В докладе рассмариваются схемы, содержащие parity gates(XOR) и threshold gates.

Определение. threshold gate с m входами определяется m весами $(w_1, \ldots, w_m)$ и порогом T. На входах $(y_1, \ldots, y_m)$ threshold gate с такими параметрами возвращает 1, если $\sum w_i y_i \geqslant T$, и 0 иначе.

Определение. Пусть $r_{n,m} = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} 2^i x_{ij}$. Тогда $U_{n,m} = sign(2r_{n,m} + 1)$

Основным результатом доклада будет следующая теорема:

Теорема 1. *Любой threshold gate может быть заменен на threshold схему полиномального размера глубины 2, где любой threshold gate имеет единичные веса.*

Схема доказательства:

Будем счиатать, что все переменные лежат в $\{-1, 1\}$, а не в $\{0, 1\}$.

Сначала доказывается, что любой threshold gate может быть заменен на threshold gate $U_{n,m}$ общего вида, где n и m не сильно больше числа входов у изначального threshold gate.

После этого можно зафиксировать параметр s, который мы выберем в самом конце, и рассмотреть следующую функцию:

$$\begin{aligned} M_l(y) = \sum_{i=-2b}^{2b} & sign(y - i \cdot 2^{l+s\log a} - 2^l + a^{-s}2^l) \\ & -sign(y - i \cdot 2^{l+s\log a} - 2^{l+1} - a^{-s}2^l) \\ & +sign(y - i \cdot 2^{l+s\log a} + 2^l - a^{-s}2^l) \\ & -sign(y - i \cdot 2^{l+s\log a} + 2^{l+1} + a^{-s}2^l) \end{aligned} \tag{1}$$

Далее рассматривается следующая функция: $N_l(x) = M_l(\Sigma_{ls}(x))$, где $\Sigma_{ls}(x) = \Sigma_{max(l+s\log a-\log b,0)}^{min(l+s\log a,a-1)}(x)$, где $\Sigma_{t1}^{t2} = \sum_{i=t1}^{t2} \sum_{j=0}^{b-1} 2^i x_{ij}$.

Утверждается, что если рассмотреть теперь функцию $N_{a,b} = \sum_{l=0}^{a+\log b} N_l(x)$, то она будет считаться схемой глубины 2. В то же самое время она равна $U_{a,b}(x)$ для почти всех x.

Далее вероятностными методами доказывается, что существует такая схема при $s = 2$, считающая ровно $U_{n,m}$, получающаяся подстановкой в схему, считающую $N_{a,b}$, где a и b не сильно больше n и m.

Приближенное равновесие Нэша для анонимных игр

17 декабря 2018 г.

1 Abstract

В сатье строится PTAS для равновесия Нэша со смешанными стратегиями в анонимных играх в случаях, когда количество стратегий постоянно. Для каждого набора смешанных стратегий мы генерируем естественное распределение и показываем, что близость распределений относительно некоторой метрики влечет близость соответствующих наборов. В дальнейшем, мы примененяем вероятностные методы, чтобы дискретизировать распределения наших наборов. После генерации дискретных наборов мы используем динамическое програмиирование и получаем требуемый PTAS.

Определение 1 *Анон. игрой называется* $G = (n, S, (u_l^i))$, n*-кол-во игроков,* S*-мн-во стратегий,* u_i^l:$\Pi_{n-1}^S \to [0,1], \Pi_{n-1}^S = ((x_1, ..., x_S)$: x_i*-неотрицательные целые числа* $\wedge \sum_{m=1}^{|S|} x_m = n - 1$ *)*

Определение 2 *Смешанный набор стратегий - это отображение* δ: $[n] \to \Delta^S$*, где* Δ^S*-множество распределений на* S*.*

Определение 3 *Смешанное равновесие Нэша - это отображение* δ *т.ч.* $E_{x\sim\delta_{-i}, l\sim\delta_i} u_l^i(x) \geq E_{x\sim\delta_{-i}} u_t^i(x), \forall i \in [n], t \in S$

Теорема 1 *Существует PTAS для задачи поиска смешанного равновесия Нэша для анонимных игр с постоянным числом стратегий. Точнее, существует функция* g *такая, что для всех* $\epsilon \geq 0$*,* ϵ*-равновесие Нэша анонимной игры* $G = (n, S, (u_i^l))$ *может быть вычислено за время* $n^{g(S, 1/\epsilon)}$

Практически для любого графа Reconstruction Number(RN) равно трем.

В данной статье речь шла о некотором общем утверждении, касающемся характеристики, определяющий графы с точностью до изоморфизма.

Определение. *G* - граф, где $\{x_1,..x_n\}$ - множество его вершин, будем ***deck*** называть совокупность $\{G_i|G_i = G\backslash x_i, i = 1,..,n\}$

Определение. *Reconstruction Number(RN)* графа *G* будем называть минимальное число *k*, такое что существует набор $\{i_1,..i_k\}, 1 \le i_1 < \cdots < i_k \le n$, такой, что если для графа *H* вместе с $deck_H$ существует набор размера *k* - J = $\{j_1,..,j_k\}$, такой что $G_{i_l} \cong H_{j_l}$ для любого $l = 1,.,k$, то *H* изоморфен *G*.

Определение. $\Im(n, p)$- вероятностное пространство графов на *n* вершинах, где каждое ребро выбирается с вероятностью $p = p(n)$. Случайный элемент этого пространства обозначаем $G_p\left(G_{p,n}\right)$.

Определение. Будем говорить, что случайный элемент G_pобладает некоторым свойством *Q*, если вероятность этого события стремится к 1 при $n \to \infty$.

Лемма(Приведена без подробного доказательства).

$k \in \mathbb{N}, c > \frac{k+2}{2}$ – фиксированы,

$$\frac{c\log n}{n} \le p = p(n) \le 1 - \frac{c\log n}{n}.$$

G из $\Im(n+k, p)$ - такое что если $W \subset V$, мощности *n*, а ρ- инъекция, индуцирующая изоморфизм подграфов: $G[W] \to G[\rho(W)]$, где $G[W]$ - подграф графа *G* на множестве вершин *W*, тогда $\rho(w = w$ для любого w из *W*.

Теорема. Пусть $c > \frac{5}{2}$ и $\frac{c\log n}{n} \le p = p(n) \le 1 - \frac{c\log n}{n}$.

1. Тогда G_p, *n* - такое что любые три элемента *deck* определяют его.
2. Более того любые два элемента *deck* G_iи G_j определяют его граф с точностью до ребра между x_i и x_j.

Для доказательства, выделим множество графов, подходящих под условие Леммы и докажем, что для графов из этого множества, верно выполнение второго пункта теоремы, что влечет выполнение и первого.

Заметим, что из Леммы следует, что вероятность этого множества стремится к единице, то есть почти все графы обладают свойством принадлежности данному множеству, что и доказывает теорему.

Следствие. Почти для любого графа *RN=3*.

Алгоритм Мура

Допустим, у нас есть конечный автомат с n состояниями и алфавитом размера k. Вспомним, что такое алгоритм Мура (а именно – алгоритм минимизации DFA).

Пусть у нас есть некоторый автомат. На первом шаге разделим множество состояний на 2: принимающие и не принимающие. На каждом последующем шаге делаем проверку: если из одного множества по разным состояниям и одной и той же букве мы переходим в разные m множеств, то делим исходное множество на соответствующие m подмножеств. Продолжается алгоритм до тех пор, пока множества не перестанут делиться. Не сложно заметить, что в худшем случае время алгоритма $O(kn^2)$.

Определение. Структурой перехода называется четвёрка вида $(\Sigma, Q, \cdot, q_0)$.

Определение. Пусть E - непустое конечное множество. Для любого $\gamma \in (0,1)$ вероятность r, определенная на подмножествах X из E, есть $r(X) = \gamma^{|X|}(1-\gamma)^{|E|-|X|}$, которая называется распределением Бернулли параметра γ на элементах E.

Определение. Для любого $n \geqslant 1$ вероятность p, определенная на D_n (множество DFA с n состояниями), является моделью Бернулли, когда существует вероятность q, определенная на T_n (множество структур перехода с n состояниями), и действительное число $\gamma \in (0,1)$ такое, что для любого $A \in D_n$, $p(A) = q(T_A)r(F_A)$, где $A = (T_A, F_A)$ и r - распределение Бернулли по параметру γ на состояниях.

Определение. Пусть $\gamma \in (0,1)$. Пусть $p : D \to [0,1]$ - такое отображение, что для любого $n \geqslant 1$ ограничение p на D_n является моделью Бернулли параметра γ. Тогда p - модель Бернулли на D (параметра γ).

Теорема 1. *Для любой моделли Бернулли на D средняя временная сложность (ожидаемое количество итераций в алгоритме Мура для случайно выбранного автомата) алгоритма минимизации равна $O(kn \log n)$.*

То есть в среднем время работы алгоритма значительно лучше. Для доказательства этого факта мы находили условия для того, чтобы алгоритм Мура делал больше, чем l итераций и замечали, что вероятность данного события экспоненциально мала. Затем мы выбирали $l = \log n$.

Пусть X-граф на вершинах $x_1....x_n$.

Обозначим за X_i - граф, который мы получили из графа удалением всех рёбер, инцидентных x_i и добавлением всех нерёбер, инцидентных x_i.

Пусть у нас есть имеется множество непомеченных графов $X_1...X_n$ (Под непомеченными графами имеется ввиду, что мы знаем их с точностью до изоморфизма).

Мы хотим узнать, при каких n мы всегда можем однозначно восстановить X по этому множеству.

При $n = 4$ это неверно.

Основной результат:

Теорема 1. *Пусть $n \neq 0 \ (mod\ 4)$. Тогда если X и X' - графы на вершинах $x_1, ..., x_n$, причём при $1 \leq i \leq n$ $X_i \cong X'_i$, то $X \cong X'$.*

С помощью чего мы это всё доказываем?

Пусть $f : \mathbb{Z}_2^k \to \mathbb{R}$, тогда её преобразование Фурье - это $\widehat{f} : \mathbb{Z}_2^k \to \mathbb{R}$, определённое как

$$\widehat{f}(X) = \sum_Y (-1)^{XY} f(Y)$$

Также при $\Gamma \subset \mathbb{Z}_2^k$ определим $\overline{f} : \mathbb{Z}_2^k \to \mathbb{R}$ как

$$\overline{f}(Y) = \sum_{X \in Y + \Gamma} f(X)$$

Лемма 1. *Линейное преобразование $f \mapsto \overline{f}$ обратимо тогда и только тогда, когда $\widehat{\chi_\Gamma}(X) \neq 0$ при всех $X \in \mathbb{Z}_2^k$. (χ_Γ - это характеристическая функция множества Γ)*

Тепрь пусть V_n - это множество всех формальных линейных комбинаций $\sum_X a_x X$, $a_x \in \mathbb{R}$, где X пробегает множество всех графов на вершинах $x_1, ..., x_n$.

Пусть $\phi : V_n \to V_n$ - это линейное преобразование, определённое как

$$\phi(X) = X_1 + ... + X_n,$$

где X_i - помеченные графы, определённые выше.

Лемма 2. *ϕ обратимо тогда и только тогда, когда $n \neq 0 \ (mod\ 4)$*

Открытые вопросы:

Верно ли это при $n = 0 \ (mod\ 4)$ и $x \geq 8$?

Есть ли доказательство нашего основного результата, которое явно строит X?

Похожая гипотеза:

Верно ли тоже то же самое для графов $X_1, ..., X_n$, которые мы получаем удалением вершин $x_1, ..., x_n$ соответственно из графа X?

Даны s полиномов с n переменными степени 2 над полем $\mathbb{F}_q$. Задача определить существует ли у этой системы решение называется MQS (Multivariable Quadratic Systems) и является NP-трудной. Мы будем решать задачу #MQS, то есть считать количество (на самом деле долю) решений. Эта задача очевидно не проще, поэтому мы будем пытаться решать ее приближенно, то есть мы разрешаем себе ошибаться не более, чем на ε. Мы хотим построить алгоритм, который будет детерминированным (то есть рандома нет, это важно!), и будет работать за время $poly(n, s, \log q)/\varepsilon^2$.

Теорема 1. *При $s = 1$ задача решается точно за полиномиальное время.*

Эта теорема без доказательства, мы ей только пользовались.

Определение 1. *Множество $S \subset \mathbb{F}_q^n$ называется ε-biased (ε-скошенным, наверное), если выполняется следующее условие*

$$\forall u \in \mathbb{F}_q^n, r \in \mathbb{F}_q \; |Pr_{v \in S}\{\langle u, v\rangle = r\} - Pr_{v \in \mathbb{F}_q^n}\{\langle u, v\rangle = r\}| \leq \varepsilon$$

Теорема 2. *Можно построить ε-biased множество размера $O(\frac{n^2}{\varepsilon^2})$ за время $poly(n, \log q)/\varepsilon^2$*

Эта теорема тоже без доказательства.

Теорема 3. *Можно получить ε-приближение для задачи #MQS за время $poly(n, s, \log q)/\varepsilon^2$*

Для доказательства нужно построить ε-biased подмножество множества $\mathbb{F}_q^s$ (внимание, здесь s!) и рассмотреть формулы $P_i(y) := \sum v_i p_i(y)$, где v_i - элементы построенного множества, а p_i - полиномы из задачи. Тогда если y - решение, то $P_i(y) = 0$, а иначе $P_i(y)$ принимает значения 0 и 1 с вероятностью примерно (с точностью до ε) $\frac{1}{q}$. Тогда приближением количества решений #MQS будет разность количеств решений уравнений $P_i(y) = 0$ и $P_i(y) = 1$. Эти количества мы считать умеем по теореме 1.

Следствие 1. *Если $\varepsilon > \frac{1}{q^n}$ и система имеет хотя бы εq^n решений, то за время $poly(n, s, \log q)/\varepsilon^2$ можно найти решение явно.*

Значение переменных выбираем, подставив все возможные значения и выбрав то, которое дает большее число решений, а когда мы уже больше не сможем сделать следующий шаг так, чтобы гарантировать, что решения остались, просто переберем значения всех оставшихся переменных.

Теорема 4. *Можно свести #k-SAT с n переменными к вычислению количества решений полинома степени $q(k/\varepsilon)^{O(k)}$ с n переменными за время $O(q^{\varepsilon n})$.*

Нам хочется попросить, чтобы в формуле было не более $(k/\varepsilon)^{O(k)}n$ клозов. Это сделает sparsification lemma (на самом деле нет, она сделает нечто другое, но нам сейчас это не важно). Теперь КНФ достаточно просто сводится к системе полиномов: or можно симулировать через умножение, а and - через and (нам же нужно, чтобы все полиномы выполнились). Также, надо дополнительно добавить n полиномов вида $x(1-x)$, которые запишут условие, что наши переменные булевы. Получили систему уравнений G. Разобъем ее на εn систем уравнений размера не более $(k/\varepsilon)^{O(k)}$. Теперь из каждой G_j построим полином P_j.

$$P_j(x) := 1 - \prod_{i=1}^{(k/\varepsilon)^{O(k)}} (1 - (p_i(x))^{q-1})$$

P_j имеет нужную степень, но полинимов много, а нам нужен один. Но мы уже учились с этим бороться в теореме 3. Разница в том, что там мы только приближали, а нам нужно посчитать точно. Но мы знаем одно 0-biased множество (все линейные комбинации), его и возьмем.

В докладе рассматривается частный случай задачи о выполнимости булевой схемы — Unique Circuit-SAT, в котором заранее известно, что число удовлетворяющих схеме наборов переменных не превосходит единицы. Представлен алгоритм, решающий эту частную задачу за время $\mathcal{O}(2^{.374589m})$, где m - число гейтов.
Основные понятия: Булева схема представляет собой ориентированный ациклический граф. Ее входы (вершины, у которых входящая степень 0) это переменные, в которые можно подставлять значения 0 или 1. В каждой из внутренних вершин схемы написана одна из шестнадцати двухместных булевых функций - их мы называем гейты. Кроме того есть один гейт с исходящей степенью 0, который мы называем выходом схемы. Соответственно схема выполнима, если можно назначить переменным такие значения, что в выходном гейте получится 1.

Анализ времени работы происходит так:Данный алгоритм относится к типичным расщепляющим алгоритмам. Суть метода в том, что мы делим нашу задачу на несколько более простых. Например, если у нас была схема с числом гейтов L, и мы каким-то образом сумели расщепить ее на две размера $L-2$ и $L-4$, то чтобы найти время работы алгоритма нам необходимо решить такую рекурренту $T(L) = T(L-2)+T(L-4)$. Тогда время $T(L)$ с точностью до полиномиального фактора это γ^L, где γ решение рекурренты, т. е. корень уравнения $x^(-2)+x^(-4)=1$.
Алгоритм: Мы разбираем кучу случаев, где самый худший (самый большой) γ появляется в последнем случае и равен 1.29647. Таким образом, оценка на время работы алгоритма $\mathcal{O}(2^{.374589m})$ (просто взяли логарифм).

Идея разбора в худшем случае примерно такая: мы делаем сразу нескольео расщеплений, после чего мы точно понимаем, что в некоторых ветвях, если и есть решения то их не меньше двух. Однако, по условию задачи решений не более чем одно поэтому такие ветви сразу моргут быть отброшены из рассмотрения. Откуда оценка получается лучше чем в общем случае Circuit SAT.

Множество $S \subset \mathbb{N}_0 := \{0, 1, 2, 3, \ldots\}$ называется 2-автоматным если существует конечный автомат, распознающий язык двоичных записей чисел из S. Например, 2-автоматны множества всех степеней двоек, всех чётных чисел и всех чисел, в двоичной записи которых нечётное число единиц, но не 2-автоматны множества всех степеней троек, всех точных квадратов и всех простых чисел.

Пусть $\mathbb{F}_2$ — поле из двух элементов, а $\mathbb{F}_2[t]$ и $\mathbb{F}_2[[t]]$ — кольца многочленов и формальных степенных рядов над $\mathbb{F}_2$ соответственно. Формальный степенной ряд $\sum\limits_{n=0}^{+\infty} f_n t^n = f \in \mathbb{F}_2[[t]]$ называется алгебраическим, если он является корнем какого-нибудь ненулевого многочлена с коэффициентами из $\mathbb{F}_2[t]$. Например, алгебраичны решения уравнений $f^2 + f + t = 0, (1+t)f + t^4 = 0, (t^2+t+1)f^3 + (t^4+t)f + t^8 = 0$ из кольца $\mathbb{F}_2[[t]]$ формальных степенных рядов над полем $\mathbb{F}_2$.

Theorem 0.1 (Теорема Кристеля). *Формальный степенной ряд $\sum\limits_{n=0}^{+\infty} f_n t^n$ алгебраичен тогда и только тогда, когда множество $\{n \mid f_n = 1\} \subset \mathbb{N} \cup \{0\}$ 2-автоматно.*

План доказательства 2-автоматности каждого алгебраического степенного ряда.

1) В определении 2-автоматности, можно считать, что распознаются развёрнутые двоичные записи чисел из S. (то есть читаем от младших разрядов к старшим).

2) $(f+g)^2 = f^2 + g^2$ в $\mathbb{F}_2, \mathbb{F}_2[t]$ и $\mathbb{F}_2[[t]]$ — будем этим нагло пользоваться.

3) Введём операцию Λ_r для $r = 0, 1$ на подмножествах $\mathbb{N}_0$ и на элементах $\mathbb{F}_2[[t]]$: $\Lambda_r(S) := \{n \mid n \in \mathbb{N}_0, 2n + r \in S\}$ (эта операция соответствует "прочитыванию" цифры r в терминах правых контекстов. Аналогично для любого $f \in \mathbb{F}_2[[t]]$ существуют и единственны такие $f_0, f_1 \in \mathbb{F}_2[[t]]$, что $f = f_0^2 + t f_1^2$. Определим $\Lambda_r(f) := f_r$. Это как раз соответствует прочитыванию цифры r в терминах 2-автоматных множеств.

4) Теперь, чтобы доказать 2-автоматность алгебраического ряда $f \in \mathbb{F}_2[[t]]$, достаточно предъявить конечное множество Q степенных рядов, замкнутое относительно операций Λ_0 и Λ_1 и содержащее f.

5) Любой алгебраический степенной ряд — корень уравнения вида $p_0 f^{2^0} + p_1 f^{2^1} + \ldots + p_n f^{2^n} = 0$, где $n \in \mathbb{N}$, а $p_i \in \mathbb{F}_2[t]$ и не все равны 0 (какая-то алгебра).

6) Не сильно умаляя общности, можно считать, что $p_0 = 1$ (какая-то алгебра). Тогда утверждается, что для достаточно большого d (d зависит от степеней p_i) подойдёт множество всех формальных степенных рядов вида $q_0 f^{2^0} + q_1 f^{2^1} + \ldots + q_n f^{2^n}$, где каждое q_i независимо пробегает множество всех многочленов над $\mathbb{F}_2$ степени не больше d (их всего 2^{d+1} штук — конечное число).

7) GGWP. В другую сторону я не успел толком рассказать на паре.

□

Абстракт

Alexander Morakhovski

17 декабря 2018 г.

Однозначный конечный автомат (UFA), это недетерминированный конечный автомат с не более чем одним принимающим вычислением для любой строки. В этой статье, доказывается суперполиномиальная оценка на количество состояний, на операцию дополнения для UFA. То есть, приводится пример однозначного конечного автомата над односимвольным алфавитом; даже бесконечный набор автоматов $(A_d)_{d\in N}$ такие, что количество состояний в любом недетерминированном автомате который распознает дополнение A_d имеет не менее $|A_d|^d$ состояний, где $|A_d|$ это количество состояний в автомате A_d.

В начале мы строим наш автомат, он будет выглядить как начальное состояние + цепочка состояний (на самом деле цепочку мы уберем в ходе доказательства) а потом n циклов не пересекающихся (там все принимающие состояния)(смотрим Рисунок внизу). Длина цикла любого у нас будет равна перемножению какого-то фиксированного числа простых чисел. (мы выбираем фиксированное число разных простых чисел из набора простых который мы изначально как-то выбрали и перемножаем. Так выбираем для каждого цикла).

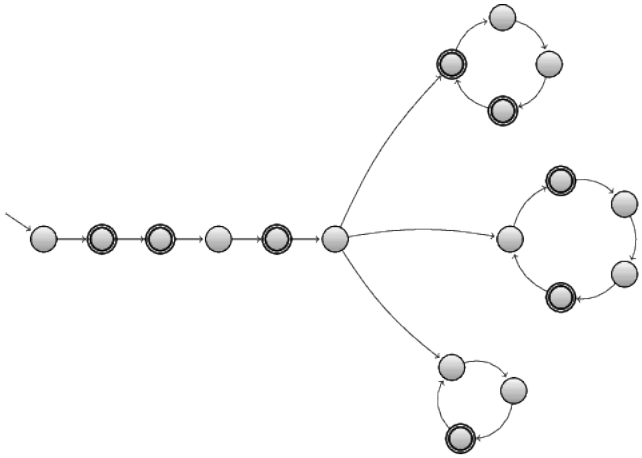
Доказывается, что автомат однозначный.

Дальше каждый цикл рассматриваем как точку в графе. Рисуем полный граф, где на ребрах написано по каким простым циклы пересекаются.

Доказывается лемма, Для всех натуральных чисел k, существует турнир R такой, что свойство выполняется: для всех E в R , если для всех вершин x существует такая вершина y, что xEy, то E содержит по крайней мере k различных ребер, которые не пересекаются по вершинам. Можно выбрать турнир с этим свойством размером $n = 12k^2 2^{2k}$

По теореме о простых числах, говорим что все простые можно выбрать относительно близко друг к другу.

Потом замечаем, что слово длины $p_1...p_l$, где p_i это все наши простые, не принимаются автоматом, значит принимаются дополнением.



Чтобы не было слишком много состояний в дополнение сразу, нужен большой цикл. С помощью леммы доказываем, что все равно состояний нужно суперполиномиальное число.

Алгоритм Копперсмита-Винограда

Мрыхин Михаил

Хотим быстро множить матрицы.

1 Тензорная форма записи

Пишем умножение матриц как $\sum_{i,j,k=1}^{m,n,p} a_{ij}b_{jk}c_{ki}$ для удобства последующих выкладок. Здесь c_{ki} - формальные переменные, двойственные $(AB)_{ik}$ (т.е. коэффициенты при них - элементы матрицы-произведения).

2 τ-теорема Шёнхаге

Если есть такие коэффициенты $\alpha_{i,j,h,l}, \beta_{j,k,h,l}, \gamma_{k,i,h,l}$, что

$$\sum_{l=1}^{L}\left(\sum_{i,j,h}\alpha_{i,j,h,l}x_{i,j}^{(h)}\right)\left(\sum_{j,k,h}\beta_{j,k,h,l}y_{j,k}^{(h)}\right)\left(\sum_{k,i,h}\beta_{k,i,h,l}z_{k,i}^{(h)}\right)=\sum_{h}\left(\sum_{i,j,k=1}^{m_h,n_h,p_h}x_{i,j}^{(h)}y_{j,k}^{(h)}z_{k,i}^{(h)}\right),$$

и при этом $L=\sum_h (m_h n_h p_h)^\tau$, то $\omega \leq 3\tau$. (ω - это нижний предел асимптотической скорости умножения матриц, т.е. мы не обязаны умножать за $O(n^\omega)$, но за $O(n^{\omega+\epsilon})$ для любого $\epsilon > 0$).

Это обобщённая форма того, что происходит в Штрассене - мы пытаемся через небольшое количество произведений линейных комбинаций элементов выразить несколько произведений матриц. Суть доказательства тоже примерно такая же - рекурсивно оцениваем, сколько умножений нам понадобится для асимптотически больших матриц.

Теорема обобщается дальше, если α, β и γ сделать функциями от λ, а произведение матриц считать с точностью до $O(\lambda)$.

3 Теорема Салема-Спенсера

Это было без доказательства:

Для любого $\epsilon > 0$ существует такое M_ϵ, что для $M > M_\epsilon$ найдётся подмножество $B \subset [1, ..., \left\lfloor \frac{M-1}{2} \right\rfloor]$, свободное от арифметических последовательностей длины 3, и при этом $|B| \geq M^{1-\epsilon}$.

4 Суть самой теоремы

Фиксируем q. Будем координатам x, y и z сопоставлять верхние индексы в зависимости от того, нулевая эта координата или нет (0 и 1 соответственно).

Берётся алгоритм, который за $q + 2$ умножений приблизительно считает кривоватое произведение векторов $(\sum x_0^{[0]} y_i^{[1]} z_i^{[1]} + x_i^{[1]} y_0^{[0]} z_i^{[1]} + x_i^{[1]} y_i^{[1]} z_0^{[0]})$. Эта конструкция тензорно множится с собой $3N$ раз.

После этого мы хэшируем все тройки в итоговом произведении с помощью случайных весов, помноженных на верхние индексы. Выкидываем (присваиванием одному из множителей 0) все тройки, у которых не ровно треть нулей в каждом индексе, те, у которых хэши не лежат в множестве из теоремы Салема-Спенсера, и те, у которых хэши совпали. После чего замечаем, что оставшиеся тройки не имеют общих переменных, каждая из них считает произведение двух матриц $q^N \times q^N$, а их количество для какого-то выбора весов в хэше достаточно высоко, и применяем теорему Шёнхаге.

1 On the power of threshold circuits with small weights

1.1 Что происходит

Рассматриваются схемы, состоящие из линейных пороговых элементов (linear threshold functions; LTF) вида $sgn(\sum w_i x_i + w_0),\ w_i \in Z, x_i \in \{-1, 1\}$; класс таких схем глубины d называем LT_d. Мотивация в изучении таких схем - нейронные сети, а именно многослойные перцептроны, которые фактически состоят из таких элементов. Ставится вопрос: можно ли в таких элементах ограничить веса полиномом от N (размера входного вектора), чтобы их было проще реализовывать. Схемы из таких элементов образуют классы $\widehat{LT_d}$.

1.2 Основные результаты

- Любой LTF может быть реализован схемой глубины 3, состоящей из элементов с ограниченным весом (т.е. $LT_1 \subset \widehat{LT_3}$).

- Обобщение: любую схему глубины d можно переделать в схему из элементов с ограниченным весом: $LT_d \subset \widehat{LT_{2d+1}}$

- Даны верхние оценки на реализацию некоторых арифметических функций:

  - $COMPARISON(X, Y) \in \widehat{LT_2}$
  - $SUM(X, Y) \in \widehat{LT_2}$
  - $MAXIMUM(X_1..X_m) \in \widehat{LT_4}$

1.3 Как доказывается

Доказываем в такой последовательности:

1. $COMPARISON \in \widehat{LT_2}$
   Пользуемся гармоническим анализом: представляем исходную функцию как $f(x) = \sum_\alpha a_\alpha X^\alpha, a_\alpha \in Q$, где α - всевозможные мономы из переменных $x_1..x_n$. Факт из гармонического анализа: если спектральная норма функции $||f(x)|| = \sum |a_\alpha|$ ограничена полиномом, то $f(x) \in \widehat{LT_2}$. Показываем, что, норма $COMPARISON$ растёт линейно с ростом n и, следовательно, $COMPARISON \in \widehat{LT_2}$.

2. $LT_1 \subset \widehat{LT_3}$
   Сводим вычисление произвольной взвешенной суммы под sgn к сумме двух чисел, которая, как показано выше, считается за 2 слоя. Можно показать, что сведение можно выполнить за 1 слой.

Дальше идут примеры:

1. $SUM(X, Y) \in \widehat{LT_2}$
   Замечаем, что сумма отличается от сравнения только знаком Y и, следовательно, лежит в том же классе.

2. $MAXIMUM(X_1..X_m) \in \widehat{LT_4}$
   Вводим вспомогательные переменные c_{ij}, которые обозначают результат сравнения чисел $i,\ j$. Все они считаются параллельно за 2 слоя. Затем для всех чисел требуем, чтобы оно было больше остальных. Взяв дизъюнкцию по всем числам, получаем итоговый максимум. Эти две операции реализуются конъюнкцией и дизъюнкцией, а они реализуются одним слоем каждая.

В конце доклада без доказательств приводилось обобщение: $LT_d \subset \widehat{LT_{2d+1}}$. Доказывается индукцией по глубине исходной схемы.

Шпаргалка по докладу на теорсеминаре по статье «The First and Fourth Public-Key Cryptosystems with Worst-Case/Average-Case Equivalence»

Олейников Иван

17 декабря 2018 г.

1 Предварительные определения

Определение 1 (Криптосистема с публичным ключом)**.** *Криптосистемной с публичным ключом называется тройка вероятностных полиномиальных по времени алгоримов* (G, E, D).

Алиса и Боб могут использовать такую криптосистему для секретной передачи сообщений.

Прежде, чем криптосистемой можно будет шифровать сообщения, нужно сгенерировать пару из публичного и секретного ключей: $(pk, sk) \leftarrow G(1^n)$*, здесь* $n \in \mathbb{N}$ *— это параметр безопасности, от него завясит время работы алгоритмов криптосистемы и трудность взлома криптосистемы. Генерацией ключей занимается получатель сообщений — Алиса, публичный ключ она раздаёт всем, кто в будущем захочет отправлять ей шифрованные сообщения, а секретный оставляет себе и держит в тайне.*

Отправитель Боб, имеющий открытый ключ pk*, может зашифровать сообщение* $b_0 \dots b_l$ *шифрующим алгоритмом:* $x \leftarrow E(pk, b_0 \dots b_l)$*, получив* x *— шифротекст (ещё его могут называть «код») сообщения. После того, как он передаст шифротекст Алисе, имеющей секретный ключ, та сможет восстановить сообщение дешифрующим алгоритмом:* $b_0 \dots b_l \leftarrow D(sk, x)$.

Все алгоритмы криптосистемы вероятностные и мы разрешаем им ошибаться. То есть может оказаться так, что D не сможет восстановить зашифрованное E сообщение. Мы потребуем лишь, чтобы вероятность успешного восстановления сообщения была не меньше какой-то константы от n. Вероятность в этом утверждении берётся по случайным битам всех трёх алгоритмов:

$$\Pr_{G,E,D}[D(sk, E(pk, m)) = m] \geq c > 0,$$
$$\text{where } (pk, sk) \leftarrow G(1^n)$$

для любого сообщения m, где c — не зависящая от n константа.

(Если это выполняется, то можно добиться сколь угодно близкой к 1 вероятности успеха. Должно быть очевидно, как это сделать.)

Наших требований хватает для того, чтобы легитимный пользователь, имеющий сектретный ключ, смог корректно восстановить зашифрованные данные. Теперь введём требование, которое, в некотором смысле, гарантирует нам, что не имеющий секретного ключа противник не сможет прочесть зашифрованное сообщение.

Определение 2 (Взлом криптосистемы)**.** *Пусть* A *— полиномиальный по времени вероятностный алгоритм. Рассмотрим такой эксперимент:*

(1) Сгенерируем пару ключей: $(pk, sk) \leftarrow G(1^n)$.

(2) Подадим A публичный ключ и попросим сгенерировать два различных сообщения: $(m_0, m_1) \leftarrow A(pk)$.

(3) Зашифруем случайное из этих сообщений: $i \leftarrow \{0,1\}$, $x \leftarrow E(pk, m_i)$.

(4) Передадим его A и попросим угадать, какое из сообщений было зашифровано: $i' \leftarrow A(m_0, m_1, x)$.

Если в резльтате такого эксперимента окзалось $i = i'$*, то будет считать, что взлом удался.*

Нам бы хотелось, чтобы вероятность удачного взлома уменьшалась с увеличением n. Для этого введём следующее опредедение.

Определение 3 (Надёжная криптосистема)**.** *Криптосистема называется надёжной, если для любого полиномиального вероятностного алгоритма, вероятность успешного взлома криптосистемы этим алгоритмом становится меньше любого обратного полинома при достаточно больших* n*.*

(Обратный полином — это функция вида $1/p(n)$*, где* $p(n)$ *— полином.)*

Если взломщик научится взламывать криптосистему с вероятностью, ограниченной снизу обратным полиномом, то он сможет амплифицировать корректность до сколь угодно близкой к единице. (Очевидно, как.)

2 Цель и результат доклада

Цель. Целью доклада было определить криптосистему Ajtai-Dwork (читается «Айтай-Дворк») — задать три её алгоритма G, E, D. После чего доказать два утверждения:

(a) для этой криптосистемы выполняется заданное нами выше требование: вероятность корректного дешифрования ограничена снизу ненулевой константной от n;

(b) если криптосистема ненадёжна, то задача Unique Shortest Vector Problem (uSVP) решается за полиномиальное время в худшем случае.

Многим кажется неправдоподобным, что задача uSVP решаема, поэтому пункт (b) является (условным) доказательством надёжности нашей криптосистемы.

Результат. За доклад мы успели определить саму криптосистему, но до доказательств корректности (a) и надёжности (b) мы не дошли. И даже с заданием криптосистемы были проблемы — у меня были проблемы с алгоритмом дешифрования.

3 Памятка о содержании доклада

Напоминим, чем оперировала криптосистема:

Сообщение — Набор из $l+1$ векторов $b_0 \dots b_l$.

Секретный ключ — Набор из $l+1$ ортогональных векторов $u_0 \dots u_l$.

Публичный ключ — Три множества векторов из R^{n+l}: V — $l+1$ штук, D — m' штук, P — $n+l$ штук.

Шифротекст — Вектор $x \in \mathbb{R}^{n+l}$.

Генерация секретного ключа. Векторы публичного ключа выберем поочерёдно: направление u_i выберем из равномерного распределения на сфере радиуса 1 в пространстве $\operatorname{span}\{u_0 \dots u_{i-1}\}^\perp$. Длина каждого из них полагалась независимо равной длине случайного вектора из равномерного распределения на $n+l$-мерном шаре.

Генерация публичного ключа. Все вектора публичного ключа V, D и P будем брать из распределения $\mathcal{H}\mathrm{Samp}^{R,n+l}_{u_0\dots u_l} + \mathcal{N}_{n+l}(0,\rho^2)$ на векторах из $\mathbb{R}^{n+l}$ (распределение зависит от секретного ключа $u_0 \dots u_l$). Здесь $\mathcal{N}_{n+l}(0,\rho^2)$ — $(n+l)$-мерное нормальное распределение, а $\mathcal{H}\mathrm{Samp}^{R,n+l}_{u_0\dots u_l}$ — это нормальное распределение $\mathcal{N}(0,R^2)$, у которого для каждого i сонаправленную u_i компоненту округлили вниз до ближайшего кратного $1/\|u_i\|$.

Шифрование.

$$x = \sum_{i=0}^{l} v_i b_i + \sum_{i=0}^{m'} d_i \delta_i \mod \mathcal{L}(P),$$

где δ_i — случайные биты, выбираемые для шифрования, а $\mathcal{L}(P)$ — решётка, порождённая векторами P (то есть все их целочисленные линейные комбинации).

Дешифрование. На этой части я сломался и дальше мы не двинулись.

Опр: G — ориентированный граф, $g \mapsto |g|$ — отображение его вершин в целые числа. Если для любого ребра $g_1 g_2$ выполняется $|g_2| = |g_1| + 1$, то G называется градуированным графом.

Опр: Градуированный граф G называется модулярным, если для любых двух вершин g_1, g_2 множества $\{g \mid gg_1, gg_2 \in E(G)\}$ и $\{g \mid g_1 g, g_2 g \in E(G)\}$ либо оба пусты, либо каждое состоит из одной вершины.

Опр: Граф G называется ***Y*-графом**, если он модулярен и для каждой вершины количество последователей на один больше, чем предков.

Опр: Косая диаграмма — выпуклое конечное подмножество решетки $\mathbb{Z}^2$ с градуировкой $|(x, y)| = x + y$. Внимание! Тут подразумевается другое понятие выпуклости, которое можно воспринимать так: косая диаграмма должна являться объединением клеток 1×1.

//Далее косая диаграмма всегда обозначается S

Опр: Верхняя и нижняя границы косой диаграммы S определяются формулами:

$$\partial_+(S) = \{(x, y) \in S : (x, y+1) \wedge (x+1, y) \wedge (x+1, y+1) \notin S\}$$

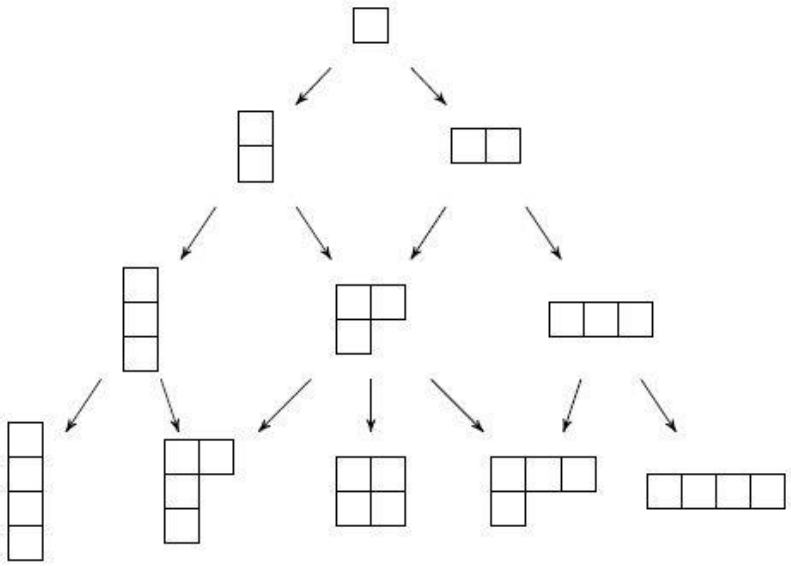
$$\partial_-(S) = \{(x, y) \in S : (x, y-1) \wedge (x-1, y) \wedge (x-1, y-1) \notin S\}$$

Основные примеры графов, которые рассматривались в докладе:

1. *Граф Юнга* **Y**

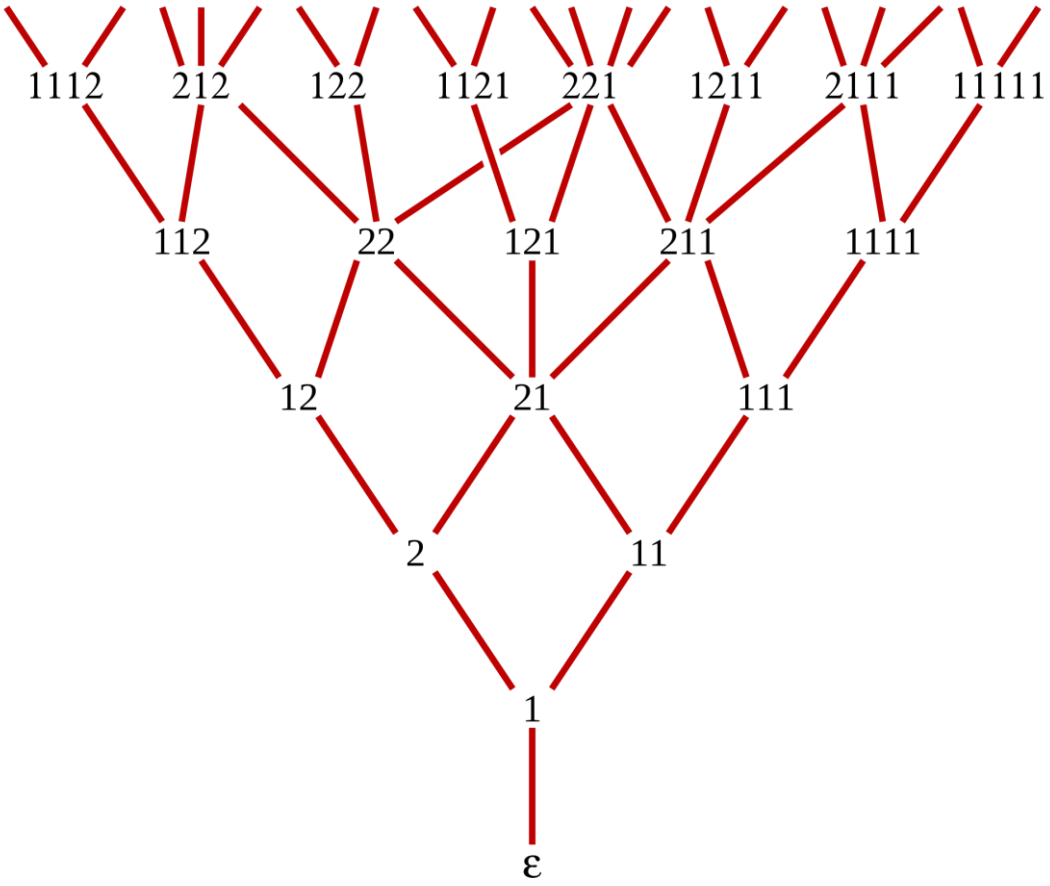
   Вершины — диаграммы Юнга.

   Последовательности $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ соответствует диаграмма Юнга, у которой n строк, в i-ой строке λ_i клеток, клетки выравнены по левой стороне. Градуировка — количество клеток. Предки — диаграммы Юнга, которые могут быть получены удалением одной клетки.



2. *Граф Юнга-Фибоначчи* ***YF***

Вершины — конечные слова в алфавите $\{1,2\}$. Градуировка — сумма цифр. Слову g предшествуют слова, которые получаются из g удалением первой единицы, а также все слова, которые получаются заменой любой двойки, состоящей перед первой единицей, на 1.



Лемма: Графы ***Y*** и ***YF*** являются Y-графами.

Опр: Ростом будем называть отображение $grow: T \to G$ одного градуированного графа в другой такой, что:

$$t_1 t_2 \in E(T) \vee t_1 = t_2 \Rightarrow grow(t_1)grow(t_2) \in E(G) \vee grow(t_1) = grow(t_2)$$

Такое отображение может склеивать две связанные вершины в одну.

Опр: Обобщенная перестановка σ — это конечное множество клеток диаграммы S (никакие две из них не лежат в одной строке или столбце).

Обычный RSK устанавливает соответствие между множествами двустрочных лексикографических упорядоченных массивов и парами таблиц Юнга (или другой вариант формулировки: RSK связывает с любой перестановкой пару путей в графе Юнга). Обобщенный RSK верен не только для графа Юнга, но и для любого модулярного графа. Его связь с обычным RSK: обобщенный можно конкретизировать для графа Юнга и получить классическое соответствие.

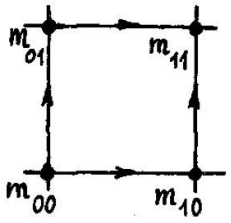
Сейчас будет *очень кратко* сформулировано обобщенное RSK-соответствие для модулярных графов, ибо для того, чтобы сформулировать его хоть немного подробнее, потребуется в два раза больше определений ☹. В целом, я надеюсь, что Гиршу хватит только теоремы ниже, ибо и так достаточно сложно вышло.

Теорема (Обобщенное RSK-соответствие для модулярного графа G):

Пусть S – косая диаграмма. Тогда обобщенное RSK-соответствие – это сквозное отображение, сопоставляющее произвольному росту $\partial_+(S) \to G$ обобщенную перестановку.

Опр: Пусть S – косая диаграмма. Тогда рост $grow: S \to G$ называется двумерным.

Опр: Двумерный рост $M: S \to \mathbb{Z}$ называется полумодулярным, если значения $m_{00}, m_{10}, m_{01}, m_{11}$, которые он принимает в вершинах произвольной клетки диаграммы S, связаны неравенством $m_{00} + m_{11} \geq m_{01} + m_{10}$.



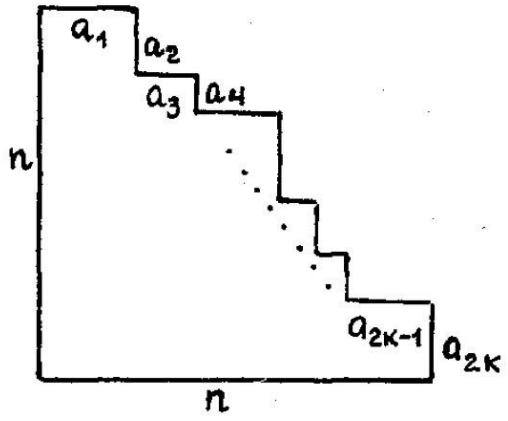
Опр: Если $grow: T \to G$ – рост, то отображение $t \mapsto |grow(t)|$ является ростом $T \to \mathbb{Z}$, называется модулем роста $grow$ и обозначается $|grow|$.

Теорема (Обобщенное RSK-соответствие для $\boldsymbol{Y}$-графа G):

Пусть G является $\boldsymbol{Y}$-графом, S – косая диаграмма. Тогда существует биективное соответствие, сопоставляющее каждому росту $grow^+: \partial_+(S) \to G$ пару $(grow^-, M)$, состоящую из роста $grow^-: \partial_-(S) \to G$ и полумодулярного роста $M: S \to \mathbb{Z}$, сужение которого на $\partial_-(S)$ совпадает с $|grow^-|$.

Теорема (применение RSK):

Пусть диаграмма S имеет вид:



(Рядом с каждым отрезком указана его длина).

Количество неориентированных замкнутых путей в $\boldsymbol{Y}$-графе G, начинающихся в нуле графа и имеющих следующую структуру: a_1 ребер "вверх" (в направлении ориентации графа G; напомню, что G ориентированный граф), затем a_2 ребер "вниз" (против ориентации графа), далее a_3 ребер "вверх" и т.д. равно числу n-клеточных перестановок, состоящих из клеток диаграммы S.

Современные методы в теоретической информатике
Конечные автоматы и разрешимость монадических теорий

Золотов Б.

Монадическая логика — это логика, расширяющая логику первого порядка, но являющаяся фрагментом логики второго порядка. В её формулах и предложениях разрешены кванторы по одноместным предикатам: уже не только по элементам, но ещё не по всем предикатам. Например, формула монадической логики в языке, содержащем предикат $\leq$ —

$$\exists X\ \exists x_0 \in X\ \forall y \in X\ x_0 \leq y.$$

Теорема: Глядя на конечный автомат, можно эффективно выяснить, принимает ли он хоть одно непустое слово.

Конечные цепи

Монадический язык одного потомка — язык первого порядка с предикатами $\subseteq$ и SUC:

$\mathrm{SUC}(X,Y) = \exists x, y\ X = \{x\},\ Y = \{y\}$, y — следующий за x элемент.

Любое конечное вполне упорядоченное множество (ВУМ) естественным образом является моделью этого языка.

Пусть дано конечное ВУМ $C = \{1, \ldots, N\}$ и n его подмножеств $X_1, \ldots, X_n$. Построим по ним слово $\mathrm{Word}(C, X_1, \ldots, X_n)$ длины N над алфавитом $\Sigma_n = \{0,1\}^n$:

1	$\begin{pmatrix}1\\0\\1\\\vdots\\1\end{pmatrix}$	$\begin{pmatrix}0\\1\\1\\\vdots\\0\end{pmatrix}$	$\ldots$	$\begin{pmatrix}1\\1\\0\\\vdots\\1\end{pmatrix}$
2				
3				
$\vdots$				
n				
	1	2	1, если $j \in C_i$; 0 иначе	N

Теорема: Есть алгоритм, который по Σ_n–автомату A строит формулу $\varphi(X_1, \ldots, X_n)$ монадического языка одного потомка, такую что для любой конечной цепи C и любых n её подмножеств $X_1, \ldots, X_n$

$C \models \varphi(X_1, \ldots, X_n)$ если и только если A принимает $\mathrm{Word}(C, X_1, \ldots, X_n)$.

Теорема: Есть алгоритм, который по формуле $\varphi(X_1, \ldots, X_n)$ строит автомат A над Σ_n, такой что

то же самое.

Теорема: Монадическая теория конечных цепей разрешима. Это потому, что по предложению (формуле без свободных переменных) строится автомат над $\Sigma_0 = \{\circ\}$, и предложение выполняется в какой-то из цепей, если автомат примет хотя бы одно слово.

Цепь $\mathbb{N}$

Очевидно, является моделью монадического языка одного потомка.

Нам потребуются NDFA, работающие на бесконечных строках: $A = (S, T, s_{\text{in}}, F)$, где T — таблица переходов, а $F \subset 2^S$ — множество *финальных наборов* состояний. Запуск является принимающим, если

$$\{s \mid \text{автомат посещает } s \text{ беск. много раз}\} \in F.$$

Результаты этого раздела аналогичны результатам предыдущего, с тем лишь отличием, что вместо $\text{Word}(C, X_1, \ldots, X_n)$ по тому же принципу строится $\text{SEQ}(\mathbb{N}, X_1, \ldots, X_n)$.

Бесконечное двоичное дерево

В этом разделе я успел рассказать только определения.

Двоичное дерево — это $\{l, r\}^*$. Монадический язык двух потомков содержит предикаты $\subseteq$, Left и Right. Σ–оценка дерева — запихивание символов алфавита Σ в его узлы.

Древесный Σ–автомат — четвёрка (S, T, T_{in}, F).

$$T \subseteq S \times \{l, r\} \times \Sigma \times S \text{ — таблица переходов}$$
$$T_{\text{in}} \subseteq \Sigma \times S \text{ — таблица начальных состояний}$$
$$F \subset 2^S \text{ — все финальные наборы состояний}$$

То, принимает ли автомат слово, выясняется по итогам *игры* между автоматом и «Сусаниным»: на нечётном ходу автомат выбирает состояние, а на чётном ходу «Сусанин» говорит, в какую сторону спускаться из текущего узла:

A chooses:	Pathfinder chooses:
s_0	
	d_1
s_1	
	d_2
s_2	
	d_3
s_3	
	...

Here each $s_n \in S$ and each $d_n \in \{l, r\}$. The choices of A are restricted by the following conditions:

$$(V(e), s_0) \in T_{\text{in}} \quad \text{and} \quad (s_n, d_{n+1}, V(d_1 \ldots d_{n+1}), s_{n+1}) \in T.$$

Автомат выигрывает, если $\{s \mid s \text{ выбиралось беск. много раз}\} \in F$, и принимает дерево с символами из Σ в узлах, если обладает выигрышной стратегией.

Наконец, last appearance record для позиции в дереве — строка из символов, находящихся в предках данной позиции, где каждый символ встречается ровно один раз — на месте своего самого нижнего появления в предках. То есть, строке *abacca* будет соответствовать *bca*.

Теорема: Монадическая теория двоичного дерева разрешима.

Теорема: По древесному автомату A можно эффективно построить автомат A', принимающий в точности деревья, отвергаемые автоматом A. *(«Теорема о дополнении»)*