1 Сложность задачи об эквивалентности регулярных и расширенных регулярных выражений

1.1 Регулярные выражения

Мы говорим что машина Тьюринга работает за зоне, ограниченной функцией $s(n)$, если на любом входе длины n машина заканчивает свою работу, не удаляясь от начала ленты далее, чем на $s(n)$ ячеек. Через PSPACE мы обозначаем семейство языков, распознаваемых на зоне, ограниченной некоторым полиномом.

Теорема 1. *Задача эквивалентности регулярных выражений PSPACE-полна.*

Утверждение теоремы означает, что язык, состоящий из пар эквивалентных регулярных выражений, принадлежит PSPACE и любой язык из PSPACE сводится к нему. То есть, для любого языка L из PSPACE можно указать полиномиальный по времени алгоритм, который по данному слову x находит регулярные выражения R'_x, R''_x с таким свойством: R'_x эквивалентно R''_x тогда и только тогда, когда $x \in L$.

Доказательство. Сначала докажем, что эта задача принадлежит PSPACE. По регулярному выражению за полиномиальное время можно построить эквивалентый ему недетерминированный автомат, то есть, автомат допускающий тот же язык. Поэтому достаточно на полиномиальной зоне уметь распознавать эквивалентность двух недетерминированных автоматов A и B. Для этого обычным образом построим детерминированный автомат C, допускающий симметрическую разность множеств, допускаемых автоматами A и B. Затем выясним, допускает ли полученный автомат хотя бы одно слово. Здесь имеется следующая трудность. Автомат C имеет экспоненциальное число состояний, точнее 2^{m+n} состояний, где m, n числа состояний A и B. Поэтому выписать таблицу переходов C мы не можем. Однако это нам и не нужно. Состояния автомата C естественным образом представляются двоичными словами длины $m + n$. При этом для каждой буквы σ из алфавита автомата отношение $E_\sigma(a, b)$ "прочитав σ в состоянии a автомат C переходит в состояние b" разрешимо за полиномиальное время. Отношения "быть начальным состоянием" и "быть допускающим состоянием" также полиномиально разрешимы.

Поэтому нам достаточно научиться на полиномиальной зоне решать следующую задачу. Пусть имеется отношение E на множестве двоичных слов, которое разрешимо на полиномиальной зоне. Рассмотрим следующую задачу: даны два слова s, t одной длины. Надо узнать существует ли последовательность слов $x_1, \ldots, x_k$ той же длины, такая, что $E(s, x_1)$, $E(x_1, x_2)$, $\ldots$, $E(x_{k-1}, x_k)$, $E(x_k, t)$. Иными словами, надо узнать, есть ли путь из вершины s в вершину t в ориентированном графе, вершины которого — слова длины n, а ребра задаются отношением E. Будем называть эту задачу "задачей s-t-достижимости".

Допустим для любого E, разрешимого на полиномиальной зоне, мы умеем решать задачу s-t-достижимости на полиномиальной зоне. Тогда возьмем в качестве E объединение по $\sigma \in \Sigma$ отношений $E_\sigma(a, b)$. В качестве s возьмем начальное состояние, а в качестве t одно из заключительных. Тогда существование пути из s в t означает, что автомат допускает хоть одно слово, приходя из начального состояния в состояние t. Перебирая все допускающие состояния t и решая задачу достижимости t из s, мы узна́ем, допускает ли автомат C хотя бы одно слово.

Теорема 2 (Сэвич). *Для любого E, разрешимого на полиномиальной зоне, задача s-t-достижимости принадлежит PSPACE.*

Доказательство. С каждым натуральным числом n и словами s, t длины n свяжем следующую игру двух игроков. В игре имеются позиции двух видов — $\langle x, y, i\rangle$ и $\langle x, z, y, i\rangle$, где x, y, z — слова длины n, а i — натуральное число, не превосходящее n. Во всех позициях первого вида ход первого игрока и он имеет право добавить в позицию любое слово z длины n, не изменяя i, то есть перейти в позицию $\langle x, z, y, i\rangle$. В позициях второго вида ход второго игрока, и он имеет право удалить либо x, либо y, уменьшив при этом i на единицу, то есть перейти в любую из двух позиций $\langle x, y, i - 1\rangle$, $\langle y, z, i - 1\rangle$.

Игра начинается в позиции $\langle s, t, n\rangle$ и заканчивается, когда после хода второго игрока, уменьшающего число i, оно станет равным нулю. При этом выигрывает первый, если $x = y$ или $E(x, y)$, а иначе — второй.

Докажем, что путь из s в t существует тогда и только тогда, когда в игре выигрывает первый игрок. Пусть путь есть. Тогда первый игрок может играть, обеспечивая перед своим ходом (когда текущая позиция имеет вид $\langle x, y, i\rangle$) истинность такого инварианта: "существует путь из x в y длины не более 2^i". В графе 2^n вершин, поэтому существует путь из s в t длины не более 2^n. Значит в начальный момент инвариант выполнен. Чтобы сохранить инвариант, первый игрок выбирает любой путь указанной длины, соединяющий x и y и в качестве z берет вершину в середине этого пути. Ясно, что какую бы вершину x или z ни удалил второй игрок, оставшиеся вершины будут соединены путем длины не более 2^{i-1}. Когда игра закончится, и i станет равным 0, из вершины x в вершину y будет путь длины 0 или 1, что и означает выигрыш первого игрока. Заметим, что указанная стратегия не является полиномиальной ни по времени, ни по зоне, однако этого и не требуется.

Обратно, пусть нет пути из s в t. Тогда второй игрок перед любым ходом противника может обеспечить истинность такого утверждения: "не существует никакого пути из x в y". По условию в начальной позиции это верно. Пусть это верно в позиции $\langle x, y, i\rangle$ перед ходом первого игрока. Тогда какую бы вершину z ни выбрал первый игрок, либо из x в z нет пути, либо из z в y нет пути (либо и то, и другое одновременно). В первом случае второй игрок стирает y, а иначе x.

Нетрудно написать рекурсивную процедуру, получающую на вход позицию p, и определяющую, кто из игроков имеет выигрышную стратегию в игре, начинающейся с позиции p. Если позиция заключительная, то процедура определяет, кто выиграл, просто запустив полиномиальный по зоне алгоритм разрешения отношения E. Иначе, процедура перебирает все возможные ходы того игрока i, который должен ходить, и, вызывая рекурсивно себя на полученной позиции p', узнает, кто в ней выигрывает. Если в хотя бы одной такой позиции p' выигрывает игрок i, то она объявляет эту позицию выигрышной для него, иначе — для другого. Поскольку каждый из игроков делает n ходов, количество рекурсивных вызовов на исходной позиции равно $2n$. При каждом вызове надо запоминать только текущую позицию p' и константу битов дополнительной информации. Поэтому при работе на исходной позиции программе понадобится $O(n^2)$ ячеек плюс количество ячеек, необходимое для разрешения отношения E на словах длины n. □

Замечание. Формулировку теоремы можно усилить следующим образом. Пусть отношение E дается машине в виде оракула, то есть в любой момент вычисления машина может на специальной ленте записать пару слов одной длины, после чего "оракул" сообщает ей, смежны ли они в графе. Такие машины называются "машинами с оракулом". Усиленная формулировка такова: существует одна полиномальная по зоне машина с оракулом, которая решает задачу о s-t-достижимости для всех E. (При этом мы уже не предполагаем, что E разрешимо на полиномиальной зоне.) В сущности, в доказательстве мы построили машину, решающую эту задачу на зоне $O(n^2)$. Совсем недавно О. Рейнгольдом доказана разрешимость этой задачи *для неориентированных графов* на зоне $O(n)$.

Осталось доказать PSPACE-трудность задачи эквивалентности регулярных выражений. На самом деле мы установим PSPACE-трудность более частной задачи распознавания универсальности регулярных выражений: по данному регулярному выражению над данным алфавитом Σ выяснить, задает ли оно множество всех слов над алфавитом Σ.

Пусть дан произвольный язык из PSPACE и машина M, распознающая его на полиномиальной зоне. Будем считать, что машина M должна остановиться в некотором состоянии q_0, если исходное слово x принадлежит L и в любом другом состоянии, иначе. Для каждого слова x мы найдем за полиномиальное время регулярное выражение R_x, которое универсально тогда и только тогда, когда M не допускает x, то есть вычисление M на x заканчивается не в состоянии q_0.

Пусть $p(n)$ — полином ограничивающий количество использованных ячеек машиной M на входах длины n. Запустим мысленно M на слове x длины n и запишем конфигурации $C_0, C_1, \ldots, C_T$ машины во все моменты времени вплоть до остановки. Конфигурация C_t машины в данный момент времени — это по определению слово длины $p(n) + 1$, состоящее из символов, записанных на ленте машины в первых $p(n)$ ячейках (в остальных одни пробелы) и текущего состояния машины. Состояние вставим

перед символом, обозреваемым в данный момент головкой. Слово $\#C_0\#C_1\#\ldots\#C_T\#$ будем называть протоколом вычисления M на x. Регулярное выражение R_x будет задавать все слова над алфафитом Δ, кроме протокола M на x, если M допускает x. Алфавит Δ состоит из ленточного алфавита M, множества состояний машины и разделителя $\#$.

Сначала заметим следующее. Символ номер $i+p(n)+2$ в слове $\#C_0\#C_1\#\ldots\#C_T\#$ однозначно определяется четверкой символов с номерами $i-1, i, i+1, i+2$ этого слова. Функция f, определяющая эту зависимость, задается программой машины M и нам известна.

В регулярном выражении R_x будут перечисляться все причины, по которым данное слово w может не являться протоколом допускающего вычисления на x. Вот эти причины.

1. Слово не начинается или не заканчивается на $\#$.
2. Количество букв между какими-то соседними вхождениями символа $\#$ не равно $p(n)+1$.
3. Слово, записанное между первыми двумя вхождениями $\#$ не равно начальной конфигурации M на x.
4. Слово, записанное между последними двумя вхождениями $\#$ не содержит символа q_0.
5. В w найдется подслово вида $abcdve$, где a, b, c, d, e симвлолы, а v слово длины $p(n)-2$, и e отлично от $f(a, b, c, d)$.

Нетрудно для каждой причины написать регулярное выражение длины $\mathrm{poly}(n)$, задающее соответствующее множество слов. Например, для последней причины оно выглядит так:

$$\Delta^* \cdot \Big(\bigcup_{abcd \in \Delta^4} abcd \cdot \Delta^{p(n)-2} \cdot (\Delta \setminus \{f(a,b,c,d)\}) \Big) \cdot \Delta^*.$$

Здесь мы использовали сокращение $\Delta^{p(n)-2}$ для записи многократной конкатенации, и другие сокращения. Без сокращений выражение имеет длину, пропорциональную $p(n)$.

Написав объединение всех этих выражений, мы получим искомое выражение R_x. □

1.2 Расширенные регулярные выражения

Теперь разрешим в регулярных выражениях наряду с операциями $\cup, \cdot, *$ использовать еще и операцию пересечения множеств. Такие регулярные выражения будем называть расширенными. Мы докажем, что задача распознавания универсальности расширенных выражений уже не принадлежит PSPACE. В доказательстве мы будем использовать теорему о иерархии по зоне (частный случай).

Теорема 3 (о иерархии). *Существует язык, распознаваемый некоторой машиной Тьюринга на зоне 2^n, но не распознаваемый никакой машиной Тьюринга на зоне $o(2^n)$.*

Доказательство. Ослабим немного требования к языку и построим сначала разрешимый язык, не распознаваемый никакой машиной на зоне $o(2^n)$. При этом не будем заботиться, чтобы он распознавался на зоне 2^n. Это делается диагональным методом. Будем записывать программы всевозможных машин Тьюринга в виде слов какого-нибудь фиксированного алфавита Σ. Рассмотрим язык

$$L = \{px \in \Sigma^* \mid p \text{ — программа некоторой машины, которая допускает вход } px, \text{ не выходя за пределы первых } 2^{|px|)} \text{ ячеек на ленте}\}.$$

Мы предполагаем, что программы записываютя в таком виде, что читая слово px слева направо, можно понять, где заканчивается p и начинается x. Дополнение языка L не распознается никакой машиной, работающей на зоне $o(2^n)$: машина с программой p ошибается на всех достаточно длинных входах вида px. Действительно, при длинных x машина p на входе px будет использовать не более $2^{|px|}$ ячеек и поэтому $px \in L$ тогда и только тогда, когда машина допускает px.

С другой стороны язык L (следовательно, и его дополнение) разрешим. Чтобы выяснить, принадлежит ли исходное слово w языку L, мы сначала выясняем, имеет ли оно вид px, где p является записью программы некоторой машины. Если нет, то отвергаем вход. Иначе запускаем эту машину на входе px и моделируем ее работу в течение не более, чем $T = 2^{c2^{|px|}}$ шагов. Константу c подберем так, чтобы T превосходило общее число конфигураций машины длины $2^{|px|}$. Если машина попытается выйти из зоны $2^{|px|}$ или не остановится в отведенное ей время, то отвергаем вход (если она не остановилась, но уложилась в зону, то какая-то конфигурация повторилась дважды в процессе вычисления, а значит машина уже не остановится никогда). Если этого не случится и машина допустит вход, то и мы его допу́стим, а иначе отвергнем.

Теперь оценим, сколько места на ленте требуется для распознавания языка L. Для хранения текущей конфигурации машины p длины 2^n нужно $c(p)2^n$ ячеек, где $c(p)$ некоторая функция от p. Умножение на константу необходимо, поскольку ленточный алфавит машины p может быть произвольным и нам придется кодировать его символы несколькими символами нашего алфавита. Можно так записывать в виде слов программы машин Тьюринга, что для моделирования такой памяти и достаточно. Кроме этого, нам нужно хранить счетчик числа шагов, к которому мы добавляем единицу после каждого шага моделирования. Для хранения счетчика в двоичной системе счисления понадобится $\log 2^{c2^{|px|}} = 2^{|px|}$ ячеек, где c также зависит от p. Таким образом нам требуется $c(p)2^{|px|}$ ячеек, что больше, чем разрешенные $2^{|px|}$

Поэтому изменим немного язык, сохранив основную идею. Опишем машину, распознающую измененный язык, задав тем самым язык. Получив на вход слово w длины n, с самого начала отмеряем на ленте 2^n ячеек, и если в какой-то момент нам захочется выйти за отмеренное пространство, тут же останавливаемся и отвергаем входное слово. Тем самым определяемый язык будет распознаваться на зоне 2^n. Кроме того, заводим бинарный счетчик шагов длины 2^n, который сначала равен нулю. Затем, как и раньше, выясняем, имеет ли входное слово w вид px, где p является записью программы некоторой машины. При разумной форме записи программ для этого понадобится места не больше, чем занимает сама программа p. Если исходное слово не имеет такого вида, то отвергаем вход. Иначе запускаем машину p на входе w и моделируем ее работу в течение не более, чем 2^{2^n} шагов. Выполнив каждый шаг работы p, увеличиваем счетчик на 1. Если при моделировании нам не хватит 2^n ячеек на ленте или программа p будет работать более 2^{2^n} шагов, то отвергаем вход. Если этого не случится и машина допустит вход, то и мы его допустим, а иначе отвергнем.

Дополнение языка $\tilde{L}$, распознаваемого этой машиной, не распознается никакой машиной на зоне $o(2^n)$. Действительно, рассмотрим произвольную машину с программой p и зоной $o(2^n)$. Для доведения моделирования p на входе px до конца, нам достаточно $c(p)o(2^n)$ ячеек на ленте. При достаточно больших n это меньше 2^n. Время работы p не превосходит эспоненты от $c(p)o(2^n)$, что опять же меньше 2^{2^n} при достаточно больших n. Значит для достаточно длинных x слово px принадлежит $\tilde{L}$ тогда и только тогда, когда оно принимается машиной p. Следовательно, машина p не может распознавать дополнение к $\tilde{L}$. □

Теорема 4. *Не существует машины Тьюринга, распознающей универсальность расширенных регулярных выражений над двоичным алфавитом на зоне* $2^{o(\sqrt{n/\log n})}$.

Доказательство. Наш план таков. Рассмотрим любой язык L распознаваемый на зоне 2^n, но не распознаваемый на зоне $o(2^n)$, и машину M распознающую его на зоне 2^n. По слову x длины n мы построим за полиномиальное время расширенное регулярное выражение R_x длины $O(n^2 \log n)$, которое универсально, тогда и только тогда, когда машина отвергает x. Почему отсюда следует, что задача распознавания универсальности расширенных регулярных выражений сложна? Допустим можно решать эту задачу на зоне $2^{o(\sqrt{n/\log n})}$. Тогда по слову x длины n за время $\mathrm{poly}(n)$ найдем выражение R_x длины $O(n^2 \log n)$. Затем на зоне

$$2^{o(\sqrt{(n^2 \log n)/\log(n^2 \log n))}} = 2^{o(n)}$$

выясняем, универсально ли выражение R_x. Таким образом, мы распознаем L на зоне $2^{o(n)} = o(2^n)$.

Полученное противоречие показывает, что невозможно распознавать универсальность расширенных регулярных выражений над двоичным алфавитом на зоне $2^{o(\sqrt{n/\log n})}$.

Осталось объяснить, как построить выражение R_x. Для этого нам понадобится построить некоторое короткое регулярное выражение, задающее все слова кроме одного длинного слова. Точнее, мы построим регулярное выражение длины $O(n^2)$ в некотором алфавите A из n символов, которое содержит все слова над алфавитом A, кроме некоторого слова d_n длины больше 2^n. Кроме того, мы построим расширенное регулярное длины $O(n^2)$ для множества всех циклических сдвигов слова $d_n\#$. Допустим, у нас уже есть такие выражения. Тогда выражение R_x строится так. Для данного слова x длины n опять рассмотрим протокол $\#C_0\#C_1\#\ldots\#C_T\#$ вычисления M на x. На этот раз слова C_i имеют ту же длину, что и слово d_n, что больше 2^n. Подпишем в протоколе слово d_n под каждым из слов $C_0, C_1, \ldots, C_T$. Полученное слово в новом алфавите Σ, состоящем из $\#$ и "двухэтажных" букв, будем называть, модифицированным протоколом. Регулярное выражение R_x будет задавать все слова над этим алвафитом, кроме модифицированного протокола вычисления M на x, если это вычисление допускающее.

По каким причинам слово w может не быть модифицированным допускающим протоколом? Для ответа на этот вопрос рассмотрим два отображения: g, h, которые определены на множестве слов над алфавитом Σ следующим образом. Отображение g стирает все буквы нижнего этажа, а отображение h — все буквы верхнего этажа. Символ $\#$ ни одно из них не меняет. Вот причины, по которым слово w может не быть модификацией допускающего протокола.

1. Слово w не начинается или не заканчивается на $\#$.
2. В слове $h(w)$ между какими-то двумя последовательными вхождениями $\#$ стоит слово отличное от d_n.
3. В слове w одна из первых $n+2$ букв не такая, как в модифицированном протоколе (эти буквы соответствуют нетривиальной части начальной конфигурации, содержащей слово x и начальное состояние машины).
4. В слове $g(w)$ между первыми двумя вхождениями $\#$ какая-то из букв с номером, большим $n+2$, отлична от пробела.
5. В слове $g(w)$ между последними двумя вхождениями $\#$ нет вхождения символа q_0.
6. В слове w найдется подслово вида $abcdve$, где a, b, c, d, e символы, а v слово, такие, что $h(cdve)$ есть циклический сдвиг слова $d_n\#$ и $g(e)$ отлично от $f(g(a), g(b), g(c), g(d))$.

Как же написать регулярное выражения для множеств, задаваемых каждой причиной? Для этого полезно сделать следующее наблюдение. Если имеется отображение g из множества слов в алфавите Σ в множество слов в алфавите Δ, которое действует побуквенно, то есть, заменяет каждую букву алфавита Σ на некоторую букву алфавита Δ и имеется расширенное регулярное выражение R длины l над Δ, то можно построить расширенное регулярное выражение длины $O(l)$, задающее $g^{-1}(S)$ — прообраз множества слов S, задаваемых R. Действительно, отображение g^{-1} коммутирует с операциями объединения, пересечения, конкатенации и итерации, поэтому достаточно в выражении R заменить каждую букву δ на $g^{-1}(\delta)$. Будем полученное таким образом выражение обозначать через $g^{-1}(R)$.

Пусть g, h — отображения определенные выше (g стирает буквы нижнего этажа, а h — верхнего). Тогда выражение $g^{-1}(R)$ имеет длину в n раз большую, чем R, а выражение $h^{-1}(R)$ — в константу раз большую, чем R.

Расширенное выражение для множества из второго пункта есть $\Sigma^* \cdot \# \cdot g^{-1}(R) \cdot \# \cdot \Sigma^*$, где R есть регулярное выражение, задающее все слова в алфавите A, кроме d_n.

Выражение для третьего пункта есть объединение $n+2$ выражений (по одному на каждую букву) следующего вида. Выражение номер i есть $b_1 b_2 \ldots b_{i-1}(\Sigma - b_i)\Sigma^*$, где b_j обозначает j-ую букву модифицированного протокола вычисления на x. Его длина есть $O(n)$ и оно задает множество всех слов, у которых первые $i-1$ букв правильные, но i-ая неправильна.

Выражение для четвертого пункта есть $\#(\Sigma - \#)^{n+1}(\Sigma - \#)^* g^{-1}(\Delta - \text{пробел})\Sigma^*$. Его длина равна $O(n^2)$ (длина выражения для $\Sigma - \#$ равна $O(n)$ и это выражение повторяется $O(n)$ раз; остальные составные части имеют длину $O(n)$).

Аналогично строится выражение для пятого пункта.

А для шестого пункта годится такое выражение:

$$\Sigma^* \cdot \Big(\bigcup_{abcd \in \Delta^4} \big(g^{-1}(abcd \cdot \Delta^* \cdot (\Delta \setminus \{f(a,b,c,d)\})) \cap h^{-1}(A \cdot A \cdot R')\big) \cdot \Sigma^*,$$

где R' выражение, задающее множество всех циклических сдвигов слова $d_n\#$, а Δ есть алфавит протоколов. Самая длинная составная часть этого выражения $h^{-1}(A \cdot A \cdot R')$ имеет длину $O(n^2)$.

Построенное выражение R_x имеет длину $O(n^2)$ и алфавит из $O(n)$ символов. При переходе к двоичному алфавиту его длина увеличится в $\log n$ раз из-за необходимости кодировать символы алфавита размера $O(n)$ нулями и единицами.

Осталось указать слово d_n и построить короткие выражения для множества всех слов, кроме d_n и для множества всех циклических сдвигов слова $d_n\#$. Положим $A = \{a_1, \ldots, a_n\}$ и определим d_n по индукции. Положим d_0 равным пустому слову и $d_{n+1} = d_n a_{n+1} d_n a_{n+1}$.

Сначала зададим само слово d_n выражением длины $O(n^2)$. Это выражение есть пересечение выражений $R_1, \ldots, R_n$. Здесь

$$R_i = \big((A_{<i})^+ \cdot a_i \cdot (A_{<i})^+ \cdot a_i \cdot (A_{>i})^+\big)^*,$$

где $A_{<i}$ обозначает выражение $a_1 \cup \cdots \cup a_{i-1}$ и аналогично понимается $A_{>i}$. Выражение B^+ есть сокращение для B^*B. Отдельно определим

$$R_n = (A_{<n})^+ \cdot a_n \cdot (A_{<i})^+ \cdot a_n.$$

Говоря неформально, выражение R_i описывает, как выглядит d_n с точки зрения вхождений буквы a_i. Докажем, что выражение $R_1 \cap \cdots \cap R_n$ задает d_n. Ясно, что d_n принадлежит этому языку (это легко доказать индукцией по n). Чтобы доказать, что в нем нет других слов, нам понадобится следующая

Лемма 1. *Пусть слово x принадлежит языку $R_1 \cap \cdots \cap R_i$, где $i < n$. Тогда слово x принадлежит и языку*

$$\big(d_i \cdot (A_{>i})^+\big)^*.$$

Доказательство. Для $i = 1$ это очевидно. Пусть утверждение выполнено для $i-1$ и пусть x — произвольное слово из $R_1 \cap \cdots \cap R_i$. Так как x принадлежит множеству

$$R_i = \big((A_{<i})^+ \cdot a_i \cdot (A_{<i})^+ \cdot a_i \cdot (A_{>i})^+\big)^*,$$

символы с номерами, большими i, разделяют его на блоки, причем каждый блок u принадлежит множеству $(A_{<i})^+ \cdot a_i \cdot (A_{<i})^+ \cdot a_i$ По индуктивному предположению x принадлежит множеству

$$\big(d_{i-1} \cdot (A_{\geq i})^+\big)^*.$$

Следовательно, каждый блок u разделяется буквами a_i на блоки d_{i-1}, то есть u принадлежит $a_i^*(d_{i-1}(a_i)^+)^* a_i^*$. Поскольку в каждом блоке u только две буквы a_i, причем после последняя в конце, а перед первой и между первой и второй непустые слова, блок u имеет вид $d_{i-1}a_i d_{i-1}a_i$, то есть совпадает с d_i. Лемма доказана. □

По лемме, любое слово x из $R_1 \cap \cdots \cap R_{n-1}$ имеет вид

$$\big(d_{n-1}a_n^+\big)^*.$$

Сравнивая это выражение с R_n, мы видим, что x равно d_n. С другой стороны, d_n принадлежит выражению $R_1 \cap \cdots \cap R_n$, поэтому это выражение задает d_n. Длина выражения R_i есть $O(n)$, поэтому общая длина выражения $R_1 \cap \cdots \cap R_n$ равна $O(n^2)$.

Теперь легко построить регулярное выражение для множества $A^* \setminus \{d_n\}$. Для этого достаточно написать регулярное выражение для дополнения R_i. Пусть $i < n$. В каком случае слово x не принадлежит R_i? На это есть четыре причины.

1. Количество вхождений символа a_i нечетно.
2. Между каким-то нечетным вхождением a_i и следующим за ним четным вхождением a_i (или между началом слова и первым вхождением a_i) есть символы из $A_{\geq i}$ или нет ничего.
3. Между каким-то четным вхождением a_i и следующим за ним нечетным вхождением a_i какой-то символ $A_{<i}$ встречается левее какого-то символа из $A_{>i}$, или нет ни одного вхождения символов из $A_{<i}$, или нет ни одного вхождения символов из $A_{>i}$.
4. После последнего вхождением a_i есть вхождение какого-то символа из $A_{<i}$ или нет ни одного вхождения символов из $A_{>i}$.

Множество слов, удовлетворяющих каждому из четырех пунктов, задается регулярным выражением длины $O(n)$. Аналогично для дополнения до R_n.

Осталось задать множество слов, получающееся всевозможными циклическими сдвигами слова $d_n\#$. Для этого рассмотрим определявшиеся ранее выражения $R_1, \ldots, R_n$. Будем в их определении $A_{>i}$ понимать как $a_{i+1} \cup \cdots \cup a_n \cup \#$, и изменим R_n следующим образом

$$R_n = (A_{<n})^* \cdot a_n \cdot (A_{<n})^* \cdot a_n \cdot \#.$$

Очевидно, что полученное таким образом выражение $R_1 \cap \cdots \cap R_{n-1} \cap R_n$ задает слово $d_n\#$. Напишем теперь для каждого i выражение R'_i длины $O(n)$, задающее множество всех циклических сдвигов слов из R_i. Для этого рассмотрим любое слово x из R_i. Пусть $x' = uv$, где $x = vu$, некоторая циклическая перестановка слова x. Граница между u и v может попасть между двумя блоками вида $(A_{<i})^+ \cdot a_i \cdot (A_{<i})^+ \cdot a_i \cdot (A_{>i})^+$, внутрь первого слова $(A_{<i})^+$, между первым словом $(A_{<i})^+$ и a_i, и так далее. Всего имеется лишь конечное число вариантов для положения границы, и для каждого варианта можно написать выражение длины не более, чем в константу раз, превышающей длину R_i для множества всех циклических сдвигов этого типа. Например, для сдвигов второго типа (граница попала внутрь внутрь первого слова $(A_{<i})^+$) выражение будет таким

$$\big((A_{<i})^+ \cdot a_i \cdot (A_{<i})^+ \cdot a_i \cdot (A_{>i})^+\big)^* (A_{<i})^+.$$

Аналогичным образом можно написать выражение для множества слов, получающегося циклическими сдвигами слов из R_n. Докажем, что пересечение выражений $R'_1 \cap \cdots \cap R'_n$ и задает множество циклических сдвигов $d_n\#$. Заметим, что операции пересечения и перехода к множеству циклических сдвигов не коммутируют. Например, пересечение синглетонов $\{ab\}$ и $\{ba\}$ пусто. Однако после перехода к множеству циклических сдвигов получим два одинаковых множества $\{ab, ba\}$. Тем не менее, в одну сторону включение верно: пересечение циклических сдвигов включает циклический сдвиг пересечения. Поэтому язык $R'_1 \cap \cdots \cap R'_n$ содержит d_n и все его циклические сдвиги. Чтобы доказать, что он больше ничего не содержит, нам понадобится следующий аналог леммы 1.

Лемма 2. *Пусть слово x принадлежит языку $R'_1 \cap \cdots \cap R'_i$, где $i < n$. Тогда некоторый циклический сдвиг слова x принадлежит языку*

$$\big(d_i \cdot (A_{>i})^+\big)^*.$$

Доказательство. Для $i = 1$ это очевидно. Пусть утверждение выполнено для $i - 1$ и слово x принадлежит $R'_1 \cap \cdots \cap R'_i$. Тогда его некоторый циклических сдвиг x' принадлежит множеству

$$\big(d_{i-1} \cdot (A_{\geq i})^+\big)^*$$

а какой-то другой сдвиг x'' принадлежит языку

$$R_i = \big((A_{<i})^+ \cdot a_i \cdot (A_{<i})^+ \cdot a_i \cdot (A_{>i})^+\big)^*.$$

Ясно, что слово x'' является циклическим сдвигом слова x'. Но все сдвиги слов из $\left(d_{i-1} \cdot (A_{\geq i})^+\right)^*$, принадлежащие R_i, сами принадлежат $\left(d_{i-1} \cdot (A_{\geq i})^+\right)^*$ (если мы перенесем, часть букв из d_{i-1} в конец, то получим слово, заканчивающееся на букву из $A_{<i}$, а слова из R_i могут заканчиваться только на буквы из $A_{>i}$). Поэтому x'' принадлежат языку $\left(d_{i-1} \cdot (A_{\geq i})^+\right)^*$. Отсюда, как и в лемме 1, получаем, что x'' принадлежит множеству

$$\left(d_{i-1}a_i d_{i-1}a_i \cdot (A_{>i})^+\right)^* = \left(d_i \cdot (A_{>i})^+\right)^*.$$

Лемма доказана. □

Таким образом, из того, что x принадлежит $R'_1 \cap \cdots \cap R'_n$, следует, что некоторый его циклический сдвиг x' принадлежит множеству

$$\left(d_{n-1} \cdot (A_{\geq n})^+\right)^*$$

и некоторый другой сдвиг x'' принадлежит множеству R_n. Аналогичными рассуждениями устанавливается, что и x'' принадлежит R_n, а значит $x'' = d_n\#$. Теорема доказана. □

В доказательстве теоремы мы построили регулярное выражение длины $O(n^2)$, допускающее все слова длины не более 2^n, но не универсальное. Преобразовав это выражение в недетерминированный автомат, мы получим автомат с $\mathrm{poly}(n)$ состоянимями, допускающий все слова длины не больше 2^n, но не все слова вообще. Заметим, что недетерминированность здесь существенна: если детерминированный автомат с k состояниями отвергает некоторое слово, то он отвергает некоторое слово длины меньше k (рассмотрим в автомате кратчайший путь из начального состояния в некоторое отвергающее состояние). И вообще, для детерминированных автоматов верна следующая оценка.

Лемма 3. *Если у детерминированного автомата A k состояний, а у детерминированного автомата B m состояний, и они эквивалентны на множестве слов длины не больше $k + m - 2$, то они эквивалентны на множестве всех слов.*

Доказательство. Создадим из автоматов A, B один автомат, поставив их рядом, не смешивая множества их состояний и не проводя никаких стрелок между ними. Назовем два состояния s и t полученного автомата эквивалентными, если множество слов, допускаемых по путям, выходящим из s, совпадает с множеством слов, допускаемых по путям, выходящим из t. Пользуясь этой терминологией, можно сказать, что исходные автоматы эквивалентны, если их начальные состояния эквивалентны. Также назовем два состояния i-эквивалентыми, если то же самое верно для слов длины не превосходящей i. Количество классов 0-эквивалентности равно 2: все допускающие состояния образуют один класс, а все отвергающие — второй класс. $(i+1)$-эквивалентность можно определять рекуррентно: состояния s и t $(i+1)$-эквивалентны тогда и только тогда, когда для любой буквы состояния, в которые переходит автомат из s и t, читая эту букву, i-эквивалентны. Поэтому, если отношение i-эквивалентности совпадает с отношением $(i+1)$-эквивалентности, то оно совпадает и с отношением j-эквивалентности для всех $j > i$, то есть совпадает с отношением (просто) эквивалентности. Каждый смежный класс i-эквивалентности разбивается на один или несколько классов $(i+1)$-эквивалентности. Поэтому число классов не убывает с ростом i. При этом, если число классов i-эквивалентности равно числу классов $(i+1)$-эквивалентности, то i-эквивалентность совпадает с просто эквивалентностью. Поскольку имеется два класса 0-эквивалентности и при любом i классов не может быть больше общего числа состояний, $(k+m-2)$-эквивалентность совпадает с просто эквивалентностью. Лемма доказана. □