in \mathbb{R}\}$.

$$G_{id} = \left\{ g_{a,b}(x) = \begin{cases} 1, & \text{если } x \leqslant a; \\ 2, & \text{если, } a < x \leqslant b; \text{ , } (a,b) \in K \\ 3, & \text{иначе.} \end{cases} \right\}; \qquad (1)$$

$$F_{id} = \left\{ f_{=,a}(x) = \begin{cases} 1, & \text{если } x = a; \\ 0, & \text{если } x \neq a. \end{cases} , a \in J \right\}; \qquad (2)$$

$$\mathcal{F}^{id} = \langle F_{id} \cup G_{id} \rangle. \qquad (3)$$

Таким образом каждому предикату $f_{=,a} \in F$ сопоставлен индекс $(a) \in J$, а каждому переключателю $g_{a,b} \in G$ сопоставлен индекс $(a,b) \in K$.

ИГ U_{id} над базовым множеством $\mathcal{F}^{id}$ строится по аналогии с 2–3 деревом следующим образом. В вершинах в которых стоят числа $A : B$ назовем переключательными и припишем им переключатели $g_{A,B} \in G$. К концевым вершинам проведем соответствующие предикатные ребра $f_{=,y} \in F$ (см. рис. 6).

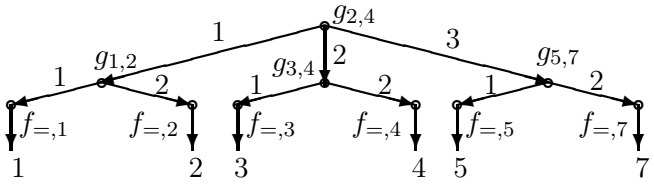

Рис. 6. ИГ построенный по аналогии с 2–3 деревом.

Приведем пример как формально описывается преобразование описанное выше (добавление 6 к 2–3 дереву в виде ИГ).

1.3.3. Формулы над переменными из вариационного ряда

Пусть $Z = \{z_1, z_2, \ldots, z_k\}$ — множество переменных, принимающих значения из $\mathbb{R}$. Пусть $\widehat{Z} = (z_{(1)}, z_{(2)}, \ldots, z_{(k)})$ — вариационный ряд соответствующий множеству Z, то есть $\widehat{Z}$ это вектор, компоненты которого есть упорядоченные по неубыванию значения переменных множества Z.

Обозначим $\mu_i(Z) = z_{(i)}$, то есть $\mu_i(Z)$ — i-ый элемент вариационного ряда, соответствующего Z. Введем следующие обозначения для функций:

$$r^{i,j}(Z) = (\mu_i(Z), \mu_j(Z)); r^i(Z) = (\mu_i(Z)); r^i_L(Z) = \mu_i(Z), i, j \in \mathbb{N} \quad (4)$$

1.3.4. Пример преобразования ИГ

Вершина выделенная жирным («•») на всех рисунках в этом примере будет соответствовать вершине прикрепления.

Рассмотрим ИГ U_1 (рис. 7).

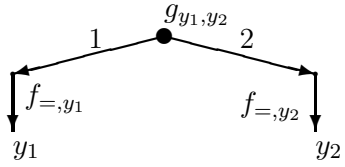

Рис. 7. ИГ $U_1, (y_1 < y_2)$.

Нужно найти такое преобразование C_e, чтобы $C_e(U_1) = U_2$ (рис. 8), в результате добавления новой записи y_3 к ИГ U_1, где $(y_{(1)}, y_{(2)}, y_{(3)})$ — вариационный ряд, соответствующий множеству $\{y_1, y_2, y_3\}$.

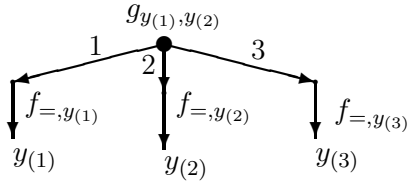

Рис. 8. ИГ U_2.

Предъявим простой шаблон $\mathcal{T}_1(U_1, I, V_{U_1})$, согласованный с U_1 (рис. 9), а соответствующая этому простому шаблону интерпретация I: $I(p^1_K) = (y_1, y_2) = g_{y_1,y_2}; I(p^1_J) = (y_1) = f_{=,y_1}; I(p^2_J) = (y_2) = f_{=,y_2}; I(p^1_L) = y_1; I(p^2_L) = y_2$.

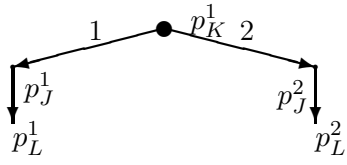

Рис. 9. Простой шаблон $\mathcal{T}_1$.

Следуя введенным обозначениям, $M(\mathcal{T}_1) = \{p_J^1, p_J^2, p_K^1, p_L^1, p_L^2\}$ и $p_L^0 \notin M(\mathcal{T}_1)$. Рассмотрим интерпретацию $I^{'}$ множества переменных $M \cup p_L^0$, причем

- $I^{'} = I$ на множестве переменных M;
- $I^{'}(p_L^0) = y_3$ (запрос на добавление).

Предъявим теперь шаблон $\mathcal{T}_2(\mathcal{T}, M_L, V_\mathcal{T})$ (рис. 10), $M_L = \{p_L^0, p_L^1, p_L^2\}$.

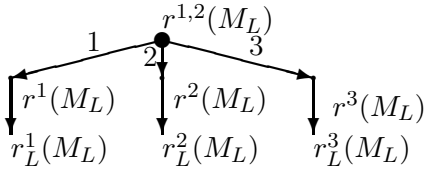

Рис. 10. Шаблон $\mathcal{T}_2$.

Исходя из определения формул преобразования, получаем:
$I^{'}(r^{1,2}(M_L)) = (y_{(1)}, y_{(2)}) = g_{y_{(1)},y_{(2)}}$; $I^{'}(r^1(M_L)) = (y_{(1)}) = f_{=,y_{(1)}}$; $I^{'}(r^2(M_L)) = (y_{(2)}) = f_{=,y_{(2)}}$; $I^{'}(r^3(M_L)) = (y_{(3)}) = f_{=,y_{(3)}}$; $I^{'}(r_L^1(M_L)) = y_{(1)}$; $I^{'}(r_L^2(M_L)) = y_{(2)}$; $I^{'}(r_L^3(M_L)) = y_{(3)}$.

Если положить $y_1 = 5, y_3 = 6, y_2 = 7$, то получим преобразование, используемое в 2–3 дереве, которое было описано выше (см. рис. 4 и рис. 5).

2. Динамический информационный граф

2.1. Определения

Рассмотрим ИГ U, $N(U)$ — множество входящих в него вершин. Пусть $\beta, \beta^{'} \in N(U)$, путем $\pi(\beta, \beta^{'})$ назовем последовательность вершин и ребер ИГ U, которая начинается в вершине β, заканчивается в вершине $\beta^{'}$, и в этой последовательности два любых соседних элемента инциденты. Длиной пути $l(\pi(\beta, \beta^{'}))$ назовем количество ребер, участвующих в нем. Пусть $\Pi(\beta, \beta^{'})$ — множество всех путей из β в $\beta^{'}$. Тогда расстоянием между вершинами $\beta, \beta^{'} \in N(U)$ назовем минимальную из всех путей длину $l(\pi(\beta, \beta^{'})), \pi(\beta, \beta^{'}) \in \Pi(\beta, \beta^{'})$ и обозначим $(\beta, \beta^{'}) = \min\{l(\pi(\beta, \beta^{'})) : \pi(\beta, \beta^{'}) \in \Pi(\beta, \beta^{'})\}$. Эксцентриситетом

вершины $\beta \in N(U)$ назовем число $\varepsilon(\beta) = \max\limits_{\beta' \in N(U)} (\beta, \beta')$. **Радиусом** ИГ U назовем число $r(U) = \min\limits_{\beta \in N(U)} \varepsilon(\beta)$.

Через $\mathcal{G}(N, R)$, $N, R \in \mathbb{N}$ обозначим класс ИГ радиуса не больше R таких, что количество ребер инцидентных любой вершине графа не превосходит N.

Рассмотрим ИГ U, $E(U)$ — множество входящих в него ребер. Ребро инцидентное вершинам $\beta_1, \beta_2 \in N(U)$ будем обозначать $e(\beta_1, \beta_2)$.

Рассмотрим ИГ U_2 и его подграф (подмножество ребер и инцидентным им вершин) U_1 (пишем $U_1 \subseteq U_2$). Множество $\Sigma(U_1, U_2) = \{\beta : \beta \in N(U_1)$ и $\exists \beta_1 \in N(U_2 \backslash U_1), \exists\ \beta_2 \in N(U_1),\ e(\beta_1, \beta) \in E(U_2),\ e(\beta_2, \beta) \in E(U_1)\}$ назовем множеством **граничных вершин** ИГ U_1 и ИГ U_2.

Пусть $U_1, U_2 \in \mathcal{G}(N, \infty)$ и $U_1 \subseteq U_2$ $(N < \infty)$.

Назовем **кодом вершины** на запросе $x \in X$ **типа поиск** относительно пары (U_1, U_2) четверку (k_1, k_2, k_3, k_4), где $k_1 = 0$, если вершина предикатная; $k_1 = 1$, если она переключательная; $k_2 = 0$, если вершина корень; $k_2 = 1$, если вершина листовая; $k_2 = 2$ в остальных случаях; $k_3 \in \{1, \ldots, N+1\}$ и в случае переключательной вершины принимает значение соответствующего ей переключателя на запросе $x \in X$, если это значение принадлежит $\{1,\ldots,\mathrm{N}\}$, и $k_3 = N + 1$ во всех остальных случаях; $k_4 \in \{0, 1\}$, $k_4 = 1$, если вершина принадлжет множеству граничных вершин $\Sigma(U_1, U_2)$ и $k_4 = 0$, если не принадлежит.

Кодом ребра на запросе $x \in X$ типа поиск относительно пары (U_1, U_2) назовем пару (k_1^r, k_2^r), где $k_1^r = 0$, если ребро предикатное; $k_1^r = 1$, если ребро переключательное; $k_2^r \in \{0, 1, \ldots, N\}$. У предикатного ребра k_2^r равен значению переключателя на запросе $x \in X$ (то есть предикатным ребрам приписан код $(0, 0)$ или $(0, 1)$), а каждому переключательному ребру приписан код $(1, i)$, где $i \in \{1, \ldots, N\}$ номер переключательного ребра.

В общем случае множества запросов X и множество записей Y могут быть разные. Поэтому для того, чтобы определить код вершин и ребер ИГ на запросах типа вставка и удаление введем дополнительную функцию $\eta : Y \to X^l$, $l \in \mathbb{N}$, $l < \infty$, $\eta = (\eta_1, \ldots, \eta_l)$.

Кодом вершины на запросе $y \in Y$ **типа вставка, удаление** относительно пары (U_1, U_2) будет l четверок кодов вершин, которые

соответствуют запросам $\eta_1(y), \ldots, \eta_l(y)$ типа поиск. Кодом ребра на запросе $x \in Y$ будет значение l пар кодов ребер, которые соответствуют запросам $\eta_1(y), \ldots, \eta_l(y)$ типа поиск.

Рассмотрим ИГ U, в нем каждая вершина и ребро имеет нагрузку. Граф $K(U)$ полученный из ИГ U удалением нагрузок вершин и ребер, назовем **каркасом** ИГ U.

Кодом ИГ U_1 относительно ИГ U_2 **на запросе**, назовем нагруженный каркас $K(U_1)$, где нагрузка каждой вершины $K(U_1)$ это код данной вершин на данном запросе относительно пары (U_1, U_2), а на нагрузка каждого ребра $K(U_1)$ это код данного ребра на данном запросе относительно пары (U_1, U_2). Через $\mathcal{K}(N, R)$ обозначим множество кодов ИГ U_1 относительно ИГ U_2, где U_1 пробегает класс ИГ $\mathcal{G}(N, R)$, U_2 пробегает класс ИГ $\mathcal{G}(N, R+1)$ и $U_1 \subseteq U_2$. Понятно, что $|\mathcal{K}(N, R)| < \infty$.

Пусть $N, R \in \mathbb{N}$, $\mathcal{P}$ — некоторое конечное множество преобразований, шаблоны которых порождены ИГ из $\mathcal{G}(N, R)$ и $C_0 \in \mathcal{P}$, где C_0 — тождественное преобразование такое, что $C_0(U) = U$ для произвольного ИГ U из $\mathcal{G}(N, R)$.

Пусть $N(U_1)$ множество вершин ИГ U_1. Рассмотрим множество вершин $\mathfrak{B} \subseteq 2^{N(U_1)}$ и обозначим через $U(\mathfrak{B}, U_1)$ — ИГ, полученный из ИГ U_1 как подграф на вершинах $\mathfrak{B}$.

Пусть $\mathcal{A}$ — конечный автомат (подробно об определении конечного автомата в [2]). Будем считать, что автомат $\mathcal{A}$ перемещается по вершинам ИГ $U_2 \in \mathcal{G}(N, \infty)$ и в каждый момент времени обозревает окрестность U_1 текущей вершины радиуса R, то есть будем считать, что входным символом автомата $\mathcal{A}$ является код обозреваемой окрестности относительно U_2. Понятно, что эти коды будут принадлежать $\mathcal{K}(N, R)$ и значит входным алфавитом автомата $\mathcal{A}$ является конечное множество $\mathcal{K}(N, R)$. Результатом этого автомата $\mathcal{A}$ на обозреваемой окрестности текущей вершины будет некоторое преобразование этой окрестности и перемещение в некоторую вершину преобразованной окрестности. Тем самым выходным символом автомата $\mathcal{A}$ будет пятерка $b = (\mathfrak{B}, \pi, C, \beta, e)$, где

- $\mathfrak{B} \subseteq 2^{N(U_1)}$ определяет множество вершин обозреваемой окрестности к которому будет применено преобразование;
- π — функция нумерации граничных вершин $U(\mathfrak{B}, U_1)$ и U_2 (содержательно это множество граничных вершин $U(\mathfrak{B}, U_1)$ и U_1

объединенное со множеством вершин в коде которых $k_4 = 1$ и одновременно принадлежащих $U(\mathfrak{B}, U_1)$). Пронумерованные граничные вершины назовем вершинами прикрепления;

- C — преобразование из конечного множества $\mathcal{P}$ применяемое к ИГ $U(\mathfrak{B}, U_1)$, причем ИГ $U(\mathfrak{B}, U_1)$ с заданной функцией π нумерацией вершин прикрепления согласован с левой частью преобразования C;
- $\beta \in N(C(U(\mathfrak{B}, U_1))) \cup \{U_1 \setminus U(\mathfrak{B}, U_1)\}$ — следущая текущая вершина автомата $\mathcal{A}$;
- $e \in \{0, 1, 2\}$ отвечает за продолжение и остановку передвижения автомата $\mathcal{A}$.

Множество всех таких выходных символов b образует выходной алфавит B автомата $\mathcal{A}$.

Пусть U — ИГ над базовым множеством $\langle F, G\rangle$ из класса $\mathcal{G}(N, \infty)$, тогда пару $(\mathcal{A}, U)$ назовем динамическим информационным графом (ДИГ) типа (N, R) над $\mathcal{F}{=}\langle F, G, \mathcal{P}, \eta\rangle$ и обозначим $\mathcal{D} = (\mathcal{A}, U)$.

2.2. Функционирование ДИГ

Определим функционирование ДИГ $\mathcal{D} = (\mathcal{A}, U)$ типа (N, R) над $\mathcal{F}{=}\langle F, G, \mathcal{P}, \eta\rangle$ на запросе. Если запрос есть запрос на поиск, то функционирование ДИГ $\mathcal{D}$ совпадает с функционированием ИГ U и не задействует автомат $\mathcal{A}$ (подробное описание функционирования ИГ в [1]).

Если запрос является запросом на вставку или удаление, то функционирование ДИГ $\mathcal{D}$ происходит следующим образом. В начальный момент текущей вершиной объявляется корень ИГ U. Рассматривается ИГ U_1 как подграф U, с центром в текущей вершине и радиуса R.

На вход автомата $\mathcal{A}$ подается код ИГ U_1 относительно ИГ U на запросе. Пусть $b = (\mathfrak{B}, \pi, C, \beta, e) \in B$ выходная буква автомата $\mathcal{A}$, тогда ИГ U меняется по следующему правилу. В ИГ U ИГ $U(\mathfrak{B}, U_1)$ заменяется на ИГ U', полученной в результате применения преобразования C к ИГ $U(\mathfrak{B}, U_1)$ $(C(U(\mathfrak{B}, U_1)) = U')$ и «прикрепляется» к ИГ U посредством функции π и функции сопоставления вершин прикрепления ξ из преобразования C. После этого текущее положение автомата $\mathcal{A}$ меняется на β и в зависимости от значения последней

компоненты выходной буквы b ДИГ $\mathcal{D}$ либо продолжает функционирование ($e = 0$), либо завершает успешно ($e = 1$), либо завершает функционирование с ошибкой ($e = 2$), где под ошибкой понимается те случаи, когда был запрос на вставку записи существующей в U или был запрос на удаление записи не существующей в U.

Замечание: Операция замены ИГ $U(\mathfrak{B}, U_1) \subseteq U$ на ИГ $C(U(\mathfrak{B}, U_1)) = U''$ корректная, то есть ДИГ $\mathcal{D} = (\mathcal{A}, U'')$ останется типа (N, R), где ИГ U'' это ИГ полученный из ИГ U заменой подграфа $U(\mathfrak{B}, U_1)$ на U'. Этот факт следует из определения множества преобразований $\mathcal{P}$, а именно, что все преобразования из $\mathcal{P}$ порождены шаблонами из $\mathcal{G}(N, R)$, то есть после преобразования получим, что $U'' \in \mathcal{G}(N, \infty)$.

2.3. Сложность и объем ДИГ

Сложность и объем ДИГ $\mathcal{D} = (\mathcal{A}, U)$ типа (N, R) над базовым множеством $\mathcal{F}=\langle F, G, \mathcal{P}, \eta\rangle$ определяется для фиксированного ИГ U. Объемом ДИГ $\mathcal{D}$ будем называть объем ИГ U (количество ребер в графе) и обозначим $Q(\mathcal{D})$. Сложность ДИГ $\mathcal{D}$ на запросе $x \in X$ типа поиск положим равной $T(U, x)$ — сложность ИГ U на запросе x. Подробно об определении сложности ИГ U на запросе x можно прочитать в [1].

Определим сложность ДИГ $\mathcal{D}$ на запросе $x \in X$ типа вставка (удаление) и обозначим их соответственно $T_{in}(\mathcal{D}, x)$ ($T_{del}(\mathcal{D}, x)$). Для этого введем сложность выходного действия автомата $\mathcal{A}$.

Пусть $L : \mathcal{P} \to \mathbb{N}$, входным параметром, которой является $C \in \mathcal{P}$, а значением сложность выполненного преобразования. Другими словами с помощью L вводится сложность каждого элемента из множества преобразований $\mathcal{P}$.

Последовательность преобразований, выполненных автоматом $\mathcal{A}$, в ходе функционирования ДИГ $\mathcal{D}$ на запросе $x \in X$ типа вставка обозначим $f_{\mathcal{A}}(x, 1)$ (типа удаление $f_{\mathcal{A}}(x, 2)$).

Сложностью ДИГ $\mathcal{D}$ на запросе $x \in X$ типа вставка (удаление) назовем

- $T_{in}(\mathcal{D}, x) = \sum\limits_{C \in f_{\mathcal{A}}(x,1)} L(C)$ ($T_{del}(\mathcal{D}, x) = \sum\limits_{C \in f_{\mathcal{A}}(x,2)} L(C)$).

Стандартным образом вводится понятие сложности в худшем случае. Опишем подробнее сложность в худшем для запроса типа поиск.

Величину $\widehat{T}_{in}(\mathcal{D}) = \max_{x \in X} T_{in}(\mathcal{D}, x)$ назовем верхней сложностью (или сложностью в худшем случае) вставки в ДИГ $\mathcal{D}$.

Верхней сложностью ДИГ $\mathcal{D}$ назовем $\max\{\widehat{T}_{in}(\mathcal{D}), \widehat{T}_{del}(\mathcal{D}), \widehat{T}(U)\}$ и обозначим $\widehat{T}(\mathcal{D})$, где $\widehat{T}(U) = \max_{x \in X} T(U, x)$.

3. Применение модели

3.1. Постановка задачи

Пусть $V \subset Y \subseteq \mathbb{R}$, $|V| < \infty$, $X = Y$. Скажем, что ДИГ $\mathcal{D}$ решает задачу поиска идентичных объектов (ЗПИО) $\langle X, V, = \rangle$, если ответ на произвольный запрос $x \in X$ типа поиска равен $\{x\}$, если $x \in V$, и пуст в противном случае, и если на произвольном запросе типа вставки (удаления) записи $y \in Y$ результирующий ДИГ $\mathcal{D}$ выдает ответ $\{x\}$ на произвольный запрос $x \in X$ типа поиск, если $x \in V \cup \{y\}$ ($x \in V \setminus \{y\}$), и пуст в противном случае.

3.2. Решение задачи

3.2.1. Описание базового множества $\mathcal{F}_{id}$

В качестве базового множества будем рассматривать множество

$$\mathcal{F}_{id} = \langle F_{id}, G_{id}, \mathcal{P}_{id}, \eta_{id} \rangle, \tag{5}$$

где F_{id}, G_{id} определены соотношениями (1), (2). В качестве функции η_{id} рассматриваем тождественную функцию $\eta_{id} : Y \to X$ (в ЗПИО $Y = X$). Базовое множество преобразований $\mathcal{P}_{id}$ состоит из 55 преобразований, которые изображены на рис. 7–30, а также включает в себя тождественное преобразование $C_0 \in \mathcal{P}_{id}$.

Во всех преобразованиях из $\mathcal{P}_{id}$ шаблоны порождены ИГ из $\mathcal{G}(5, 3)$, множество индексов $J = \{(a) : a \in \mathbb{R}\}$, $K = \{(a, b) : a, b \in \mathbb{R}\}$, и во всех преобразованиях использованы формулы, реализующие функции из $\mathcal{R}_{id}$, которое определим ниже.

Введем обозначения для краткости записи формул в преобразованиях. Пусть $\mathcal{T}$ — простой шаблон и $M(\mathcal{T})$ — множество переменных, входящих в этот простой шаблон. Обозначим $M_J(\mathcal{T}) = M(\mathcal{T}) \cap P_J$, $M'_K(\mathcal{T}) = M(\mathcal{T}) \cap P_K$, $M'_L(\mathcal{T}) = M(\mathcal{T}) \cap P_L$. В случае когда шаблон $\mathcal{T}$ определен однозначно, будем писать просто множества M_J, M'_K и M'_L.

Если $p \in P_K$, тогда значением этой переменной будет пара из K. Через p^1 и p^2 обозначим первую и вторую компоненту переменной p, то есть $p = (p^1, p^2)$, p^1 и p^2 принимают значения из $\mathbb{R}$. Пусть $M \subseteq P_K$, тогда $\widehat{M} = \bigcup\limits_{p \in M} p^1 \cup \bigcup\limits_{p \in M} p^2$.

Пусть $M_L = M_L^{'} \sqcup p_L^0$ ($p_L^0 \in P_L, p_L^0 \notin M_L^{'}$), $M_K = \widehat{M_K^{'}}$, $M \subseteq M_J \cup M_K \cup M_L$, тогда $\mathcal{R}_{id}$ состоит из функций $r^{i,j}(M), r^i(M), r_L^i(M)$, где $i, j \in \{1, 2, 3, 4\}$ и функции определены соотношением (4), а так же еще одной функции

$$r_{sp}(p_L^0, (p_J^1), (a, b)) = \begin{cases} r^{1,2}(p_J^1, b), & \text{если } p_L^0 = a; \\ r^{1,2}(a, p_J^1), & \text{если } p_L^0 = b; \ , (a, b) \in K \\ r^{1,2}(a, b), & \text{иначе.} \end{cases} \qquad (6)$$

.

3.2.2. Формулировка и доказательство теоремы

Теорема 1. *Существует ДИГ $\mathcal{D}_{id} = (\mathcal{A}_{id}, U_{id})$ типа $(5, 3)$ над базовым множеством $\mathcal{F}_{id}$, определяемым соотношением (5), который решает ЗПИО $\langle X, V, = \rangle$, причем*

$$\widehat{T}(D_{id}) \leqslant (\lambda_{max} + 1)([\log_2 |V|] + 1);$$
$$Q(D_{id}) \leqslant 3|V|,$$

где $\lambda_{max} = \max\{L(C) : C \in \mathcal{P}_{id}\}$.

Доказательство основывается на алгоритме **A** (алгоритм 2–3 дерева), описанного выше. ИГ U_{id} будет графом построенный по аналогии с 2–3 деревом (см. рис. 6), а автомат $\mathcal{A}_{id}$, используя преобразования из $\mathcal{P}_{id}$, в некотором смысле будет повторять алгоритм **A**.

Доказательство разобьем на три подпункта, соответствующие запросам типа поиск, вставка и удаление. Каждый подпункт описывается в виде алгоритма, который выполняет автомат (при получении запроса автомат $\mathcal{A}_{id}$ переходит в соответствующее состояние).

Сложность и объем ДИГ $\mathcal{D}_{id}$ определяется при фиксированной библиотеке V.

Поиск

ДИГ $\mathcal{D}_{id}$ функционирует как ИГ U_{id}. Сложность ИГ U_{id} на запросе не превосходит $[\log_2 |V|] + 1$. Этот факт является следствием

определения сложности ИГ, а так же очевидного свойства 2–3 дерева (высота дерева $h :] \log_3 |V| [\leqslant h \leqslant [\log_2 |V|])$. Поэтому, чтобы найти нужную запись ИГ вычислит не более чем $[\log_2 N]$ переключателей, а так же один предикат, откуда и следует что $\widehat{T}(U_{id}) \leqslant [\log_2 |V|] + 1$.

Прежде чем перейти к вставке и удалению, разберем отдельно частные случаи (база данных пустая, состоит из 1, 2 или 3 записей).

Во время доказательства в скобках будем указывать номер шага алгоритма **A**, проводя аналогию с этим алгоритмом.

Когда база данных пустая, ИГ U_{id} состоит из одной вершины корень. Если поступил запрос на удаление, то функционирование ДИГ завершается ошибкой. Если поступил запрос $x \in Y$ на **вставку**, то выходом автомата является преобразование $C_1 \in \mathcal{P}_{id}$ (см. рис. 11). На этом рисунке слева направо изображены соответственно простой шаблон $\mathcal{T}_1$, шаблон $\mathcal{T}_2$, ИГ U_1 (ИГ к которому применяем преобразование), и последним на рисунке изображен пример ИГ $U_2{=}C_1(U_1)$.

В дальнейшем все рисунки, изображающие преобразования из $\mathcal{P}_{id}$ будут содержать последовательно те же объекты (слева направо). Так же буква «к» находящаяся рядом с вершиной обозначает корень, вершина обведенная кругом «$\odot$» означает центр рассматриваемой окрестности (текущее положение автомата $\mathcal{A}_{id}$), а так же новое текущее положение автомата $\mathcal{A}_{id}$, которое будет соответствовать компоненте β выходного символа $b \in B$ автомата $\mathcal{A}_{id}$. Граничные вершины на одних и тех же преобразованиях могут отличаться (в зависимости от текущего ИГ U_{id}), самая верхняя вершина будет почти всегда граничной (за исключением начальных графов, когда окрестность радиуса 3 больше графа), а остальные будут интуитивно понятны. В неочевидных случаях, обозначим упорядоченное множество вершин прикрепления соответствующими символами «ex_i», $i \in \{1, \ldots, m\}$, где m количество вершин прикрепления.

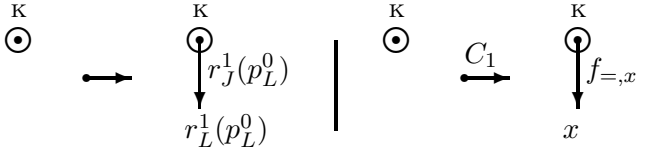

Рис. 11. Преобразование C_1 и его применение.

Преобразование C_2 (см. рис. 12) предназначено для вставки $x \in Y$ ($A_{1.0}$), а преобразование C_3 (см. рис. 13) для удаления $x \in Y$ в ИГ U_{id}, когда количество записей в нем равняется 1 ($A_{2.0}$). В качестве примера ИГ U_2 выбран случай, когда $y < x$. В дальнейшем будем приводить пример ИГ U_2 без пояснений.

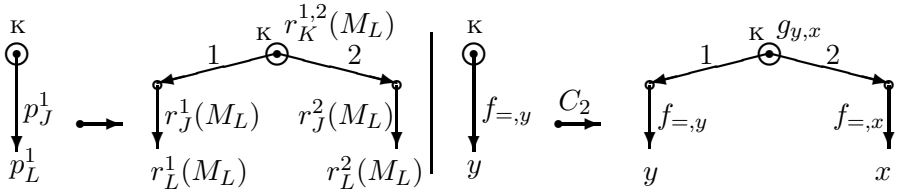

Рис. 12. Преобразование C_2 и его применение.

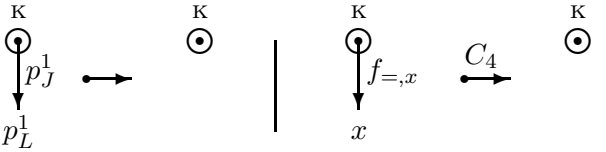

Рис. 13. Преобразование C_3 и его применение.

Преобразование C_e (пример преобразования ИГ 1.3.4, рис. 7–10) предназначено для вставки $x \in Y$, а преобразования C_4^L, C_4^R (см. рис. 14) для удаления $x \in Y$ в ИГ U_{id}, когда количество записей в нем равняется 2 ($A_{2.2.1}$).

Преобразования C_4^L, C_4^R отличаются тем, что удаляя $x \in Y$ мы должно разделить (в случае вставки можно воспользоваться вариационным рядом) одно по сути преобразование на два. Иногда на рисунке будем приводить только одно из преобразований вида C^L, C^M, C^R (буквы означают удаление левого поддерева, среднего и правого поддерева).

Так же преобразования C_4^L, C_4^R будут использованы в случаях большого значения N, поэтому на рисунке не отображено правое и левое поддерево соответственно (при $|V| = 2$ это поддерево есть предикатное ребро и запись), вместо этого возле вершины указано, что она является вершиной прикрепления и имеет номер 1 (ex_1).

В этом и во всех последующих примерах функция ξ соответствия вершин прикрепления из применяемого преобразования будет иметь

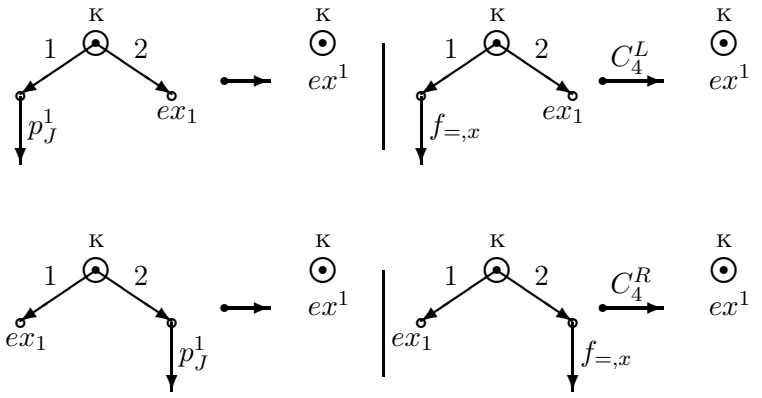

Рис. 14. Преобразования C^L_4, C^R_4 и их применения.

следующий вид $\xi(ex_i) = ex^i$, $i \in \{1, \ldots, m\}$, где m количество вершин прикрепления.

Для **вставки** $x \in Y$, когда ИГ U_{id} содержит 3 записи, потребуется выполнить 2 преобразования. Сначала автомат $\mathcal{A}_{id}$ применяет преобразование C_5 (см. рис. 15), которое необходимо как промежуточное преобразование, после чего автомат применяет преобразование C_6 (см. рис. 16) и завершает функционирование. Чтобы не усложнять рисунки, иллюстрирующие преобразования, все формулы, встречающиеся в шаблоне $\mathcal{T}_2$, будем описывать во время доказательства. В преобразовании C_5 все формулы зависят от M_L, в C_6 все формулы зависят от M_J.

Некоторые преобразования отличаются только в том, что на месте предикатной вершины стоит переключательная, но для сути преобразования это не важно, а важно знать максимальный параметр у этих функций. Изображать важную информацию в примерах ИГ будем символом с информативным параметром. Например возможно ситуация, когда ex_i может быть предикатной вершиной и предикатному ребру (исходящему из этой вершины) будет приписан предикат $f_{=,b} \in F_{id}$, или быть переключательной вершиной и приписан переключатель $g_{a,b} \in G_{id}$. В обоих случаях для сути преобразования нужен только параметр b, поэтому для таких ситуаций на рисунках будем изображать символ «fg_b» рядом с этой вершиной (в формальном описании это два разных преобразования, так как будут использованы разные формулы). В таких ситуациях будем писать два множества

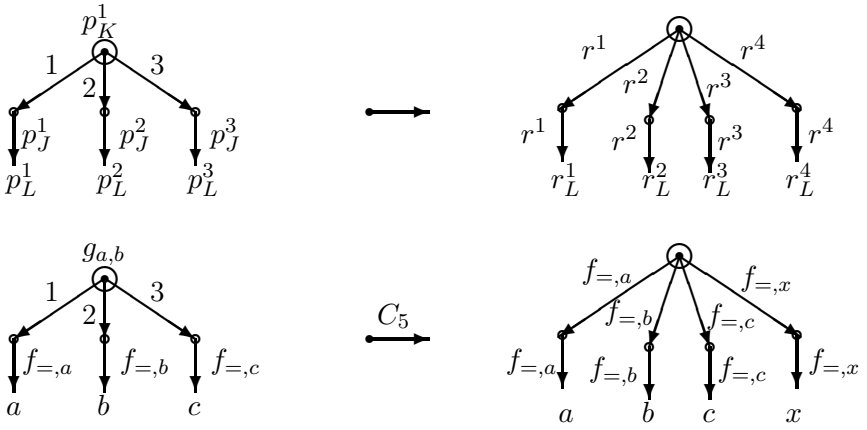

Рис. 15. Преобразование C_5 и его применение.

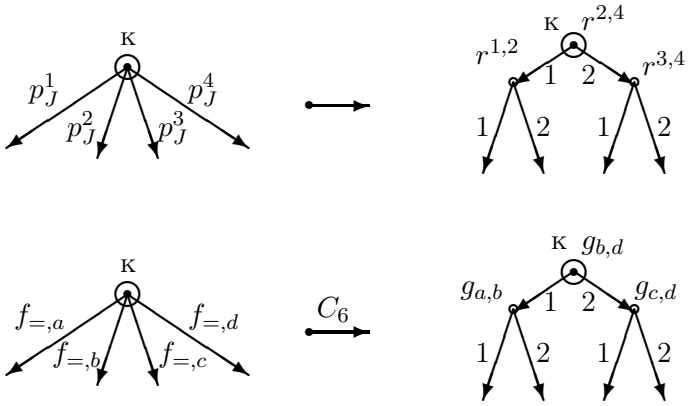

Рис. 16. Преобразование C_6 и его применение.

переменных, от которых зависят формулы (подразумевая при этом два разных преобразования) (см. рис. 17).

Для **удаления** $x \in Y$, когда ИГ U_{id} содержит 3 записи, автомат $\mathcal{A}_{id}$ выполнит 1 преобразование из трех C_7^L, C_7^M или C_8^k (см. рис. 17, 18) в зависимости от входного символа и завершит функционирование. В преобразовании $C_7^L \in \mathcal{P}_{id}$ в простом шаблоне $\mathcal{T}_1$ вершина соответствующая ex_1 может быть предикатной и предикатному ребру, выходящему из нее, приписана переменная p_J^2, или быть переключательной и ей приписана переменная p_K^2. Аналогично для вершины ex_2 могут приписаны в том же смысле переменные p_J^3 или p_K^3. Это раздвоение позволит не загромождать доказательство рисунками, и

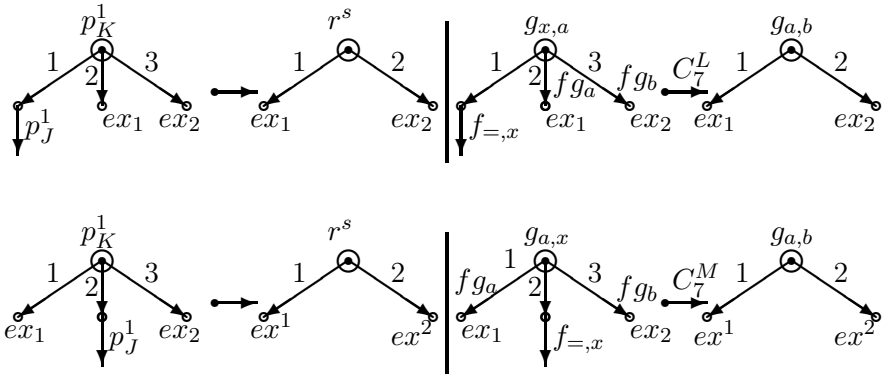

Рис. 17. Преобразования C^L_7, C^M_7 и их применения.

под C^L_7 надо понимать два разных преобразования из $\mathcal{P}_{id}$: $C^L_{7.1}$ — это C^L_7 с предикатными вершинами ex_1,ex_2, $C^L_{7.2}$ — это C^L_7 с переключательными вершинами ϵx_1,ϵx_2.

Формула r^s в преобразовании $C^L_{7.1}$ есть формула $r^{1,2}(p^1_J, p^2_J)$, а в преобразовании $C^L_{7.2}$ есть формула $r^{2,4}(p^2_K, p^3_K)$. В дальнейшем такие ситуации будем коротко описывать $C^L_7(r^s = r^{1,2}(p^1_J, p^2_J)$ или $r^s = r^{2,4}(p^2_K, p^3_K))$. $C^M_7(r^s = r^{1,2}(p^1_J, p^2_J)$ или $r^s = r^{2,4}(p^2_K, p^3_K))$, $C^k_8(r^s = r^{1,2}(p^1_J, p^2_J)$ или $r^s = r^{2,4}(p^2_K, p^3_K))$.

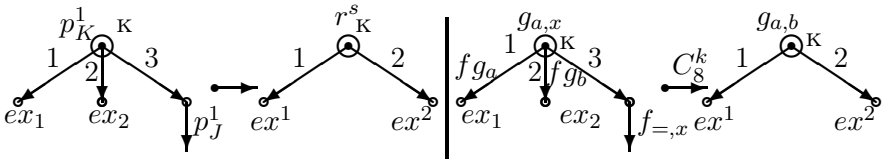

Рис. 18. Преобразование C^k_8 и его применение.

Итак рассмотрим общий случай $|V| \geqslant 4$ (высота ИГ $U_{id} \geqslant 3$).

Вставка $x \in Y$

Опишем алгоритм автомата **P**.

P_0: Автомат $\mathcal{A}_{id}$ находится в состояние поиска (A_0) в котором на выход он выдает тождественное преобразование, а новую текущую вершину выбирает согласно третьей компоненте кода текущей вершины входного символа (k_3). Дойдя до того момента, что расстояние до записи стало равно 2, останавливает процедуру поиска. Чтобы дой-

ти до этого места автомат прошел не более чем по $[\log_2 |V|]$ ребрам графа. Автомат $\mathcal{A}_{id}$ переходит в состояние P_1.

P_1: Если автомат $\mathcal{A}_{id}$ находится в корне ИГ U_{id}, то если корень предикатная вершина и выходит из него 4 предикатных ребра, тогда выходом автомата $\mathcal{A}_{id}$ будет преобразование C_6 $(A_{1.4})$ и завершение функционирования, иначе просто завершение функционирования. Если автомат находится не корне ИГ U_{id}, то в зависимости от входа (код ИГ на запросе $x \in Y$ радиуса 3 с центром в той точке, где находится автомат) на выход он выдает соответствующее преобразование C_e $(A_{1.2})$ или C_5. Если выходом было преобразование C_e, то автомат переходит в состояние P_3 (правки параметров). Если выходом было преобразование C_5, то автомат переходит в состояние P_2.

P_2: В зависимости от входа, автомат на выход выдает одно из преобразований C_9–C_{13} (см. рис. 19–23) $(A_{1.3}$ или $A_{1.4})$. C_9(все формулы зависят от M_J), $C_{10}(r^{1,2}(M_J), r^{3,4}(M_J)$, $r^s = r^{2,3}(p^2_K, p^2_J))$, $C_{11}(r^{1,2}, r^{3,4}, r^2, r^4$ зависят от M_J, $r^{s_1} = r^{2,4}(p^2_K, p^3_K))$, $C_{12}(r^{1,2}, r^{3,4}, r^2, r^4$ зависят от M_J, $r^{s_1} = r^2(p^2_K), r^{s_2} = r^2(p^3_K))$, $C_{13}(r^{1,2}, r^{3,4}, r^2, r^4$ зависят от M_J, $r^{s_1} = r^2(p^2_K), r^{s_2} = r^2(p^3_K))$.

Если выходом было преобразование C_9 или C_{10}, то автомат переходит в состояние P_3.

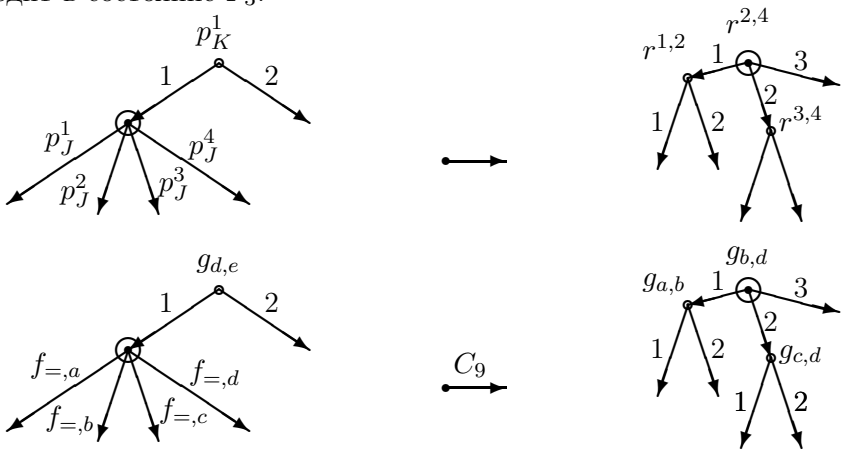

Рис. 19. Преобразование C_9.

Если выходом было одно из преобразований C_{11}, C_{12} или C_{13}, то автомат переходит в состояние P_1.

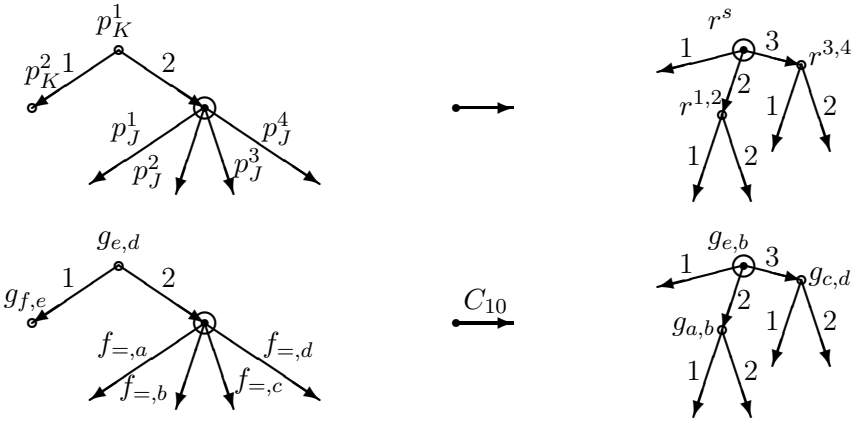

Рис. 20. Преобразование C_{10}.

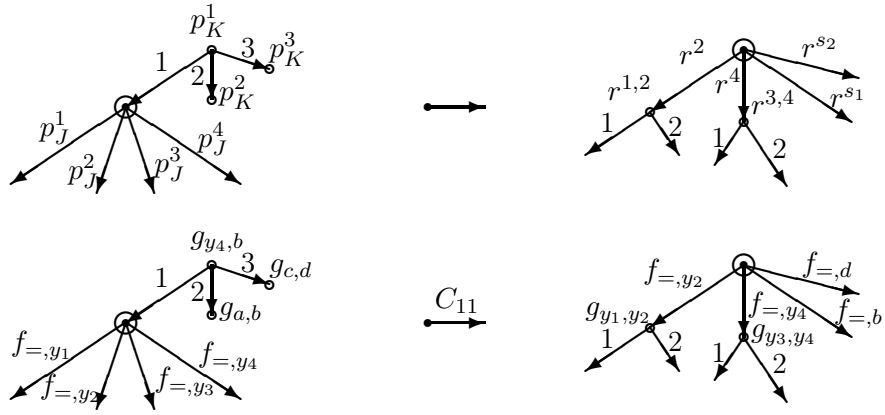

Рис. 21. Преобразование C_{11}.

P_3: В случае правки параметров структура самого ИГ U_{id} больше не измениться, могут измениться только параметры в переключателях.

Если автомат находится в корне ИГ U_{id}, то функционирование прекращается. Иначе в зависимости от входа (важно по какому ребру он перейдет к родителю текущей вершины), выходом будет одно из преобразований C_{14}^1, C_{14}^2 или C_{14}^3, и переходит в состояние P_3. $C_{14}^1(r^s =$

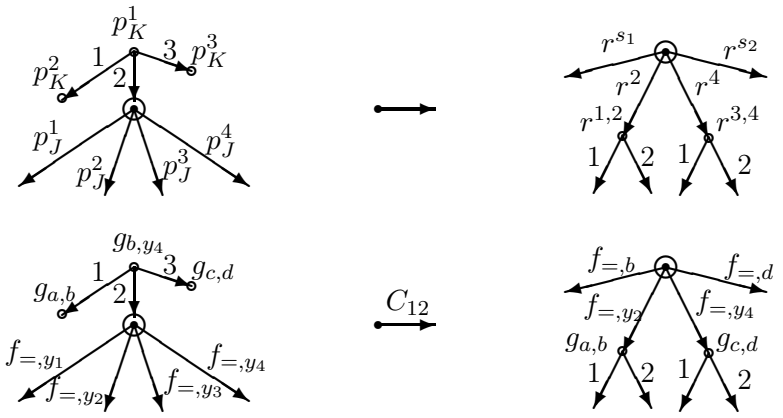

Рис. 22. Преобразование C_{12}.

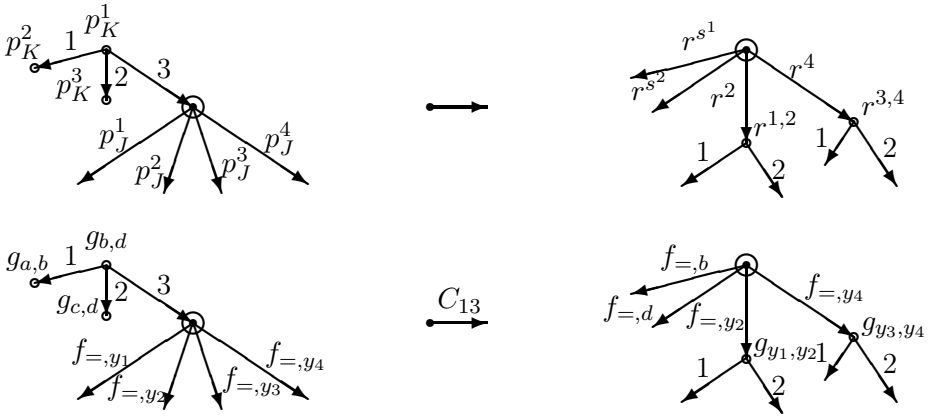

Рис. 23. Преобразование C_{13}.

$r^{1,2}(r^2(r^1(p_K^1), p_L^0), r^2(p_K^1)))$, $C_{14}^2(r^{s_1} = r^{1,2}(r^2(r^2(p_K^1), p_L^0), r^1(p_K^1)))$, $C_{14}^3(r^{s_2} = r^{1,2}(p_K^1))$.

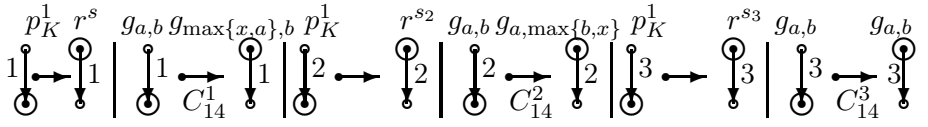

Рис. 24. Преобразования C_{14}^1, C_{14}^2, C_{14}^3 и их применения.

Очевидно, что сложность обратного прохода (P_1–P_3) можно оценить максимальной сложностью из всех преобразований умноженную на высоту дерева, так как после каждого преобразования текущее положение автомата уменьшает расстояние до корня как минимум на 1. Поэтому $\widehat{T}_{in}(\mathcal{D}_{id}) \leqslant (\lambda_{max}+1)([\log_2|V|]+1)$.

При этом несложно убедиться непосредственной проверкой, что каждый переключатель до и после преобразования сохранял свойство переключателей ИГ U_{id}: первый параметр переключателя обозначает максимальную запись в левом поддереве, а второй в среднем (правом если у переключателя двое сыновей). Поэтому полученный после завершения функционирования ДИГ $\mathcal{D}_{id} = (\mathcal{A}_{id}, U_{id})$, ДИГ $\mathcal{D}_{id} = (\mathcal{A}_{id}, U'_{id})$ будет решать ЗПИО $\langle X, V \cup x, = \rangle$.

Удаление $x \in Y$

Опишем алгоритм автомата **D**.

D_0:

Аналогично P_0 автомат $\mathcal{A}_{id}$ переходит в состояние поиска (A_0). Дойдя до листовой вершины (до удаляемой записи x), останавливает процедуру поиска и следующее выходное действие подняться к родителю текущей вершины. Чтобы дойти до этого места автомат прошел не более чем по $[\log_2|V|]+1$ ребрам графа. Если он не дошел до такой вершины, то функционирование завершается с ошибкой. После чего автомат переходит в состояние D_1.

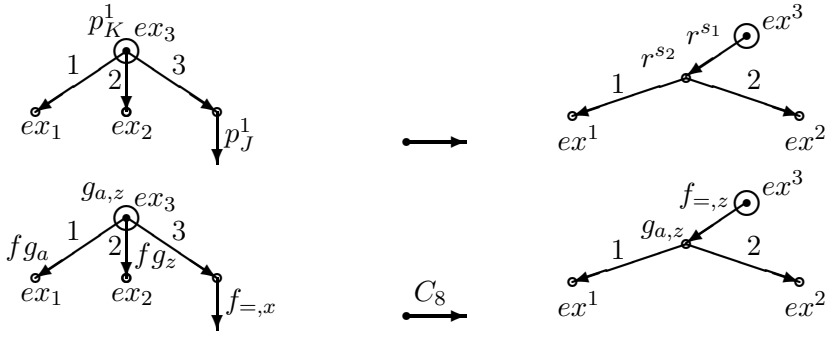

Рис. 25. Преобразование C_8 и его применение.

D_1:

Автомат $\mathcal{A}_{id}$ поднимается к родителю текущей вершины. Если у родителя текущей вершины выходит 3 ребра ($A_{2.1}$), то выходом ав-

томата $\mathcal{A}_{id}$ будет применить одно из преобразований C_7^L,C_7^M,C_8, иначе автомат $\mathcal{A}_{id}$ переходит в состояние D_2. $C_8(r^{s_2} = r^{1,2}(p_J^2, p_J^3)$ или $r^{s_2} = r^{2,4}(p_K^2, p_K^3)$, $r^{s_1} = r^1(p_J^3)$ или $r^{s_1} = r^2(p_K^3))$, где p_J^2, p_K^2 — соответствуют ex_1, p_J^3, p_K^3 — соответствуют ex_2 (здесь описано два разных преобразования). После этого автомат $\mathcal{A}_{id}$ переходит в состояние правки параметров D_5.

D_2:

Автомат $\mathcal{A}_{id}$ поднимается к родителю текущей вершины. Если текущая вершина корень, то применяется в зависимости от входа одно из преобразований C_4^L, C_4^R, C_7^L, C_7^M, C_8^k, после чего алгоритм заканчивает работу ($A_{2.2.1}$). Иначе переходит к D_3.

D_3:

Если у текущей вершины есть соседний брат с тремя сыновьями, то автомат $\mathcal{A}_{id}$ применяет одно из преобразований C_{14}^L,C_{14}^R,C_{15}^L,C_{15}^R (см. рис. 26, 27) в зависимости от входа, иначе он переходит в состояние D_4. C_{14}^R — это 4 разных преобразования в зависимости от (i, j) (при $(i, j) = (1, 2)$ предполагается, что из верхнего переключателя выходит 2 ребра). $C_{14}^R(r^{s_1} = r^{1,2}(p_K^2); r^{s_2} = r^{1,2}(p_J^2, p_K^2)$ или $r^{s_2} = r^{2,3}(p_K^4, p_K^2); r^{s_{1,2}} = r^{1,4}(p_K^3, p_K^2), r^{s_{2,3}} = r^{1,2}(p_K^3); r^s = r^2(p_K^2))$, где p_J^2, p_K^4 соответствуют вершине ex_3 (два разных преобразования). C_{14}^L отличается от C_{14}^R тем, что предикатное ребро $f_{=,x}$ выходит из ex_3. В C_{14}^L нет дополнительного предикатного ребра r^s, а смысл всех формул сделать все переключатели правильными после переброски брата, удаляемой предикатной вершины (под правильным понимаем свойство параметров переключателя содержать информацию о максимальной записи в поддереве). Преобразования C_{15}^L, C_{15}^R аналогичны преобразованию C_{14}^L.

При этом дополнительное предикатное ребро участвуют в новом ИГ (см. рис. 26) только в случае, когда удаляемая запись была максимальна в этом (удаляемом) графе, и с помощью этого предикатного ребра хранится информация о том, на что нужно менять x во время правки параметров.

D_4:

Автомат $\mathcal{A}_{id}$ в зависимости от входа применяет одно из преобразований C_{16}^L,C_{16}^R,C_{17}^L,C_{17}^R. После чего переходит в состояние D_1 ($A_{2.3}$). При этом в этих преобразованиях остается предикатное ребро $f_{=,x}$, чтобы в последующем можно было применить уже введенные преобразования. C_{16}^R отличается от C_{16}^L тем, что предикатное ребро p_J^1

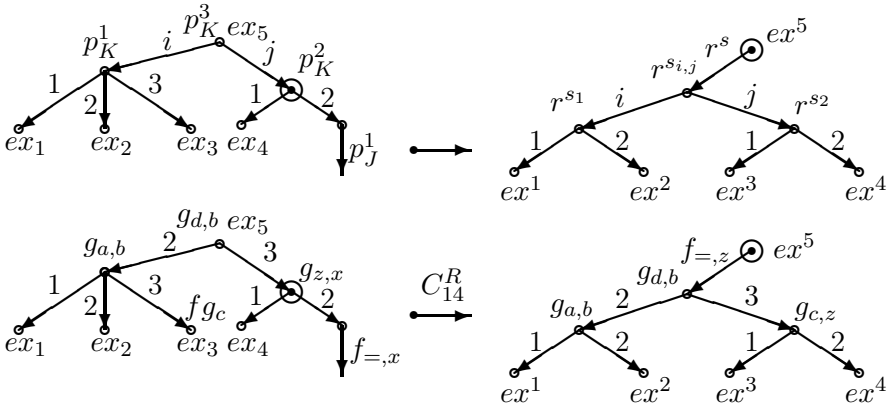

Рис. 26. Преобразование C_{14}^R и его применение: $(i,j) \in \{(1,2),(2,3)\}$.

выходит из ex_3; C_{17}^L отличается от C_{17}^R тем, что предикатное ребро p_J^1 выходит из ex_1. В $C_{16}^L(r^{s_1} = r^{1,2}(p_K^1);\ r^{s_{1,2}} = r^{2,3}(p_K^3, p_L^0);\ r^{s_{2,3}} = r^{1,3}(p_K^3, p_K^2))$. В $C_{17}^R(r^{s_1} = r^{1,3}(p_K^1, p_K^2);\ r^{s_{1,2}} = r^{2,1}(p_K^2);\ r^{s_{2,3}} = r^{1,4}(p_K^3, p_K^2))$. В $r^{s_{1,2}}$ из C_{17}^R нарушен порядок параметров (второй параметр меньше первого, это сделано для сохранения свойства переключателей в ИГ: содержание информации о максимальной записи в соответствующем поддереве).

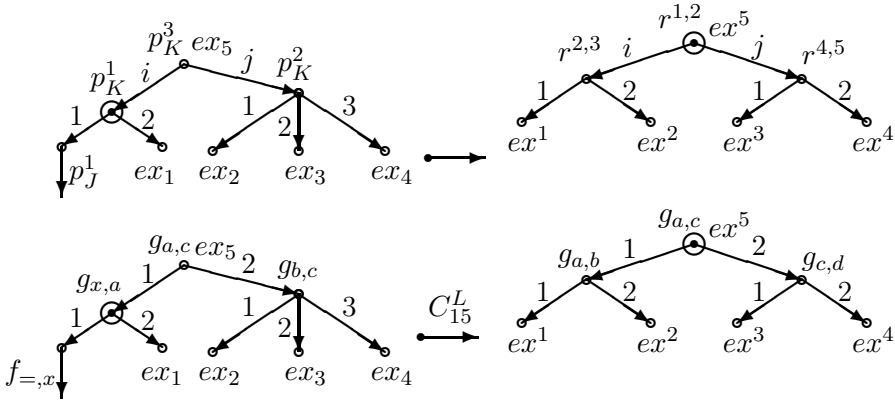

Рис. 27. Преобразование C_{15}^L и его применение: $(i,j) \in \{(1,2),(2,3)\}$.

D_5:

Если родитель текущей вершины корень, то применяется преобразование C_{19} (см. рис. 28) и автомат $\mathcal{A}_{id}$ завершает функционирование, иначе автомат $\mathcal{A}_{id}$ переходит к состоянию D_6. На рис. 28 изображены все случаи ИГ U_2 в зависимости от значений формул. В C_{18} и C_{19} ($r_{sp}(p_L^0, p_J^1, p_K^1)$ определяется соотношением (6)).

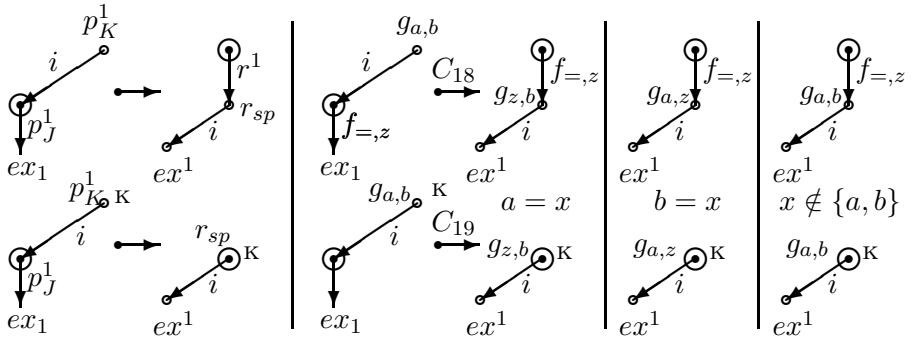

Рис. 28. Преобразования C_{18}, C_{19} и их применения: $i \in \{1, 2, 3\}$.

D_6:

Автомат $\mathcal{A}_{id}$ на выходе выдает преобразование C_{18} и переходит к состоянию D_5.

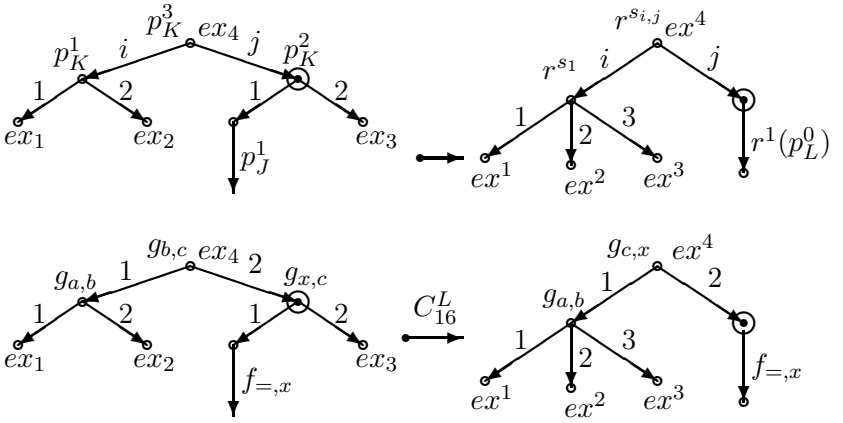

Рис. 29. Преобразование C_{16}^L и его применение: $(i, j) \in \{(1, 2), (2, 3)\}$.

Алгоритм **D** повторяет рассуждения доказательства работы алгоритма **A** для удаления [3]. За исключением D_5 и D_6, которые с

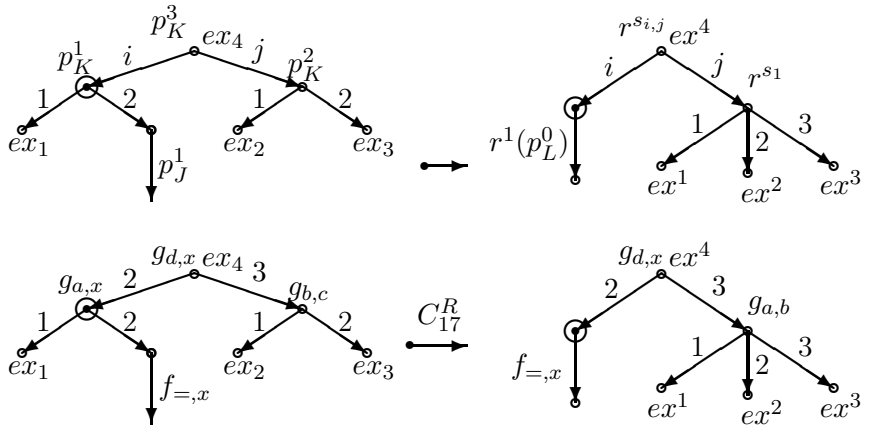

Рис. 30. Преобразование C_{17}^R и его применение: $(i,j) \in \{(1,2),(2,3)\}$.

помощью преобразований C_{18} и C_{19} (информация о заменяемых параметров передается в дополнительных предикатах) правильно переставляют параметры переключателей после того как удалили запись. В этом не сложно убедиться, если учесть, что если встречался параметр x, то он должен будет заменен на переменную z (максимальный оставшийся брат x) во всех предикатах ведущих по родителям удаляемой вершины к корню. Этот параметр z передается с помощью дополнительного предикатного ребра, которое в итоге удаляется, когда автомат доходит до корня.

Очевидно, что сложность обратного прохода можно оценить максимальной сложностью из всех преобразований умноженную на высоту дерева. Поэтому $\widehat{T}_{out}(\mathcal{D}_{id}) \leqslant (\lambda_{max}+1)([\log_2|V|]+1)$, откуда и следует доказательство теоремы, так как несложно заметить, что $Q(\mathcal{D}_{id}) \leqslant 3|V|$.

Список литературы

[1] Гасанов Э. Э., Кудрявцев В. Б. Теория хранения и поиска информации. — М.: ФИЗМАТЛИТ, 2002.

[2] Кудрявцев В. Б., Алешин С. В., Подколзин А. С. Введение в теорию автоматов. — М.: Наука, 1985.

[3] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. — М., 1979.