Да припомним, че въведохме понятието 1-управляващ граматика, чрез което може да се представят функции $f : \Sigma_0^* \to \Sigma_1^*$. С този апарат досега успяхме да покажем, че функциите, които се представят с 1-управляващи граматики имат следните свойства:

1. затворени са относно композиция, IF-THEN-ELSE и WHILE-DO-DONE, където условията са по първата буква на думата.

2. функцията +1 (или SUCC) е представима.

3. $\textsc{Fetch}_h$ е 1-представима за всяка функция $h : \Sigma \to \mathcal{P}(\Sigma)$, където $\textsc{Fetch}_h(w)$, в зависимост от първата буква $w[1]$ на w, намира най-малкото $i > 1$, за което $w[i] \in h(w[1])$ и в този случай $\textsc{Fetch}_h(w) = w[i]w[2..n]$, а ако такъв символ няма $\textsc{Fetch}_h(w) = \bot w$.

4. $\textsc{Apply}_\rho$ е 1-представима за всяка функция $\rho : \Sigma \to (\Sigma^* \times \Sigma^*)\{(\varepsilon,\varepsilon),(\#,\#)\}$, където $\textsc{Apply}_\rho(w)$ прилага правилото $\rho(w[1])$ върху най-лявото срещане на лявата страна на това правило в $w[2..n]\#$, ако такова и резултатът тогава е $\top w'$ и връща $\bot w$, ако такова срещане няма.

Забележка 0.1. Да забележим, че ако имаме няколко $\rho_i : \Sigma \to (\Sigma^* \times \Sigma^*)\{(\varepsilon,\varepsilon),(\#,\#)\}$ за $i \leq k$, където k е фиксирано, то използвайки IF-THEN-ELSE, може да композираме $\textsc{Apply}_{\rho_i}$ по следния начин:

$w_1 \leftarrow \texttt{APPLY}_{\rho_1}(w)$
`if` $w_1[1] = \top$ `then return` w_1
`else` $w_1' \leftarrow w_1[2..n]$`;` $w_2 \leftarrow \texttt{APPLY}_{\rho_2}(w_1')$
....
`if` $w_i[1] = \top$ `then return` w_i
`else` $w_i' \leftarrow w_i[2..n]$`;` $w_{i+1} \leftarrow \texttt{APPLY}_{\rho_i}(w_i')$
....
`if` $w_{k-1}[1] = \top$ `then return` w_{k-1}
`else` $w_{k-1}' \leftarrow w_{k-1}[2..n]$`;` $w_k \leftarrow \texttt{APPLY}_{\rho_i}(w_{k-1}')$
`return` w_k

Тогава резултатът ще бъде първото $\textsc{Apply}_{\rho_i}(w)$, където $\rho_i(w[1])$ е първото правило, което може да бъде успешно приложено към w, ако такова има, и $\bot w$, ако такова няма. Това следва непосредствено като забележим, че $w_i' = w_{i-1}$, ако w_i' е дефинирано. Оттук по индукция $w_i' = w$, ако е дефинирано.

Ще означаваме горната конструкция с $\textsc{ApplySet}_\rho$, като за $\rho(w[1])$ ще си мислим като за краен брой от правила, които са подредени по някакъв (фиксиран) начин.

Дефиниция 0.1. *Казваме, че 1-управляваща граматика $G = \langle \Sigma, \mathcal{N}, P, S, F, \# \rangle$ е* еднозначна*, ако едновременно:*

1. *всяко правило $\alpha \to \beta \in P$ има свойството, че $\alpha \in \Sigma^*\mathcal{N} \cup \mathcal{N}\Sigma^*\{\varepsilon, \#\}$.*

2. *за всяка дума $w \in \Sigma^*\mathcal{N}\Sigma^*\#$ има най-много едно правило $\alpha \to \beta \in P$, за което $w = x\alpha y$.*

Смисълът на тази дефиниция е, че всеки път когато има извод от $Sw\# \Rightarrow_G^* Fv\#$ той е единствен и на всяка конкретна стъпка от този извод има най-много едно действие, което може да бъде извършено.

Забележка 0.2. Тъй като при конструкциите за композиция, IF-THEN-ELSE, WHILE-DO-DONE, +1, APPLY и FETCH добавените правила удовлетворяват условията за еднозначност, то тези операции запазват еднозначните 1-управляващи граматики. Оттук следва, че и APPLYSET също запазва еднозначните 1-управляващи граматики.

Теорема 0.1. *Нека $G = \langle \Sigma, \mathcal{N}, P, S, F, \# \rangle$ е 1-управляваща граматика, която представя функция $f : \Sigma_0^* \to \Sigma_1^*$. Тогава има еднозначна 1-управляваща граматика $G = \langle \Sigma', \mathcal{N}', P', S', F,' \#' \rangle$, която:*

1. *представя f.*

2. *има константа $c = c(G)$, за която за всяка дума $w \in \Sigma_0^*$ и естествено число n е вярно, че:*

$$\text{ако } Sw\# \Rightarrow_G^{(n)} Ff(w)\#, \text{ то } S'w\#' \Rightarrow^{\leq c^{n+|w|}} F'f(w)\#.$$

Казано с думи, еднозначността на 1-управляващите граматики не е съществено ограничение, но цената, която плащаме е експоненциално удължаване на най-късите изводи.

Доказателство. Доказателството ще използва забележка 0.2, в частност няма да конструираме експлицитно граматики, а ще работим с тях само на посредством тези конструкции.

Ще опишем как систематично да генерираме редица от правила в граматиката G, които да проверяваме дали определят успешно изпълнение върху входа w. Това ще правим до намирането на първата такава редица. Ако редицата от правила не определя успешно изпълнение, възстановяваме началната конфигурация, и генерираме следваща редица от правила.

Ето как това може да бъде описано с помощта на горните операции. Ще означаваме с $p_i \in P$ правилата в P и ще гледаме на тях като на букви. Някои от тези букви ще може да бъдат маркирани, което ще бъде друг символ $(p_i, \boxdot) \in P \times \{\boxdot\}$. Неформално това ще бъде актуалното правило, което трябва да бъде приложено.

1. При вход w, преобразуваме w до $w_1 = \#_1\#w$. Граматика с единствено правило $S_1 \to F_1\#_1\#$ представя тази функция.

   Сегментът до $\#_1$ ще представлява редица от правила. Сегментът между $\#_1$ и $\#$ ще бъде мястото, където ще се симулират тези правила.

2. Докато $w_1[1] \neq \top$ правим следното. (тогава предполагаме, че $w_1 = p_1 \dots p_n\#_1\#w$)

   (а) Копираме w в сегмента между $\#_1$ и $\#$, вж. задача 6 от Машини на Тюрниг, получаваме:

   $$w_2 = p_1 \dots p_n\#_1 w\#w.$$

   (б) Прилагаме правилото $\#_1 \to \#_1 S$, (Apply). Получаваме:

   $$w_3 = p_1 \dots p_n\#_1 Sw\#w.$$

   (в) Маркираме първото правило, може и като се използва Apply, а може и директно:

   $$S_3 p \to F_3(p, \boxdot) \text{ и } S_3\#_1 \to \#_1.$$

   В резултат:

   $$w_4 = (p_1, \boxdot)p_2 \dots p_n\#_1 Sw\#w \text{ или } w_4 = \#_1 Sw\#w \text{ при } n = 0.$$

   Върху $u_1 = w_4$ започваме да симулираме редицата $p_1 p_2 \dots p_n$ върху сегмента $Sw\#$. Нека $v_0 = Sw$ Ще поддържаме инварианта:

   $$u_i = p_1 \dots p_{i-1}(p_i, \boxdot)p_{i+1} \dots p_n\#_1 v_i\#w \text{ и } v_{j-1}\# \overset{p_j}{\Rightarrow}_G v_j\# \text{ за } j < i.$$

   Съответно при $i = n + 1$:

   $$u_i = p_1 \dots p_n\#_1 v_n\#w \text{ и } v_{j-1} \overset{p_j}{\Rightarrow}_G v_j \text{ за } j < n + 1.$$

   (г) Намираме първото маркирано правило $u'_i = \text{Fetch}_{P\times\{\boxdot\}}(u_i)$. Получаваме:

   $$u'_i = \begin{cases} (p_i, \boxdot)p_1 p_2 \dots p_n\#_1 v_i\#w \text{ за } i < n + 1 \\ \bot p_1 \dots p_n\#_1 v_n\#w. \end{cases}$$

   (д) Докато $u'_i[1] \neq \bot$, прилагаме p_i към $u'_i[2..N]$, (това е Apply_ρ, за $\rho(p_i, \boxdot) = p_i)$). Получаваме:

   $$u''_i = \begin{cases} \top p_1 \dots (p_i, \boxdot), \dots p_n\#_1 v_{i+1}\#w \text{ ако } p_i \text{ може да се приложи към } v_i \\ \bot(p_i, \boxdot)p_1 p_2 \dots p_n\#_1 v_i\#w = \bot(p_i, \boxdot)u_i \text{ иначе.} \end{cases}$$

   (е) Ако $u''_i[1] = \top$, прилагаме множеството от правила $M = \{(p, \boxdot)p' \to p(p', \boxdot) \,|\, p, p' \in P\} \cup \{(p, \boxdot)\#_1 \to p\#_1\}$ към u''. (това е ApplySet_ρ с $\rho(\top) = M$). Иначе, ако $u''_i[1] = \bot$, изтриваме $u''_i[2]$ и махаме маркирания символ като прилагаме $\{(p, \boxdot) \to p\}$. В резултат получаваме:

   $$u'''_i = \begin{cases} \top u_{i+1} = \top p_1 \dots p_i(p_{i+1}, \boxdot) \dots p_n\#_1 v_{i+1}\#w \text{ ако } u''_i[1] = \top \\ \bot p_1 \dots p_n\#_1 v_i\#w \text{ иначе.} \end{cases}$$

   (ж) За да затворим цикъла по i, ако $u'''_i[1] = \top$, Fetch_h с $h(\top) = P \times \{\boxdot\}$. Получаваме:

   $$u'_{i+1} = \begin{cases} (p_i, \boxdot)p_1 p_2 \dots p_n\#_1 v_i\#w \text{ за } i < n + 1 \text{ и } u'''_i = \top u_{i+1} \\ \bot p_1 \dots p_n\#_1 v_n\#w \text{ за } i = n + 1 \text{ и } u'''_i = \top u_{i+1} \\ \bot p_1 \dots p_n\#_1 v_i\#w \text{ за } i < n + 1 \text{ и } u'''_i = \bot u_i. \end{cases}$$

   (з) Когато $u'_i[1] = \bot$ проверяваме дали v_i започва с F, тоест прилагаме $\#_1 F \to \#_1 F$. Ако това прилагане е успешно, изтриваме сегмента $\#w$ и $p_1 \dots p_n\#_1$ и приключваме.

(и) Когато $u'_i[1] = \perp$ проверяваме дали v_i започва с F, тоест прилагаме $\#_1 F \to \#_1 F$. Ако това прилагане не е успешно, изтриваме сегмента между $\#_1$ и $\#$, тоест v_i и увеличаваме с 1 сегмента $p_1 \dots p_n \#_1$ като число в $|P|$-ична бройна система.

Да забележим, че едно правило в G може да увеличи дължината v_i най-много с фиксиран брой символи $c' = c'(G)$. Поради това дължината на $|v_i| \leq ic' + |w| + 1$. Оттук следва, че $|u_i| \leq n + ic' + |w| + 1 \leq (c'+1)(n+|w|)$. Тъй като итерацията по прилагане на едно конкретно правило изисква $\leq c''|u_i|$ прехода, то получаваме, че прилагането на правилото p_i към v_i отнема по-малко от $c''(c'+1)(n+|w|)$ стъпки. Следователно симулацията на всички правила $p_1 p_2 \dots p_n$ изисква не повече от $\widetilde{c}(n+|w|)^2$ стъпки.

Сега да допуснем, че в G има извод $Sw\# \Rightarrow_G^{(n)} Ff(w)\#$. Тогава той се определя от редица от правила $p_1 \dots p_n$ в $|P|$. При симулацията в G' преди симулирането на тази редица ще бъдат симулирани не повече от всички редици с дължина по-малка или равна на n. Тоест дължината на извода в G' на $S'w\#' \Rightarrow_{G'}^{(m)} F'f(w)\#'$ ще бъде:

$$m \leq \sum_{k=0}^{n} \sum_{p_1 \dots p_k \in P^k} \widetilde{c}(k+|w|)^2 \leq \sum_{k=0}^{n} |P|^k \widetilde{c}(n+|w|)^2 \leq \widetilde{c}(n+|w|)^2 |P|^{n+1} \leq c^{n+|w|}$$

за подходяща константа $c = c(G)$, която зависи от P и $\widetilde{c}$. $\square$

Теорема 0.2. *За всяко $k \geq 1$ и k-лентова машина на Тюринг $M = \langle \Sigma, \Gamma, Q, s, \Delta, F, \_ \rangle$ има 1-управляваща граматика $G_M = \langle \Sigma', \mathcal{N}, P, S_M, F_M, \# \rangle$, която представя езика $\mathcal{L}(M) \subseteq \Sigma^*$. Нещо повече:*

1. *има константа $c = c(M)$, за която за всяка дума w и всяко $n \in \mathbb{N}$:*

$$\text{ако има } f \in F : (s, (w\_, \_, \dots, \_), (1, 1, \dots, 1)) \Rightarrow_M^{(n)} (f, W', J'), \text{ то } S_M w\# \Rightarrow^{\leq c(1+n+|w|)^2} F_M v\#.$$

2. *ако M е детерминирана, то G_M е еднозначна.*

Доказателство. Идеята отново е да симулираме изпълнението на машината M чрез конструкциите за 1-управляващи граматики. За да постигнем това, ще осигурим, че всяка конфигурация $\kappa = (p, W, J)$ на M съответства на дума в граматиката от вида:

$$w_\kappa = p\#_k W'_k \#_{k-1} W'_{k-1} \#_{k-2} \dots \#_1 W'_1, \text{ където } W'_i = W_i[1..J_i - 1](W[J_i], i) W_i[J_i + 1..|W_i|],$$

тоест J_i-тият символ в W'_i е маркиран. Този маркер показва, къде е главата на i-та лента. Преминаването от едно представяне на конфигурация κ към следващо, което представя κ' със свойството $\kappa \Rightarrow_M \kappa'$ е свързано със следните стъпки:

1. Намираме маркираните символи $\overline{a} = (a_1, \dots, a_k)$ за всяко $i \leq k$.
2. Върху така намерените символи заедно със състоянието p прилагаме преход(а) $\tau = \big\langle (p, \overline{a}), (q, \overline{b}) \big\rangle \in \Delta$, който определя $\kappa \overset{\tau}{\Rightarrow}_M \kappa'$.
3. След това за всяко $i \leq k$, ако $b_i \in \Gamma$, заместваме маркирания символ (a_i, i) с (b_i, i), а ако $b_i \in \{\leftarrow, \rightarrow\}$ преместваме i-тия маркер в съответната посока.

Сега ще опишем този процес в термините на операциите върху 1-управляващи граматики.

1. (Инициализация 1) При вход w преобразуваме w до $w_1 = s\#_k \#_{k-1} \#_{k-2} \dots \#_1 w$. Това съответства на граматика с единствено правило $S_1 \to F_1 s\#_k \#_{k-1} \#_{k-2} \dots \#_1$.
2. (Инициализация 2) Добавяме $(\_, i+1)$ пред всеки $\#_i$ включително. Това правим като с $\textsc{Apply}_{\#_i \to (i+1, \#_i)}$. Получаваме:
$$w_2 = s\#_k(\_, k)\#_{k-1}(\_, k-1)\#_{k-2} \dots (\_, 2)\#_1 w.$$
3. (Инициализация 3) Поставяме $\_$ в края на w_2 и маркираме първият символ на $w\_$. Това правим с композиция на $\textsc{Apply}_{\# \to \_\#}$ и $\textsc{ApplySet}_R$, където $R = \{\#_1 b \to \#_1(b, 1) \,|\, b \in \Gamma\}$. Получаваме:
$$w_3 = \begin{cases} s\#_k(\_, k)\#_{k-1}(\_, k-1)\#_{k-2} \dots (\_, 2)\#_1(a, 1)w'\_, & \text{ако } w = aw' \text{ е непразна} \\ s\#_k(\_, k)\#_{k-1}(\_, k-1)\#_{k-2} \dots (\_, 2)\#_1(\_, 1), & \text{ако } w = \varepsilon. \end{cases}$$

   Дотук w_3 е представянето на началната конфигурация $(s, (w\_, \_, \dots, \_), (1, 1, \dots, 1))$ на M при вход w. Оттук нататък преобразуваме една конфигурация в следваща, използвайки преходите на M докато конфигурацията не стане финална. Нека $u_0 = w_3$ и $j = 0$.

   Ще поддържаме инварианта, че u_j е представяне на конфигурация $\kappa_j = (q_j, W^{(j)}, J^{(j)})$ и съответно за $j > 0$, $\kappa_{j-1} \Rightarrow_M \kappa_j$.

4. Докато $u_j[1] \notin F \cup \{\perp\}$ правим следното.

   (а) Композираме $\textsc{Fetch}_{\Gamma\times i}$ за $1 \leq i \leq k$ в нарастващ ред. Тогава от:

   $$u_j = q_j\#_k W'_k \dots \#_1 W'_1, \text{ където } W'_i = W^{(j)}[1..J_i^{(j)} - 1](a_i, i)W^{(j)}[J_i^{(j)} + 1..|W^{(j)}|],$$

   получаваме:
   $$u'_j = (a_k, k)(a_{k-1}, k-1)\dots(a_1, 1)q_j\#_k W'_k\#_{k-1}W'_{k-1}\dots\#_1 W'_1.$$

   (б) Прилагаме към u'_j граматика с единствени правила:

   $$\{S(a_k, k)(a_{k-1}, k-1)\dots(a_1, 1)q \to F(b_k, k)\dots(b_1, 1)p \,|\, \langle(q, (a_1, \dots, a_k)), (p, (b_1, \dots, b_k)\rangle \in \Delta\}.$$

   В частност, ако M е детерминирана, то за всяко α има най-много едно правило с лява страна α и тъй като всички правила имат една и съща дължина на левите си страни, тези правила ще са еднозначни. Получаваме:
   $$u''_j = (b_k, k)(b_{k-1}, k-1)\dots(b_1, 1)q_{j+1}\#_k W'_k\#_{k-1}W'_{k-1}\dots\#_1 W'_1.$$

   (в) За $i = k$ до $i = 1$ прилагаме множеството R_i от правила:

   $$\begin{array}{rcl} R_i & = & \{(a, i) \to (b_i, i) \,|\, a \in \Gamma\}, \text{ ако } b_i \in \Gamma \\ R_i & = & \{c(a, i) \to (c, i)a \,|\, a, c \in \Gamma, a \neq \_\} \cup \\ & & \{c(\_, i)d \to (c, i)\_d \,|\, c, d \in \Gamma\} \cup \{c(\_, i)\#_{i-1} \to (c, i)\#_{i-1} \,|\, c \in \Gamma\}, \text{ ако } b_i = \leftarrow \\ R_i & = & \{(a, i)d \to a(d, i) \,|\, a, d \in \Gamma\} \cup \{(a, i)\#_{i-1} \to a(\_, i)\#_{i-1} \,|\, a \in \Gamma\}, \text{ ако } b_i = \rightarrow. \end{array}$$

   (г) С това получаваме u_{j+1} и продължаваме със следващата итерация на цикъла.

5. Ако $u_j[1] \in F$, приемаме думата, тоест заменяме $u_j[1]$ с F_M.

Както и в предишната теорема, за n прехода, всяка от думите W_j може да увеличи дължината си най-много с n. Поради това $|u_j| \leq c(|w| + k + kn)$. Една конкретна итерация на цикъла изисква $\leq c'(|u_j| + k + 1)$ прехода. Следователно ако M приема w с изпълнение с дължина n, то в G_M ще има извод $S_M w\# \Rightarrow_{G_M}^{(m)} F_M v'\#$, с дължина:

$$m \leq n(c(|w| + k + kn)) \leq c'(|w| + n + 1)^2,$$

където $c' = ck$ зависи единствено от k и от машината на Тюринг M. $\square$