МИНИСТЕРСТВО ОБЩЕГО И ПРОФЕССИОНАЛЬНОГО
ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ТВЕРСКОЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

М.А.Тайцлин

ЯЗЫКИ ЗАПРОСОВ ДЛЯ БАЗ ДАННЫХ

Тверь 1999 год

УДК 519.7

Брошюра посвящена изучению языков запросов для баз данных.

В первой главе содержатся определения классов сложности глобальных предикатов и полноты языка запросов для данного класса сложности. Показывается, что язык логики предикатов первого порядка задает только полиномиально распознаваемые на детерминированных машинах Тьюринга запросы, но не является полным для PTIME. Строятся полные для PTIME и NPTIME языки запросов при условии, что базы данных упорядочены.

Во второй главе приводится пример Гуревича, показывающий, что использование линейного порядка существенно увеличивает выразительные возможности языка логики предикатов, даже для случая генерических (сохраняющихся при перестановках универсума) запросов.

Содержание

Введение

Брошюра посвящена изучению языков запросов для баз данных.

В первой главе содержатся определения классов сложности глобальных предикатов и полноты языка запросов для данного класса сложности. Показывается, что язык логики предикатов первого порядка задает только полиномиально распознаваемые на детерминированных машинах Тьюринга запросы, но не является полным для PTIME. Строятся полные для PTIME и NPTIME языки запросов при условии, что базы данных упорядочены.

Во второй главе приводится пример Гуревича, показывающий, что использование линейного порядка существенно увеличивает выразительные возможности языка логики предикатов, даже для случая генерических (сохраняющихся при перестановках универсума) запросов.

Мы предполагаем, что читатель предварительно изучил книгу "Математические основания информатики"А.П.Столбоушкина и М.А.Тайцлина (части 1, 2, 3; Тверь, 1998, издание Тверского государственного университета). Ссылка [СТ, Определение 7.1.1] имеет в виду определение 7.1.1 из этой книги.

Используемая нами система ссылок аналогична той, которая используется в [СТ]. Символ ■ обозначает конец доказательства или отсутствие доказательства.

1 ЯЗЫКИ ЗАПРОСОВ ДЛЯ УПОРЯДОЧЕННЫХ БАЗ ДАННЫХ

1.1 Глобальные предикаты и языки запросов

Мы будем рассматривать реляционные базы данных.

Под *реляционной* сигнатурой мы будем понимать сигнатуру, в которой нет ненульместных символов операций, другими словами, каждый символ этой сигнатуры либо является символом отношения, либо является константой (нульместным символом операции).

Как и в [CT, §7.1], под схемой баз данных мы будем понимать фиксированную реляционную сигнатуру, но всегда будем рассматривать также обогащенную схему, в которую дополнительно включены константа 0 для обозначения нуля, символ $<$ бинарного отношения для обозначения стандартного линейного упорядочения натуральных чисел, символ $'$ унарной операции перехода к следующему натуральному числу и константа $LAST$ для обозначения наибольшего элемента.

Определение 1.1.1 (стандартной базы данных). *Под стандартной базой данных мы будем понимать любое состояние этой обогащенной схемы, в котором основное множество является начальным отрезком натуральных чисел, $LAST$ интерпретируется как наибольший элемент основного множества, а $<$ — как стандартное отношение линейного порядка на этом начальном отрезке натуральных чисел. Другими словами, мы рассматриваем некоторую реляционную сигнатуру*

$$\langle P_1, \ldots, P_r,\ c_1, \ldots, c_m \rangle \tag{1}$$

и конечные алгебраические системы $\mathcal{A}$ вида

$$\langle \{0,\ 1, \ldots, LAST\}, P_1^{\mathcal{A}},\ P_2^{\mathcal{A}}, \ldots, P_r^{\mathcal{A}},\ c_1^{\mathcal{A}}, \ldots, c_m^{\mathcal{A}},\ <,\ ',\ 0,\ LAST \rangle,$$

в которых $c_i^{\mathcal{A}}$ является выделенным элементом (значением константы c_i), $P_i^{\mathcal{A}}$ является значением в $\mathcal{A}$ символа P_i и является отношением соответствующей местности на множестве $\{0, 1, \ldots, LAST\}$, $<$ является обычным отношением линейного порядка, а одноместная операция $'$ задаётся обычным образом:

$$0' = 1;\ 1' = 2; \ldots; (LAST - 1)' = LAST;\ LAST' = LAST.$$

При этом для $i = 1,\ 2, \ldots, r$ местности $P_i^{\mathcal{A}}$ и P_i совпадают.

$\mathcal{A}$ называется стандартной базой данных для схемы (1).

Надо заметить, что определение стандартной базы данных отличается от определения арифметической системы из [СТ, определение 7.4.2] тем, что, кроме наименьшего элемента и операции, дающей следующий элемент, стандартная база данных содержит также отношение, задающее упорядочение элементов. Понятно, что это упорядочение может быть задано некоторой простой программой, использующей наименьший элемент и операцию получения следующего элемента. Но это упорядочение не может быть задано никакой формулой логики предикатов, использующей только символы наименьшего элемента и операции получения следующего элемента. С другой стороны, существуют формулы логики предикатов сигнатуры, содержащей только символ для отношения порядка, определяющие наибольший элемент, наименьший элемент и следующий элемент для заданного элемента. Действительно, следующим является такой элемент, который больше заданного, но между которым и заданным нет других элементов. Поэтому при наличии отношения линейного порядка мы могли бы и не рассматривать операцию получения следующего элемента и константы, выделяющие наибольший и наименьший элементы, но рассматриваем эту операцию и эти константы только для сокращения обозначений.

Определение 1.1.2 (глобального предиката). *Пусть Ω — произвольная схема баз данных, $\mathfrak{K}$ — некоторый класс ее стандартных баз данных. $\mathfrak{K}$-глобальным n-местным предикатом называется отображение, которое каждой системе $\mathcal{A} \in \mathfrak{K}$ ставит в соответствие некоторый n-местный предикат, заданный на основном множестве системы $\mathcal{A}$. Мы будем опускать $\mathfrak{K}$, когда ясно, о каком классе систем идет речь.*

Нульместные глобальные предикаты называются булевыми.

Обогащая начальную реляционную сигнатуру дополнительными константами

$$a_1, \ldots, a_n$$

и рассматривая вместо первоначального класса $\mathfrak{K}$ класс $\mathfrak{K}_n$ стандартных баз данных обогащенной сигнатуры вида

$$(\mathcal{A},\ b_1, \ldots, b_n),$$

где $\mathcal{A} \in \mathfrak{K}$, мы можем свести n-местный $\mathfrak{K}$-глобальный предикат к булеву $\mathfrak{K}_n$-глобальному предикату.

Действительно, n-местному $\mathfrak{K}$-глобальному предикату $\mathfrak{G}$ соответствует такой булев $\mathfrak{K}_n$-глобальный предикат $\mathfrak{G}_n$, что

$$\langle b_1, \ldots, b_n \rangle \in \mathfrak{G}(\mathcal{A}) \Longleftrightarrow \mathfrak{G}_n((\mathcal{A},\ b_1, \ldots, b_n))$$

для любой $\mathcal{A} \in \mathfrak{K}$.

По этой причине далее мы будем рассматривать только булевы глобальные предикаты.

Каждый булев $\mathfrak{K}$-глобальный предикат определяет подкласс класса $\mathfrak{K}$, составленный из тех и только тех систем из $\mathfrak{K}$, на которых этот глобальный предикат истинен. Этот подкласс мы будем отождествлять с этим булевым глобальным предикатом.

Для задания глобальных предикатов, как уже отмечалось в [СТ, §7.1], используются языки запросов, простейшим из которых является язык логики предикатов. Этот язык действительно является практически используемым. Например, некоторой модификацией этого языка является язык **SQL**, используемый в системе **ORACLE** и во многих других промышленных программных продуктах.

Для доказательства невыразимости некоторых глобальных предикатов в языке логики предикатов нам понадобится некоторое достаточное условие неразличимости двух алгебраических систем заданной формулой логики предикатов. Для формулировки этого условия используют так называемые игры Эренфойхта.

Рассмотрим реляционную сигнатуру Ω, состоящую из констант

$$c_1,\ c_2, \ldots, c_m$$

и символов отношений. Рассмотрим две алгебраических системы $\mathcal{A}$ и $\mathcal{B}$ этой сигнатуры. Через $c_i^{\mathcal{A}}$ и $c_i^{\mathcal{B}}$ будем обозначать значения константы c_i в системах $\mathcal{A}$ и $\mathcal{B}$ соответственно.

Определение 1.1.3 (локального мономорфизма $\mathcal{A}$ в $\mathcal{B}$). *Пусть $a_1, \ldots, a_n$ взяты из $\mathcal{A}$ и $b_1, \ldots, b_n$ взяты из $\mathcal{B}$. Пусть τ одно-однозначно отображает*

$$\{a_1, \ldots, a_n,\ c_1^{\mathcal{A}}, \ldots, c_m^{\mathcal{A}}\}$$

на

$$\{b_1, \ldots, b_n,\ c_1^{\mathcal{B}}, \ldots, c_m^{\mathcal{B}}\}$$

так, что

$$\tau(a_1) = b_1, \ldots, \tau(a_n) = b_n,\ \tau(c_1^{\mathcal{A}}) = c_1^{\mathcal{B}}, \ldots, \tau(c_m^{\mathcal{A}}) = c_m^{\mathcal{B}}.$$

Это отображение τ называется локальным мономорфизмом $\mathcal{A}$ в $\mathcal{B}$, если для каждого символа отношения P из Ω и каждых $d_1, \ldots, d_k$ из

$$\{a_1, \ldots, a_n,\ c_1^{\mathcal{A}}, \ldots, c_m^{\mathcal{A}}\}$$

выполняется $P^{\mathcal{A}}(d_1, \ldots, d_k)$ *в* $\mathcal{A}$ *тогда и только тогда, когда выполняется* $P^{\mathcal{B}}(\tau(d_1), \ldots, \tau(d_k))$ *в* $\mathcal{B}$*, где* k *есть число аргументных мест* P.

Определение 1.1.4 (n-игры Эренфойхта на $\mathcal{A}$ и $\mathcal{B}$). *В n-игре Эренфойхта участвуют два игрока. Первый называется Разрушителем, а второй — Повторителем. Игра состоит из n ходов. На каждом ходе Разрушитель выбирает один элемент в одной из этих систем, а Повторитель после этого выбирает один элемент в другой из этих систем. Пусть в результате игры в системах $\mathcal{A}$ и $\mathcal{B}$ выбраны соответственно элементы $a_1, \ldots, a_n$ и $b_1, \ldots, b_n$. Пусть τ одно-однозначно отображает*

$$\{a_1, \ldots, a_n,\ c_1^{\mathcal{A}}, \ldots, c_m^{\mathcal{A}}\}$$

на

$$\{b_1, \ldots, b_n,\ c_1^{\mathcal{B}}, \ldots, c_m^{\mathcal{B}}\}$$

так, что

$$\tau(a_1) = b_1, \ldots, \tau(a_n) = b_n,\ \tau(c_1^{\mathcal{A}}) = c_1^{\mathcal{B}}, \ldots, \tau(c_m^{\mathcal{A}}) = c_m^{\mathcal{B}}.$$

Повторитель выиграл, если это отображение τ является локальным мономорфизмом. В противном случае выиграл Разрушитель. Повторитель имеет выигрышную стратегию в n-игре Эренфойхта на $\mathcal{A}$ и $\mathcal{B}$, если он может выиграть при любой игре Разрушителя.

Теорема 1.1 (об n-игре Эренфойхта). *Если Повторитель имеет выигрышную стратегию в n-игре Эренфойхта на $\mathcal{A}$ и $\mathcal{B}$, то $\mathcal{A}$ и $\mathcal{B}$ не различаются замкнутыми предваренными формулами, содержащими не более n кванторов.*

Доказательство. Индукцией по n.

При $n = 0$, если Повторитель имеет выигрышную стратегию в 0-игре, то τ является локальным мономорфизмом. Это означает, что рассматриваемые системы не различаются замкнутыми атомными формулами (такая формула имеет вид $P(d_1, \ldots, d_k)$ для какого-то P из Ω и каких-то $d_1, \ldots, d_k$ из $\{c_1, \ldots, c_m\}$, где k обозначает число аргументных мест у P, либо имеет вид $c_i = c_j$), значит, $\mathcal{A}$ и $\mathcal{B}$ не различаются и замкнутыми бескванторными формулами.

Пусть теорема верна для n. Докажем её для $n+1$. Пусть Повторитель имеет выигрышную стратегию в $(n+1)$-игре на $\mathcal{A}$ и $\mathcal{B}$. Замкнутая предваренная формула Ψ с $n+1$ квантором, неразличимость относительно которой систем $\mathcal{A}$ и $\mathcal{B}$ требуется доказывать, для некоторого переменного x и некоторой предваренной формулы Φ с n кванторами либо сама имеет вид $(\exists x)\Phi$, либо такой вид эквивалентен отрицанию этой формулы Ψ. Так как неразличимость относительно отрицания совпадает с неразличимостью относительно самой формулы, то мы будем предполагать, что такой вид имеет сама формула. Будем предполагать, что Ψ истинна в $\mathcal{A}$, и докажем, что эта формула истинна и в $\mathcal{B}$. Обратный случай сводится к этому простым переименованием $\mathcal{A}$ и $\mathcal{B}$.

Так как $(\exists x)\Phi$ истинна в $\mathcal{A}$, то существует такой элемент a_0 в $\mathcal{A}$, что Φ истинна для каждого такого состояния σ системы $\mathcal{A}$, для которого $\sigma(x) = a_0$. Пусть константа c_0 не входит в Ω. Это означает, что $(\Phi)^x_{c_0}$ истинна в обогащении $(\mathcal{A},\ a_0)$ системы $\mathcal{A}$, в котором c_0 интерпретируется как a_0. $(\Phi)^x_{c_0}$ получается из Φ заменой свободных вхождений x на c_0.

Пусть Разрушитель выбирает на первом шаге игры элемент a_0 в $\mathcal{A}$. Используя свою выигрышную стратегию, Повторитель может так выбрать элемент b_0 в $\mathcal{B}$, что дальше Повторитель имеет выигрышную стратегию в n-игре на $(\mathcal{A},\ a_0)$ и $(\mathcal{B},\ b_0)$.

По индукционному предположению, последние две системы не различаются замкнутыми предваренными формулами с n кванторами. Значит, $(\Phi)^x_{c_0}$ истинна в $(\mathcal{B},\ b_0)$. Это означает, что Ψ истинна в $\mathcal{B}$. ∎

Справедлива и обратная теорема о том, что если $\mathcal{A}$ и $\mathcal{B}$ не различаются замкнутыми предваренными формулами, содержащими не более n кванторов, то Повторитель имеет выигрышную стратегию в n-игре Эренфойхта на $\mathcal{A}$ и $\mathcal{B}$. Но нам эта обратная теорема не понадобится, и мы не будем её доказывать.

Напомним точные определения $\mathcal{P}$-вычислимого глобального предиката, и в частности, определение кода стандартной базы данных.

Определение 1.1.5 (кода стандартной базы данных). *Рассмотрим схему баз данных*

$$\Omega = \langle R_1^{(j_1)},\ R_2^{(j_2)}, \ldots, R_r^{(j_r)},\ c_1, \ldots, c_m \rangle,$$

где $c_1, c_2, \ldots, c_m$ — константы, а $R_1, R_2, \ldots, R_r$ — символы отношений местности, соответственно, $j_1,\ j_2, \ldots, j_r$. Мы считаем, что все элементы Ω упорядочены в приведенном порядке.

Пусть $\mathcal{A}$ — некоторая стандартная база данных для схемы Ω, то есть конечное основное множество $|\mathcal{A}|$ вместе с отображением, ко-

торое каждому предикатному символу $R_k \in \Omega$ *ставит в соответствие* j_k *-местный предикат*

$$R_k^{\mathcal{A}} \subseteq |\mathcal{A}|^{j_k}$$

на множестве $|\mathcal{A}|$, *а каждой константе* $c_k \in \Omega$ — *выделенный элемент* $c_k^{\mathcal{A}}$ *из* $|\mathcal{A}|$.

Как мы договорились, множество $|\mathcal{A}|$ *представляет собой начальный отрезок натурального ряда, то есть что, для* $n = ||\mathcal{A}||$, $|\mathcal{A}| = \{0,\ 1, \dots, n-1\}$. *При этом все элементы* $|\mathcal{A}|$ *естественным образом упорядочены. Этот порядок на элементах индуцирует и порядок на кортежах элементов, именно, для произвольного* $q \in \{0, 1, \dots\}$ *и для произвольных*

$$\alpha_1, \alpha_2, \dots, \alpha_q, \beta_1, \beta_2, \dots, \beta_q \in |\mathcal{A}|$$

считаем, что

$$\langle \alpha_1, \alpha_2, \dots, \alpha_q \rangle$$

меньше

$$\langle \beta_1, \beta_2, \dots, \beta_q \rangle,$$

если существует такое $s \le q$, *что*

$$\alpha_1 = \beta_1, \alpha_2 = \beta_2, \dots, \alpha_{s-1} = \beta_{s-1},$$

но $\alpha_s < \beta_s$. *Пусть*

$$\vartheta_1^q, \vartheta_2^q, \dots, \vartheta_{n^q}^q$$

перечисляют все кортежи из q *элементов множества*

$$|\mathcal{A}| = \{0,\ 1, \dots, n-1\},$$

взятые в определенном выше порядке, то есть

$$\vartheta_1^q = \langle 0,\ 0, \dots, 0 \rangle, \quad \vartheta_2^q = \langle 0,\ 0, \dots, 0,\ 1 \rangle, \dots,$$

$$\vartheta_{n^q}^q = \langle n-1,\ n-1, \dots, n-1 \rangle.$$

Кодом натурального числа $t \in \{0, 1, \dots\}$ *будем называть слово* $\#(t)$ *в алфавите* $\{0, 1\}$, *представляющее собой двоичную запись числа* t.

Кодом отношения $R_k^{\mathcal{A}} \subseteq |\mathcal{A}|^{j_k}$, *будем называть слово*

$$\#(R_k^{\mathcal{A}}) = \xi_1 \xi_2 \dots \xi_{n^{j_k}} \in \{0, 1\}^*$$

длины n^{j_k}, *в котором для* $u \le n^{j_k}$

$$\xi_u = \begin{cases} 1, & \text{если } \vartheta_u^{j_k} \in R_k^{\mathcal{A}}; \\ 0, & \text{если } \vartheta_u^{j_k} \notin R_k^{\mathcal{A}}. \end{cases}$$

Наконец, кодом самой стандартной базы данных $\mathcal{A}$ *будем называть слово*

$$\#(\mathcal{A}) \in \{0, 1, \wedge\}^*,$$

где

$$\#(\mathcal{A}) = \wedge\#(LAST)\wedge\#(c_1^{\mathcal{A}})\wedge\#(c_2^{\mathcal{A}})\wedge\cdots\wedge\#(c_m^{\mathcal{A}})\wedge\#(R_1^{\mathcal{A}})\wedge\cdots\wedge\#(R_r^{\mathcal{A}})\wedge$$

Отличие от [CT] состоит в том, что в качестве разделителя между кодами отношений и чисел мы теперь используем пробел, а не звёздочку. Ясно, что это различие несущественно.

Напомним теперь определение сложности глобального предиката.

Определение 1.1.6 (сложности глобального предиката).

Пусть Ω *— некоторая схема баз данных,* $\mathfrak{A}$ *— некоторый класс стандартных баз данных для схемы* Ω. *Пусть* Ξ *— булев* $\mathfrak{A}$*-глобальный предикат. Пусть* $\mathfrak{P}$ *— такой язык в алфавите* $\{0, 1, *\}$, *что*

$$\mathfrak{P} = \{\#(\mathcal{A}) \mid \mathcal{A} \in \Xi\}.$$

Если $\mathcal{C}$ *— некоторый такой класс сложности, что* $\mathfrak{P} \in \mathcal{C}$, *то будем говорить, что* $\mathfrak{A}$*-глобальный предикат* Ξ *имеет сложность* $\mathcal{C}$.

Напомним определение классов сложности $PTIME$ и $NPTIME$.

Пусть $f, g : \{0, 1, ..\} \to \{0, 1, ..\}$. Будем говорить, что функция f *асимптотически мажорируется* функцией g (или что g есть *асимптотическая мажоранта* для f), тогда и только тогда, когда

$$(\exists n)(\forall m \ge n) f(m) \le g(m).$$

Пусть $L \subseteq (\Sigma \setminus \{\wedge\})^*$ — некоторый язык, $u : \{0, 1, \dots\} \to \{0, 1, \dots\}$. Будем говорить, что u есть *верхняя оценка детерминированной (недетерминированной) временной сложности* языка L, если существует такой 1-МТ (соответственно, 1-НМТ) распознаватель M, что M определяет язык L, и функция t_M асимптотически мажорируется функцией u.

Здесь $t_M(n)$ есть время работы M на слове длины не более n в наихудшем останавливающемся случае. Другими словами, для определения $t_M(n)$ мы должны перебрать все останавливающиеся на словах длины не более n пути вычислений M, при которых M не проходит дважды одной и той же конфигурации, и взять максимум по числу тактов.

Класс всевозможных языков в алфавите $(\Sigma \setminus \{\wedge\})$ с верхней оценкой детерминированной временной сложности $f : \{0, 1, \dots\} \to \{0, 1, \dots\}$ обозначается $DTIME(f)$.

Класс всевозможных языков в алфавите $(\Sigma \setminus \{\wedge\})$ с верхней оценкой недетерминированной временной сложности $f : \{0, 1, \dots\} \to \{0, 1, \dots\}$ обозначается $NTIME(f)$.

$$\mathcal{P} = PTIME = \bigcup_{k>0} DTIME(n^k);$$

$$NP = NPTIME = \bigcup_{k>0} NTIME(n^k).$$

Подробное определение 1-МТ и 1-НМТ распознавателей приведено в [СТ, §6.1].

Грубо говоря, язык принадлежит к классу $PTIME$, если существует такая машина Тьюринга M с входными, рабочими и выходной лентами и такое натуральное число k, что

- M останавливается на любом входе,
- M делает не более n^k шагов, если длина входа не превосходит n,
- M воспринимает вход в том и только в том случае, когда вход представляет собой слово, принадлежащее рассматриваемому языку.

Язык принадлежит к классу $NPTIME$, если существует такая недетерминированная машина Тьюринга M с входными, рабочими и выходной лентами и такое натуральное число k, что

- если M воспринимает вход, то M воспринимает вход, сделав не более n^k шагов, если длина входа не превосходит n,
- M воспринимает вход в том и только в том случае, когда вход представляет собой слово, принадлежащее рассматриваемому языку.

Теперь мы хотим показать, что глобальные предикаты, задаваемые формулами логики предикатов, $\mathcal{P}$-вычислимы.

Итак, пусть Ω — реляционная сигнатура, Ω' — расширенная сигнатура, полученная добавлением символа для отношения линейного порядка к сигнатуре Ω. $L_{\Omega'}$ — язык логики предикатов в сигнатуре Ω', $\varphi \in L_{\Omega'}$ — формула этого языка, а список $x_1, x_2, \ldots, x_n$ задает все свободные переменные формулы φ (мы предполагаем, что множество всех переменных языка $L_{\Omega'}$ линейно упорядочено и перечисляем переменные в этом порядке). Всякая такая формула φ называется *элементарным запросом* (к базе данных со схемой Ω). При этом замкнутые формулы задают булевы элементарные запросы.

Пусть теперь $\mathcal{A}$ — некоторая стандартная база данных для схемы Ω, $|\mathcal{A}|$ — ее основное множество (носитель).

Напомним, что оценкой в системе $\mathcal{A}$ или состоянием системы $\mathcal{A}$ называется произвольное отображение σ множества переменных в носитель системы $\mathcal{A}$.

Ответом на запрос φ для базы данных $\mathcal{A}$ называется n-местный предикат $\varphi_{\mathcal{A}}$, определенный так:

$$\langle a_1, a_2, \ldots, a_n \rangle \in \varphi_{\mathcal{A}} \Leftrightarrow \sigma_{a_1, a_2, \ldots, a_n} \models \varphi$$

для произвольных $a_1, a_2, \ldots, a_n \in |\mathcal{A}|$, где

$$\sigma_{a_1, a_2, \ldots, a_n}(x) = \begin{cases} a_i, \text{ если } x \text{ есть } x_i \text{ и } 0 < i \leq n; \\ a_1 \text{ в остальных случаях.} \end{cases}$$

Другими словами, предикат $\varphi_{\mathcal{A}}$ определяется как множество точек истинности формулы φ в системе $\mathcal{A}$.

Ясно, что формула φ определяет предикат во всякой стандартной базе данных для схемы Ω, то есть глобальный предикат. Каждый глобальный предикат, который можно так задать некоторой формулой логики предикатов, называется *элементарным.*

Теорема 1.2. *Пусть* $\mathfrak{A}$ *— класс всех стандартных баз данных для схемы* Ω *и для булевого элементарного запроса* ϕ *пусть*

$$\mathfrak{P}_\phi = \{\#(\mathcal{A}) \mid \mathcal{A} \in \mathfrak{A}, \mathcal{A} \models \phi\}.$$

Тогда $\mathfrak{P}_\phi \in \mathcal{P} = PTIME$.

Неформальное доказательство. Не ограничивая общности, можно считать, что формула ϕ предваренная. Индукция по числу кванторов. Если кванторов нет, формула ϕ эквивалентна дизъюнкции конъюнкций базисных формул. Базисная формула — это либо атомная формула, либо отрицание атомной формулы. Проверка истинности сводится к проверке истинности нескольких конъюнкций базисных формул. Если хотя бы одна из конъюнкций истинна, то истинна и рассматриваемая формула. Проверка каждой конъюнкции сводится к проверке истинности нескольких базисных формул. Если все эти базисные формулы истинны, то истинна и их конъюнкция. В противном случае эта конъюнкция ложна. Для вычисления истинности $P(d_1, \ldots, d_k)$, где P — символ отношения местности k из схемы базы данных $\mathcal{A}$, надо использовать хранящийся на входной ленте код этой базы данных.

Для определения истинности формулы $(\exists x)\psi$ следует проверить истинность формул $(\psi)_j^x$ для $j = 0, \ldots, LAST$. Если хотя бы одна из этих формул истинна, то истинна и формула $(\exists x)\psi$. В противном случае эта формула ложна. ∎

1.2 Транзитивное замыкание. Подалгебра

Определение 1.2.1 (транзитивного отношения). *Бинарное отношение Q на множестве A называется транзитивным, если для любых элементов a, b и c множества A из истинности $Q(a, b)$ и $Q(b, c)$ следует истинность $Q(a, c)$.*

Теорема 1.3. *Пересечение любого множества транзитивных на A отношений является транзитивным на A отношением.*

Доказательство. Если для каких-то трех элементов пара первых из них и пара последних из них принадлежат пересечению, то обе эти пары принадлежат каждому отношению и, в силу транзитивности, пара, составленная из первого и последнего элементов, тоже принадлежит каждому отношению и, значит, пересечению всех рассматриваемых отношений. ∎

Всегда существует транзитивное отношение на A, содержащее заданное бинарное отношение Q на A. Действительно, тождественное отношение на A, которое истинно для любой пары элементов A, является транзитивным. Из теоремы следует, что пересечение всех транзитивных отношений, содержащих заданное бинарное отношение, является транзитивным отношением. Это пересечение является наименьшим транзитивным отношением, содержащим заданное отношение. Если заданное отношение само транзитивно, то это пересечение совпадает с заданным отношением.

Определение 1.2.2 (транзитивного замыкания). *Пересечение всех транзитивных отношений, содержащих заданное бинарное отношение, называется транзитивным замыканием заданного отношения.*

Пусть теперь множество A конечно, а Q является бинарным отношением на A. Тогда следующая конструкция строит транзитивное замыкание отношения Q.

Теорема 1.4 (о транзитивном замыкании). *Пусть R_0 — пусто, а R_{i+1} содержит те и только те пары (a, b), для которых существует такой элемент c множества A, что либо истинно $Q(a, b)$, либо одновременно $R_i(a, c)$ и $Q(c, b)$ истинны.*

Существует такое i, что $R_i = R_{i+1}$. Для этого i отношение R_i является транзитивным замыканием отношения Q.

Доказательство. Индукцией по j докажем, что $R_j \subseteq R_{j+1}$.

Это верно для $j = 0$, так как R_0 является пустым.

Если $R_j \subseteq R_{j+1}$, то $R_{j+1} \subseteq R_{j+2}$.

Так как $R_j \subseteq R_{j+1}$ для любого j, то последовательность

$$R_0,\ R_1, \ldots, R_j, \ldots$$

является неубывающей последовательностью подмножеств конечного множества A^2. По этой причине существует такое i, что $R_i = R_{i+1}$. Для этого i и любого j выполняется равенство

$$R_i = R_{i+j}.$$

Покажем, что R_i является транзитивным замыканием отношения Q.

Ясно, что R_j содержится в транзитивном замыкании Q для любого j.

Действительно, это верно для $j = 0$ в силу пустоты R_0. Пусть T является транзитивным замыканием отношения Q и R_j содержится в T. Покажем, что R_{j+1} тоже содержится в T. Действительно, $R_{j+1}(a, b)$ эквивалентно существованию такого c, что либо $Q(a, b)$, либо одновременно $R_j(a, c)$ и $Q(c, b)$. Если $Q(a, b)$, то $T(a, b)$. Пусть выполняется вторая возможность. По индукционному предположению и определению транзитивного замыкания, из этого следует, что $T(a, c)$ и $T(c, b)$. Из транзитивности T и этого следует $T(a, b)$.

Покажем, что $T \subseteq R_i$. По определению, T является пересечением всех транзитивных отношений, содержащих Q. Остается заметить, что R_i является транзитивным отношением.

Пусть $R_i(a,d)$ и $R_j(d,b)$. Индукцией по j докажем, что тогда $R_i(a,b)$.

При $j=0$ утверждение очевидно, так как R_0 пусто. Пусть утверждение верно для некоторого j и любых $a,\ d,\ b$.

$R_{j+1}(d,b)$ означает, что либо $Q(d,b)$, либо одновременно $R_j(d,c)$ и $Q(c,b)$ для некоторого c. В первом случае получаем $R_{i+1}(a,b)$ и $R_i(a,b)$. Во втором случае получаем $R_i(a,c)$ и $Q(c,b)$, что влечет $R_{i+1}(a,b)$. Но $R_{i+1}=R_i$. Значит, $R_i(a,b)$.

Итак, $R_i = T$. ∎

Будем теперь рассматривать схему базы данных

$$\langle P,\ c_1,\ c_2\rangle,$$

где P является символом бинарного отношения, а $c_1,\ c_2$ — константами. Базы данных для этой схемы являются упорядоченными графами с двумя выделенными элементами. Для такого графа $\mathcal{G}$ через $T^{\mathcal{G}}$ обозначим транзитивное замыкание отношения, которое соответствует символу P в $\mathcal{G}$.

Рассмотрим глобальный предикат $\mathfrak{T}$ на упорядоченных графах с двумя выделенными элементами, истинный на таком графе $\mathcal{G}$ тогда и только тогда, когда $T^{\mathcal{G}}(c_1^{\mathcal{G}},c_2^{\mathcal{G}})$ истинно.

Теорема 1.5 (полиномиальность транзитивного замыкания).

$$\mathfrak{T} \in \mathcal{P} = PTIME.$$

Неформальное доказательство. При построении транзитивного замыкания фактически делается не более квадрата числа элементов шагов. На каждом шаге добавляются все такие пары (a,b), для которых существует такой элемент c, что пара (a,c) уже получена на предыдущем шаге, а пара (c,b) лежит в $P^{\mathcal{G}}$. Имея код отношения, вычисленного на предыдущем шаге, и код $P^{\mathcal{G}}$, за полиномиальное время делаем следующий шаг. ∎

Пусть f является операцией местности n на множестве A. С этой операцией можно связать $(n+1)$-местное отношение P_f на этом же множестве A следующим образом:

для произвольного набора $a_1, \ldots, a_n, b$ элементов множества A истинность $P_f(a_1, \ldots, a_n, b)$ равносильна истинности равенства

$$f(a_1, \ldots, a_n) = b.$$

Разумеется, не всякое $(n+1)$-отношение P на A задает некоторую n-местную операцию на A. Необходимыми и достаточными условиями для этого являются:

- для любых $a_1, \ldots, a_n, b, c$ из $P(a_1, \ldots, a_n, b)$ и $P(a_1, \ldots, a_n, c)$ следует, что $b = c$;

- для любых $a_1, \ldots, a_n$ существует такой элемент b в A, для которого выполняется $P(a_1, \ldots, a_n, b)$.

Мы будем говорить, что отношение *представляет операцию*, если оно удовлетворяет указанным необходимым и достаточным условиям.

Так как эти условия можно задать формулами логики предикатов, то с точки зрения изучения выразительных возможностей можно не рассматривать вообще сигнатур, содержащих символы операций, заменяя их символами отношений и рассматривая только такие интерпретации этих добавленных символов отношений, для которых указанные формулы истинны. Однако с других точек зрения наличие символов операций существенно меняет дело.

Если наоборот сигнатура не содержит вообще символов отношений, то алгебраическая система такой сигнатуры называется *алгеброй*.

Подмножество алгебраической системы *замкнуто относительно некоторой операции*, определенной на основном множестве этой системы, если из того, что какие-то аргументы лежат в этом подмножестве, следует, что и значение рассматриваемой операции на этом наборе аргументов лежит в этом подмножестве. В частности, для константы это означает, что рассматриваемое подмножество содержит значение этой константы.

Операции, соответствующие сигнатурным символам операций в рассматриваемой алгебраической системе, называются *основными операциями* этой алгебраической системы. При этом элементы, соответствующие сигнатурным константам, называются также выделенными элементами этой алгебраической системы.

Определение 1.2.3 (подалгебры, подсистемы). *Непустое подмножество алгебры называется подалгеброй, если оно замкнуто относительно всех основных операций этой алгебры. Непустое подмноже-*

ство алгебраической системы называется подсистемой, если оно замкнуто относительно всех основных операций этой системы.

Если сигнатура не содержит символов операций, всякое непустое подмножество является подсистемой. В общем же случае всегда существует наименьшая подсистема, содержащая заданное непустое подмножество основного множества рассматриваемой алгебраической системы. Эта подсистема определяется как пересечение всех подсистем, содержащих рассматриваемое подмножество, и называется *подсистемой, порожденной этим подмножеством.* Для понимания корректности этого определения достаточно доказать следующую теорему.

Теорема 1.6. *Пересечение любого множества подсистем, содержащих заданное непустое подмножество рассматриваемой алгебраической системы, является подсистемой этой системы.*

Доказательство. Если какой-то набор элементов принадлежат пересечению, то этот набор принадлежат каждой подсистеме. Значит, значение некоторой основной операции на этом наборе принадлежит каждой подсистеме и потому принадлежит пересечению всех рассматриваемых подсистем. ■

Пусть теперь множество A конечно, а $\mathcal{A}$ является алгебраической системой, у которой A является основным множеством. Пусть X является непустым подмножеством множества A. Тогда следующая конструкция строит подсистему, порожденную X в $\mathcal{A}$.

Теорема 1.7 (о порожденной подсистеме). *Пусть R_0 — пусто, а R_{i+1} содержит те и только те элементы x из A, которые либо входят в X, либо являются выделенными элементами в $\mathcal{A}$, либо для которых существуют n, такая n-местная основная операция f в $\mathcal{A}$ и такие элементы $a_1, \dots, a_n$ из R_i, что $x = f(a_1, \dots, a_n)$.*

Существует такое i, что $R_i = R_{i+1}$. Для этого i множество R_i является подсистемой, порожденной X в $\mathcal{A}$.

Доказательство. Индукцией по j докажем, что $R_j \subseteq R_{j+1}$.

Это верно для $j = 0$, так как R_0 является пустым.

Если $R_j \subseteq R_{j+1}$, то $R_{j+1} \subseteq R_{j+2}$.

Так как $R_j \subseteq R_{j+1}$ для любого j, то последовательность

$$R_0,\ R_1, \dots, R_j, \dots$$

является неубывающей последовательностью подмножеств конечного множества A. По этой причине существует такое i, что $R_i = R_{i+1}$. Для этого i и любого j выполняется равенство

$$R_i = R_{i+j}.$$

Покажем, что R_i является подсистемой, порожденной X в $\mathcal{A}$.

Ясно, что R_j содержится в этой подсистеме для любого j.

Действительно, это верно для $j = 0$ в силу пустоты R_0. Пусть T является подсистемой, порожденной X, и R_j содержится в T. Покажем, что R_{j+1} тоже содержится в T. Действительно, $R_{j+1}(a)$ эквивалентно существованию такого n и таких $a_1, \ldots, a_n$ в R_j, что либо a является выделенным элементом или элементом X, либо $a = f(a_1, \ldots, a_n)$ для некоторой основной n-местной операции f. Если выполняется первая возможность, $a \in T$ по определению подсистемы, порожденной X. Пусть выполняется вторая возможность. По индукционному предположению и определению подсистемы, $a \in T$.

Покажем, что $T \subseteq R_i$. По определению, T является пересечением всех подсистем, содержащих X. Остается заметить, что R_i является подсистемой.

R_i содержит все выделенные элементы $\mathcal{A}$. Пусть $a_1, \ldots, a_n$ содержатся в R_i и f является n-местной основной операцией.

Тогда $f(a_1, \ldots, a_n)$ лежит в $R_{i+1} = R_i$.

Итак, $R_i = T$. ∎

Рассмотрим глобальный предикат $\mathfrak{T}_m$ на стандартных базах данных для схемы, полученной из какой-то сигнатуры заменой символов ненульместных операций на символы представляющих их отношений и обогащенной $(m+1)$-ой дополнительной константой. Интерпретации этих дополнительных констант будем называть дополнительными выделенными элементами. Каждой такой базе данных соответствует конечная алгебраическая система первоначальной сигнатуры, полученная заменой представляющих отношений на представляемые ими операции и отбрасыванием упорядочения, первого и последнего элементов, операции получения следующего элемента и дополнительных выделенных элементов. Этот глобальный предикат истинен на такой базе данных тогда и только тогда, когда в соответствующей алгебраической системе первоначальной сигнатуры последний дополнительный выделенный элемент принадлежит подсистеме, порожденной остальными дополнительными выделенными элементами.

Теорема 1.8 (полиномиальность порожденной подсистемы).

$$\mathfrak{T}_m \in \mathcal{P} = PTIME.$$

Неформальное доказательство. При построении порожденной подсистемы фактически делается не более числа элементов шагов. На каждом шаге добавляются все такие элементы, которые получаются как значения основных операций на наборах аргументов, принадлежащих подмножеству, полученному на предыдущем шаге. Имея код подмножества, вычисленного на предыдущем шаге, и код стандартной базы данных, за полиномиальное время делаем следующий шаг. ∎

1.3 Неполнота FO для $PTIME$

Целью настоящего параграфа является доказательство теоремы о том, что транзитивное замыкание не выразимо в языке FO логики предикатов. Для краткости мы будем формулы логики предикатов называть FO формулами.

Рассмотрим для положительного r две конечные стандартные базы данных $\mathcal{A}_r$ и $\mathcal{B}_r$ для схемы, содержащей символ P бинарного отношения и две константы c_0 и c_1.

Основное множество обеих баз данных представляет собой множество

$$A_r \;=\; B_r \;=\; \{\langle i,j\rangle \mid i,j \leqslant 2^r\}.$$

Таким образом, каждая из баз данных содержит ровно $(2^r+1)(2^r+1)$ элементов.

Формально мы нарушаем при этом условие, что основным множеством стандартной базы данных должен быть начальный отрезок натуральных чисел. Однако это условие нужно только для того, чтобы стандартным образом определить наименьший, наибольший элементы и линейный порядок. Если это сделать отдельно, то условие становиться несущественным.

Интерпретацией c_0 в обеих базах данных является $a_0 \;=\; b_0 \;=\; \langle 2^{r-1},0\rangle$. Интерпретацией c_1 в $\mathcal{A}_r$ является $a_1 \;=\; \langle 2^{r-1},2^r\rangle$, а интерпретацией c_1 в $\mathcal{B}_r$ является $b_1 \;=\; \langle 2^{r-1}-1,2^r\rangle$. Линейный порядок в обеих базах данных определяется одинаково и задаётся лексикографически:

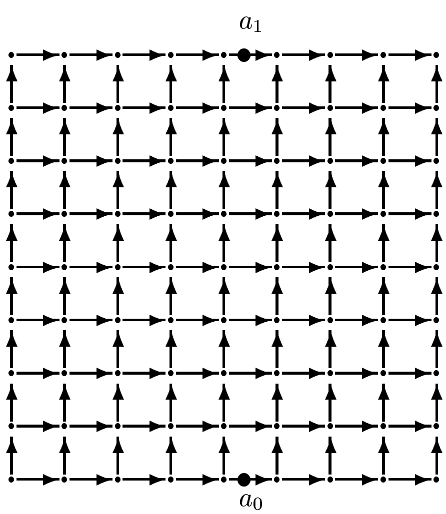
$$\langle i,j\rangle < \langle u,v\rangle \iff (i<u \text{ или } (i=u \text{ и } j<v)).$$

Это означает, что точка, лежащая выше, больше, а из точек, лежащих на одном уровне, больше та, которая правее. Отношение P определяется в обеих базах данных одинаково:

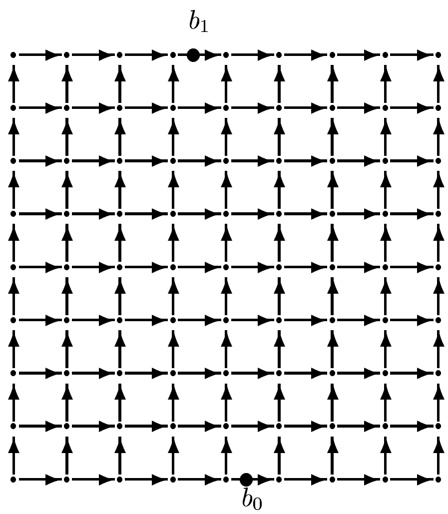
$$P^{\mathcal{A}}(\langle i,j\rangle,\langle u,v\rangle) \;\Leftrightarrow\; P^{\mathcal{B}}(\langle i,j\rangle,\langle u,v\rangle) \;\Leftrightarrow$$
$$((i=u \text{ и } v=j+1) \text{ или } (u=i+1 \text{ и } v=j)).$$

Это означает, что вторая точка либо лежит в той же строке, но в следующем столбце, либо лежит в том же столбце, но в следующей строке.

Для $r=3$ обе системы приведены на следующем рисунке:



Система $\mathcal{A}_3$



Система $\mathcal{B}_3$

Очевидно, что $\mathfrak{T}(\mathcal{A}_r)$ истинно, но $\mathfrak{T}(\mathcal{B}_r)$ ложно для любого положительного r.

Теорема 1.9 (неэлементарность транзитивного замыкания).
Глобальный предикат $\mathfrak{T}$ не является элементарным.

Доказательство. От противного. Пусть этот глобальный предикат задается FO формулой ϕ. Тогда $\mathcal{A}_r \models \phi$ и $\mathcal{B}_r \models \neg\phi$ для любого положительного r.

Можно считать, что формула ϕ является предваренной и содержит s кванторов [CT, теоремы 3.24 и 3.28]. Мы получим противоречие, доказав, что $\mathcal{A}_{2s}$ и $\mathcal{B}_{2s}$ на самом деле не различаются никакой предваренной формулой, содержащей не более s кванторов. По теореме 1.1, для этого достаточно доказать, что Повторитель имеет выигрышную стратегию в s-игре Эренфойхта на $\mathcal{A}_{2s}$ и $\mathcal{B}_{2s}$.

Опишем эту стратегию. Расстоянием между $\langle\alpha,\gamma\rangle$ и $\langle\beta,\delta\rangle$ равно

$$|\alpha-\beta|+|\gamma-\delta|.$$

Мы будем использовать на каждом шаге i игры два числа — радиус $\rho(i)$ для этого шага и сдвиг $\sigma(i)$ для выбираемых на этом шаге элементов. Мы полагаем: $\rho(i)=2^{s-i}$. На шаге i выбираются элементы $a_{i+1}=\langle\alpha_{i+1},\gamma_{i+1}\rangle$ из $\mathcal{A}_{2s}$ и $b_{i+1}=\langle\beta_{i+1},\delta_{i+1}\rangle$ из $\mathcal{B}_{2s}$. Элементы a_i и b_i для $i=0,1,\ldots,s,s+1$ называются соответствующими друг другу. Стратегия состоит в том, что $\gamma_{i+1}=\delta_{i+1}$, а $\alpha_{i+1}=\beta_{i+1}+\sigma(i)$. Таким образом стратегия сводится к определению $\sigma(i)$. Сразу скажем, что сдвиг на каждом шаге равен либо 0, либо 1. Сдвиг равен 1 в том и только том случае, когда выбранный на этом шаге Разрушителем элемент находится на расстоянии, не превышающем радиуса для этого шага, от какого-то такого ранее выбранного элемента этой же базы данных, сдвиг которого тоже равен 1. При этом сдвиг a_0 и b_0 равен 0, а сдвиг a_1 и b_1 равен 1. Эти элементы и только они выбраны до начала игры.

Для корректности этой стратегии надо понять, что если на каком-то шаге в $\mathcal{A}_{2s}$ выбирается элемент первого столбца, то сдвиг на этом шаге равен 0. В самом деле, пусть до шага i каждый выбранный элемент, для которого сдвиг равен 1, находится на расстоянии, превышающем $2^{2s-i}-1$, от любого элемента первого столбца. Тогда после шага i (перед шагом $i+1$) каждый элемент, для которого сдвиг равен 1, находится на расстоянии, превышающем

$$2^{2s-i}-1-2^{s-i}>2^{2s-1}-2^{2s-i-1}-1=2^{2s-i-1}-1.$$

Это объясняется тем, что каждый элемент, сдвиг которого равен 1, отстоит на расстоянии, не превышающем 2^{s-i}, от какого-то ранее выбранного элемента, сдвиг которого тоже равен 1.

Лемма 1.10. *Если выбранные после какого-то шага игры элементы одной из рассматриваемых баз данных находятся на расстоянии, не превышающем радиуса для этого шага, то соответствующие элементы другой из рассматриваемых баз данных находятся на таком же расстоянии.*

Если один из выбранных после какого-то шага игры элементов одной из рассматриваемых баз данных меньше другого выбранного после этого шага элемента, то соответствующий первому элементу в другой из рассматриваемых баз данных тоже меньше элемента, соответствующего второму из рассматриваемых выбранных элементов в другой из рассматриваемых баз данных.

Доказательство. Индукцией по номеру шага игры. Надо заметить, что выбранные элементы, расстояние между которыми не превосходит радиуса, имеют одинаковый сдвиг. ∎

Из леммы сразу следует, что отображение, переводящее выбранные элементы в им соответствующие, является локальным мономорфизмом. ∎

Определение 1.3.1 (полноты класса глобальных предикатов). *Пусть Ω — некоторая реляционная сигнатура, $\mathfrak{A}$ — некоторый класс стандартных баз данных для схемы Ω, $\mathcal{C}$ — некоторый класс сложности.*

Пусть $\mathfrak{B}$ — некоторый такой класс $\mathfrak{A}$-глобальных предикатов, что каждый $\mathfrak{A}$-глобальный предикат $\mathfrak{P} \in \mathfrak{B}$ имеет сложность $\mathcal{C}$. Будем говорить, что $\mathfrak{B}$ является $\mathcal{C}$-полным для $\mathfrak{A}$, если для всякого $\mathfrak{A}$-глобального предиката $\Xi : \mathcal{A} \longmapsto \Xi^{\mathcal{A}}$, имеющего сложность $\mathcal{C}$, найдется такой $\mathfrak{P} \in \mathfrak{B}$, что для всякой системы $\mathcal{A} \in \mathfrak{A}$ предикаты $\Xi^{\mathcal{A}}$ и $\mathfrak{P}^{\mathcal{A}}$ совпадают.

Из теорем 1.2, 1.5, и 1.9 следует

Теорема 1.11 (неполнота элементарных запросов для $PTIME$). *Класс элементарных глобальных предикатов лежит в $PTIME$, но не является $PTIME$-полным.* ∎

1.4 Язык неподвижных точек

Конструкция, использованная для получения транзитивного замыкания и порожденной подалгебры, может быть использована и в более общем случае.

Определение 1.4.1 (положительного вхождения). *Символ отношения P входит положительно в атомную формулу $P(t_1, \ldots, t_n)$ и не входит в эту атомную формулу отрицательно. Никакой другой символ отношения не входит в эту атомную формулу ни положительно, ни отрицательно. В атомную формулу $t_1 = t_2$ никакой символ* *отношения не входит ни положительно, ни отрицательно. Символ отношения входит положительно (соответственно, отрицательно) в конъюнкцию или дизъюнкцию двух формул, если этот символ входит положительно (соответственно, отрицательно) в одну из этих формул. Символ отношения входит положительно (отрицательно) в отрицание формулы, если этот символ входит отрицательно (положительно) в саму эту формулу. Навешивание квантора по предметному переменному не меняет положительных и отрицательных вхождений символов отношений.*

Будем теперь предполагать, что n-местный символ отношения Q не входит в рассматриваемую схему баз данных. Рассмотрим формулу Φ, сигнатура которой получается из сигнатуры рассматриваемых стандартных баз данных добавлением этого символа Q. Будем предполагать, что эта формула не содержит импликаций, символ Q входит в неё только положительно, а все свободные переменные этой формулы содержатся среди $x_1, \ldots, x_n$. Другими словами, кроме символов, входящих в рассматриваемую схему баз данных, и символа для отношения порядка, в записи этой формулы может также использоваться и дополнительный символ Q, но Q входит в эту формулу только положительно. Такую формулу будем называть *расширенной n-формулой*. Для каждого n *расширенной формулой* будем называть каждую расширенную n-формулу.

Через $(\mathcal{A}, R, a_1, \ldots, a_n)$ для стандартной базы данных $\mathcal{A}$ для рассматриваемой схемы баз данных будем обозначать обогащение $\mathcal{A}$, в котором Q интерпретируется n-местным отношением R на основном множестве базы данных A, а переменным $x_1, \ldots, x_n$ приписаны соответственно значения $a_1, \ldots, a_n$.

Лемма 1.12. *Если $R_1 \subseteq R_2$ и $(\mathcal{A}, R_1, a_1, \ldots, a_n) \models \Phi$, то $(\mathcal{A}, R_2, a_1, \ldots, a_n) \models \Phi$.*

Доказательство. Если Φ есть $Q(x_1, \ldots, x_n)$ или вообще не содержит Q, утверждение очевидно. Если утверждение верно для двух формул, то оно верно для конъюнкции и дизъюнкции этих формул и для формулы, полученной из одной из них навешиванием квантора. ∎

Пусть теперь $\mathcal{A}$ является стандартной базой данных для рассматриваемой схемы баз данных.

Определение 1.4.2 (неподвижной точки формулы). *Полагаем R_0 пустым и полагаем*

$$R_{i+1} = \{\langle a_1, \ldots, a_n \rangle \mid (\mathcal{A}, R_i, a_1, \ldots, a_n) \models \Phi\}.$$

Существует такое i, что $R_i = R_{i+1}$. Для этого i отношение R_i называется неподвижной точкой формулы Φ на $\mathcal{A}$ и обозначается через $LFP(\Phi)(\mathcal{A})$.

Теорема 1.13 (о неподвижной точке формулы).

Определенный таким образом глобальный предикат $LFP(\Phi)$ на множестве всех стандартных баз данных для рассматриваемой схемы баз данных лежит в $PTIME$.

Неформальное доказательство. Существование указанного i следует из конечности $\mathcal{A}$ и леммы 1.12. При построении неподвижной точки фактически делается не более n-той степени от числа элементов шагов. На шаге $j+1$ вычисляется R_{j+1} как множество всех точек истинности формулы Φ на $(\mathcal{A}, R_j, a_1, \ldots, a_n)$. По коду этой системы код R_{j+1} вычисляется за полиномиальное время. ∎

Примерами глобальных предикатов, представимых как неподвижные точки расширенных n-формул, являются определенные ранее транзитивное замыкание и порожденная подсистема.

В самом деле, транзитивное замыкание определяется для баз данных для схемы $\langle P \rangle$ как неподвижная точка формулы

$$(P(x,y) \vee (\exists z)(Q(x,z) \& P(z,y))).$$

Пусть рассматриваемая сигнатура содержит, например, константы

$$c_1, \ldots, c_m$$

и символы операций

$$f^{(1)}, g^{(2)}, h^{(3)}.$$

Пусть

$$X = \{x_1, \ldots, x_r\}.$$

Тогда порожденная X подсистема определяется как неподвижная точка формулы

$$\begin{gathered}(x = c_1 \vee \cdots \vee x = c_m \vee x = x_1 \vee \cdots \vee x = x_r \vee \\ (\exists z_1)(\exists z_2)(\exists z_3) \\ (Q(z_1) \& Q(z_2) \& Q(z_3) \& (x = f(z_1) \vee x = g(z_1, z_2) \vee x = h(z_1, z_2, z_3)))).\end{gathered}$$

Вместо символов операций можно рассматривать символы представляющих их отношений P_f, P_g, P_h. Тогда формула примет вид

$$
\begin{gathered}
(x = c_1 \vee \dots \vee x = c_m \vee x = x_1 \vee \dots \vee x = x_r \vee \\
(\exists z_1)(\exists z_2)(\exists z_3) \\
(Q(z_1)\&Q(z_2)\&Q(z_3)\&(P_f(x, z_1) \vee P_g(x, z_1, z_2) \vee P_h(x, z_1, z_2, z_3)))).
\end{gathered}
$$

Приведем теперь в некотором смысле примеры другого рода.

Рассмотрим расширенную формулу

$$((y = 0\&z = x) \vee (\exists v)(\exists u)(y = v'\&Q(x, v, u)\&z = u')). \tag{2}$$

Неподвижная точка $LFP(\phi)$ этой формулы ϕ задаёт на каждой стандартной базе данных трехместное отношение, истинное для элементов a, b, c этой базы данных тогда и только тогда, когда

$$c = \begin{cases} a + b, \text{ если } a + b \leqslant LAST, \\ LAST \text{ в остальных случаях.} \end{cases}$$

Это утверждение легко доказывается индукцией по b.

Неподвижная точка расширенной формулы

$$
\begin{gathered}
((x = 0\&y = 0\&z = 0) \vee (x = 1\&y = 0\&z = 1) \vee \\
((x = 0 \vee x = 1)\&y > 0\&z = 2) \vee \\
(\exists v)(\exists u)(x = u'\&y < v\&Q(u, v, 0)\&Q(u, y, z)) \vee \\
(\exists v)(\exists u)(x = u'\&Q(u, v, 0)\&Q(u, y, 2)\&z = 2) \vee \\
(\exists u)(\forall w)(x = u'\& \\
(w > y \to (Q(u, w, 1) \vee Q(u, w, 2)))\&Q(u, y, 0)\&z = 1) \vee \\
(\exists u)(\exists v)(\forall w)(x = u'\& \\
(w > y \to (Q(u, w, 1) \vee Q(u, w, 2)))\&Q(u, y, 1)\&v < y\&Q(u, v, 0)\&z = 0) \vee \\
(\exists u)(\forall w)(x = u'\&(Q(u, w, 1) \vee Q(u, w, 2))\& \\
((y = 0\&z = 1) \vee (\exists v)(y = v'\&Q(u, v, 1)\&z = 0) \vee \\
(\exists v)(y = v'\&Q(u, v, 2)\&z = 2)))
\end{gathered}
$$

задаёт на каждой стандартной базе данных трехместное отношение, истинное для элементов a, b, c этой базы данных тогда и только тогда, когда

$$c = \begin{cases} b\text{-ой от начала цифре в двоичной записи } a, \text{ если } b < len(a), \\ 2 \text{ в остальных случаях,} \end{cases}$$

где $len(a)$ обозначает число цифр в двоичной записи числа a, 2 обозначает $0''$, а 1 обозначает $0'$.

Определение 1.4.3 (языка FP). *FP формулами рассматриваемой сигнатуры являются:*

- *каждая FO формула,*
- *неподвижная точка каждой расширенной формулы,*
- *отрицание FP формулы,*
- *конъюнкция и дизъюнкция FP формул,*
- *формулы $(\exists x)\Phi$ и $(\forall x)\Phi$ для каждой FP формулы Φ и каждой переменной x.*

Каждая FP формула определяет предикат во всякой стандартной базе данных для рассматриваемой схемы, то есть глобальный предикат, если истинность FO формул и истинность конъюнкции, дизъюнкции и отрицания понимать обычным образом, кванторам приписывать обычный смысл, а истинность неподвижной точки формулы понимать по определению неподвижной точки. Каждый глобальный предикат, который можно так задать некоторой FP формулой, называется *FP глобальным предикатом.*

Приведем пример. Тройку предметных переменных, относящихся к стандартной базе данных, можно рассматривать как задающую трёхзначное число в $(LAST+1)$-ичной системе счисления. При этом первая переменная задаёт старший разряд или равна нулю, если число однозначное или двузначное, вторая переменная задаёт средний разряд или равна нулю, если число однозначное, а третья переменная задаёт младший разряд. Аналогично задаёт число и четверка переменных.

Мы желаем иметь FP формулу, утверждающую, что четвёрка переменных задаёт число, являющееся суммой чисел, задаваемых двумя тройками переменных.

Построим такую формулу. Она должна иметь

$$x_1, x_2, x_3, y_1, y_2, y_3, z_0, z_1, z_2, z_3$$

в качестве свободных переменных и не иметь других свободных переменных. Мы будем использовать

$$s_1, s_2, s_3, s_4, u, v, w$$

в качестве связанных переменных. $\bar{x}$ и $\bar{y}$ обозначают вход, $\bar{z}$ обозначает результат, $\bar{s}$ обозначает переполнение. Переполнение в каждом разряде равно нулю или равно единице.

Пусть ϕ задана (2) и ψ есть $LFP(\phi)$. Через $\psi(u, v, w)$ как обычно будем обозначать формулу, которая получена из ψ подстановкой термов u, v, w вместо свободных переменных в указанном порядке. Вместо $\psi(u, v, w)$ пишем $u + v = w$. Более сложные выражения являются понятными сокращениями. Например,

$$u + v + w = t$$

заменяет

$$(\exists x)(u + v = x \& x + w = t).$$

Нужной формулой будет

$$\begin{gathered}(\exists s_1)(\exists s_2)(\exists s_3)(\exists s_4)(s_4 = 0 \& z_0 = s_1 \& \\ \&_{i=1}^{3}((\exists u)(\exists v)(\exists w)(s_{i+1} + x_i + u < LAST \& \\ s_{i+1} + x_i + v = LAST \& v = u' \& v < y_i \& v' + w = y_i \& \\ v + w < y_i \& z_i = w \& s_i = 1) \vee \\ (\exists u)(s_{i+1} + x_i + u < LAST \& y_i = u' \& z_i = s_{i+1} + x_i + y_i \& s_i = 0)))\end{gathered}$$

Аналогично можно построить формулу, которая утверждает, что набор длины $k + 1$ из переменных задаёт сумму чисел, заданных двумя наборами из k переменных.

Из теорем 1.2 о полиномиальности распознавания элементарных запросов и 1.13 о полиномиальности вычисления неподвижной точки следует

Теорема 1.14 (о полиномиальном распознавании FP запросов). *Каждый FP глобальный предикат лежит в $PTIME$.* ∎

1.5 Полнота FP для $PTIME$

Теорема 1.15 (Сазонов, Варди, Иммерман, Ливчак).

Каждый глобальный предикат из $PTIME$ является FP *глобальным предикатом.*

Доказательство. Пусть k — некоторое положительное натуральное число. Рассмотрим 1-МТ распознаватель M [CT, §6.1], делающий на любом входе, длина которого не превосходит n, не более n^k тактов работы и определяющий рассматриваемый глобальный предикат. Напомним, что входом является код стандартной базы данных. Так как имеется только конечное число одноэлементных и двухэлементных стандартных баз данных, каждая из которых описывается некоторой FO формулой, то будем предполагать, что входом является база данных, число элементов которой больше двух. Так как длина входа является полиномом от числа элементов этой базы данных, а число элементов равно $(LAST+1)$, то, увеличив при необходимости k, можно считать, что M делает не более $LAST^k$ тактов работы, если вход является кодом стандартной базы данных, основное множество которой есть

$$\{0, 1, \ldots, LAST\}.$$

Построим FP формулу ϕ_M, задающую тот же глобальный предикат.

При построении мы используем идеи, похожие на те, которые использовались в [CT, доказательство теоремы 6.7].

Итак, пусть $M = \langle \{0, 1, \wedge\}, Q, \pi, q_0 \rangle$, где:

Q — алфавит состояний, $Q = \{q_0, q_1, \ldots, q_m\}$,

$q_0 \in Q$ — начальное состояние,

$$\begin{aligned} \pi : (\{\wedge, 0, 1\} \times \{\wedge, 0, 1\} \times Q) \to \\ (\{\wedge, 0, 1\} \times Q \times \{L, R, N, S\} \times \{L, R, N\} \times \{\wedge, 0, 1\}) \end{aligned} \quad (3)$$

есть функция переходов. Можно предполагать, что получив результат, M сразу же останавливается.

Описание работы 1-МТ распознавателя M будет задаваться расширенной формулой ψ. Дополнительный символ Q при этом будет $(2k+5)$-местным. В этой формуле номер ячейки на ленте и номер такта работы распознавателя будут кодироваться наборами предметных переменных $x_1, \ldots, x_k$ и $t_1, \ldots, t_k$ длины k. Всего возможных значений такого набора имеется $(LAST+1)^k$. Так как M делает не более n^k тактов работы на входе с числом элементов не более $n+1$, то этих значений достаточно. Остальными свободными переменными будут s, o, q, l, d для содержимого ячейки, обозреваемости ячейки, номеров состояния и ленты и результата работы. При этом число элементов рассматриваемой стандартной базы данных не должно быть меньше трёх и не должно быть меньше числа состояний M. Так как имеется только конечное число стандартных баз данных, число элементов которых не превосходит заданного числа, а каждая из этих баз данных описывается некоторой FO формулой, то можно предполагать, что число элементов рассматриваемой стандартной базы данных не меньше трех и не меньше числа состояний 1-МТ распознавателя M. $s = 0$, $s = 1$, $s = 2$ соответственно означают, что обозревается $\wedge$, 1 или 0. $o = 0$, $o = 1$ соответственно означают, что ячейка обозревается или не обозревается. Номер входной ленты равен 0, а номер рабочей ленты равен 1. Результат работы равен 2, если результата пока нет.

Формула ψ является дизъюнкцией двух расширенных формул ψ_b и ψ_n, описывающих начальную конфигурацию и правила для получения следующей конфигурации.

Для наглядности мы будем использовать некоторые сокращения. Для натурального числа i и переменной x формула $x = i$ определяется по индукции. $x = 0$ является формулой по определению стандартных баз данных (стандартные базы данных для схемы Ω являются конечными алгебраическими системами обогащенной сигнатуры, включающей, кроме символов из Ω, также дополнительные символы $<$, 0, $LAST$, $'$ для линейного порядка, наименьшего и наибольшего элементов и для операции получения следующего элемента). $x = i + 1$ есть $(\exists y)(y = i \& x = y')$. $\bar{x}$ обозначает $x_1, \ldots, x_k$. $\bar{x}$ рассматривается как запись числа в $(LAST + 1)$-ичной системе счисления младший разряд которого есть x_k, а старший есть x_1. При этом $\bar{x}'$ обозначает запись следующего числа. $\bar{t}$ обозначает $t_1, \ldots, t_k$. $\bar{t}$ рассматривается как запись числа в $(LAST+1)$-ичной системе счисления младший разряд которого есть t_k, а старший есть t_1. При этом $\bar{t}'$ обозначает запись следующего числа. Аналогично понимаются записи $\bar{u}$ и для других переменных u. Наши рассмотрения организованы таким образом, что число, все цифры которого есть $LAST$, никогда не встречается.

Формула ψ_b не зависит от программы M, а зависит только от рассматриваемой схемы баз данных. Формула ψ_b утверждает, что $d = 2$ (результат пока не получен), номер такта равен 0 и M находится в начальном состоянии, обозревает ячейки с номером 2 на входной и номером 0 на рабочей лентах, на входной ленте первые две ячейки пусты, а затем написан код стандартной базы данных, а рабочая лента пуста. Другими словами, $\bar{t} = 0$, $q = 0$, и либо $l = 0$, $\bar{x} = 2$ и $o = 1$, а s задаёт первый символ кода рассматриваемой стандартной базы данных, либо $l = 0$, $o = 0$, $\bar{x} \neq 2$, а s задаёт соответствующий символ кода, либо $l = 1$, $\bar{x} = 0$, $o = 1$ и $s = 0$, либо $l = 1$, $\bar{x} \neq 0$, $o = 0$ и $s = 0$.

Единственный не вполне тривиальный момент при написании ψ_b состоит в описании входной ленты, другими словами, в определении s для заданного значения $\bar{x}$, если $l = 0$. Надо предположить, что k и число элементов рассматриваемой стандартной базы данных достаточно велики и использовать Q с большими значениями d для вычисления двоичных цифр констант и вычисления сложения. На входной ленте после трёх пробелов написаны двоичные цифры $LAST$ и затем после пробела двоичные цифры первой константы, пробел и двоичные цифры следующей константы и так далее. После цифр последней константы написан пробел, а затем разделенные одним пробелом коды основных отношений. Пусть $Q(\bar{y}, \bar{0}, 0, 0, 0, j, 3)$ задаёт номер ячейки на входной ленте, с которой начинается запись j-ой константы, а $Q(\bar{z}, \bar{0}, 0, 0, 0, j, 4)$ задаёт номер ячейки на входной ленте, с которой начинается запись j-ого отношения. Тогда в ячейке входной ленты с номером $\bar{y}+\bar{u}$ записана $\bar{u}$-ая цифра j-ой константы, если $\bar{u} < len(c_j^{\mathcal{A}})$, а в ячейке входной ленты с номером $\bar{z}+\bar{u}$ записано 1 или 0 в зависимости от истинности или ложности

$$R_j^{\mathcal{A}}(u_{k-k_j+1}, u_{k-k_j+2}, \ldots, u_{k-k_j+k_j})$$

при условии, что

$$u_1 = \cdots = u_{k-k_j} = 0.$$

Здесь R_j является именем j-ого по порядку основного отношения рассматриваемой схемы баз данных и k_j обозначает число аргументных мест у R_j, а $len(c_j^{\mathcal{A}})$ обозначает число цифр в двоичной записи числа $c_j^{\mathcal{A}}$.

Мы предлагаем читателю в качестве упражнения явно построить ψ_b для случая, когда схема базы данных есть

$$\langle P^{(2)}, R^{(3)}, c_1, c_2 \rangle.$$

Более простой вариант построения ψ_b получился бы, если бы мы допускали распознаватели с несколькими входными и несколькими рабочими лентами. При этом каждую константу и каждое основное отношение можно задавать на отдельной входной ленте. При задании константы можно при этом задавать не её двоичные цифры, а просто писать соответствующее число единиц. Легко заметить, что имея указанное представление, его можно сначала преобразовать за полиномиальное время в код стандартной базы данных. По этой причине, если глобальный предикат распознаётся за полиномиальное время 1-МТ распознавателем, то он распознаётся за полиномиальное время и некоторым таким распознавателем. Как замечено в [CT, теорема 6.2], верно и обратное утверждение. Построение ψ_n для новых распознавателей усложнилось бы незначительно.

Формула ψ_n является дизъюнкцией формул ψ_r, где r пробегает все команды 1-МТ M. Формула ψ_r предполагает существование предыдущего такта работы. Если правило r задаёт движение головки влево на рабочей ленте, то формула ψ_r рассматривает отдельно случай, когда на рабочей ленте на предыдущем такте обозревается ячейка с номером 0. В этом случае предварительно рабочая лента сдвигается на одну ячейку вправо (заменяется содержимое ячейки с номером 0 согласно выполняемому правилу, затем утверждается, что содержимым некоторой ячейки в рассматриваемый момент является содержимое предыдущей ячейки в предыдущий момент, в рассматриваемый момент ячейка номера 0 обозревается и является пустой). В остальных случаях заменяется содержимое обозреваемой ячейки на рабочей ленте на предыдущем такте и утверждается, что содержимое ячейки в рассматриваемый момент равняется содержимому этой же ячейки в предыдущий момент с учетом произведенной замены, а обозреваемой является на рабочей ленте ячейка, расположенная непосредственно левее, правее обозреваемой или совпадающая с обозреваемой на предыдущем такте работы, согласно правилу r. Во всех случаях аналогично описывается содержимое ячеек и обозреваемая ячейка на входной ленте. Входную ленту ни в каких случаях сдвигать не надо.

Рассмотрим, для примера, команду

$$01q_5 \Rightarrow \wedge q_4 RL\wedge$$

Эта команда означает, что M, обозревая в состоянии q_5 символ 0 на входной ленте и символ 1 на рабочей ленте, печатает на рабочей ленте $\wedge$, сдвигается вправо на рабочей ленте, сдвигается влево на входной ленте и переходит в состояние q_4 (мы сохраняем обозначения, принятые в [CT, §6.1]). При этом M не останавливается и не выдаёт результат.

Для этой команды r в качестве ψ_r выбираем следующую формулу:

$$\begin{gathered}
(\exists \bar{t}_1)(\exists s_1)(\exists o_1)(\exists q_1)(\exists d_1)(Q(\bar{x}, \bar{t}_1, s_1, o_1, q_1, l, d_1)\& \\
\bar{t} = (\bar{t}_1)'\& q_1 = 5\& q = 4\& d_1 = 2\& d = 2\& \\
(\exists \bar{x}_2)(\exists s_2)(\exists o_2)(\exists l_2)(\exists l_3)(\exists \bar{x}_3)(\exists s_3) \\
(Q(\bar{x}_2, \bar{t}_1, s_2, o_2, q_1, l_2, d_1)\& Q(\bar{x}_3, \bar{t}_1, s_3, o_2, q_1, l_3, d_1)\& \\
s_2 = 2\& l_2 = 0\& o_2 = 1\& s_3 = 1\& l_3 = 1\& \\
((o_1 = 1\& l = 1\& s = 0) \vee ((o_1 = 0 \vee l = 0)\& s = s_1))\& \\
((l = 1\& \bar{x} = (\bar{x}_3)'\& o = 1) \vee (l = 0\& \bar{x}' = \bar{x}_2\& o = 1) \vee \\
(l = 1\& \bar{x} \neq (\bar{x}_3)'\& o = 0) \vee (l = 0\& \bar{x}' \neq \bar{x}_2\& o = 0)))) \qquad (4)
\end{gathered}$$

Неподвижная точка построенной так формулы ψ истинна в стандартной базе данных $\mathcal{A}$ на состоянии σ тогда и только тогда, когда

1-МТ M, работая на коде этой базы данных, в момент $\sigma(\bar{t})$ находится в состоянии номера $\sigma(q)$ и ячейка номера $\sigma(\bar{x})$ на ленте номера $\sigma(l)$ в этот момент содержит символ номера $\sigma(s)$. Эта ячейка обозревается в том и только том случае, когда $\sigma(o) = 1$. При этом $\sigma(d)$ есть результат работы M на рассматриваемом входе, если этот результат уже получен, или есть 2.

Доказательство проводится индукцией по номеру шага построения неподвижной точки. На шаге 1 в неподвижную точку добавляется описание начальной конфигурации и утверждение верно для $\sigma(\bar{t}) = 0$. На следующем шаге добавляется описание следующей конфигурации. Неподвижная точка, таким образом, содержит описание всех конфигураций и ничего больше.

По доказанному, в качестве ϕ_M можно взять формулу

$$(\exists\bar{x})(\exists\bar{t})(\exists s)(\exists o)(\exists q)(\exists l)(\exists d)(LFP(\psi)\&d = 1).$$

∎

1.6 Язык ∃-FO. Полнота ∃-FO для $NPTIME$

Определение 1.6.1 (∃-FO формулы). *Если ϕ является расширенной n-формулой, то $(\exists Q)\phi$ называется ∃-FO формулой. Эта формула истинна на состоянии σ в стандартной базе данных $\mathcal{A}$ тогда и только тогда, когда существует такое n-местное отношение R на основном множестве этой базы данных, что ϕ истинна на этом же состоянии σ в алгебраической системе $(\mathcal{A},\ R)$, обогащающей $\mathcal{A}$, в которой R является интерпретацией Q.*

Можно было бы разрешать навешивать и несколько кванторов существования по отношениям. Однако этот случай немедленно сводится к случаю одного квантора, так как несколько дополнительных символов отношений можно заменить одним, местность которого на 1 больше самой большой из местностей рассматриваемых отношений. При этом этот дополнительный аргумент просто служит для нумерации рассматриваемых отношений. Если этот аргумент равен i, то это на самом деле означает, что рассматривается i-ое дополнительное отношение.

Теорема 1.16. ∃-FO $\subseteq$ *NPTIME.*

Неформальное доказательство. Нужный 2-НМТ распознаватель для $(\exists Q)\phi$ сначала случайным образом дописывает код Q к коду $\mathcal{A}$, а затем использует полиномиальный 1-МТ распознаватель для ϕ. Для этого этот 2-НМТ распознаватель сначала копирует входную ленту на вторую рабочую, после этого дописывает на второй рабочей ленте случайным образом код Q и затем использует эту вторую рабочую ленту в качестве ленты данных, копируя работу 1-МТ распознавателя для ϕ. Этот 2-НМТ распознаватель работает полиномиальное время и его можно моделировать некоторым 1-НМТ распознавателем, тоже работающим полиномиальное время [CT, теорема 6.2]. ∎

Теорема 1.17 (Фейгин). *NPTIME* $\subseteq \exists$-FO.

Неформальное доказательство. Рассмотрим произвольный 1-НМТ распознаватель, работающий полиномиальное время. Можно считать, что для каждого набора из обозреваемых символов на входной и рабочей лентах и состояния этот распознаватель имеет ровно две команды. Если имеется одна команда, её можно просто повторить, а если имеется более двух команд, можно добавить новое состояние и либо выполнять первую команду, либо стоять на месте на обеих лентах, не менять символ на рабочей ленте и перейти в это новое состояние. Для этого нового состояния и рассматриваемого набора обозреваемых на лентах символов предусмотреть оставшиеся команды. После конечного числа таких преобразований получим 1-НМТ распознаватель, который для каждого набора из обозреваемых символов на входной и рабочей лентах и состояния имеет ровно две команды.

Рассмотрим дополнительное отношение Q_2 соответствующей местности, которое истинно на наборе своих аргументов тогда и только тогда, когда на шаге, номер которого задаётся этим набором, выбирается первая из этих двух возможных команд.

Полученный 1-НМТ распознаватель можно переделать в такой 1-МТ распознаватель M с двумя входными лентами, на первой из которых содержится код $\mathcal{A}$, а на второй помещается код введенного дополнительного отношения Q_2, что команды M получаются из команд рассматриваемого 1-НМТ распознавателя добавлением в левую часть команды обозреваемого на второй входной ленте символа и добавлением в правую часть сдвига вправо на второй входной ленте. При этом для каждого набора из обозреваемых символов на первой входной и рабочей лентах и состояния и обозреваемого нуля на второй входной ленте выбирается правая часть соответствующего первого правила, а для каждого набора из обозреваемых символов на первой входной и рабочей лентах и состояния и обозреваемой единицы на второй входной ленте выбирается правая часть соответствующего второго правила.

Пусть расширенная формула ψ описывает работу этого 1-МТ распознавателя M. Дополнительный символ Q_1 при этом будет $(2k+5)$-местным. Входные ленты имеют номера 0 и 1, а рабочая лента имеет номер 2. В этой формуле номер ячейки на ленте и номер такта работы распознавателя будут кодироваться наборами предметных переменных $x_1, \dots, x_k$ и $t_1, \dots, t_k$ длины k. Остальными свободными переменными будут s, o, q, l, d для содержимого ячейки, обозреваемости ячейки, номеров состояния и ленты и результата работы. Пусть

$$\bar{v} = (\bar{x}, \bar{t}, s, o, q, l, d).$$

Неподвижная точка построенной так формулы ψ истинна в стандартной базе данных $(\mathcal{A}, Q_2)$ на состоянии σ тогда и только тогда, когда 1-МТ M, работая на коде этой базы данных, в момент $\sigma(\bar{t})$ находится в состоянии номера $\sigma(q)$ и ячейка номера $\sigma(\bar{x})$ на ленте номера $\sigma(l)$ в этот момент содержит символ номера $\sigma(s)$. Эта ячейка обозревается в том и только том случае, когда $\sigma(o) = 1$. При этом $\sigma(d)$ есть результат работы M на рассматриваемом входе, если этот результат уже получен, или есть 2.

Переделаем эту формулу ψ следующим образом. Рассмотрим кортеж переменных $\bar{z}$, не входящих в ψ, длины $2k+5$ и символ отношения Q_3 местности $4k+10$. Заменим в ψ каждое вхождение подформулы $Q_1(\bar{u})$ на вхождение подформулы $Q_3(\bar{z}, \bar{u})$. Полученную формулу обозначим через ψ_1. Очевидно, что формула

$$((\exists \bar{z}) Q_3(\bar{z}, \bar{v}) \& \\ (\forall \bar{z})(\forall \bar{v})(\neg Q_3(0, \bar{v}) \& (Q_3((\bar{z})', \bar{v}) \leftrightarrow \psi_1))) \quad (5)$$

задаёт неподвижную точку формулы ψ. Обозначим эту формулу (5) через ϕ.

По доказанному, в качестве ∃-FO формулы, задающей глобальный предикат, задаваемый рассматриваемым 1-НМТ распознавателем, можно взять формулу

$$(\exists Q_2)(\exists Q_3)(\exists \bar{x})(\exists \bar{t})(\exists s)(\exists o)(\exists q)(\exists l)(\exists d)(\phi \& d = 1).$$

∎

Следствие 1.18. FP⊆ ∃-FO.

Можно подумать, что теоремы 1.16 и 1.17 и следствие 1.18 доказаны только для стандартных баз данных, другими словами, для линейно упорядоченных конечных алгебраических систем. Однако на самом деле наличие упорядочения задаётся $\exists$-FO формулой (говорящей, что существует дополнительное бинарное отношение, задающее линейный порядок). По этой причине эти теоремы и следствие верны для класса произвольных конечных алгебраических систем.

2 РОЛЬ УПОРЯДОЧЕНИЯ

2.1 Введение

База данных представляет собой несколько конечных таблиц. Обычно удобно выбирать элементы строк этих таблиц из фиксированной бесконечной области — универсума. Но тогда можно использовать в запросах и общие знания об универсуме. При этом возникают интересные вопросы о конечности ответа на запрос и о зависимости выразительной силы языка запросов от сигнатуры универсума. Обсуждению этих вопросов посвящен этот раздел.

2.2 Бесконечные области

Базы данных предназначены для хранения текущей информации о как-то структурированной предметной области. В каждый момент времени эта информация является *конечной* и представляет собой *конечный* набор *конечных* таблиц ([Cod70, Cod72]). Обычно число таблиц и устройство каждой таблицы не меняются с течением времени, но меняются строки таблиц. Могут добавляться новые строки и удаляться некоторые старые. Строки хранящихся таблиц представляют собой конечные последовательности элементов. Число элементов каждой последовательности фиксировано для фиксированной таблицы. Устройство таблицы практически и есть число элементов в каждой строке этой таблицы. Более формально (или более научно) каждая таблица — это конечноместное конечное отношение, а сама база данных — это конечный набор конечноместных конечных отношений. Для удобства разговора о базе данных каждому ее отношению приписывают некоторое имя с указанием числа аргументов (или местности) этого имени отношения. *Схема* (или сигнатура) базы данных и есть конечная последовательность этих имен отношений с указанием местности каждого имени. Схема базы данных фиксируется раз и навсегда. В каждый момент времени именам отношений из этой схемы присвоены некоторые конечные отношения соответствующих местностей. Это — *состояние* базы данных в данный момент.

Например, база данных "Отец-сын"представляется в виде одного бинарного отношения (или таблицы с двумя столбцами).

Постепенно мы собираем все больше и больше информации об отцах и сыновьях, состояния этой базы данных меняются, но ее схема (имя одного бинарного отношения) остается неизменной.

Для осмысления хранящейся информации, разумеется, безразлично, чем являются элементы таблиц. Другими словами, если переобозначить элементы хранящихся таблиц, заменяя в каждой таблице каждую строку на строку образов элементов первоначальной строки при этом переобозначении, то новое состояние будет представлять ту же информацию.

По этой причине обычно удобно предполагать, что существует некоторая фиксированная бесконечная область, из которой выбираются элементы строчек таблиц для любого состояния базы данных.

В качестве такой области обычно выбирают целые или рациональные числа, или строчки символов некоторого фиксированного алфавита, или что-то еще подобное. Эту выбранную область мы называем *универсумом.*

Но как только такое соглашение принимается, возникают новые проблемы, обсуждению которых и посвящен этот раздел.

Прежде всего, универсум может быть снабжен некоторыми отношениями, константами и операциями. Например, натуральные числа можно рассматривать вместе с отношением порядка и операциями сложения и умножения (или только с одной операцией сложения, или с еще какими-то другими отношениями и операциями). В каждом конкретном случае должно быть ясно указано, какие константы, отношения и операции рассматриваются на универсуме. Перечисление их имен с указанием местностей составляет *сигнатуру* универсума. Эти дополнительные отношения и операции по своей природе бесконечны. Они представляют собой общие знания, которые тоже можно использовать, наряду с конечной хранящейся в данный момент специальной информацией. Интерпретация этих дополнительных символов (сигнатурных символов универсума), следовательно, постоянна и не зависит от состояния базы данных.

Смысл хранения информации в базе данных состоит в возможности ответов на различные запросы к базе данных. Эти запросы должны формулироваться на некотором специальном языке — языке запросов. В качестве простейшего языка запросов обычно используют язык FO логики предикатов первого порядка ([End72]). Запросами при этом являются формулы логики предикатов первого порядка с равенством, в которых можно использовать как имена констант, отношений и операций универсума, так и имена отношений из схемы рассматриваемой базы данных.

Вернемся к нашему примеру базы данных "Отец-сын". В качестве универсума выберем область всех слов некоторого алфавита. Пусть F

будет имя используемого в базе данных бинарного отношения. Тогда формула $M(x)$:

$$\exists y, z(y \neq z \wedge F(x,y) \wedge F(x,z))$$

представляет унарное отношение (таблицу с одним столбцом), состоящее из тех x, которые имеют более одного сына. В то же время формула $G(x,z)$:

$$\exists y(F(x,y) \wedge F(y,z))$$

представляет таблицу "Сын сына".

Мы бы хотели, чтобы результирующие отношения (ответы на запросы) были бы всегда конечными. Однако некоторые FO формулы дают бесконечные ответы. Понятно, что такой формулой является $\neg F(x,y)$. Хуже, что формула $M(x) \vee G(x,z)$ тоже может давать бесконечный ответ, так как ответом будет любая пара (x,z), для которой выполняется $M(x)$. Бесконечный ответ могут давать и другие запросы.

Формула, которая *может* давать бесконечный ответ, называется *бесконечной* или *опасной* в противоположность *конечным* или *безопасным* формулам, которые всегда дают конечный ответ.

В частности, формула $M(x) \vee G(x,z)$ является опасной. Эта формула дает бесконечный ответ в случае, когда в базе данных имеется лицо, имеющее более одного сына.

2.3 Конечно заданные отношения

Таким образом, некоторые запросы могут давать бесконечный ответ даже для конечных состояний базы данных. По этой причине мы вынуждены рассматривать и бесконечные состояния (в которых некоторые хранящиеся таблицы являются бесконечными).

Но как задать бесконечную таблицу?

Простейший способ — использовать для этого формулы логики предикатов первого порядка. Мы уже отмечали, что такие формулы могут задавать бесконечные отношения.

Трудность в том, что в общем случае не существует способа для проверки истинности такой формулы на заданном наборе значений ее переменных. По этой причине лучше ограничиться бескванторными формулами, для которых в случае рекурсивного универсума (операции и отношения которого эффективно заданы) такая проверка возможна.

Канелакис и его соавторы ([KKR90, KKR95]) предложили рассматривать только линейно упорядоченные универсумы и для них задавать отношения базы данных бескванторными формулами логики предикатов первого порядка, использующими, кроме символов предметных переменных, отношение порядка и символы для элементов универсума. Таким образом, в данном состоянии базы данных каждому символу отношения из схемы этой базы данных сопоставлена бескванторная формула, в записи которой используются только пропозициональные связки, символы предметных переменных, символы для изображения конкретных элементов универсума, символ равенства и символ отношения порядка. Ясно, что в таких формулах можно не использовать отрицание и импликацию (можно ограничиться использованием из пропозициональных связок только конъюнкции и дизъюнкции; такие бескванторные формулы называются *положительными*). Число предметных переменных в сопоставляемой формуле должно равняться местности рассматриваемого символа из рассматриваемой схемы базы данных. При этом предметные переменные должны быть линейно упорядочены, так что при задании кортежа элементов универсума той же длины, что и местность рассматриваемого символа отношения из этой схемы базы данных, ясно, какое значение приписывается каждому предметному переменному (первый элемент кортежа — это значение первого переменного, второй элемент кортежа — это значение второго переменного и так далее).

Данный кортеж элементов универсума принадлежит отношению, задаваемому рассматриваемой бескванторной формулой, если эта формула истинна на этом кортеже.

Отношения, задаваемые таким образом бескванторными формулами, называются *конечно заданными.* Состояние базы данных называется *конечно заданным*, если все его отношения конечно заданы.

Понятно, что каждое конечное отношение может быть конечно задано, значит, каждое конечное состояние может быть конечно задано.

Канелакис и его соавторы ([KKR90, KKR95]) также заметили, что если элементарная теория линейного порядка рассматриваемого универсума допускает элиминацию кванторов, то *для любого конечно заданного состояния ответ на запрос представляется в виде конечно заданного отношения, если этот запрос представлен в виде* FO *формулы, использующей, кроме связок и кванторов, только символы из схемы базы данных, символы равенства и порядка, символы для элементов универсума и символы предметных переменных.*

Примером универсума, порядок на котором допускает элиминацию кванторов, являются рациональные числа.

Представляя логическую необходимость, конечно заданные базы данных вместе с тем необходимы и во многих приложениях, например, в географических базах данных ([KKR90, KKR95]).

Проиллюстрируем сказанное одним простым примером.

Рассмотрим в качестве универсума рациональные числа вместе с обычным отношением порядка. Пусть схема базы данных состоит из одного бинарного символа отношения F. Пусть формула

$$x = 3 \vee (x < 8 \wedge x > 5 \wedge y < x)$$

задает F, предполагая, что x — первое переменное, а y — второе. Рассмотрим запрос

$$\exists z(\neg F(x, z) \wedge F(z, y)).$$

Можно вычислить, что ответ на этот запрос задается формулой

$$(x \leqslant 5 \wedge x > 3) \vee x \geqslant 8 \vee x < 3 \vee (x < 8 \wedge x > 5 \wedge y < 8).$$

2.4 Безопасные формулы

Мы в этом разделе ограничимся рассмотрением только счетных универсумов.

Будем предполагать, что среди имен констант универсума имеются имена для всех элементов универсума (при этом интерпретацией имени элемента является сам этот элемент). Предположим также, что элементарная теория универсума разрешима. Тогда ясно, что каждый безопасный запрос является вычислимым.

Действительно, пусть $F(\vec{x})$ задает безопасный запрос. Каждую конечную таблицу можно задать бескванторной формулой, используя только имена констант универсума. Поэтому, подставляя вместо имен отношений базы данных задающие их формулы, мы преобразуем формулу $F(\vec{x})$ в формулу $F'(\vec{x})$, в которой уже не используются символы из схемы базы данных.

Например, если схема базы данных состоит из одного бинарного имени R, интерпретация которого в рассматриваемом состоянии есть

$$\langle a_1, b_1 \rangle, \langle a_2, b_2 \rangle, \ldots, \langle a_r, b_r \rangle,$$

то можно каждое вхождение $R(x, y)$ заменить на

$$((x = a_1 \wedge y = b_1) \vee (x = a_2 \wedge y = b_2) \vee \cdots \vee (x = a_r \wedge y = b_r)).$$

Если формула $\exists\vec{x}(F'(\vec{x}))$ ложна, ответом на запрос является пустое отношение. Пусть эта формула истина. Перечисляя в каком-то порядке все наборы элементов универсума подходящей длины, будем выделять из них те наборы, которые удовлетворяют формуле $F'(\vec{x})$. Пусть $\vec{a}$ — это первый набор, для которого $F'(\vec{x})$ истинно. Это дает первую строку ответа.

Далее рассмотрим формулу

$$\exists\vec{x}(\vec{x} \neq \vec{a} \wedge (F'(\vec{x}))).$$

Если она ложна, ответ построен. В противном случае продолжаем перечислять наборы и находим второй набор, на котором формула $F'(\vec{x})$ истина. И так далее.

Так как запрос безопасен, через конечное число шагов мы получим искомый ответ.

Итак, полезно знать, является ли заданный запрос безопасным. Это было понято в [Ull82], где Ульман поставил вопрос, существует ли алгоритм для различения опасных и безопасных формул. Эта проблема известна как *проблема безопасности.* Она решена отрицательно в [Di 69], [Var81] и [АГСШ86]. Ответ отрицателен для любого бесконечного универсума и любой схемы базы данных, содержащей неодноместное отношение.

Более специализированной проблемой является проблема определения безопасных формул для *заданного* конечного состояния. Эта проблема известна как *проблема безопасности для состояния* ([АГСШ86]). Ответ зависит от универсума. В [АГСШ86] и [AH91] показано, что для многих универсумов эта проблема разрешима.

В [Var81] и [Ull82] предложен, а затем развит во многих публикациях (например, [Ull88], [Van91] и [Hir91]) следующий подход. Так как невозможно определять, является ли запрос безопасным, может быть можно ввести синтаксические ограничения на вид допускаемых формул таким образом, чтобы каждый безопасный запрос представлялся в виде формулы рассматриваемого вида, а каждая формула этого вида была бы безопасной. Более научно, существует ли такой рекурсивный подкласс безопасных формул, что каждая безопасная формула эквивалентна относительно всех конечных состояний рассматриваемой схемы базы данных одной из формул этого подкласса? Ответ, конечно, может зависеть от рассматриваемых универсума и схемы. Верно ли это для всякого рекурсивного универсума и всякой схемы базы данных?

Эта проблема называется проблемой *эффективного синтаксиса* для безопасных запросов. Можно было бы интересоваться не только рекурсивными, но и рекурсивно перечислимыми подклассами. Однако давно и хорошо известно, что существование рекурсивно перечислимых подклассов влечет существование и рекурсивных подклассов тоже.

В [Ull82] и [Ull88]) замечено, что для универсумов с пустой сигнатурой эффективный синтаксис существует. Вопрос о существовании синтаксиса, эффективного в смысле вычислительной сложности, обсуждался в [Van91].

В [ST95] предложен пример рекурсивного универсума с разрешимой элементарной теорией, в котором нет эффективного синтаксиса для безопасных запросов и в котором проблема безопасности для состояния неразрешима.

2.5 Упорядоченные универсумы и генерические запросы

Далее будем рассматривать линейно упорядоченные универсумы. Это означает, что в сигнатуре универсума выделено специальное бинарное отношение, интерпретацией которого является некоторый линейный порядок. Мы будем использовать $<$ для обозначения этого линейного порядка.

Примером такого универсума являются упорядоченные рациональные числа. В этом случае сигнатура универсума состоит только из символа, обозначающего линейный порядок.

Рассмотрим для этого универсума некоторое конечно заданное состояние s некоторой схемы базы данных. Как уже упоминалось в разделе 2.3, каждый FO запрос переводит это состояние в некоторое конечно заданное отношение R. Рассмотрим некоторое одно-однозначное отображение ϕ рациональных чисел на рациональные числа, которое сохраняет порядок (если $x < y$, то $\phi(x) < \phi(y)$; для каждого рационального числа x существует такое рациональное число y, что $x = \phi(y)$). Такие отображения называются *$<$-автоморфизмами*. Для каждого символа отношения из рассматриваемой схемы базы данных заменим в бескванторной формуле, конечно задающей это отношение, каждую константу c на $\phi(c)$. Получим новое конечно заданное состояние $\phi(s)$ этой же схемы. Все, кто хоть немного изучал математическую логику, знают, что ответом на рассматриваемый запрос для состояния $\phi(s)$ будет отношение $\phi(R)$, конечное задание для которого получается из конечного задания R заменой каждой константы c на $\phi(c)$. В этом смысле все FO запросы являются *$<$-генерическими*. Это означает, что эти запросы сохраняются при $<$-автоморфизмах универсума. Это верно не только для конечных или конечно заданных состояний, но и для любых состояний вообще.

Однако существуют очень простые генерические запросы, которые невозможно задать FO формулами. Даже для конечных состояний. Например, для схемы, состоящей из одного имени отношения, которое унарно, нельзя задать FO формулой запрос, говорящий, что общее число элементов этого отношения нечетно. Более содержательные примеры можно найти в [KKR90, KKR95, KG94].

Мы рассматриваем некоторый упорядоченный универсум сигнатуры $<$ и некоторое обогащение его сигнатуры. Проще говоря, используем либо только знания о порядке на этом универсуме, либо также некоторые дополнительные общие знания.

Хотя мы дальше сосредоточим внимание на $<$-генеричности, надо заметить, что первоначальное понятие генеричности касалось $=$-генеричности ([CH80]) над конечными состояниями. Речь шла о запросах, которые сохраняются при любых перестановках элементов универсума (некоторые из этих перестановок нарушают упорядочение; например, для $x < y < z$ существует такая перестановка ϕ, что $\phi(x) = z, \phi(z) = x, \phi(y) = y$).

Заметим, что хотя специалисты по базам данных используют термин "генеричность"(по-английски, "genericity"), Ю.Ш.Гуревич (который не публиковал работ, но одним из первых обратил внимание на эти проблемы) предпочитает термин "инвариантность", возможно, лучше отражающий суть дела.

Фактически речь идет о сравнении универсума с пустой сигнатурой и линейно упорядоченного универсума сигнатуры $<$. В случае пустой сигнатуры FO запросы сохраняются при любых перестановках универсума и, значит, являются $=$-генерическими. Это замечено в [CH80].

Ю.Ш.Гуревич ([Gur90]) показал для любого бесконечного линейно упорядоченного универсума, что существуют схема базы данных и $=$-генерический запрос, который над конечными состояниями FO выразим с использованием $<$, но не является FO выразимым в этом же универсуме, но с пустой сигнатурой. Пример Гуревича был сообщен одному из авторов еще в 1990 году, но до сих пор не опубликован.

Рассмотрим такую структуру S над универсумом U с непересекающимися конечными одноместными отношениями P, Q и двухместным отношением $E \subseteq P \times Q$, что для каждого подмножества X множества P существует и только один такой элемент y в Q, что

$$X = \{x : \ E(x, y)\}.$$

Таким образом, E отождествляет Q с множеством всех подмножеств множества P. Пусть K будет совокупность всех таких структур S. Класс K является FO выразимым над конечными состояниями. В самом деле, конечная структура S принадлежит к классу K тогда и только тогда, когда выполнены следующие условия:

- $E \subseteq P \times Q$;
- существует y в Q, для которого $\{x : xEy\}$ пусто;
- если $y \in Q$, $X = \{x : xEy\}$, $u \in P$ и $u \notin X$, то существует z в Q, для которого $\{x : xEz\}$ есть объединение X и $\{u\}$;
- y и z из Q равны тогда и только тогда, когда $\{x : xEy\} = \{x : xEz\}$.

Можно проверить, что запрос "Число элементов P четно" не является FO выразимым над K без использования какого-либо линейного порядка. Но легко понять, что этот запрос является =-генерическим и FO выразимым над K с использованием произвольного фиксированного линейного порядка. Действительно, формула "существует такое подмножество X множества P (иначе, существует такой y в Q), что (для $X = \{x : xEy\}$)

- X содержит первый элемент множества P;
- X содержит некоторый элемент из P тогда и только тогда, когда X не содержит предшественника этого элемента в P;
- X не содержит последний элемент множества P"

выражает этот запрос.

Таким образом, обогащение линейным порядком универсума с пустой сигнатурой для некоторых схем баз данных позволяет FO выразить больше =-генерических запросов. Но, как уже отмечалось, нечетность числа элементов унарного отношения для схемы, содержащей только имя этого отношения, продолжает оставаться невыразимой.

Тем более удивительно, что для многих универсумов дальнейшее обогащение сигнатуры уже не увеличивает FO выразимость =-генерических запросов. Мы доказываем, что при этом даже не увеличивается FO выразимость <-генерических запросов. Так как каждый =-генерический запрос является также <-генерическим, то из второго утверждения первое следует непосредственно.

Для краткости, утверждение, что рассматриваемое обогащение универсума дополнительными отношениями, операциями или константами не увеличивает FO выразимость <-генерических запросов, называется трансляционным результатом или результатом о трансляции.

В некоторых случаях результат о трансляции не имеет места. Например, допуская использование сложения и умножения, на натуральных, целых или рациональных числах мы можем FO выразить все вычислимые запросы, что, как уже отмечалось, нельзя сделать, используя только порядок. Но для некоторых других универсумов вопрос о справедливости результата о трансляции может быть очень трудным.

Пусть $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ и $\mathbb{R}$ обозначают множества всех натуральных, всех целых, всех рациональных и всех вещественных чисел. Наиболее интересными вопросами о трансляционных результатах были вопросы для следующих универсумов:

1. $(\mathbb{Q}, <, +)$ и $(\mathbb{R}, <, +)$;
2. $(\mathbb{Z}, <, +)$ и $(\mathbb{N}, <, +)$;
3. $(\mathbb{R}, <, +, \times)$.

Например, в [GST95, GS95] рассматривались $<$-генерические FO запросы для универсума $\mathbb{Q}$, рассматриваемого вместе с отношением $<$ обычного линейного порядка и обычной операцией $+$ сложения.

Было показано, что FO запросы, не использующие сложения, имеют сложность AC^0 ([KG94]). Поэтому пытались, используя сложение, определить какой-то $<$-генерический запрос, не лежащий в AC^0, в частности, одно из известных таких свойств графов.

Однако все эти попытки оказались безуспешными. Наконец, в [GST95, GS95] была доказана AC^0 сложность всех расширенных (использующих сложение) FO запросов над конечно заданными состояниями, но только для случая, когда в рассматриваемом конечном задании все константы являются *целыми.* Вопрос же о трансляционном результате оставался открытым. В [GS94] даже отмечалось, что отсутствует техника для такого рода доказательств в этой области.

Это уже история. Теперь настало время сформулировать нашу проблему в общем виде.

Рассмотрим некоторый список Ω имен отношений. Рассмотрим две сигнатуры $L_0 = \{<\}$ и $L = \{<\} \bigcup \Omega$. Для схемы базы данных $SC = \{R_1, \ldots, R_n\}$ пусть $L_0^+ = L_0 \bigcup SC$ и $L^+ = L \bigcup SC$. FO формулы сигнатуры L_0^+ и задаваемые ими запросы назовем *ограниченными,* а FO формулы сигнатуры L^+ и задаваемые ими запросы назовем *расширенными.* Ограниченные запросы относятся к универсуму сигнатуры $<$, в котором интерпретацией $<$ является линейное упорядочение универсума, а расширенные запросы относятся к этому универсуму, обогащенному дополнительными отношениями, операциями и константами. $<$-генерические запросы будем называть генерическими.

Проблемы, которые мы изучаем, состоят в следующем:

- Каким условиям должен удовлетворять универсум, чтобы для него имел место трансляционный результат (другими словами, чтобы для любой схемы базы данных каждый расширенный локально генерический запрос был эквивалентен над конечными состояниями какому-то ограниченному запросу)?

- Можно ли так обогатить сигнатуру рассматриваемого линейного порядка с разрешимой элементарной теорией, чтобы обогащенный универсум имел разрешимую элементарную теорию и чтобы существовал для этого обогащения такой локально генерический расширенный запрос, который не был бы эквивалентен над конечными состояниями никакому ограниченному запросу (другими словами, чтобы трансляционный результат не имел места для этого обогащенного универсума)?

- Справедлив ли результат о трансляции для рассматриваемого обогащения сигнатуры рассматриваемого универсума?

Для обычного линейного порядка на рациональных числах (или на целых числах, или на натуральных числах) ответ на второй вопрос нам неизвестен. Мы предполагаем, что в этих случаях такого обогащения сигнатуры не существует. Насколько мы понимаем, это — один из наиболее интересных открытых вопросов.

Первый вопрос подробно обсуждается в этой статье.

Эта общая проблема допускает *различные интерпретации.*

- "Генерический"можно понимать как локально генерический, называя локально генерическими для конечно заданных состояний запросы, которые сохраняются при любых локальных $<$-мономорфизмах (отображениях, переводящих одно конечное подмножество универсума в другое с сохранением порядка) с условием, что это конечное подмножество содержит как набор элементов, для которого проверяется истинность запроса, так и все константы, которые встречаются в рассматриваемом конечном задании рассматриваемого состояния. Но можно использовать и другие понятия генеричности, например, сохранение при любых $<$-автоморфизмах универсума. Именно эти запросы, как и ранее, мы называем генерическими, называя локально генерическими для конечно заданных состояний запросы, которые сохраняются при локальных $<$-мономорфизмах. Понятно, что любой локально генерический запрос является генерическим (для конечно заданных состояний). Для многих универсумов эти два понятия генеричности вообще совпадают. Так будет для вещественных и рациональных чисел и вообще для любых дважды транзитивных линейных порядков. Но для некоторых универсумом эти понятия различны. Например, для натуральных чисел вообще нет нетривиальных перестановок, сохраняющих порядок. Значит, любой запрос является генерическим, но, разумеется, далеко не любой будет локально генерическим. Для целых чисел каждое сохраняющее порядок отображение задается сложением с некоторым фиксированным числом. Поэтому запрос, утверждающий, что существуют как четные, так и нечетные числа в одноместном отношении, является генерическим, но не локально генерическим.

- Эквивалентность можно понимать как существование алгоритма для построения эквивалентной формулы, но можно понимать просто как существование эквивалентной формулы.

- Проблема может рассматриваться для всех состояний, или только для конечно заданных состояний, или даже только для конечных состояний.

Если рассматривать любые состояния, то результатов о трансляции нет даже для очень простых универсумов. Примеры приведены в [ST96, BST97, BST96] .

Однако практически интересны только случаи конечных или конечно заданных состояний. В работах [ST96, BST97, BST96] мы детально изучаем эти случаи и показываем, как случай конечно заданных состояний сводится к случаю конечных. В частности, если трансляционный результат верен (верен в эффективном смысле) для некоторого универсума для конечных состояний, то он верен для этого универсума в таком же смысле и для конечно заданных состояний. По этой причине дальше рассматриваются только конечные состояния.

2.6 Результаты о трансляции

В [PVV95] рассматриваются вещественные числа с порядком и сложением и для этого универсума доказывается трансляционный результат.

Из доказательства усматривается алгоритм, позволяющий транслировать генерический расширенный запрос в эквивалентный над конечными состояниями ограниченный. В [ST96] этот результат распространяется на любые полные упорядоченные абелевы группы.

Далее рассматривается случай локальной генеричности.

Выдающимся продвижением в рассматриваемой области явилась работа [BDLW96], в которой результат о трансляции получен для всех так называемых *о-минимальных* универсумов. Это понятие является важным в современной теории моделей и введено в [PS86, KPS86, PS88]. Обогащенный универсум называется o-минимальным, если его подмножества, определяемые FO формулами с параметрами из универсума, исчерпываются объединениями конечного числа открытых интервалов и точек.

К числу o-минимальных структур, в частности, относятся вещественные или рациональные числа с порядком и сложением (и, более обще, все полные упорядоченные абелевы группы), вещественные числа с порядком, сложением и умножением (и, более обще, все упорядоченные вещественно замкнутые поля). Гораздо более глубокий результат - недавняя теорема Уилки (A.Wilkie) об o-минимальности упорядоченного поля вещественных чисел (вещественных чисел с порядком, сложением и умножением), обогащенного одноместной операцией $\exp(x)$.

В некотором смысле трансляционные результаты для o-минимальных универсумов перекрывают упомянутые результаты для полных упорядоченных абелевых групп. Однако разница в том, что если o-минимальная структура имеет неразрешимую элементарную теорию, но элементарная теория ее линейного порядка разрешима (например, обогащение упорядоченного поля вещественных чисел невычислимой константой), то алгоритм, дающий по расширенному генерическому запросу эквивалентный ему ограниченный запрос, невозможен в принципе. По этой причине, из факта справедливости трансляционного результата алгоритма трансляции не извлечешь.

В [BST97, BST96], используя глубокие теоретико-модельные методы, мы предлагаем существенно более сильные трансляционные результаты (но по-прежнему в неэффективном смысле).

Ключевая идея, позволившая нам использовать эти методы, состоит в том, что справедливость результата о трансляции для обогащенного универсума на самом деле есть свойство элементарной теории этого универсума. Другими словами, если другой универсум той же сигнатуры не отличим от рассматриваемого никакой замкнутой FO формулой, то трансляционный результат для него справедлив тогда и только тогда, когда этот результат справедлив для рассматриваемого универсума. Эта идея позволяет использовать технику теории моделей. Рассмотрение "нестандартных"моделей элементарной теории рассматриваемого универсума и псевдоконечных состояний над ними — предельного случая конечных состояний — позволило прояснить ситуацию.

Наши основные результаты состоят в следующем.

Во-первых, мы предлагаем некоторое необходимое и достаточное условие на расширенный запрос, при выполнении которого этот запрос эквивалентен относительно конечных состояний ограниченному.

Во-вторых, для локально генерических запросов в качестве следствия получается полезное достаточное условие эквивалентности ограниченным запросам.

В-третьих, мы находим весьма общее достаточное для трансляционного результата условие на элементарную теорию рассматриваемого универсума, которые мы называем *однородностью*.

Говоря очень неформально, это понятие состоит в следующем. Состояние называется псевдоконечным, если для него выполняются все FO свойства, которые выполняются для всех конечных состояний. Универсум называется λ-насыщенным, если в любом его обогащении дополнительными константами, число которых меньше λ, каждое конечно выполнимое множество FO формул с одной общей свободной переменной целиком выполнимо в этом обогащении. Для универсума выполняется свойство однородности, если можно так обогатить сигнатуру этого универсума новыми отношениями, которые FO определимы в старой сигнатуре, что для обогащенного универсума M' существует такой кардинал λ, что любой такой частичный изоморфизм h двух псевдоконечных подмножеств A и B обогащенного универсума, для которого структура (M', A, B, h) является λ-насыщенной, можно продолжить на произвольный дополнительный элемент этого универсума.

Хотя однородность можно прямо использовать для доказательства трансляционных результатов, проверка этого условия для конкретных универсумов может требовать весьма сложных рассмотрений.

По этой причине, в-четвертых, мы выделяем широкий класс универсумов — так называемые квази o-минимальные структуры. Эти универсумы все удовлетворяют условию однородности.

Грубо говоря, структура является квази o-минимальной, если каждое ее FO определимое в константном обогащении подмножество является конечным объединением интервалов, из которых вырезаны FO определимые в первоначальной структуре подмножества.

Примерами квази o-минимальных универсумов являются:

- все o-минимальные универсумы;
- целые или натуральные числа со сложением и порядком;
- вещественные числа с порядком и с подмножеством, выделяющим рациональные числа.

Класс квази o-минимальных универсумов замкнут относительно естественной операции упорядоченной суммы двух упорядоченных универсумов. Таким образом, из известных квази o-минимальных структур можно строить новые квази o-минимальные составные структуры. Заметим, что рассмотрение множеств с различными типами элементов (например, составленных из строчек, чисел разной природы и чего-то еще) обычно для практического программирования.

Однако однородность шире квази o-минимальности. Вот примеры универсумов, которые не квази o-минимальны, но удовлетворяют условию однородности и, следовательно, для них верен результат о трансляции.

Мы замечаем, что структура $(A, <, E)$, где $(A, <)$ — плотно линейно упорядоченное множество без концевых элементов, а E — это эквивалентность на A с ровно двумя классами эквивалентности, каждый из которых является плотным подмножеством множества A, удовлетворяет условию однородности, но не является квази o-минимальной.

Мы доказываем, что

- структура $(\mathbb{R}, +, <, \mathbb{Q})$ удовлетворяет условию однородности, значит, для нее выполняется трансляционный результат;
- эта структура не является квази o-минимальной.

В частности, трансляционный результат получается для любой структуры вида $(A, <, E)$, где $(A, <)$ — плотно линейно упорядоченное множество без концевых элементов, а E — это эквивалентность на A с бесконечным множеством классов эквивалентности, каждый из которых является плотным подмножеством множества A.

Наше общее рассмотрение в действительности позволяет указать и другие универсумы, для которых имеет место результат о трансляции. В частности, трансляционный результат получается для любого универсума вида $(\mathbb{R}, +, <, F, f_\alpha)_{\alpha \in F}$, где F — это подполе поля $\mathbb{R}$, а f_α — это имя для унарной операции умножения на α.

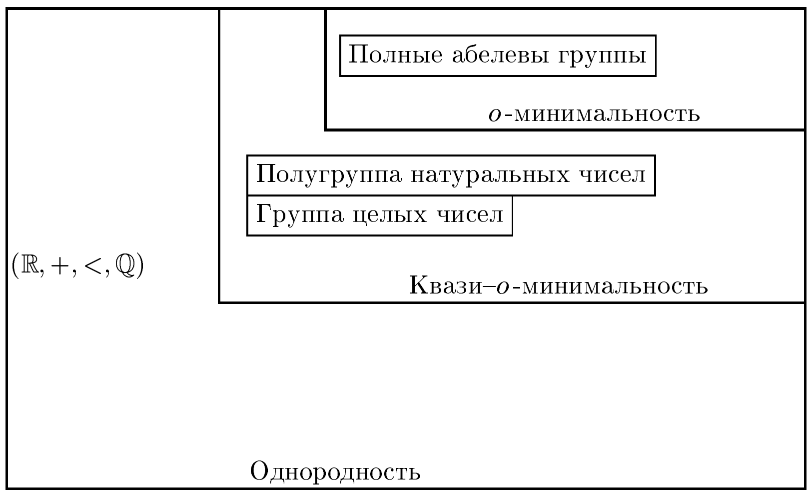
Так как для универсума $(\mathbb{R}, +, \times, <, \mathbb{Q})$ результат о трансляции не имеет места, то мы действительно приближаемся к некоторой границе возможного.

Эти результаты содержатся в [BST99].

Те, кого интересуют подробные определения и доказательства, могут получить ps-файл, содержащий эту работу, используя адрес:

`homepages.tversu.ru/~p000101`

Следующий рисунок изображает часть приведенных трансляционных результатов.



Список литературы

[AH91] A. Avron and J. Hirshfeld. On first order database query languages. In *Proc. 6th IEEE Symp. on Logic in Computer Science*, pages 226–231, 1991.

[BDLW96] M. Benedikt, G. Dong, L. Libkin, and L. Wong. Relational expressive power of constraint query languages. In *Proc. 15th ACM Symp. on Principles of Database Systems*, pages 5–16, 1996.

[BST96] O.V. Belegradek, A.P. Stolboushkin, and M.A. Taitslin. On order-generic queries. Technical report 96–01, DIMACS, 1996.

[BST97] O.V. Belegradek, A.P. Stolboushkin, and M.A. Taitslin. Generic queries over quasi-o-minimal domains. In *Logical Foundation of Computer Science, 4th International Symposium, LFCS'97,* *Yaroslavl, Russia, July 1997, Proceedings*, volume 1234 of *LNCS*, pages 21–32. Springer-Verlag, 1997.

[BST99] O.V. Belegradek, A.P. Stolboushkin, and M.A. Taitslin. Extended order-generic queries. *Annals of Pure and Applied Logic*, 97(1–3):85–125, 1999.

[CH80] A. Chandra and D. Harel. Computable queries for relational databases. *Journal of Computer and System Sciences*, 21:156–178, 1980.

[Cod70] E.F. Codd. A relational model for large shared data banks. *Communications of the ACM*, 13:377–387, 1970.

[Cod72] E.F. Codd. Relational completeness of data base sublanguages. In R. Rustin, editor, *Database Systems*, pages 33–64. Prentice-Hall, 1972.

[Di 69] R.A. Di Paola. The recursive unsolvability of the decision problem for the class of definite formulas. *Journal of the ACM*, 16(2):324–327, 1969.

[End72] H.B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, New York, 1972.

[GS94] S. Grumbach and J. Su. Finitely representable databases. In *Proc. 13th ACM Symp. on Principles of Database Systems*, 1994.

[GS95] S. Grumbach and J. Su. Dense-order constraint databases. In *Proc. 14th ACM Symp. on Principles of Database Systems*, pages 66–77, 1995.

[GST95] S. Grumbach, J. Su, and C. Tollu. Linear constraint databases. In *Proc. Logic and Computational Complexity (LCC'94)*, LNCS. Springer-Verlag, 1995.

[Gur90] Yu. Gurevich. $<$-invariant first-order vs. plain first order. Private communication, 1990.

[Hir91] J. Hirshfeld. Safe queries in relational databases with functions. In *CSL '91: 5th Workshop on Computer Science Logic*, LNCS, pages 173–183. Springer-Verlag, 1991.

[KG94] P.C. Kanellakis and D.Q. Goldin. Constraint programming and database query languages. In *Proc. International Symposium on Theoretical Aspects of Computer Software (TACS'94)*, pages 96–120, 1994.

[KKR90] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. In *Proc. 9th ACM Symp. on Principles of Database Systems*, pages 299–313, 1990.

[KKR95] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, August 1995.

[KPS86] J.F. Knight, A. Pillay, and C. Steinhorn. Definable sets in ordered structures, II. *Transactions of American Mathematical Society*, 295(2):593–605, 1986.

[PS86] A. Pillay and C. Steinhorn. Definable sets in ordered structures, I. *Transactions of American Mathematical Society*, 295(2):565–592, 1986.

[PS88] A. Pillay and C. Steinhorn. Definable sets in ordered structures, III. *Transactions of American Mathematical Society*, 309(2):469–476, 1988.

[PVV95] J. Paradaens, J. Van den Bussche, and D. Van Gucht. First-order queries on finite structures over reals. In *Proc. 10th IEEE Symp. on Logic in Computer Science*, pages 79–87. IEEE Computer Society Press, 1995.

[ST95] A.P. Stolboushkin and M.A. Taitslin. Finite queries do not have effective syntax. In *Proc. 14th ACM Symp. on Principles of Database Systems*, pages 277–285, 1995.

[ST96] A.P. Stolboushkin and M.A. Taitslin. Linear vs. order constraint queries over rational databases. In *Proc. 15th ACM Symp. on Principles of Database Systems*, pages 17–27, 1996.

[Ull82] J.D. Ullman. *Principles of Database Systems.* Computer Science Press, 2 edition, 1982.

[Ull88] J.D. Ullman. *Principles of Database and Knowledge-Base Systems, Volumes I and II.* Computer Science Press, 1988.

[Van91] Van Gelder, A. and R.W. Topor. Safety and translation of relational calculus queries. *ACM Trans. on Database Systems*, 16(2):235–278, June 1991.

[Var81] M.Y. Vardi. The decision problem for database dependencies. *Information Processing Letters*, pages 251–254, 1981.

[АГСШ86] А.К. Айламазян, М.М. Гилула, А.П. Столбоушкин, and Г.Ф. Шварц. Сведение реляционной модели с бесконечными областями к случаю конечных областей. *Доклады Академии Наук СССР*, 286(2):308–311, 1986.

Предметный указатель