Нижняя оценка на область видимости автомата, обрабатывающего произвольный поток запросов к динамической базе данных

А. А. Плетнев

Рассматривается динамическая задача поиска идентичных объектов (ДЗПИО). В работах [1, 2, 3] автор показал, как эту задачу можно решать с помощью многоавтоматного динамического информационного графа (МДИГ) для любого потока запросов. В данной работе показано, что не существует конечного МДИГ с радиусом видимости один, и степенью ветвления два, обрабатывающий произвольный поток запросов.

Ключевые слова: динамические базы данных, поиск по ключу, информационный граф, автомат, потоки запросов.

Введение

В современном мире нас окружает огромное количество информации, поэтому нахождение среди нее необходимых данных является особенно актуальной задачей. Математический интерес к данной проблеме заключается в описании модели, с помощью которой можно получать необходимые алгоритмы поиска искомой информации. К решению этой задачи подходят с различных сторон. Огромный вклад в моделировании баз данных внес Гасанов Эльяр Эльдарович [4]. Он предложил

информационно-графовый подход. В настоящее время этот подход используют многие ученые. Одни из последних работ по этой тематике можно найти в [5, 6, 7, 8].

В данной работе будет рассмотрена постановка задачи, совпадающая с описанной в [2]. Она занимает большой объем, поэтому здесь опишем только ее смысл, а точную формулировку можно прочитать в [2].

Рассматривается поток запросов. Под потоком запросов будем понимать бесконечную последовательность запросов на поиск, вставку и удаление к базе данных, которые поступают через равные промежутки времени — такты. При этом в один такт может поступить не более одного запроса. Также считаем, что за один такт может быть выполнено любое локальное преобразование структуры базы данных, предназначенное для поддержания базы данных в актуальном состоянии. Другими словами, каждый процесс, который перестраивает или ищет запрос, может сделать за один такт только одно преобразование над базой данных, при этом на следующий такт к базе данных может поступить уже новый запрос. Ограничения на разные процессы состоит в том, что они не могут изменять одновременно один участок памяти, но при этом могут считывать такие участки. Такие структуры данных называют ОЧЭЗ (одновременное чтение эксклюзивная запись) структурами. В иностранной литературе они описываются как CREW структуры [9].

Будем считать, что у нас есть неограниченное количество доступных процессов. Каждый процесс обрабатывает один запрос, который может быть одним из трех типов: поиск, вставка или удаление. В работах [1, 2, 3] были предложены ОЧЭЗ структуры данных, решающие ДЗПИО. Идея построения искомой структуры данных основывается на информационно-графовом подходе [4] с использованием конечного автомата [10, 11, 12, 13, 14, 15, 16, 17]. Такая структура данных названа многоавтоматным динамическим информационным графом (МДИГ), и она зависит от нескольких параметров. В данной работе будет доказана нижняя оценка на параметры, при кото-

рой не существует алгоритма для решения ДЗПИО с помощью МДИГ.

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

Постановка задачи и результаты

Пусть H — поток запросов, $H : \mathbb{N} \to X$, где $X = \{\text{П}, \text{В}, \text{У}, \Lambda\} \times \mathbb{N}$ — множество запросов. Буква в запросе означает его действие: поиск (П), вставка (В), удаление (У) или пустой запрос (Λ).

Множество всех потоков запросов обозначим через $\mathcal{H}$. Рассмотрим функцию множества записей $V : \{\mathbb{N} \cup \{0\}\} \times \mathcal{H} \to 2^{\mathbb{N}}$, которая удовлетворяет следующим условиям:

$$V(i, H) = \begin{cases} \emptyset, & \text{если } i = 0; \\ V(i-1, H), & \text{если } i > 0 \text{ и } H(i) \text{ запрос на поиск} \\ & \text{или пустой запрос;} \\ V(i-1, H) \cup \{x\}, & \text{если } i > 0 \text{ и } H(i) = (\text{В}, x); \\ V(i-1, H) \setminus \{x\}, & \text{если } i > 0 \text{ и } H(i) = (\text{У}, x). \end{cases}$$

Функция V составляет каждому такту i и потоку запросов H — множество записей, которые образуют базу данных после завершения функционирования МДИГ для всех запросов до такта i включительно, если обработка любого запроса происходила бы мгновенно (за 1 такт).

Будем рассматривать две различные постановки ДЗПИО: с выдачей ответа, когда ответом на запрос типа $(\text{П}, x)$ будет $\emptyset$ или $\{x\}$, и логическая — когда алгоритм поиска отвечает на вопрос о том, есть ли искомая запись в базе данных или нет.

Пусть дана функция $L : \mathbb{N} \to \mathbb{N}$. Будем говорить, что МДИГ решает ДЗПИО с выдачей ответа (логическую ДЗПИО) со сложностью L, если для любого потока H и любого такта i выполняются следующие условия

- если $H(i) = (\text{П}, x)$, то результатом функционирования МДИГ должен быть $\{x\}$ ("да"), если $x \in V(i, H)$ и пустое множество ("нет") в противном случае. При этом количество тактов, необходимое на выдачу ответа, должно не превосходить $L(|V(i, H)|)$;

- если $H(i) = (\text{В}, x)$, то для любого запроса на поиск (П, z), поступившего в такт $i + 1$, результат функционирования МДИГ должен быть $\{z\}$ ("да"), если $z \in V(i + 1, H) = V(i, H) \cup \{x\}$, и пустое множество ("нет") в противном случае;

- если $H(i) = (\text{У}, x)$, то для любого запроса на поиск (П,z), поступившего в такт $i + 1$, результат функционирования МДИГ должен быть $\{z\}$ ("да"), если $z \in V(i + 1, H) = V(i, H) \setminus \{x\}$, и пустое множество ("нет") в противном случае.

Очевидно, что если МДИГ решает ДЗПИО с выдачей ответа, то он решает ее и в логической постановке, т.е. когда ответом является "да"или "нет". Также не сложно заметить, что, если МДИГ не решает задачу в логической постановке, то он не решает ее с выдачей ответа.

Напомним, что означает базовое множество для МДИГ. Множество предикатов F – это множество функций, которые являются нагрузками ребер ИГ. МДИГ базируется на конечном автомате, поэтому сами функции, приписанные ребрам, автомат увидеть не может. Автомат во время функционирования видит значение этих функций на запросе, который он обслуживает, а не сами функции. Также, помимо значений функций на запросе, автомат видит структуру подграфа ИГ. Подграф, который видит автомат, ограничен фиксированным параметром R, который означает радиус окрестности с центром в вершине, в которой находится автомат. Другими словами, если автомат находится в вершине v, то он видит все вершины, которые находятся на расстояния не большем, чем R ребер от нее. Пример радиуса видимости автомата изображен на рисунке 1. Естественно, сам ИГ не может иметь бесконечное ветвление, так как конечный

автомат даже не сможет увидеть окрестность радиуса 1 в этом случае. Поэтому, МДИГ подразумевает, что ИГ имеет ограничение на ветвление (максимальное количество ребер инцидентных вершине), которые зависит от параметра N. На рисунке 1, N равняется 3 и достигается в вершине, в которой стоит автомат. Множество преобразований $\mathcal{R}$ означает, что автомат в каждый такт своего функционирования может применить преобразование из этого множества. Цель преобразований – это перемещение по ИГ и его локальная модификация для поддержания базы данных в актуальном состоянии.

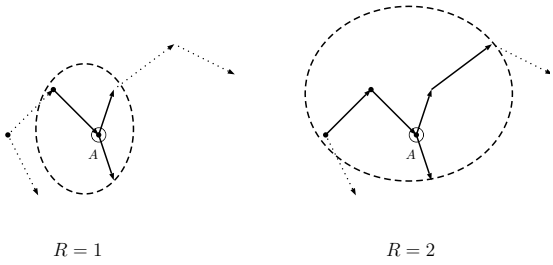

Рис. 1: Пример радиуса видимости автомата.

Формальное описание преобразования достаточно громоздкое, поэтому опишем его суть. Автомат, находясь в вершине ИГ, на вход получает код окрестности на запросе, который он обслуживает. Удобнее представлять код как граф вершинам и ребрам, которого присвоена конечная информация. Вершинам присвоены их тип: корень, внутренняя вершина или листовая. Ребрам присвоена пара – это тип предиката и его значение на запросе. Автомат на выход выдает инструкцию ИГ, на какую окрестность нужно заменить ту, которую он видит. При этом он может использовать конечный набор функций $\{\phi_1, \ldots, \phi_n\}$, которые могут создавать новые предикаты.

На рисунках, изображающих преобразования, вершину, в которой находится автомат, обозначаем через вершину в круге. Автоматы не знают значение запроса, которые они обслуживают. Поэтому запрос, который обслуживает автомат, будем обозначать $*$. На каждый такт своего функционирования автомат

на вход получает код ИГ на запросе с центром в текущей вершине и радиусом его видимости. При этом, если в правой части преобразования нет вершины, обведенной кругом, автомат завершает функционирование. На рисунках писать букву "к" для обозначения, что вершина является корнем.

Более подробно о преобразованиях и о том, как именно автомат видит окрестность, можно прочитать в [2].

Будем говорить, что МДИГ типа (N, R), если степень ветвления любой вершины ИГ не больше, чем N, а радиус видимости автомата не больше, чем R.

Рассмотрим множество предикатов

$$F(T) = \{f^0, f^1, f_a^t(x) : t \in T, a \in \mathbb{N}\}, \tag{1}$$

где $f^0(x) \equiv 0$, $f^1(x) \equiv 1$, $|T| < \infty$ и для любых $t \in T, a \in \mathbb{N}$ выполняется

$$f_a^t(x) = \begin{cases} 1, & \text{если } x = a, \\ 0, & \text{иначе.} \end{cases} \tag{2}$$

Введем понятие *конечный* МДИГ. Будем говорить, что МДИГ конечный, если для произвольного потока запросов H и произвольного такта времени i существует такая константа C,$C < \infty$, что для потока запросов $H^{'}, H^{'}(1) = H(1), \ldots H^{'}(i) = H(i), H^{'}(i+1) = \Lambda, \ldots, H^{'}(i+C) = \Lambda$, автомат обслуживающий запрос $H(i)$ завершит свое функционирование.

Неформально говоря, МДИГ будем называть конечным, если любой автомат, обслуживающий некоторый запрос, функционирует конечное время. МДИГ, который не является конечным, будем называть бесконечным.

Пусть

$$T_0 = \{\text{П}, \text{В}_1, \text{В}_2, \text{У}\},$$

то есть $F(T_0) = \{f^0, f^1, f_a^{\text{П}}, f_a^{\text{В}_1}, f_a^{\text{В}_2}, f_a^{\text{У}} : a \in \mathbb{N}\}$. Введем базовое множество преобразований $\mathcal{R}$. Базовое множество $\mathcal{R}$ состоит из 12 преобразований, изображенных на рисунках 4 - 15. Преобразования будут описаны в доказательстве теоремы 1.

В качестве базового множества для МДИГ будем рассматривать множество

$$\mathcal{F}_0 = \langle F(T_0), \mathcal{R} \rangle. \tag{3}$$

Справедлива следующая теорема.

Теорема 1. *Существует бесконечный МДИГ $\mathcal{D}_M = (\mathcal{A}, U)$ типа (2,1) над базовым множеством $\mathcal{F}_0$, определяемым соотношением (3), который решает ДЗПИО для любого потока запросов H из $\mathcal{H}$ со сложностью $L \equiv 3$.*

Замечание к теореме 1. Можно предъявить бесконечный МДИГ типа (1,1) который решает ДЗПИО со сложностью $L, L \equiv 2$. По сути он будет схож с МДИГ из теоремы 1, но для понимания основной идеи в данной работе будет представлен более понятный алгоритм.

Введем понятие *селекторный МДИГ*. Для этого напомним, что представляет из себя преобразование, которое применяет автомат. Рассмотрим произвольную функцию $\phi \in \{\phi_1, \ldots, \phi_n\}$, которую может применить автомат в своем преобразовании. Автомат, зная тип предиката, знает также, сколько у него аргументов. Функция ϕ на вход получает аргументы и типы предикатов, типы вершин, а так же запись, которую обслуживает автомат. После этого эта функция может создать новый предикат или запись. Естественно, функция ϕ может создать только предикат, принадлежащий множеству предикатов F, над которым рассматривается ИГ. Оперируя этими функциями, автомат может в новой окрестности создавать новые предикаты из F.

Например, рассмотрим ИГ над базовым множеством $F = F(T_0)$, изображенный на рисунке 2. Допустим, что автомат может использовать функцию $\phi(a, b, t_1, t_2, x) : \mathbb{N} \times \mathbb{N} \times T_0 \times T_0 \times \mathbb{N} \to F$ для преобразования предикатов. Рассмотрим пример функции ϕ. Пусть $\phi(a, b, t_1, t_2, x) = f^{\phi_T(a,b,t_1,t_2)}_{\phi_A(a,b,x)}$, где $\phi_T(a, b, t_1, t_2) =$ У, если $a = b$ и t_2 иначе, $\phi_A(a, b, x) = a + b - x * (a + b)$. Видно, что, используя функцию ϕ, автомат создает новый предикат из F, опираясь на аргументы предикатов. При этом сами аргументы автомат не знает. Используя эту функцию, автомат может

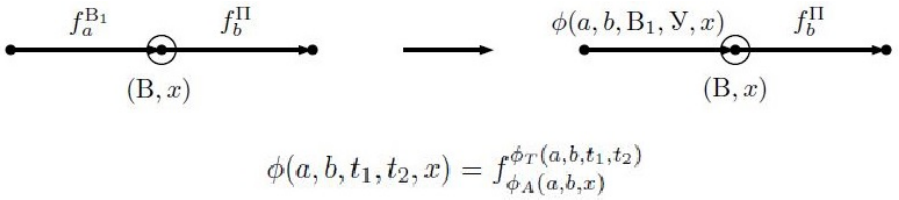

Рис. 2: Пример преобразования ИГ, которое использует функцию ϕ для создания нового предиката.

сказать ИГ сделать новую окрестность взамен той, которую он видит.

Преобразование будем называть селекторным, если оно не может создать новый аргумент, кроме тех, что есть во входной окрестности или запроса, который обслуживает автомат. Например, преобразование, описанное выше, не является селекторным, так как оно создает новый аргумент $\phi_A(a, b, x) = a + b - x * (a + b)$, который может не принадлежать множеству $\{a, b, x\}$. Преобразование $\phi(a, b, t_1, t_2, x) = f^{\phi_T(a,b,t_1,t_2)}_{\phi_A(a,b,x)}$,где $\phi_A(a, b, x) = x$, является селекторным. Другой пример не селекторного преобразования изображен на рисунке 21. Пример селекторного преобразования изображен на рисунке 4.

МДИГ будем называть селекторным, если преобразования, которые он использует, являются селекторными.

Также будем говорить, что не существует МДИГ над базовым множеством F, который решает ДЗПИО для любого потока H, подразумевая, что не существует базового множества преобразований $\mathcal{R}$ и ИГ над F, для которого МДИГ решает ДЗПИО.

Справедлива следующая теорема.

Теорема 2. *Для любой функции $L, L : \mathbb{N} \to \mathbb{N}$, и для любого множества T, $|T| < \infty$, не существует конечного, селекторного МДИГ типа (2, 1) над множеством $F(T)$, задаваемыми соотношениями (1–2), который решает логическую ДЗПИО со сложностью L.*

Обратим внимание, что в теореме 2 автомат конечный и селекторный. Как видно из теоремы 1, если выбрать функцию $L \equiv 3$, а в качестве T множество $T_0 = \{\text{П}, \text{В}_1, \text{В}_2, \text{У}\}$, то такой бесконечный МДИГ типа (2,1) существует.

Следующая теорема утверждает, что существует конечный не селекторный МДИГ типа (2,1), который решает ДЗПИО с ответом да или нет.

Пусть

$$T_1 = \{\text{да}, \text{нет}\},$$

то есть $F(T_1) = \{f^0, f^1, f_a^{\text{да}}, f_a^{\text{нет}} : a \in \mathbb{N}\}$. Введем базовое множества преобразований $\mathcal{R}_1$, которое состоит из 4 преобразований, изображенных на рисунке 21. Преобразования будут описаны в доказательстве теоремы 3.

В качестве базового множества для МДИГ будем рассматривать множество

$$\mathcal{F}_1 = \langle F(T_1), \mathcal{R}_1 \rangle. \tag{4}$$

Теорема 3. *Существует конечный, не селекторный МДИГ $\mathcal{D}_M = (\mathcal{A}, U)$ типа (1,1) над базовым множеством $\mathcal{F}_1$, определяемым соотношением (4), решающий логическую ДЗПИО со сложностью $L \equiv 2$.*

Доказательство теоремы 1

Теорема 1 утверждает, что существует бесконечный МДИГ типа (2,1), решающий ДЗПИО для любого потока запросов. Причем время поиска любого запроса всегда равно трем тактам.

Идея построения МДИГ будет следующей. Типы предикатов П,В,У будут означать, что их создал автомат на поиск, вставку или удаление. Значение индексов 1 и 2 у вставки В_1 и В_2 объясним ниже.

ИГ будет состоять из двух ребер. Пример такого ИГ изображен на рисунке 3. В ИГ всего три вершины. Корень будем

называть первой вершиной, соседнюю с корнем второй, а оставшуюся третьей. Ребро, выходящее из корня, будем называть первым ребром, а другое вторым.

В первый такт своего функционирования любой автомат меняет предикат, выходящий из корня, на новый, у которого тип совпадает с типом запроса, который он обслуживает, а аргумент равен самому запросу. Например, если поступил запрос на поиск 7, то на следующий такт из корня будет выходить ребро с предикатом f_7^{Π}. Исключение составляет автомат на вставку. Если тип предиката, выходящего из корня был B_1, то после преобразования автомат поменяет тип на B_2, а в других случаях он поменяет тип на B_1. Смысл данного преобразования будет разъяснен ниже.

Опишем алгоритм работы автоматов на удаление и поиск. Автомат на удаление после первого такта завершает функционирование. Автомат на поиск переходит во вторую вершину. На второй такт автомат на поиск переходит в третью вершину. В третий такт он смотрит значение предиката на ребре. Если оно равно единице, то в качестве ответа он выдает запись, приписанную в третьей вершине и завершает функционирование. Если значение предиката равно 0, то он выдает ответ, что записи нет, после чего завершает функционирование. Видно, что автомату на поиск при таком алгоритме нужно три такта, чтобы выдать ответ.

Итак, выше описан алгоритм работы автоматов на удаление и на поиск, осталось описать алгоритм автомата на вставку. Автомат на вставку в отличие от автомата на поиск и удаление может функционировать бесконечно долго.

Основная идея состоит в том, что все автоматы на вставку, начиная со второго такта, находятся во второй вершине и смотрят на предикат, выходящий из корня. При этом двух автоматов на вставку одного и того же элемента не будет. Как это можно сделать будет описано ниже. Все автоматы на вставку смотрят значение предиката f_a^t первого ребра. Если для какого-то автомата это значение равно 1, то это значит, что этот автомат обслуживает запись a. Возможно пять вариантов: предикат

первого ребра поменялся на $f_a^{\text{П}}$, $f_a^{\text{У}}$, $f_a^{\text{В}_1}$, $f_a^{\text{В}_2}$ или предикат не поменялся.

Рассмотрим первый вариант, когда предикат поменялся на $f_a^{\text{П}}$. Если среди автоматов на вставку, находящихся во второй вершине, есть автомат на вставку a, то этот автомат на следующий такт поменяет предикат второго ребра на $f_a^{\text{П}}$ и поместит a в третью вершину. Отметим еще раз, что автомат понимает, что его запись совпадает с запись автомата на поиск, если значение предиката на первом ребре будет равно одному.

Второй случай, когда предикат поменялся на $f_a^{\text{У}}$. Если среди автоматов на вставку, находящихся во второй вершине, есть автомат на вставку a, то этот автомат на следующий такт завершает свое функционирование, меняет предикат второго ребра на f^0.

Третий и четвертый случаи одинаковые. В первый такт своего функционирования автомат держит внутренний параметр, который равен 0, а также запоминает тип предиката, который он создаст (В_1 или В_2). Начиная со второго такта, он смотрит на тип предиката, выходящего из корня ребра. Если он поменялся, то автомат меняет свой внутренний параметр на 1. Если кроме этого тип предиката поменялся на вставку с другим индексом, и записи совпали, то автомат завершает свое функционирование (его место займет другой автомат с той же записью). Аналогично, если внутренний параметр автомата на вставку уже был равен 1, и он увидел, что пришел новый автомат на вставку той же записи, то этот автомат так же завершает свое функционирование.

Итак, выше был описан алгоритм работы МДИГ. Как видно из данного алгоритма, нагрузку первого ребра меняет только тот автомат, который находится в корне. Нагрузку второго ребра меняет только один автомат на вставку. Поэтому МДИГ работает бесконфликтно.

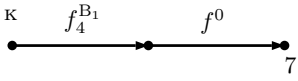

Рис. 3: Пример ИГ, участвующий в доказательстве теоремы 1.

Для формального доказательства осталось предъявить базовое множество преобразований $\mathcal{R}$,

$$\mathcal{R} = \{R_1, R_2, \ldots, R_{12}\},$$

которые будет применять автомат $\mathcal{A}$ в зависимости от входной окрестности, в соответствии с описанным выше алгоритмом. А так же доказать корректность работы МДИГ.

Для простоты объяснения будем считать, что до поступления запросов ИГ имел вид, изображенный на рисунке 3.

Все преобразования можно распределить на три группы: поиск, вставка и удаление. Рассмотрим сначала преобразования, которые делают автоматы на вставку, обрабатывая свои запросы.

В данной теореме нам не потребуется информация о типе вершины, поэтому ее не будем изображать на рисунках. Код ребра ИГ это пара (тип предиката, значение предиката на запросе). Например, рассмотрим предикат f_7^{Π} и автомат, обслуживающий запрос на вставку 6. Тогда код, который он увидит на этом ребре будет $(\Pi, 0)$. Если бы он обслуживал запрос 7, то он увидел бы код $(\Pi, 1)$.

Итак, разберем преобразования на вставку. Преобразование R_1 – это перемещение автомата на вставку в соседнюю с корнем вершину и изменение предиката ребра. Это преобразование изображено на рисунке 4. Пунктиром для наглядности обозначено ребро, предикат которого изменяется в результате преобразования. Автомат на вставку делает данное преобразование в случае, если код ребра был равен $(\mathrm{B}_1, 0)$ или $(\mathrm{B}_1, 1)$. В правой части преобразования над пунктирным ребром написано $\langle \mathrm{B}_2, * \rangle$. Это означает, что автомат говорит ИГ, что нагрузкой этого ребра нужно сделать предикат $f_*^{\mathrm{B}_2}$. Пример применения преобразования R_1 с точки зрения ИГ изображен на рисунке 5. Если код входной окрестности отличен от $(\mathrm{B}_1, 0)$ и $(\mathrm{B}_1, 1)$, то автомат на вставку применяет преобразование R_2. Оно изображено на рисунке 4. Пример применения преобразования R_2 с точки зрения ИГ изображен на рисунке 5. Как было описано в алгоритме, автомат на вставку имеет внутреннее состояние 0 в первый такт функционирования, кроме этого он запоминает тип предиката,

который он создаст. На рисунке внутреннее состояние автомата обозначено через q, а тип который он запомнит через t.

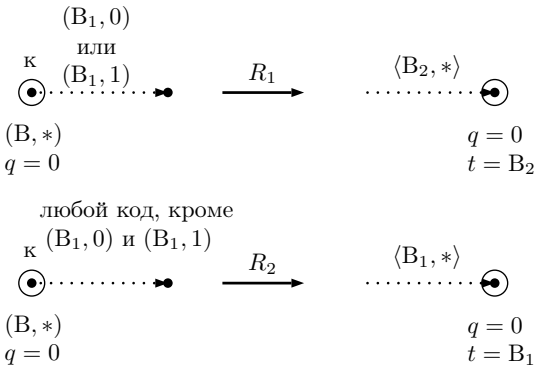

Рис. 4: Преобразования R_1 и R_2 на вставку с точки зрения автомата. q это внутренне состояние автомата, t это тип, который он запоминает.

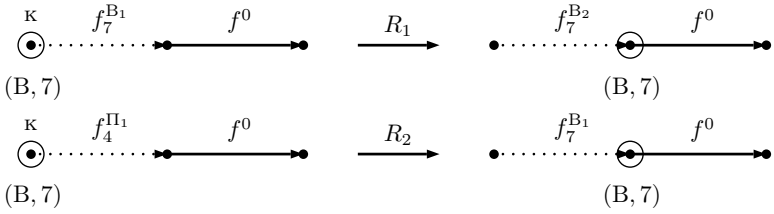

Рис. 5: Пример применения преобразований R_1 и R_2 с точки зрения ИГ.

Преобразование R_3 изображено на рисунке 6. Оно означает, что тип предиката, выходящего из корня не поменялся. Такое могло произойти, только в случае, если в потоке запросов был пустой запрос. Поэтому автомат не применяет никаких преобразований, а просто продолжает функционировать.

Преобразование R_4 изображено на рисунке 7. Оно применяется, когда тип предиката поменялся с предыдущего такта, но его значение равно нулю, либо внутреннее состояние автомата было равно единице, и значение предиката рано нулю. В этом случае автомат меняет свое состояние на единицу и забывает

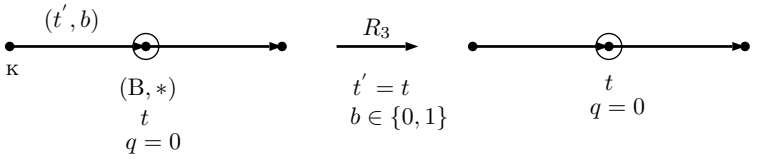

Рис. 6: Преобразование R_3.

про тип. Если состояние автомата уже было рано одному, то он его оставляет.

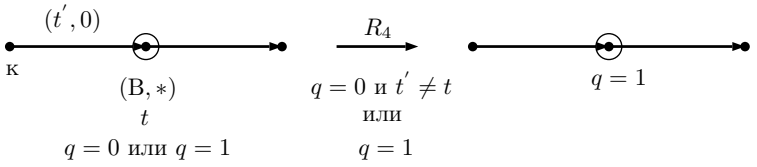

Рис. 7: Преобразование R_4. Случай когда $q = 0$ и $t' = t$ это преобразование R_3.

Преобразование R_5 изображено на рисунке 8. Оно применяется, когда тип предиката поменялся с предыдущего такта и равен либо B_1 или B_2, а значение равно 1. В этом случае автомат завершает свое функционирование.

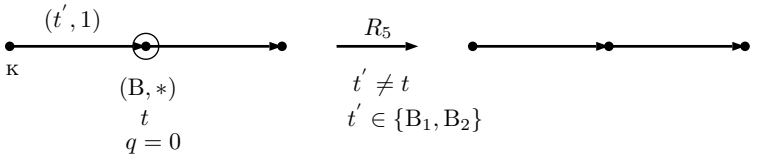

Рис. 8: Преобразования R_5.

Преобразование R_6 изображено на рисунке 9. Оно применяется, когда внутреннее состояние автомата равно одному, а тип предиката ребра, выходящего из корня, стал равен либо B_1, либо B_2, и значение равно 1. В этом случае автомат завершает свое функционирование.

В преобразованиях R_5 и R_6 автомат завершает свое функционирование, так как вместо него встанет новый автомат, обслуживающий тот же запрос.

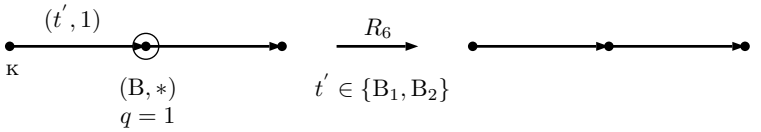

Рис. 9: Преобразования R_6.

Преобразование R_7 изображено на рисунке 10. Автомат на вставку завершает свое функционирование вне зависимости от внутреннего состояния q. При этом он меняет предикат второго ребра на f^0.

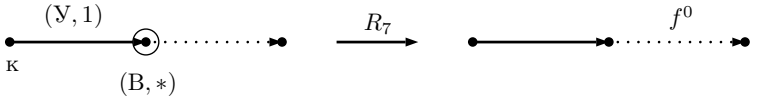

Рис. 10: Преобразования R_7.

Осталось рассмотреть последнее преобразование для вставки. Преобразование R_8 изображено на рисунке 11. Автомат на вставку видит, что пришел запрос на поиск того же элемента, что он обслуживает. Поэтому он заменяет предикат на втором ребре от корня и присваивает третьей вершине запись, которую он обслуживает.

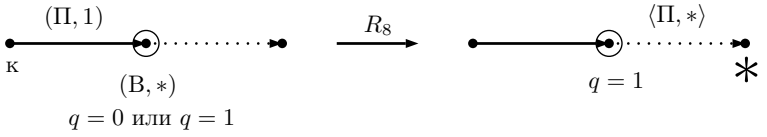

Рис. 11: Преобразования R_8.

Рассмотрим преобразования, которые применяет автомат на удаление. Как видно из алгоритма, такое преобразование всего одно (R_9). Оно изображено на рисунке 12. Автомат на удаление меняет предикат и завершает функционирование.

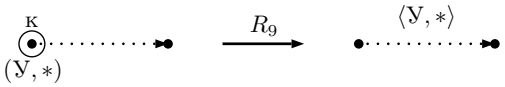

Рис. 12: Преобразования R_9.

Осталось рассмотреть преобразования, применяемые автоматом на поиск. Из алгоритма видно, что автомат на поиск применяет всего три преобразования. Первое преобразование R_{10} аналогично преобразованию R_9 на удаление и изображено на рисунке 13. Единственное отличие состоит в том, что автомат на поиск переходит в соседнюю с корнем вершину и продолжает функционировать.

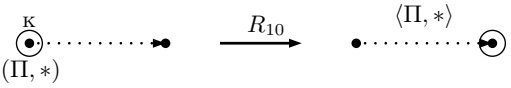

Рис. 13: Преобразования R_{10}.

Преобразование R_{11}, применяемое автоматом на поиск, изображено на рисунке 14. Это просто перемещение из второй вершины в третью.

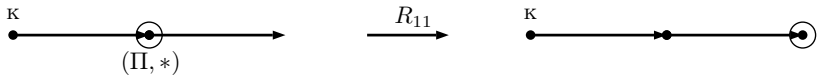

Рис. 14: Преобразования R_{11}.

Последнее преобразование R_{12} изображено на рисунке 15. Если значение предиката равно одному, то автомат на поиск выдает в ответ запись из третей вершины, а если равно нулю, то такой записи нет. В любом случае после этого автомат на поиск завершает функционирование.

Итак, описаны все преобразования, которые будут использовать автоматы. Покажем теперь, что построенный МДИГ работает бесконфликтно, корректно для любого потока запросов H, и количество тактов необходимое на поисковый запрос равно трем.

Нижняя оценка на область видимости автомата, обрабатывающего произвольный поток запросов к динамической базе данных

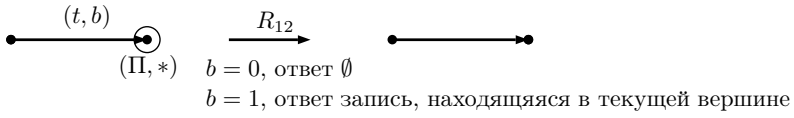

Рис. 15: Преобразования R_{12}.

Последнее утверждение очевидно, так как автомат на поиск функционирует ровно три такта, после чего выдает ответ. Осталось доказать, что МДИГ работает бесконфликтно и корректно.

Докажем, что МДИГ работает бесконфликтно. Это означает, что не бывает такой ситуации, при которой два автомата изменяют нагрузку одно и того же ребра или вершины.

Как мы уже отмечали, автоматы не могут вызвать конфликт, изменяя первое ребро. Действительно, первое ребро изменяет только автомат, находящийся в корне. В корне находится не более одного автомата, следовательно конфликтов возникнуть не может. Нагрузка первой и второй вершины не изменяется, поэтому конфликтов так же не возникает.

Осталось доказать, что конфликтов не возникает при изменении второго ребра и третьей вершины.

Как видно из алгоритма вставки, автомат после получения запроса на вставку оказывается во второй вершине. При этом, если в этой вершине уже был автомат на вставку той же записи, то он завершает функционирование. Следовательно, во второй вершине могут находиться только два автомата на вставку одной и той же записи, причем ровно один такт и на следующий такт один из автоматов обязательно завершит функционирование. Пример описанной ситуации изображен на рисунке 16 (3, 4 и 5 такты). Второе ребро и третье вершину может изменить только автомат на вставку, запись которого совпала с записью поиска. Как было доказано выше, такой автомат может быть только один, следовательно, конфликтов не возникнет.

Итак, доказано, что МДИГ работает бесконфликтно. Осталось доказать, что он работает корректно. Это означает, что когда поступает запрос на поиск записи, в ответ он должен дать

эту запись, если она принадлежит базе данных или сказать, что такой записи нет.

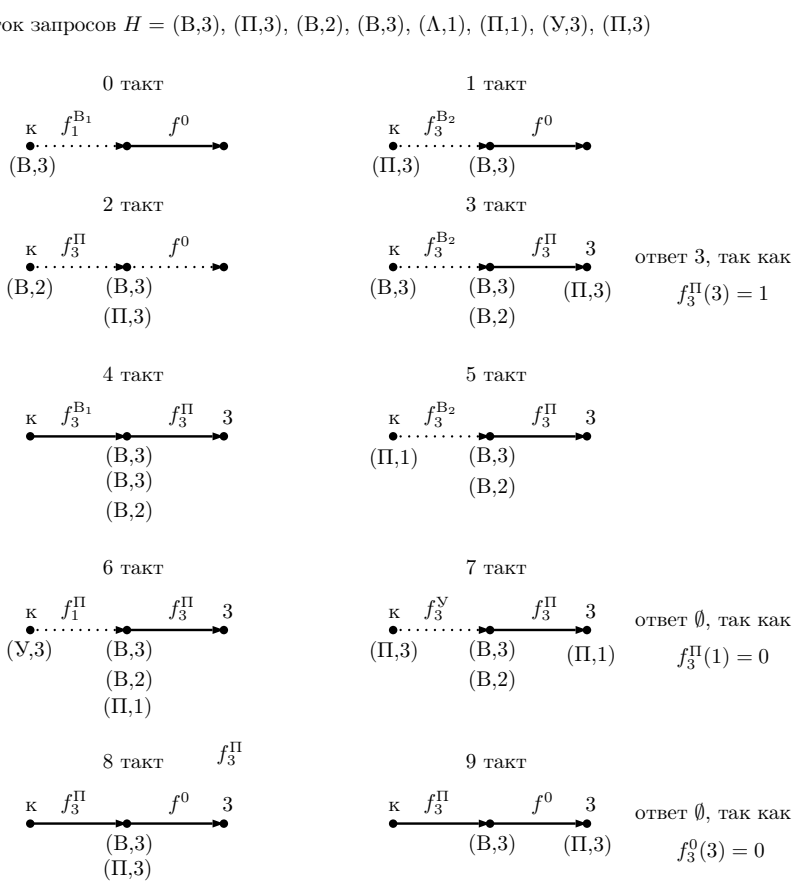

Рис. 16: Пример функционирования МДИГ, удовлетворяющий теореме 1. Пунктиром изображены ребра, нагрузка которых изменится на следующий такт.

Если к МДИГ поступал запрос на вставку, то обслуживающий его автомат постоянно функционирует во второй вершине. При этом он может завершить свое функционирование, только если поступил новый запрос на вставку той же записи, или поступил запрос на ее удаление. В других случаях он продолжает функционировать. Поэтому, если поступит запрос на поиск этой

записи после того, как был запрос на ее вставку, то во второй вершине обязательно будет функционировать автомат, который ее обслуживает. Следовательно, МДИГ будет работать корректно. Если был послан запрос на удаление, а после него шел запрос на поиск, то автомат на вставку завершит свое функционирование. В этом случае опять МДИГ будет работать корректно, так как предикат, соответствующей второму для этого запроса будет равен нулю. Следовательно, поиск выдаст ответ $\emptyset$. Для лучшего понимания на рисунке 16 изображен пример функционирования МДИГ на небольшом количестве запросов.

Теорема доказана.

Доказательство теоремы 2

МДИГ типа (2,1) означает, что ИГ, который будет преобразовывать автомат, выглядит следующим образом. Граф будет представлять из себя цепочку ребер или круг, как изображено на рисунке 17. Теорема утверждает, что нет автомата A, множества типов T и базового множества преобразований, использующих селекторные функции такого, что удается решать логическую ДЗПИО, используя такой граф над множеством предикатов $F(T)$.

Данная теорема справедлива для логической ДЗПИО. Следовательно, она справедлива для ДЗПИО. ИГ не обязан содержать сами записи, так как рассматривается логическая ДЗПИО. Далее под записью z в доказательстве будем понимать либо саму запись z, либо предикат $f_z^t, t \in T$. В доказательстве не будет существенен факт, что ИГ содержит саму запись z. Достаточно, чтобы он содержал ребро f_z^t. Например, фразу "найти запись z" можно понимать как "найти предикатное ребро $f_z^t, t \in T$, которое принимает значение один".

Идея доказательства будет следующей. Для любого фиксированного алгоритма автомата, базового множества $F(T)$ и функции L, можно привести поток запросов H, на котором не будет выполнено свойство, что количество тактов на поисковый

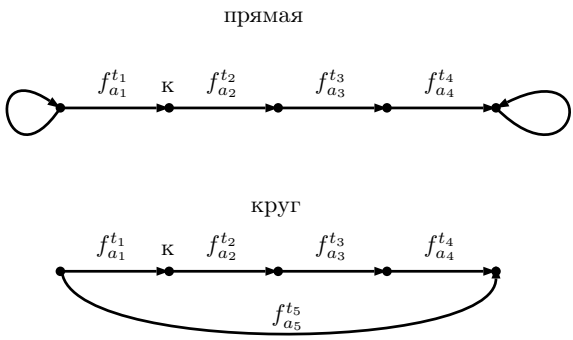

Рис. 17: Примеры ИГ, которые может построить МДИГ типа (2,1).

запрос не превосходит $L(|V(i,H)|)$, где i — такт поступления поискового запроса, $|V(i,H)|$ — объем базы данных в такт i.

Искомый поток H будет устроен следующим образом. Сначала будут идти запросы на вставку $1, 2, \ldots, M$, причем $M > 4 * 2^{L^2(3)} + 4$. Далее поток будет содержать удаление всей базы данных кроме трех записей в определенном порядке. Следующим запросом будет поиск записи, от корня до которой будет расстояние не меньше $[M/2]-1$. Доказательство существования такой записи является основным в данной теореме.

Также будет доказано, что если расстояние от корня до записи есть K ребер и K достаточно большое число, то минимальное количество тактов, которое автомат должен затратить на поиск этой записи будет не меньше $\log_2 \sqrt{\frac{K}{2}}$. После математических выкладок получим, что автомат, обслуживающий запрос на поиск записи, которая находиться на расстоянии не меньшем $[M/2]-1$ от корня, должен будет затратить больше $L(3)$ тактов, а по условии должен был затратить не больше $L(3)$ тактов, что и докажет теорему.

Итак, приступим к доказательству. Сначала будут доказаны вспомогательные утверждения, а затем будет приведен поток запросов, описанный выше.

Докажем следующую лемму.

Лемма 1. *В условиях теоремы 2, если база данных V содержит M, $M \geq 3$ записей ($|V| = M$), то существует запись из*

V, путь до которой от корня содержит не меньше $[M/2]-1$ ребер, где через $[x]$ обозначена целая часть снизу (наибольшее целое число не превосходящее x).

Доказательство. Заметим, что ИГ должен представлять из себя связный граф, так как автомат начинает свое функционирование в корне и перемещается только по ребрам, которые он видит из текущей вершины. Поэтому, если ИГ содержит компоненты связности, которые не содержат корня, то автомат в них не сможет зайти. Следовательно, все компоненты связности, которые не включает в себя корень, можно не рассматривать. Поэтому далее будем считать, что ИГ содержит только одну компоненту связности.

Очевидно, что ИГ должен содержать не меньше $M-1$ ребра, так как существуют M различных вершин, которые должны образовывать связный граф. Кроме этого, ИГ представляет из себя либо цепочку, либо круг, как изображено на рисунке 17.

Рассмотрим случай цепочки. Пусть слева от корня находиться l записей, а справа r записей, где $l, r \geq 0$ и $l + r = M$. По принципу Дирихле либо l, либо r не меньше чем $[M/2]$. Без ограничения общности, пусть $r \geq [M/2]$. Выберем самую дальнюю справа от корня запись. Расстояние до нее будет не меньше $[M/2]-1$ ребра, так как граф на $[M/2]$ вершинах должен содержать не меньше, чем $[M/2]-1$ ребро.

В случае, когда ИГ представляет из себя круг, можно свести к случаю цепочки. Для этого удалим в круге 1 ребро и получим случай цепочки, для которой уже доказано. □

Докажем теперь лемму о минимальном количестве тактов, необходимых автомату, чтобы дойти до вершины, находящейся на расстоянии в K ребер от текущей вершины автомата. Заметим только, что основная цель следующей леммы состоит в том, чтобы показать, что количество тактов будет не константной функцией, а функцией от K. Поэтому для простоты оценка в лемме будет достаточно грубой. Также заметим, что нет смысла рассматривать маленькие значения K, так как они будут большими при доказательстве теоремы 2.

Лемма 2. *В условиях теоремы 2 автомат на поиск записи, находящейся от корня на расстоянии в* $K, K \geq 100$ *ребер, затратит не меньше* $]\log_2 \sqrt{\frac{K}{2}}[$ *тактов.*

Доказательство. Лемма утверждает, что автомату на поиск записи, находящейся на расстоянии в K ребер от корня, потребуется не меньше $]\log_2 \sqrt{\frac{K}{2}}[$ тактов, чтобы ее найти.

Рассмотрим тривиальный случай, когда на ИГ нет ни одного автомата. Действительно, если на ИГ в момент поступления запроса на поиск не было ни одного другого автомата, то очевидно, что автомату на поиск потребовалось не меньше $K - 1 >]\log_2 \sqrt{\frac{K}{2}}[$ тактов, чтобы найти запись, так как расстояние уменьшиться не могло, а автомат, в силу радиуса видимости 1, может за 1 такт пройти только 1 ребро.

Пусть на ИГ функционировали другие автоматы во время поступления рассматриваемого автомата. Рассмотрим, каким образом автоматы с радиусом видимости 1 могли сократить расстояние, чтобы автомат на поиск мог дойти до искомой записи за меньшее количество тактов.

Первый способ – это поменять искомую запись с записью, которая на одно ребро ближе к корню. В этом случае расстояние до корня уменьшиться ровно на 1 ребро. Поменять с записью, которая ближе к корню больше, чем на 1 ребро, автомат не может в силу радиуса видимости 1.

Если какой-либо автомат удалит два ребра и не создаст новое, то в этом случае непрерывного пути к искомой записи либо не останется тогда, когда ИГ – цепочка, либо останется путь, который не изменится в случае круга. В случае круга такое преобразование возможно только один раз, так как после этого преобразования круг превратится в цепочку. А цепочка может превратиться в круг только тогда, когда ее длина будет не больше двух ребер, так как автомат должен соединить крайние вершины цепочки, что возможно только в случае, когда ребер не больше двух. В любом случае данное преобразование не сокращает расстояние до искомой записи, а, следовательно,

не поможет рассматриваемому автомату найти ее за меньшее количество тактов.

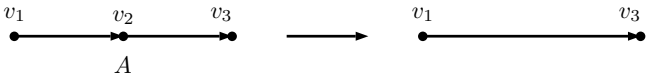

Рис. 18: Пример сокращения расстояния.

Рассмотрим последнее преобразование, которое может сократить расстояние от корня до искомой записи. Это преобразование изображено на рисунке 18. Данное преобразование заменят два ребра на одно. После его применения расстояние от корня к искомой записи сократиться на одно ребро.

Заметим, что других преобразований, сокращающих расстояние, нет, так как МДИГ типа (2,1), а, следовательно, после преобразования ИГ должен удовлетворять свойству, что степень инцидентности любой вершины не больше двух. Поэтому один автомат не может сократить расстояние больше, чем на одно ребро, так как радиус видимости у всех автоматов равен единице.

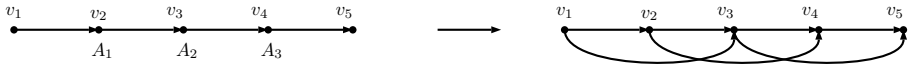

Рис. 19: Пример сокращения расстояния несколькими автоматами. В правой части нарисованы все возможные ребра. Видно, что расстояние в 4 ребра из v_1 в v_5 нельзя будет преодолеть быстрее чем за 2 такта.

Итак, рассмотрим преобразование, изображенное на рисунке 18. Посмотрим, насколько может сократиться расстояние, если несколько автоматов одновременно применят это преобразование. На рисунке 19 изображен пример, когда расстояние от вершины v_1 до вершины v_5 автоматы сокращают с помощью рассматриваемого преобразования. Видно, что расстояние в четыре ребра может сократиться максимум до двух ребер. Не трудно заметить, что если расстояние между вершинами будет p ребер, то расстояние не может стать не меньше $[p/2]$ ребер после

применения преобразования несколькими автоматами, изображенного на рисунке 18.

Итак, за один такт расстояние от корня до искомой записи может сократиться максимально на $[K/2]+1$ ребро, так как возможно применение преобразования, меняющее записи местами, которое сократит расстояние еще на одно ребро.

Доказан следующий факт. Рассматривается расстояние от автомата до записи, которую он ищет. Пусть расстояние от вершины, в которой находится этот автомат, до искомой записи равно x. Тогда в следующий такт расстояние от автомата до искомой записи не меньше, чем $[x/2]+2$. Действительно, плюс один к оценке выше мог добавить сам автомат, перейдя в вершину по направлению к искомой записи.

Для доказательства леммы 2 осталось провести вычислительные выкладки. Будут проведены достаточно грубые оценки, так как главное в данной лемме – количество тактов зависит от K и не является константой.

Очевидно, что автомат дойдет до записи, когда расстояние до нее будет равно 0. В первый такт функционирования расстояние было не меньше K. Как было доказано выше, во второй такт функционирования, расстояние не меньше $K-([K/2]+2) \geq [K/2]-2$. В третий такт – не меньше $[K/2]-2-(([K/2]-2)/2+2) \geq [K/4]-6$. Пусть для i-го такта верно, что расстояние не меньше $[K/2^{i-1}]-2^{i-1}-2$. База индукции доказана для $i=2,3$. Докажем, что это верно для $i+1$. Для $i+1$ такта расстояние не может быть меньше $[K/2^{i-1}]-2^{i-1}-(([K/2^{i-1}]-2^{i-1})/2+2) \geq [K/2^i]-2^i-2$.

Найдем большое i, $i>1$, при котором $[K/2^i]-2^i-2>0$. Умножим для простоты и левую и правую часть неравенства на 2^i. Получаем, $K>2^{2i}+2^{i+1}$. Если $i<\log_2\sqrt{\frac{K}{2}}$, то

$$2^{2i}+2^{i+1}<2^{2\log_2\sqrt{\frac{K}{2}}}+2^{\log_2\sqrt{\frac{K}{2}}+1}<K/2+2\sqrt{K}<K.$$

Следовательно, автомат затратит не менее $\log_2\sqrt{\frac{K}{2}}$ тактов на поиск записи, что и требовалось доказать. □

Докажем теперь основную лемму, необходимую для доказательства теоремы 2.

Лемма 3 (Основная)**.** *В условиях теоремы 2 пусть база данных состоит из M записей. Для любого конечного МДИГ типа (2,1) существует такой поток запросов H, что есть такт i, на котором мощность базы данных будет равна 3 ($|V(i,H) = 3|$), и в этот такт существует предикат f_z^t, $z \notin \{H(1), \ldots, H(i)\}$, расстояние от корня до которого не меньше $[M/2] - 1$ ребер.*

Доказательство. Другими словами лемма 3 утверждает, что для любого МДИГ из условия теоремы 2 можно подобрать такой поток запросов H, что через некоторое количество тактов мощность базы данных будет равна 3, и будет существовать запись, расстояние от корня до которой будет не меньше $[M/2]-1$, удаление которой не было в H.

Важно в данной лемме, что рассматривается конечный и селекторный МДИГ. Так как МДИГ конечный, то очевидно, можно найти такой такт, что ни один автомат не будет функционировать. Зафиксируем этот такт и будем посылать искомую последовательность, запросов на удаление начиная с него. Главный смысл состоит в том, что автомат на удаление не может удалить, чужую запись, так как в этом случае ее нельзя будет восстановить.

Итак, приступим к доказательству. Прежде всего, поймем, когда запись точно не может быть удалена. Всем ребрам ИГ приписаны предикаты из множества $F(T)$. Рассмотрим произвольный автомат, обслуживающий запрос z. Очевидно , что перемещаясь по ИГ, он будет видеть значение 1, только на тех ребрах, которым сопоставлены предикаты вида $f_z^t, t \in T$. На всех остальных ребрах он будет видеть значение 0.

Допустим автомат может удалить произвольную вершину. Покажем, что при этом МДИГ не будет корректно функционировать.

Рассмотрим поток запросов. В первый такт идет удаление z, а в последующих идет поиск $M-1$ записи всех элементов, кроме z. Если автомат, обслуживающий z, удалит вершину с записью

не равной z, то автомат на поиск записи, соответствующей данной вершине, завершится неправильно, следовательно, МДИГ будет работать некорректно. Действительно, как было показано выше, ни одного автомата, содержащего удаленную чужую запись, не функционирует (конечный МДИГ). Поэтому, после удаления чужой записи, она никак не сможет восстановиться, откуда получаем, что МДИГ будет работать некорректно.

Итак, показано, что если не было автомата на удаление какой либо записи, то вершина, содержащую эту запись, не может быть удалена.

Рассмотрим следующий, не трудный, но важный факт. Автомат может понять, есть ли искомая запись в какой-либо вершине, только в том случае, если он увидит предикат равный 1. Если во время своего функционирования он видит предикаты только равные 0, то он не знает есть ли искомая запись в базе данных или нет.

Рассмотрим последовательность запросов на удаление, в которой нет повторяющихся запросов (удаляются различные элементы). Для такой последовательности запросов, если автомат видит, во время своего функционирования, только предикаты равные 0, то запись, которую он удаляет не может быть удалена. Действительно, запись может быть удалена только в том случае, если хоть один из автоматов увидит предикат равный 1, соответствующей этой записи (предикат с параметром равной записи). Кроме автомата, удаляющего данную запись, никто не может увидеть значение равное 1 на таких предикатах, так как другие автоматы удаляют другие записи.

Из предыдущих выкладок следует, что если в потоке запросов все автоматы на удаление, во время своего функционирования будут видеть предикаты равные только 0, то ни одна запись не будет удалена. Покажем, что такую последовательность запросов можно привести.

Идея построения такой последовательности состоит в следующем. Каждый такт, кроме первого, своего функционирования автомат может увидеть не более одного нового ребра. В первый такт автомат может увидеть два ребра. Это следствие того, что

радиус видимости автомата равен 1, а любой вершине в ИГ инцидентно не более двух ребер (МДИГ типа (2,1)). Заметим, что новый предикат, который увидит автомат, будет равен 1 только на одном запросе. Поэтому, чтобы автомат увидел значение 0, достаточно не использовать этот запрос.

Для простоты объяснения, не ограничивая общности рассуждения, будем считать, что ни одного автомата не функционирует, и что поток запросов H, начиная с первого такта, будет содержать запросы на удаление, а не пустые запросы, чтобы дождаться, пока ни один автомат не будет функционировать.

Последовательность запросов на удаление состоит из $M-3$ запроса. Каждому автомату сопоставим множество запросов $B^i, 1 \leq i \leq M-3$, где i номер автомата в искомой последовательности удалений. В нулевой такт это множество состоит их всех элементов базы данных. Обозначать номер такта будем индексом снизу, то есть для всех $i, 1 \leq i \leq M-3 : |B_0^i| \geq M$.

Рассмотрим первый такт. В первый такт начинает функционировать только первый автомат. Пусть он увидит два предиката $f_{a_1}^{t_1}, f_{a_2}^{t_2}$. Он должен увидеть значение 0 на обоих предикатах, чтобы это гарантировать, из множества B_0^1 уберем два элемента a_1 и a_2. Получим множество элементов $B_1^1 = B_0^1 \setminus \{a_1, a_2\}$, которые может удалять первый автомат, так, чтобы во время функционирования видеть только нулевые значения предикатов.

Так как другие автоматы из последовательности еще не начали свое функционирование, то для всех $i, 2 \leq i \leq M-3 : B_0^i = B_1^i$. Следовательно в первый такт $|B_1^1| \geq M-2, 2 \leq i \leq M-3 : |B_1^i| \geq M$.

Рассмотрим второй такт. Как было доказано выше первый автомат может увидеть не более одного нового ребра. Пусть предикат, который ему соответствует f_a^t. Важно, что на ребре, которое он уже видел не может появиться новый аргумент, так как используются только селекторные преобразования. Как и на первом такте, уберем этот элемент из B_1^1, получим что $B_2^1 = B_1^1 \setminus \{a\}$, $|B_1^2| \geq M-3$. Второй автомат во второй такт, аналогично первому автомату в первый такт, может увидеть не более двух предикатов, поэтому $|B_2^2| \geq M-2$. Мощность множе-

ства оставшихся автоматов не изменится, так как они не начали еще свое функционирование.

Итак, рассмотрим $M-3$ такт. Каждый такт, начиная со второго, мощность множества B^1 могла уменьшаться не более, чем на 1. То есть $|B^1_{M-3}| \geq |B^1_{M-2}| - 1 \geq |B^1_{M-1}| - 2 \geq \cdots \geq |B^1_2| - (M-5) \geq M-3-(M-5) = 2$. Не трудно заметить, что $|B^2_{M-3}| \geq 3$, $|B^3_{M-3}| \geq 4, \ldots, |B^{M-3}_{M-3}| \geq M-2$.

Еще раз напомним смысл данных множеств. Множество B^1_{M-3} состоит не менее чем из двух элементов. Если первый автомат удалял любую запись из множества B^1_{M-3}, то он видел бы, во время своего функционирования, только нулевые значения предикатов. Аналогично для остальных множеств.

Пусть первый запрос на удаление был $z_1 \in B^1_{M-3}$, то есть $H(1) = (\text{У}, z_1)$. Автомат, который его обслуживает до $M-3$ такта, обязательно видит нулевые предикаты. Второй запрос на удаление z_2, выбираем из множества $B^2_{M-3} \setminus \{z_1\}$, мощность которого не меньше двух, так как $|B^2_{M-3}| \geq 3$. Аналогично для остальных тактов $j, j < M-3$, $H(j) = (\text{У}, z_j)$, где $z_j \in B^j_{M-3} \setminus \{z_1, z_2, \ldots, z_{j-1}\}$, $|B^j_{M-3} \setminus \{z_1, z_2, \ldots, z_{j-1}\}| \geq j+1-(j-1) \geq 2$.

Итак, для потока запросов H, в $M-2$ такт не была удалена ни одна запись, так как все автоматы на удаление видели во время своего функционирования нулевые предикаты. По лемме 1 существует запись, расстояние от корня до которой не меньше $[M/2]-1$ ребра. Пусть это запись z. Заметим, что была свобода выбора из двух записей для каждого автомата. Например, если $z_1 = z$, то в качестве $H(1)$ выберем $z_1^{'} \in B^1_{M-3} \setminus \{z\}$. Если $z_2 = z$, то $H(2)$ будет равен $z_2^{'} \in B^2_{M-3} \setminus \{z_1, z\}$, $|B^2_{M-3} \setminus \{z_1, z\}| \geq 1$. Аналогично для остальных тактов $j, j < M-3, z_j^{'} \in B^j_{M-3} \setminus \{z_1, z_2, \ldots, z_{j-1}, z\}$, $|B^j_{M-3} \setminus \{z_1, z_2, \ldots, z_{j-1}, z\}| \geq j+1-j \geq 1$.

Построенный поток H удовлетворяет условию леммы 3, так как в такт $|V(M-3), H| = 3$ и существует запись $z, (\text{У}, z) \notin H$, расстояние от корня до которой не меньше $[M/2]-1$ ребра. $\square$

Вернемся к доказательству теоремы 2. Зафиксируем произвольную функцию L и произвольный конечный МДИГ типа

(2,1). Для доказательства нужно предъявить поток запросов H, в котором существует такт i и поисковый запрос, что время на его обработку будет больше, чем $L(|V(i,H)|)$.

Как уже было сказано в идее доказательства, сначала в потоке запросов H будут идти запросы на вставку. $H(1) =$ (B,1),..., $H(M) =$ (B,M), где $M > 4 * 2^{L^2(3)} + 4$. Далее ставим столько пустых запросов, чтобы на графе не осталось ни одного работающего автомата. Номер такта, когда это случиться, обозначим M'. Применяя лемму 2 можно предъявить поток запросов $H^{'}$, такой, что будет существовать такт t, для которого мощность базы данных будет равна 3, и будет существовать запись z, расстояние от корня до которой будет не меньше $[M/2]-1$, удаление которой не было в $H^{'}$.

Пусть $H(M'+1) = H^{'}(1),\ldots,H(M'+t) = H^{'}(t), H(M'+t+1) =$ (П,z). Покажем, что время обработки запроса $H(M'+t+1)$ больше, чем $L(|V(M'+t+1,H)|)$, а тем самым докажем теорему.

Действительно, по лемме 2, $|V(M'+t+1,H)| = 3$, расстояние от корня до z не меньше $[M/2]-1 > 2^{L^2(3)}$. Будем считать, что $M > 202$, чтобы выполнилось условие леммы 2. По лемме 2 автомат затратит на поиск z не меньше, чем $\log_2 \sqrt{\frac{[M/2]-1}{2}} > \log_2 \sqrt{L^2(3)} = L(3)$, а должен был затратить не больше $L(|V(M'+t+1,H)| = L(3)$ тактов, противоречие.

Теорема 2 доказана.

Доказательство теоремы 3

Искомый МДИГ будет достаточно простым, в силу того, что разрешается в преобразованиях использовать не селекторные функции.

Рассмотрим функцию $P : \mathbb{N} \to \mathbb{N}$, которая на входе получает натуральное число n, а на выходе выдает n-е простое число. То есть $P(1) = 2, P(2) = 3, P(3) = 5, P(4) = 7, \ldots$.

Будем использовать следующие функции в преобразованиях.

$$\phi_{\text{В}}(a,x)=\begin{cases} f_a^{\text{нет}}, & \text{если } 0=a \pmod{P(x)};\\ f_{aP(x)}^{\text{нет}}, & \text{иначе;}\end{cases};\phi_{\text{У}}(a,x)=\begin{cases} f_{\frac{a}{P(x)}}^{\text{нет}}, & \text{если } 0=a \pmod{P(x)};\\ f_a^{\text{нет}}, & \text{иначе;}\end{cases};$$

$$\phi_{\text{П}}(a,x)=\begin{cases} f_a^{\text{да}}, & \text{если } 0=a \pmod{P(x)};\\ f_a^{\text{нет}}, & \text{иначе;}\end{cases};$$

ИГ будет состоять из одного ребра. Изначально считаем, что ИГ состоит из одного ребра с предикатом $f_1^{\text{нет}}$. Пример начального ИГ изображен на рисунке 20.

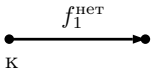

Рис. 20: Изначальный ИГ.

Основная идея состоит в следующем. Аргумент единственного предиката в ИГ каждый такт времени равен произведению простых чисел, соответствующих элементам базы данных. Например, если в базе данных находятся числа 1, 4, 7, то аргумент предиката будет равен $P(1)P(4)P(5)=2*7*11=154$. Имея такой аргумент, поиск осуществляется проверкой деления его на простое число соответствующее поисковому запросу. Например если поступает запрос на поиск 3, то проверяется делится число $154=P(1)P(4)P(5)$ на $P(3)=5$. Оно не делится, значит 3 в базе данных нет. Не сложно заметить, что ответ будет положительный только при поиске 1,4 или 5. Тем самым МДИГ будет работать корректно. Чтобы добиться такого аргумента, будут использованы преобразования на основе не селекторных функций $\phi_{\text{В}}(a,x)$ и $\phi_{\text{У}}(a,x)$.

Итак, разберем преобразования, применяемые МДИГ. Все преобразования изображены на рисунке 21. Преобразование R_1^1 это преобразование на вставку. Автомат на вставку x функционирует ровно один такт. Он применяет преобразование, изменяющее аргумент предиката на новый. Если аргумент предиката

Нижняя оценка на область видимости автомата, обрабатывающего произвольный поток запросов к динамической базе данных

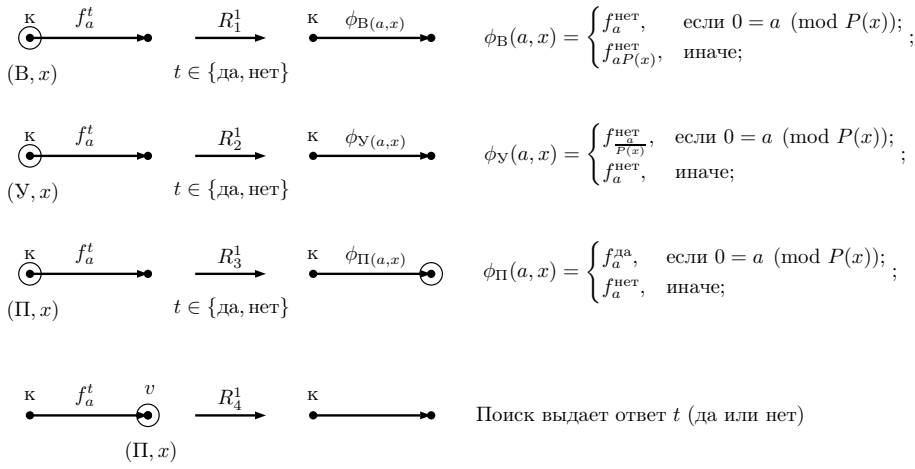

Рис. 21: Преобразования R_1^1, R_2^1, R_3^1 и R_4^1.

делился на $P(x)$, значит в базе данных уже есть x и его не нужно вставлять. Поэтому аргумент предиката остается прежним. В противном случае аргумент предиката домножается на $P(x)$.

Преобразование R_2^1 на удаление аналогично преобразованию на вставку. Аргумент предиката делится на $P(x)$, если тот делился на $P(x)$ и остается прежним в противном случае.

Преобразование R_3^1 на поиск x проверяет, делится ли аргумент предиката на $P(x)$. Если делится, то в следующий такт автомат на поиск увидит тип предиката да, иначе увидит нет. Поэтому преобразование R_4^1 – это выдача ответа. Автомат на поиск возвращает ответ да, если тип предиката равен да, и иначе ответ нет.

Приведенный алгоритм работает корректно. Пусть база данных V состоит из элементов $V = \{y_1, \dots y_k\}$. Не трудно заметить, что аргумент предиката будет равен $P(y_1) \dots P(y_k)$. Очевидно, что если поступит запрос на поиск $x, x \in V$, то $P(y_1) \times \dots P(y_k)$ разделится на $P(x)$. Если же $x \notin V$, то $P(y_1) \dots P(y_k)$ не разделится на $P(x)$.

Конфликтов также не возникает, так как только один автомат меняет нагрузку предиката в один такт. В конце приве-

дем наглядный пример, изображенный на рисунке 22, работы МДИГ.

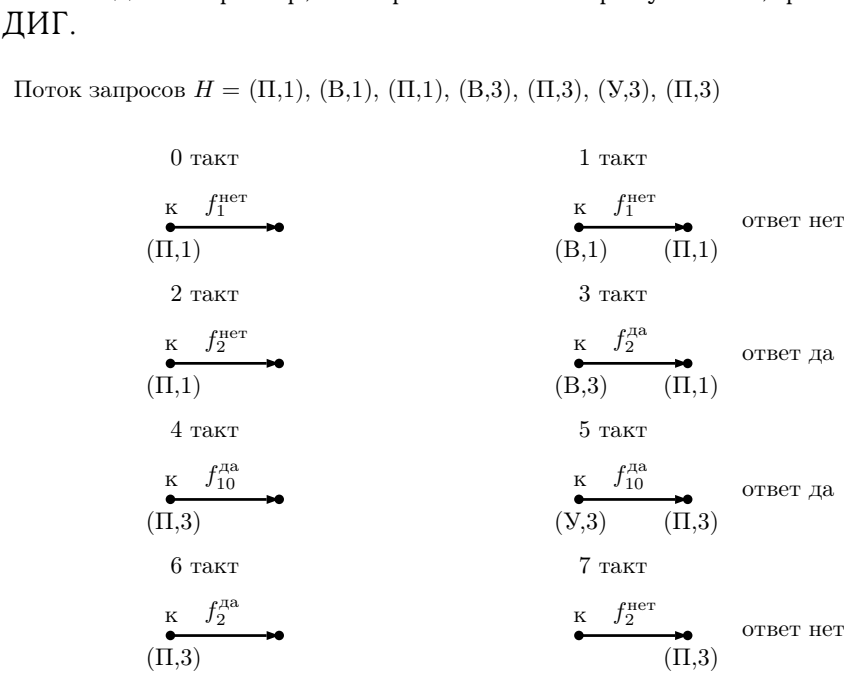

Рис. 22: Пример функционирования МДИГ.

Теорема 3 доказана.

Список литературы

[1] Плетнев А.А. Информационно-графовая модель динамических баз данных и ее применение// Интеллектуальные системы. — 2014. Т. 18, Вып. 1. — С. 111-140.

[2] Плетнев А.А. Динамическая база данных, допускающая параллельную обработку произвольных потоков запросов// Интеллектуальные системы. — 2015. Т. 19, Вып. 1. — С. 117–145.

[3] Плетнев А.А. Логарифмическая по сложности параллельная обработка автоматами произвольных потоков запросов в ди-

намической базе данных// Интеллектуальные системы. — 2015. Т. 19, Вып. 1. — С. 171–213.

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

[5] Гасанов Э.Э., Ефремов Д.В. Фоновый алгоритм решения двумерной задачи о доминировании // Интеллектуальные системы. — 2014. — Т. 18, вып. 3. — С. 133–158.

[6] Е. М. Перпер. Нижние оценки временной и объёмной сложности задачи поиска подслова // Дискретная математика, 2014, том 26:2, 58–70.

[7] Шуткин Ю.С. Моделирование схемных управляющих систем // Интеллектуальные системы. — 2014. — Т. 18, вып. 3. — С. 253–261.

[8] Перпер Е.М. Порядок сложности задачи поиска в множестве слов вхождений подслова // Интеллектуальные системы. — 2014. — Т. 19, вып. 1. — С. 99–116.

[9] E. Gafni, J. Naor, P. Ragde On Separating the EREW and CREW PRAM Models, Theoretical Computer Science 68 (1989) 343-346.

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

[11] Титова Е.Е. Конструирование движущихся изображений клеточными автоматами // Интеллектуальные системы. — 2014. — Т. 18, вып. 1. — С. 153–180.

[12] В.Б.Кудрявцев. Кафедра математической теории интеллектуальных систем (МаТИС) // Интеллектуальные системы. — 2014. — Т. 18, вып. 2. — С. 5–30.

[13] Летуновский А.А. Выразимость линейных автоматов относительно расширенной суперпозиции // Интеллектуальные системы. — 2015. — Т. 19, вып. 1. — С. 161–170.

[14] Гасанов Э.Э. Прогнозирование периодических сверхсобытий автоматами // Интеллектуальные системы. — 2015. — Т. 19, вып. 1. — С. 23–34.

[15] Бабин Д.Н. Автоматы с суперпозициями, пример нерасширяемости до предполного класса // Интеллектуальные системы. — 2015. — Т. 19, вып. 3. — С. 87 – 94.

[16] Гасанов Э.Э., Мастихина А.А. Прогнозирование общерегулярных сверхсобытий автоматами // Интеллектуальные системы. — 2015. — Т. 19, вып. 3. — С. 127–153.

[17] Часовских А.А. Критериальные системы в классах линейно-автоматных функций над конечными полями // Интеллектуальные системы. — 2015. — Т. 19, вып. 3. — С. 195 – 207.

The lower bound of automaton's view that handles arbitrary stream of requests to the dynamic databases

A. A. Pletnev

This work considers dynamic task of searching identical objects. In the articles [**?**, **?**, **?**] the author shows how this problem can be solved for arbitrary stream of requests using multiple state machines dynamic information graph (MDIG). This work also shows that there is no final MDIG with a radius of visibility one and with degree of branching two the processing arbitary stream of requests.

Keywords: Dynamic Databases, Key Searching, Information Graph, State Machine, Flow of Requests, Parallel Data Processing.