Теорема Кука

Определение 1. *Пусть s — некоторая арифметическая функция. Рассмотрим машину Тьюринга со стеком D, у которой рабочие ленты конечны, а именно, каждая рабочая лента состоит ровно из $s(|w|)+2$ клеток при работе на произвольном слове*

$$w \in (\Sigma \setminus \{*, \#, \wedge\})^*.$$

Всего имеется одна входная и k рабочих лент. Ради краткости, будем говорить, что D есть s-СМТ.

Пусть $D = \langle \Sigma, Q, \pi, q_1 \rangle$, где

Σ — конечный алфавит; $\{*, \#, \wedge\}$ различны и входят в Σ;

Q — конечный алфавит состояний;

$q_1 \in Q$ — начальное состояние;

$! \notin Q$ — заключительное состояние;

$\dagger \notin \Sigma$ — дно магазина.

Функция π есть отображение

$$\pi : (Q \times (\Sigma)^{k+1} \times (\Sigma \cup \{\dagger\})) \to \\ ((Q \cup \{!\}) \times (\Sigma)^k \times (\{L, R, C\})^{k+1} \times (\Sigma \cup \{\dagger\})^*).$$

Таким образом, π по состоянию, обозреваемым символам на лентах и верхушке магазина выдаст символы, помещаемые в обозреваемые ячейки на рабочих лентах, новое состояние, направления движения на лентах и слово, помещаемое в магазин. π задаёт правила машины со стеком.

В самых левых клетках лент помещён символ $\#$. В самых правых клетках рабочих лент помещён символ $*$. Эти символы нельзя стирать и нельзя вновь помещать на ленты. На входной ленте в начальный момент после $\#$ помещается слово $w \in (\Sigma \setminus \{*, \#, \wedge\})^*$, а после этого слова идут "пустые" ячейки (ячейки, в которых находится $\wedge$). На рабочих лентах между самым левым и самым правым символом в начальный момент находятся "пустые" ячейки. В начальный момент магазин состоит из одного дна (символа $\dagger \notin \Sigma$). Если верхушка магазина есть дно, то в магазин обязательно помещается слово вида $\alpha\dagger$, в остальных случаях в магазин помещается слово вида α, где $\alpha \in (\Sigma)^*$.

Вначале головки на всех лентах находятся в самых левых клетках. Если на какой-то ленте обозревается левый (правый) конец, то *на этой ленте головка не движется влево (вправо). Если на входной ленте обозревается "пустая" клетка, то на этой ленте головка не движется вправо. В остальном определения рутинны, и мы их опускаем. Отметим только, что на каждом такте машина выталкивает верхушку магазина и после этого заталкивает в магазин указанное в правиле слово. Так как дно всегда остаётся в магазине, то магазин никогда не опустошается.*

Мы будем рассматривать только s-СМТ машины, которые останавливаются на любом входе и ни на одной рабочей ленте никогда не обозревают $*$. Если машина останавливается, то она выдаст либо 0, либо 1. В первом случае слово w воспринимается машиной, а во втором — отвергается. Множество всех воспринимаемых слов называется языком, задаваемым D, и обозначается через L_D.

При таком варианте определения D называется s-k-СМТ или просто s-СМТ. Аналогично определяется s-NСМТ (в этом случае существует для каждого входа останавливающееся вычисление, при котором ни на одной рабочей ленте никогда не обозревается $*$).

Целью данной лекции является доказательство двух теорем, принадлежащих С.Куку.

Теорема 1 (Кука). *Пусть s — некоторая арифметическая функция, асимптотически мажорирующая функцию $log(n)$ (то есть $log(n) \ll s$). Пусть Σ — алфавит, содержащий буквы 0, 1, $\wedge$ и, возможно, другие, $L \subseteq (\Sigma \setminus \{\wedge\})^*$, M — 1-МТ распознаватель, $L = L_M$ и $t_M \ll 2^{cs}$ для некоторой константы c.*

Тогда существует s-СМТ D, что $L_D = L$.

Начнем с комментария. Теорема представляется удивительной, потому что устанавливает тонкую взаимосвязь между временной сложностью языка и "почти"ленточной сложностью. В действительности неизвестно, можно ли в формулировке теоремы 1 заменить s-СМТ D на 1-МТ M' из класса $DSPACE(s)$. Это одна из самых интригующих проблем в общей теории сложности. Современная гипотеза: нет, нельзя.

Еще более удивительно, что верно и обращение теоремы 1 (собственно, обычно теорема Кука формулируется и доказывается в обе стороны).

Доказательство теоремы Кука. Итак, пусть M — 1-МТ из теоремы, $M = \langle \Sigma, Q, \pi, q_1 \rangle$, $w \in (\Sigma \setminus \{\wedge\})^*$.

Рассмотрим работу M на w.

Конфигурацией M на w будем называть кортеж из 8-ми элементов:

$$\langle w_1, \alpha, w_2, v, l, \gamma, \beta, q\rangle,$$

где $w_1, w_2, v \in \Sigma^*$; $\alpha, \gamma, \beta \in \Sigma$; $l \in \{0, 1, 2, \dots\}$; $q \in Q$.

Содержательно, конфигурация описывает "мгновенное состояние" M на каждом такте. Здесь w_1 — часть слова на входной ленте слева от головки, w_2 — часть слова на входной ленте справа от головки (без букв $\wedge$ в конце), α — обозреваемая буква на входной ленте, γ — просматриваемая на такте буква на рабочей ленте, β — записываемая на такте на рабочую ленту буква, v — слово на рабочей ленте до выполнения такта (без "пустых" клеток в конце), q — состояние машины до выполнения такта.

Число l характеризует положение головки на рабочей ленте по отношению к той клетке, которая обозревалась в начальном состоянии.

Начальная конфигурация имеет вид:

$$\langle \lambda, \#, w, \#, 0, \#, \beta_0, q_1\rangle,$$

где λ — пустое слово,

$$\pi(q_1, \#, \#) = \langle q', \beta_0, \mu_1, \mu_2, c\rangle.$$

Фактически работу M на w можно описать последовательностью однозначно определяемых конфигураций.

Дадим теперь определение точки. Точка вычисления M на w — это четвёрка

$$\langle l, \gamma, \beta, q\rangle,$$

где $\gamma, \beta \in \Sigma$; $l \in \{0, 1, 2, \dots\}$; $q \in Q$, которая является подкортежом конфигурации с теми же значениями элементов.

То есть γ — просматриваемая на такте буква на рабочей ленте, буква β записывается на такте на рабочую ленту, число l характеризует положение головки на рабочей ленте по отношению к той клетке, которая обозревалась в начальном состоянии, q — состояние машины до выполнения такта.

Рассмотрим последовательность конфигураций C_1, $C_2, \dots, C_n$, порождаемую при работе M на слове w, где C_1 — начальная конфигурация, и полученную проецированием этой последовательности конфигураций соответствующую последовательность точек вычисления P_1, $P_2, \dots, P_n$.

Если C_n — заключительная конфигурация, то есть состояние в ней является заключительным, то следующий элемент в этой последовательности конфигураций породить нельзя.

В противном случае можно породить C_{n+1} и, значит, P_{n+1}, получаемый его проекцией.

Зная машину M и C_n, можно явно выписать C_{n+1} и P_{n+1}.

Зададимся вопросом, какой информации нехватает в последовательности P_1, $P_2, \ldots, P_n$ для выписывания P_{n+1}?

Естественно, мы должны знать, какой символ встречается на входной ленте, чтобы вычислить π для конфигурации C_n (мы из P_n знаем символ, обозреваемый на рабочей ленте, и состояние машины). Зная этот символ, мы вычислим π и определим l' и q', где

$$P_{n+1} = \langle l', \gamma', \beta', q' \rangle.$$

Остается определить γ', β'. Если, согласно вычисленному в конфигурации C_n значению n, головка на рабочей ленте остается неподвижной, то γ' совпадает с третьим элементом P_n. Однако, если головка на рабочей ленте двигается вправо или влево, то мы должны, для определения γ', знать, какой символ находится в соответствующей клетке.

Выясняется, что эта последняя информация уже содержится в последовательности P_1, $P_2, \ldots, P_{n-1}$. Чтобы установить это, найдем такое максимальное m, что $1 \leq m \leq n-1$ и P_m имеет вид

$$\langle l', \gamma'', \beta'', q'' \rangle.$$

Ясно, что если такое m есть, то $\gamma' = \beta''$. Если же такого m не сыщется, то очевидно, что $\gamma' = \wedge$.

Теперь, зная обозреваемый в конфигурации C_{n+1} символ на входной ленте, состояние q' и символ γ', мы в состоянии вычислить вновь функцию π и определить тем самым символ β'.

Это рассуждение является ключевым в доказательстве теоремы 1.

Рассмотрим процесс порождения последовательности

$$P_1, P_2, P_3, \ldots,$$

при котором мы, порождая элементы последовательности, по одному погружаем их в стек. При этом мы "позволяем" себе смотреть, что делается на входной ленте машины, но "закрываем" от себя рабочую. Сначала в этом процессе мы запишем в вершину стека P_1. Далее будем порождать P_2, $P_3, \ldots$ и записывать их в стек до тех пор, пока головка M на рабочей ленте стоит на месте. Тогда для порождения каждого элемента P_i нам нужен только элемент P_{i-1}, в силу приведенного выше рассуждения.

Предположим, что в какой-то момент времени головка M на рабочей ленте сдвигается. Допустим, что к этому моменту мы породили уже последовательность точек вычисления P_1, $P_2, \ldots, P_m$ и записали их в стек.

Чтобы породить P_{m+1}, мы должны знать, какая буква $\gamma^{(m+1)}$ записана на рабочей ленте M в клетке, на которую M переходит. При этом, если

$$P_i = \langle l^{(i)}, \gamma^{(i)}, \beta^{(i)}, q^{(i)} \rangle$$

для $i = 1, 2, \ldots, m+1$, то $l^{(m+1)}$ легко вычисляется по $l^{(m)}$ и значению функции π, а $q^{(m+1)}$ содержится в значении π.

Легко видеть, что $\gamma^{(m+1)}$ может быть извлечена из последовательности P_1, $P_2, \ldots, P_m$, а именно, $\gamma^{(m+1)} = \beta^{(j)}$, где

$$j = \max\ _{1 \leq k \leq m}\{k \mid l^{(k)} = l^{(m+1)}\},$$

если такое j существует, и $\gamma^{(m+1)} = \wedge$ в остальных случаях. Теперь $\beta^{(m+1)}$ может быть вычислена при помощи функции π. Однако мы не можем для порождения P_{m+1} взять искомый P_j прямо из стека, так как при этом, по правилам работы со стеком, мы потеряем в стеке все элементы P_m, $P_{m-1}, \ldots, P_j$, а мы хотим иметь в стеке элементы P_1, $P_2, \ldots, P_m$, P_{m+1} после выполнения данного шага.

Чтобы выйти из данной ситуации, мы вначале вновь породим в стеке все элементы P_1, $P_2, \ldots, P_m$, а потом уже будем их уничтожать по одному и в процессе этого получим $\gamma^{(m+1)} = \beta^{(j)}$ (или же исчерпаем их все и, убедившись, что подходящего элемента нет, положим $\gamma^{(m+1)} = \wedge$). Итак, стек сначала будет преобразован к виду

$$P_1, P_2, \ldots, P_{m-1}, P_m, P_1, P_2, \ldots, P_{m-1}, P_m,$$

а потом уже к искомому виду

$$P_1, P_2, \ldots, P_{m-1}, P_m, P_{m+1}.$$

Этот процесс мы будем называть *реконструкцией.*

При этом мы будем говорить, что

$$P_1, P_2, \ldots, P_{m-1}, P_m$$

получены без реконструкций, а P_{m+1} — с одной реконструкцией.

Продолжив процесс порождения последовательности

$$P_1, P_2, \dots, P_{m-1}, P_m, P_{m+1}, \dots$$

дальше, мы, через некоторое время, вновь столкнемся с ситуацией, когда для получения некоторого P_{m+k}, где $k > 1$, нам нужен будет элемент P_{j_1}, для некоторого $1 \le j_1 \le m + k - 1$. Для получения этого элемента мы вновь преобразуем стек сначала к виду

$$P_1, P_2, \dots, P_{m+k-1}, P_1, P_2, \dots, P_{m+k-1},$$

а потом к виду

$$P_1, P_2, \dots, P_{m+k-1}, P_{m+k}.$$

При этом P_{m+k} будет получен с двумя реконструкциями: одна для получения P_m, а другая — для получения P_{m+k}.

И так далее ...

Неформально процесс порождения P_1, $P_2, \dots$ и записи их в стек описан.

Опишем процесс порождения более формально.

Четверку $\langle z, u, P, v\rangle$ будем называть *маркированной точкой*, если $u \in \{0, 1, \dots\}$, $v \in \{0, 1, 2, \dots, |w| + 1\}$, $z \in \{\infty, 0, 1, \dots\}$, P — точка вычисления M. Содержательно, u будет обозначать число реконструкций, с которым получена P, z — максимально допустимое число реконструкций (точный смысл поясним позже), а v — положение головки на входной ленте.

Рассмотрим абстрактную машину $\mathfrak{A}$, моделирующую работу M описанным выше неформально способом.

$\mathfrak{A}$ снабжена входной лентой, аналогичной ленте M, с головкой на ней, и стеком для запоминания маркированных точек для машины M. Дополнительно $\mathfrak{A}$ имеет память, достаточную для запоминания двух маркированных точек и функции π машины M.

Работа абстрактной машины проходит по шагам. Если следующий шаг явно не указан, имеется в виду следующий по порядку шаг.

Параллельно (в рамочках) мы приводим набросок куска программы s-СМТ, моделирующего работу этого шага абстрактной машины. При этом моделирование шага i, после которого осуществляется переход на шаг j, начинается в состоянии q_i и заканчивается переходом в состояние q_j.

Мы используем первые $3c + 2$ рабочие ленты для записи текущей маркированной точки и следующие $3c + 2$ рабочие ленты для записи вынимаемой из стека маркированной точки. Первые c лент из них используются для записи максимально допустимого числа реконструкций, следующие c лент — для записи числа реконструкций, с которым получена рассматриваемая маркированная точка, следующие c лент — для записи положения головки на рабочей ленте машины Тьюринга, следующая лента — для записи обозреваемого на рабочей ленте машины Тьюринга символа, для записи символа, который будет записан в обозреваемую ячейку, и для записи номера (в виде последовательности нулей) состояния машины Тьюринга, следующая лента — для записи положения головки на входной ленте машины Тьюринга. При этом, если обозреваемый (значит, и печатающийся) символ есть $\#$, вместо $\#$ пишем $\times$.

Мы записываем числа в двоичной системе счисления, но начинаем с младших разрядов. При этом сначала используем первые (после начальных) ячейки всех предназначенных для записи числа лент, потом — вторые и так далее. Мы не приводим программ для увеличения или уменьшения числа на 1.

В командах s-СМТ договариваемся не писать нового состояния, если состояние не меняется, не писать записываемых символов, если они все не меняются, и не писать слово, заталкиваемое в магазин, если оно совпадает с верхушкой магазина.

Описание работы абстрактной машины.

Рассмотрим работу $\mathfrak{A}$ на входном слове w.

Шаг 1

$\mathfrak{A}$ делает маркированную точку

$$\langle \infty, 0, \langle 0, \#, \beta_0, q_1 \rangle, 0 \rangle$$

текущей (то есть формирует такую маркированную точку и запоминает ее в месте в памяти, предназначенном для текущей точки). Здесь

$$\pi(q_1, \#, \#) = \langle q'', \beta_0, \mu_1'', \mu_2'', c \rangle.$$

$$q_1\# \ldots \#\dagger \Rightarrow CRC^{c-1}RC^{c-1}RC \ldots C;$$

$$q_1\# \wedge \#^{c-1} \wedge \#^{c-1} \wedge \# \ldots \#\dagger \Rightarrow$$

$$q_{12}\infty\#^{c-1}0\#^{c-1}0\# \ldots \#CLC^{c-1}LC^{c-1}LC^{c-1}RC \ldots C;$$

$$q_{12}\#^{3c+1} \wedge \# \ldots \#\dagger \Rightarrow q_{13}\#^{3c} \times \# \ldots \#C^{3c+1}RC \ldots C;$$

$$q_{13}\#^{3c+1} \wedge \# \ldots \#\dagger \Rightarrow q_{14}\#^{3c} \times \# \ldots \#C^{3c+1}RC \ldots C;$$

$$q_{14}\#^{3c+1} \wedge \# \ldots \#\dagger \Rightarrow q_{15}\#^{3c}0\# \ldots \#C^{3c+1}LC \ldots C;$$

$$q_{15}\#^{3c+1} \times \# \ldots \#\dagger \Rightarrow q_{16}C^{3c+1}LC \ldots C;$$

$$q_{16}\#^{3c+1} \times \# \ldots \#\dagger \Rightarrow q_{17}C^{3c+1}LRC \ldots C;$$

$$q_{17}\#^{3c+2} \wedge \# \ldots \#\dagger \Rightarrow q_2\#^{3c+1}0\# \ldots \#C^{3c+2}LC \ldots C$$

Шаг 2

Пусть

$$\langle i, j, \langle l, b, d, q\rangle, v\rangle$$

есть текущая маркированная точка. Запишем ее в стек.

Пусть a — v-я буква на входной ленте (в частности, если $v = 0$, то a есть #, а если $v = |w| + 1$, то $a = \wedge$). Вычислим

$$\pi(q, a, b) = \langle q', b, \mu_1, \mu_2, c\rangle.$$

Если q' есть !, перейдём на шаг 10, если c=0, и на шаг 11, если c=1.

Записываем текущую маркированную точку в стек, а во время записи запоминаем в состоянии обозреваемый машиной Тьюринга символ на рабочей ленте и состояние машины Тьюринга, записываем

$$\pi(q, a, b) = \langle q', b, \mu_1, \mu_2, c\rangle$$

на $6c + 5$-ую рабочую ленту и переходим в состояние q_3, q_{10} или q_{11}.
Для вычисления состояния и $\pi(q, a, b)$ надо знать обозреваемый символ на входной ленте машины Тьюринга. Для его вычисления делаем v шагов на входной ленте. В зависимости от обозреваемого после этого символа на входной ленте вычисляем состояние. Перед переходом в состояние, соответствующее следующему шагу, возвращаемся в начало входной и всех рабочих лент.

Шаг 3

Теперь наша цель — вычислить следующую текущую маркированную точку.

Вычислим v'. v' определяется в зависимости от v и от μ_2 :

если $\mu_2 = C$, то $v' = v$;

если $\mu_2 = L$, то $v > 0$ и $v' = v - 1$; если $\mu_2 = R$, то $v < |w| + 1$ и $v' = v + 1$.

Шаг 4

Чтобы завершить вычисление очередной текущей точки, нам теперь осталось только определить новые координаты головки на рабочей ленте и букву, записанную под этими координатами.

Если $\mu_1 = L$ или $\mu_1 = R$, то перейдем к выполнению шага 6, в противном случае — к выполнению шага 5.

Шаг 5

Соответствует тому простому случаю, когда $\mu_1 = C$, то есть головка на рабочей ленте не сдвинулась.

Положим в качестве текущей маркированную точку

$$\langle i, j, \langle l, d, d', q' \rangle, v' \rangle,$$

где

$$\pi(q', a', d) = \langle q'', d', \mu_1'', \mu_2'', c'' \rangle,$$

a' — v'-я буква слова w, и перейдем к шагу 2.

Шаг 6

Соответствует случаю, когда головка на рабочей ленте сдвинулась.

Положим

$$l' = \begin{cases} l-1, \text{ если } \mu_1 = L; \\ l+1, \text{ если } \mu_1 = R. \end{cases}$$

Таким образом, мы вычислили координаты новой клетки на рабочей ленте. Осталось найти, какая буква записана в клетке с этими координатами.

Если $i \neq j$, то перейдем к шагу 9 (где начинается очередная реконструкция).

Иначе будем вынимать из стека маркированные точки до тех пор, пока не окажется вынутой

а) либо маркированная точка вида

$$\langle i', j, \langle l'', b'', d'', q'' \rangle, v'' \rangle$$

для некоторых $i' > i\ (= j)$, l'', b'', d'', q'', v'';

б) либо маркированная точка вида

$$\langle i, j', \langle l', b'', d'', q'' \rangle, v'' \rangle$$

для некоторых j', b'', d'', q'', v''.

а) соответствует случаю, когда при реконструкции не удалось найти нужной клетки на рабочей ленте. В случае а) идем на шаг 7.

б) соответствует случаю, когда при реконструкции удалось найти нужную клетку на рабочей ленте. В случае б) идем на шаг 8.

Шаг 7

Соответствует случаю, когда при реконструкции не удалось найти нужной клетки на рабочей ленте. Тогда искомая буква есть $\wedge$.

Итак, последней мы извлекли маркированную точку

$$\langle i', j, \langle l'', b'', d'', q'' \rangle, v'' \rangle.$$

Из построения видно, что это та самая точка, которую мы загружали в стек при выполнении шага 9 в начале реконструкции, и что $b'' = d'' = ?$.

Сделаем текущей маркированную точку

$$\langle i', j+1, \langle l'', \wedge, d''', q'' \rangle, v'' \rangle,$$

где

$$\pi(q'', a'', \wedge) = \langle q''', d''', \mu_1''', \mu_2''', c''' \rangle,$$

a'' — v''-я буква слова w, и перейдем к шагу 2.

Шаг 8

Соответствует случаю, когда при реконструкции удалось найти нужную клетку на рабочей ленте. Запомним d'' и будем вынимать из стека маркированные точки до тех пор, пока не вынем маркированную точку

$$\langle i', j, \langle l''', b''', d''', q''' \rangle, v''' \rangle$$

для некоторых $i' > i$, l''', b''', d''', q''', v'''.

Из построения видно, что это та самая точка, которую мы загружали в стек при выполнении шага 9 в начале реконструкции.

Сделаем текущей маркированную точку

$$\langle i', j+1, \langle l''', d'', d_*, q''' \rangle, v''' \rangle,$$

где

$$\pi(q''', a''', d'') = \langle q_*, d_*, \mu_1'''', \mu_2'''', c_* \rangle,$$

a''' — v'''-я буква слова w, и перейдем к шагу 2.

Шаг 9

Отметим, что, раз мы пришли к данному шагу, то $j < i$.

Запишем в стек маркированную точку

$$\langle i, j, \langle l', ?, ?, q' \rangle, v' \rangle.$$

Фактически это была бы следующая текущая точка, если бы мы могли поставить нужные буквы вместо ?.

Сделаем текущей маркированную точку

$$\langle j, 0, \langle 0, \#, \beta_0, q_1 \rangle, 0 \rangle.$$

Здесь

$$\pi(q_1, \#, \#) = \langle q'', \beta_0, \mu_1'', \mu_2'', c \rangle.$$

Перейдём к шагу 2.

Шаг 10

Остановиться и выдать 0.

Шаг 11

Остановиться и выдать 1. ∎

Приведенное выше описание $\mathfrak{A}$ формально описывает процесс порождения точек вычисления P_1, P_2, $P_3, \ldots$

При этом $\mathfrak{A}$ воспринимает входное слово w тогда и только тогда, когда $w \in L_M$.

Для завершения доказательства теоремы 1 достаточно только показать, что описанная выше машина $\mathfrak{A}$ может быть промоделирована на подходящей s-СМТ M'.

∎

Теорема 2 (Кука). *Пусть s — некоторая арифметическая функция, асимптотически мажорирующая функцию $log(n)$ (то есть $log(n) \ll s$). Пусть Σ — алфавит, содержащий буквы 0, 1, $\wedge$ и, возможно, другие, $L \subseteq (\Sigma \setminus \{\wedge\})^*$, D — такая s-NCMT, что $L_D = L$.*

Тогда существует такой 1-МТ распознаватель M, что $L = L_M$ и $t_M \ll 2^{cs}$ для некоторой константы c.

Доказательство. Конфигурацией s-k-NCMT

$$D = \langle \Sigma, Q, \pi, q_1 \rangle$$

на w будем называть кортеж

$$\langle m, v_1, l_1, \ldots, v_k, l_k, q, \alpha \rangle,$$

где $v_1, \ldots, v_k \in \Sigma^*$; $\alpha \in \Sigma$; $m, l_1, \ldots, l_k \in \{0, 1, 2, \ldots\}$; $q \in Q$.

При этом $m, l_1, \ldots, l_k$ — это позиции головок на входной и рабочих лентах; $v_1, \ldots, v_k$ — это начала рабочих лент до последней непустой клетки включительно, если выбросить крайне правые клетки; α — это верхушка магазина; а q — это состояние. Понятно, что работа s-k-NCMT D на входе w полностью описывается последовательностью конфигураций и последовательностью магазинов.

Определение 2. *Пара конфигураций называется допустимой, если существуют такие магазины одинаковой длины с указанными верхушками, что начиная работу с первой конфигурации и первого магазина, D может через конечное число тактов (возможно, не делая ни одного такта работы) попасть во вторую конфигурацию, имея при этом второй магазин, так, что никакая ячейка первого магазина, лежащая ниже верхушки, не оказывается верхушкой в процессе этой работы.*

Без ограничения общности можно считать, что в каждом правиле s-k-NCMT D слово, помещаемое в магазин, имеет длину, не превышающую 2. Это означает, что следующий магазин или имеет ту же длину, или его длина увеличивается на 1, или его длина уменьшается на 1.

Лемма 3. *Каждая допустимая пара конфигураций получается из допустимых пар вида (P, P) с помощью конечного числа следующих операций:*

(1) из пар (P_1, Q_1) и (P_2, Q_2) получается пара (P_3, Q_3), если $Q_1 = P_2$, $P_3 = P_1$ и Q_3 получается из Q_2 при помощи правила, которое не меняет длину магазина;

(2) из пар (P_1, Q_1) и (P_2, Q_2) получается пара (P_3, Q_3), если $P_3 = P_1$, P_2 получается из Q_1 при помощи правила, которое увеличивает длину магазина на 1, а Q_3 получается из Q_2 при помощи правила, которое уменьшает длину магазина на 1.

Доказательство. Индукцией по числу тактов возможной работы от P до Q. Если можно не делать тактов вообще, то пара уже получена. Если можно сделать $i+1$ такт, то за i тактов из P получается Q'. Если при этом длина магазина через i тактов не меняется, то (P, Q') получается указанными операциями, а (P, Q) получается из (P, P) и (P, Q') операцией (1). Если же через i тактов длина магазина увеличивается на 1, то найдём последний такой такт перед тактом i, на котором длина магазина увеличивается на 1. Пусть это будет такт j. Пусть на такте j имеем конфигурацию Q'', а на такте $j+1$ — конфигурацию Q'''.

Ясно, что пары (Q''', Q') и (P, Q'') допустимы. Но пара (P, Q) получается из пар (Q''', Q') и (P, Q'') при помощи операции (2). ∎

Теперь последовательно выписываем на рабочей ленте машины Тьюринга длины, начиная с 1, а на следующей рабочей ленте — сначала пары одинаковых конфигураций с рабочими лентами указанной длины, а потом добавляем без повторений пары, которые получаются из выписанных с помощью операций (1) и (2). Когда новых пар уже получить нельзя, на первой рабочей ленте увеличиваем длину на 1. Действуем так до тех пор, пока не получим пару из начальной и заключительной конфигураций. ∎