

نکته: $B \subseteq A$ اما A عضو دارد که در B نیست حروف الفبای B را از مجموعه A ناصیه اند $B \subset A$

• $A - (B \cap C) = (A - B) \cup (A - C)$

• $A - (B \cup C) = (A - B) \cap (A - C)$

• $x \notin A, x \notin B \rightarrow A \neq \bar{B} \text{ or } A \neq B$

• $x \in A, x \notin B \rightarrow A \neq B \text{ or } A \neq B$

نکته: \emptyset تک‌ای را به تقسیم

• $L_1 / L_2 = \{u \mid uv \in L_1, v \in L_2\}$

• $L_1 \setminus L_2 = \{u \mid vu \in L_1, v \in L_2\}$

• if $1 \in L_2 \rightarrow$ (رشته‌ها موجود در L_1 قطعه L_1/L_2 و $L_1 \setminus L_2$ است) $\rightarrow L_1 \subseteq L_1 / L_2$
 $L_1 \subseteq L_1 \setminus L_2$

• $L_1 / L_2 = (L_1^R \setminus L_2^R)^R$ و $L_1 \setminus L_2 = (L_1^R / L_2^R)^R$

• $C / (A \cup B) = C / A \cup C / B$, $C \setminus (A \cup B) = C \setminus A \cup C \setminus B$

$(A \cup B)^{-1} C = A^{-1} C \cup B^{-1} C$

رشته پالیندروم \leftarrow رشته‌ای که برابر معکوس خود باشد

- در این قدم رشته‌ها اگر با زوج باشد می‌توانیم $u u^R$ بنویسیم
- اگر رشته x آینه‌ای باشد x^R هم آینه‌ای است و اگر x^R آینه‌ای باشد x هم آینه‌ای است
- هر رشته پالیندروم تعداد رشته‌ها بطول $2k+1$ و $2k+2$ با هم برابر است
- تعداد رشته‌های آینه‌ای بطول n روی الفبای دوغانه‌ای برابر است با $\sum_{i=1}^n \binom{n-1}{i-1} 2^{n-i}$
- تعداد رشته‌های آینه‌ای بطول n روی الفبای Σ خاص برابر است با $\sum_{i=1}^n \binom{n-1}{i-1} 2^{n-i}$

برای رشته‌های بطول $n \leftarrow$ تعداد پسوندها $n+1$

prefix و suffix
head(w) tail(w)

تعداد زیررشته‌ها $\frac{n(n+1)}{2} + 1$ \leftarrow تعداد زیررشته‌ها $n+1$

PF $\rightarrow \exists u \in L, v \in L \mid u = \text{head}(v)$ PFX
 $L \cap L^+ \neq \emptyset$ PFX
 $x, xy \in L \rightarrow y = x$ PF \checkmark
 if $1 \in L$ PFX

SF $\rightarrow x, yx \in L \rightarrow y = x$ SF \checkmark
 $L \cap L^+ L = \emptyset$ SF \checkmark

لعم آران

۴. A و B حوزبان (نماده) و $X = AX \cup B$ همواره $X = A^*B$ کوچکترین پاسخ بدین مسئله است حال اگر $A \neq \lambda$ باشد تنها پاسخ است و اگر $A = \lambda$ و $C \in \Sigma^*$ $X = A^*(C \cup B)$ نیز جواب است.

$$X = XA \cup B \rightarrow X = B^*A$$

$$*(L_1 \cdot L_2)^R = L_2^R \cdot L_1^R$$

$$*(L_1 \cap L_2)^R = L_1^R \cap L_2^R$$

نکته

$$*(L_1 \cup L_2)^R = L_1^R \cup L_2^R$$

$$*(L_1 - L_2)^R = L_1^R - L_2^R$$

$$*(L^n)^R = (L^R)^n$$

$$*(L^*)^R = (L^R)^*$$

$$*(L^+)^R = (L^R)^+$$

$$*L_1 \cdot (L_2 \cup L_3) = L_1 L_2 \cup L_1 L_3$$

$$*(L_2 \cup L_3) \cdot L_1 = L_2 L_1 \cup L_3 L_1$$

$$*L_1 (L_2 \cap L_3) \subseteq L_1 L_2 \cap L_1 L_3$$

$$*(L_2 \cap L_3) L_1 \subseteq L_2 L_1 \cap L_3 L_1$$

$$*L_1^+ \cup L_2^+ = (L_1 \cup L_2)^+$$

$$*(L_1 \cap L_2)^+ \subseteq L_1^+ \cap L_2^+$$

$$*L^+ \cdot L = L^+ \cdot L^+ = L^+ \cdot L^*$$

$$*L^+ \cdot \lambda = L^+$$

$$*L \cdot L^+ = L^+ \cdot L$$

نکته

$$*(L^+)^+ = (L^*)^+ = L^+$$

$$*L^+ \cdot \lambda = L^+$$

$$*\overline{\Sigma^+} = \phi \quad / \quad \overline{\phi} = \Sigma^+$$

$$*\overline{\Sigma^+} = \lambda \quad / \quad \overline{\lambda} = \Sigma^+$$

$$*\phi^+ = 1$$

$$L/\phi = L \setminus \phi = L \cdot \phi = \phi$$

نکته: زبانی وجود ندارد که در تقابلی $L^+ = \overline{L^+}$ صدق کند

زبان‌هایی وجود دارد که در تقابلی $L^+ = \overline{L^+}$ صدق کند مانند λ ، Σ^+ ، ϕ ، Σ^* ، α^*

$$L = \{ \omega \mid |\omega_a| = |\omega_b| \} \quad \begin{aligned} s &\rightarrow asb \mid bsa \mid ss \mid \lambda \\ s &\rightarrow asbs \mid bsas \mid \lambda \end{aligned}$$

نکته

FSA حافظه دارد

۱) افلا

→ تدریس اشتباه کنه
→ تدریس اشتباه فضا

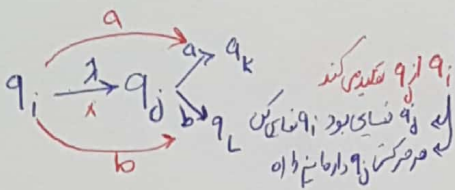
۲) تراپا

$$M = (Q, \Sigma, \delta, q_0, F) \quad \delta: Q \times \Sigma \rightarrow Q \quad \text{of DFA}$$

$$L(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$$

$$M = (Q, \Sigma, \delta, q_0, f) \quad \delta: Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q \quad \text{NFA}$$

$$L(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \emptyset\}$$



۹۱ از ۹۰ تاکید می کند
۹۲ و ۹۳ فضای بود
۹۴ هر صفت گسترده دارا

۱. حذف حرکت ۱
۲. افزودن حالت جدید برای رفع عدم قطعیت
۳. بهینه سازی

$O(m^n) = O(n) \propto \text{NFA}$
 $O(n) \propto \text{DFA}$

تعلق

DFA \sim FA \sim Reg

← تعداد حالات در حالت کلی $M_{NFA} \leq M_{DFA}$ ، $M_{NFA} \leq M_{DFA} \leq 2^{M_{NFA}}$

مکمل زبان
 برای اسم مکمل DFA حالت پایانی رویه غیر پایانی تبدیل کن و غیر پایانی رویه پایانی تبدیل کن
 $DFA \rightarrow M = (Q, \Sigma, \delta, q_0, F) \rightarrow \hat{M} = (Q, \Sigma, \delta, q_0, Q - F)$
 $NFA \rightarrow \{w \in \Sigma^* \mid \delta^*(q_0, w) \cap F = \emptyset\}$
 نکته: اگر DFA کمینه باشد همه حالات $M(L)$ و $\hat{M}(L)$ برابر است

مقاله یزدان ← تعریف حالت شروع و پیلانی و اگر دو حالت پیلانی داشتیم بین حالت اولیه جدید در نظر بگیرد از وزن حالت با ۱ برود به حالت های پای و در ادامه جفت فلش ها هم برگشتن کن

نکته: بفرض اینکه DFA کمینه زبان L دارای n حالت باشد، DFA کمینه زبان منظم L^R $2^n \leq \mu(L^R) \leq 2^n$ $\log n$ تعداد حالت DFA کمینه زبان منظم L با تعداد DFA کمینه L^R همواره برابر است.

نکته 8 - DFA معادل یک زبان منظم پذیر است اما DFA معادل منظم پذیر ولس NFA کمینه منظم پذیر نیست.

- برای هر DFA با چند حالت اولیه لزوماً یک DFA با یک حالت اولیه وجود دارد.
- برای هر DFA با چند حالت نهایی لزوماً یک DFA با یک حالت نهایی وجود ندارد.
- برای هر NFA با چند حالت اولیه لزوماً یک DFA معادل با یک حالت اولیه وجود دارد.
- برای هر DFA با چند حالت نهایی لزوماً یک DFA با فقط یک حالت نهایی وجود دارد.

- برای هر زبان منظم فاقد رشته بویج (یا) لزوماً یک NFA بدون حرکت از فقط با یک حالت نهایی وجود دارد.
- کوچکترین آمارها \rightarrow DFA کمینه بدون trap

نکته 9 - اگر حالتی که میبینی یک حالت پایانی ندارد را کنار گذاشته، اگر در گراف انتقال معادل ماشین حالت متناهی n ، حلقه یا انتقال رفت و برگشت دیده شد، زبان n لزوماً نامتناهی است.

- اگر طول کوچکترین رشته متعلق به زبان غیر تقس n برابر n باشد آنده DFA حداقل دارای $n+1$ حالت.
- یک ماشین DFA با n حالت اگر رشته بطول بزرگتر یا مساوی n را و بپذیرد.
- ۱- زبان این ماشین نامتناهی
- ۲- لزوماً رشته با طول کمتر از n رو هم خواهد پذیرفت

- اگر زبان n یک ماشین DFA با n حالت نامتناهی باشد این ماشین DFA لزوماً رشته ای با اندازه $2n-1$ را نخواهد پذیرفت

تجائز 9 \rightarrow به سبب قابل تجا \rightarrow $\begin{matrix} w, u & \text{Acc} \\ w, u & \text{Rej} \end{matrix}$ یا $\begin{matrix} \text{Rej} \\ \text{Acc} \end{matrix}$ $\exists u \in \Sigma^*$

نکته 10 - دل براس L قابل تجا اند هرگاه رشته w وجود داشته باشد که از میان w_1 و w_2 فقط یکی متعلق به L باشد به عبارتی دیگر L قابل تجا اند هرگاه $L/w_1 \neq L/w_2$

نکته 11 - هر زبان معادل یک DFA مجموعه Σ^* را به تعدادی کلاس هم ارزی افراز می کند
تعداد حالت DFA کمینه

نکته 12 - رابطه افراز پذیری یک رابطه هم ارزی است اما افراز پذیری هم ارزی نیست

نکته 13 - گراف انتقال هر ماشین متناهی معادل یک گراف انتقال کامل است

نکته 14 - حالت غیر قابل دسترس حالتی است که از حالت اولیه به آن میرسد نیست

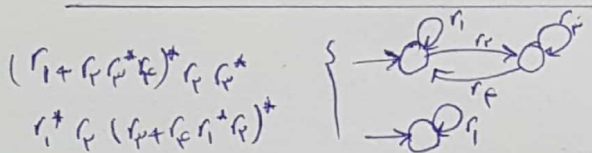
نکته 15 - $\delta^*(1-\text{closure})$ از هر حالت به خودش هم هست و مجموعه تمام حالتی است که از حالت q با قدرت n می رسد

نکته 8: $\alpha(B\alpha)^* = (\alpha B)^* \alpha$ / $\alpha + \phi = \alpha$ / $\alpha + 1 = \alpha$ (برای α برابر ϵ نیست)

$\epsilon^* \alpha \epsilon^* = \epsilon^* \alpha (\epsilon^* \alpha)^* = (\epsilon^* \alpha)^* \alpha \epsilon^*$ / $\alpha^* B^* \cap B^* \alpha^* = \alpha^* \cup B^*$

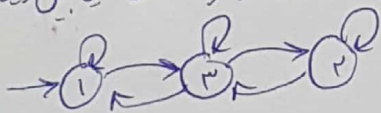
عبارات معادل $(\alpha + B)^*$

- $(\alpha + B)^* = (\alpha^* + B^*)^* = (\alpha^* B^*)^* = (B^* \alpha^*)^* = (\alpha^* B^*)^+ = (B^* \alpha^*)^+$
- $(\alpha + B)^* = \alpha^* (B \alpha^*)^* = (\alpha^* B)^* \alpha^* = (\alpha^* B)^* + (B^* \alpha^*)^* = (\alpha B^*)^* + (B \alpha^*)^* = (\alpha B^*)^* (B \alpha^*)^*$
- $(\alpha + B)^* \delta^* = (\alpha + B)^*$ $\delta \in (\alpha + B)^*$
- $(\alpha + B + \delta)^* = (\alpha + B)^*$ $\forall \delta \in (\alpha + B)^*$
- $(\alpha + \delta)^* = (\alpha + B)^*$ $B \in \delta$



GTG هر ماشین FSA (و می توان به روش دیگر تبدیل کرد)

۱. به ازای هر حالت



۱	۲	۳
۱	۲	۳
۱	۲	۳

مراحل تبدیل FSA به GTG ← در هر کدام حالت مابین دو حرف فاصله و اگر حالتی اضافی بین آن ها باشد دو راه داریم

نکته: به ازای هر گراف انتقال NFA داره شده می توان یک GTG معادل فقط با دو حالت که یکی اولیه و دیگری نهایی باشد که از یکدیگر متمایزند

۱. به ازای هر یک حالت در FSA در نظر گرفته

۲. حالت شروع

۳. اگر $x \rightarrow A$ حالت A باید با گذر α به حالت نهایی برسد

۴. یک حالت نهایی تعیین کنیم و هر کس حالت نهایی خواهد بود

تبدیل گرامر خطی از زبان به FSA

بدت آوردن گرامر خطی از زبان خطی از زبان

G_{RL} خطی از زبان $\leftarrow M \leftarrow M^R \leftarrow G_{RL} \leftarrow \begin{cases} A \rightarrow xB \\ SA \rightarrow Bx^R \\ C \rightarrow y \end{cases} \leftarrow G_{LL}$

تبدیل FSA به گرامر خطی از زبان

① $\delta(A, a) = B \sim A \rightarrow aB$

② $\delta(A, \lambda) = B \sim A \rightarrow B$

③ و برای هر حالت نهایی AEF

$A \rightarrow \lambda$ را اضافه کنید

نکته: برای هر گرامر منظم G با K قاعده یک DFA با حداکثر $K+2$ حالت

وجود دارد که پذیرنده زبان گرامر منظم باشد (قاعده تولید یافته) البته اگر تولید منظم

باشد حداکثر $K+1$ حالت دارد $A \rightarrow aB \quad A \rightarrow a \quad A \rightarrow \lambda \quad A \rightarrow xB \quad A \rightarrow x$

$$G(V, T, S, P)$$

$$V = V_1 \cup V_2 \cup \{S\}$$

$$T = T_1 \cup T_2$$

$$P = P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}$$

$$M(L_1 \cup L_2) = \{Q \times P, \epsilon, F, S, \{P_1, P_2\}\}$$

$$F_0 = \{Q \times \{F_1\} \cup \{P_1\} \times F_1 \cup \{F_1\} \times F_1\}$$

$$F_0 = \{Q \times F_1 \cup \{P_1\} \times F_1 \cup \{F_1\} \times F_1\}$$

نکته: حاصل‌تقاطع L_1 و L_2 منظم است. یعنی اگر L_1 و L_2 از زبان‌های منظم باشند، $L_1 \cap L_2$ نیز منظم است.

$$M(L_1 \cap L_2) = (Q \times P, \epsilon, F, S, \{P_1, P_2\})$$

$$F_0 = \{F_1 \times F_1\}$$

$$L_1, L_2 : G(V, T, S, P)$$

$$V = V_1 \cup V_2$$

$$T = T_1 \cup T_2$$

$$S = S_1$$

مجموعه قواعد P_1 و P_2 را اضافه می‌کنیم و هر قاعده $A \rightarrow \alpha$ در P را با $A \rightarrow S_1 \alpha$ جایگزین می‌کنیم.

$$G = (V, T, S, P)$$

$$V = V_1 \cup \{S\}$$

$$T = T_1$$

مجموعه قواعد P_1 را در P گذاشته و هر قاعده $A \rightarrow \alpha$ را با قاعده $A \rightarrow S_1 \alpha$ جایگزین می‌کنیم و قواعد تولید $S \rightarrow S_1 \alpha$ را نیز اضافه می‌کنیم.

تجزیه و تحلیل هر بخش

$$h(L) = \{h(w) : w \in L\} \quad h^{-1}(L) = \{w : h(w) \in L\}$$

$$h(h^{-1}(L)) \subseteq L \subseteq h^{-1}(h(L))$$

- اگر $h(L)$ منظم باشد آنگاه L لزوماً منظم نیست.
- اگر L یک زبان منظم باشد آنگاه $h(L)$ لزوماً منظم نیست.

$$h(L_1 \cup L_2) = h(L_1) \cup h(L_2)$$

$$h(L_1 L_2) = h(L_1) h(L_2)$$

$$h(L_1 \cap L_2) \neq h(L_1) \cap h(L_2)$$

تکمیل است

$$L_1 \cup L_2 \subseteq \text{SUF}(L_1)$$

$$L_1 / L_2 \subseteq \text{Pref}(L_1)$$

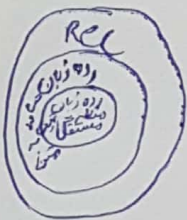
نکته: L_1 / L_2 منظم است اگر L_1 و L_2 منظم باشند.
نکته: $L_1 \cap L_2$ منظم است اگر L_1 و L_2 منظم باشند.
نکته: $(L_1 / L_2) \cap L_3$ منظم است اگر L_1, L_2, L_3 منظم باشند.
نکته: $L_1 / (L_2 \cap L_3)$ منظم است اگر L_1, L_2, L_3 منظم باشند.

نکته: $L_1 = L_1 L_2 / L_2$ isn't true for all language

نکته: زبانهای منظم تحت عمل \cap , \cup , \supset , \subset , \in همواره و * لزوماً بسته نیستند.

$$\left. \begin{array}{l} L \text{ منظم} \longleftrightarrow \bar{L} \text{ منظم} \\ L \text{ نامنظم} \longleftrightarrow L^R \text{ نامنظم} \end{array} \right\} -$$

- زبانهای نامنظم تحت اعمال \cap , \cup , \in همواره و * لزوماً بسته نیستند.
- اگر L_1 نامنظم و L_2 منظم آنگاه $L_1 \cup L_2$ و $L_1 \cap L_2$ لزوماً منظم است.
- هر زبان نامنظم زیر مجموعه یک زبان منظم است \leftarrow زبان منظم \supset^*
- هر زبان نامنظم لزوماً نامتناهی است.
- زبان L \leftarrow متناهی \leftarrow \bar{L} نامتناهی و منظم
- \leftarrow نامتناهی \leftarrow \bar{L} متناهی \leftarrow L نامنظم
- \leftarrow نامتناهی \leftarrow \bar{L} نامتناهی \leftarrow L نامتناهی ندارم



- از بین L و \bar{L} حتماً هر دو متناهی باشد و اگر یکی متناهی باشد هر دو منظم است.
- اگر L یک زبان گسسته (متمم) نامنظم باشد آنگاه L^* لزوماً منظم است.
- زبانهای منظم تحت اعمال زیر بسته اند:

$$L_1 \cup L_2 = \{x : x \in L_1 \text{ OR } x \in L_2 \text{ but } x \text{ isn't in both } L_1 \text{ and } L_2\}$$

$$\text{nor}(L_1, L_2) = \{w : w \notin L_1 \text{ and } w \notin L_2\}$$

$$\text{Cor}(L_1, L_2) = \{w : w \in \bar{L}_1 \text{ or } w \in \bar{L}_2\}$$

(۱۲) الگوریتم وجود دارد که $L = \text{tail}(L)$

۱) تصمیم بودن

۲) عضویت

۳) وجود رشته ۱

۴) کامل بودن (خ)

۵) متناهی بودن

۶) بستار بسته گیرایی

۷) L^R برابر L

۸) $L_1 \subseteq L_2$ یا $L_2 \subseteq L_1$

۹) $L_1 = L_2$ یا $L_1 \subseteq L_2$ یا $L_2 \subseteq L_1$

۱۰) داشتن رشته ای بدون زوج بداس زبان منظم $L \leftarrow L' = (\epsilon \epsilon)^+$

$L' \cap L$

۱۱) الگوریتم وجود دارد که تعیین کند زبان صایا لینه روم است $\leftarrow L = L^R$

تقسیم پذیری هر زبانهای منظم

نمونه تشخیص زبان منظم ←
 یافتن عبارت منظم ←
 استفاده از خواص بستاری ←
 روش بیکریمندها که منقضی ← $\Sigma^+ \text{ یا } \Sigma^*$
 لغو ترازها ← شرط کافی برای اثبات نامنظم بودن است

مثال ۱ $\forall n \geq 0$

$$\exists w \in L, |w| \geq n$$

$$\forall x, y, z \in \Sigma^+ \quad w = xyz \quad |x| \leq n, |y| \geq 1, |z| \geq 1$$

$$\exists i \in 0, 1, \dots : w_i = x y^i z \notin L$$

مثال ۲

$$\forall m \geq 0$$

$$\exists w \in L, |w| \leq m$$

$$\forall x, y, z \in \Sigma^+ \quad w = xyz \quad |x| \leq m, |y| \geq 1, |z| \geq 1$$

$$\exists i \in 0, 1, \dots : w_i = x y^i z \notin L$$

مثال ۳

$$\forall m \geq 0$$

$$\exists w \in L$$

$$\forall w = u, v, u_p, u_1, u_2 \in \Sigma^+ \quad |u| \geq m, |v| \geq 1, |u_p| \geq 1$$

$$\exists i \in 0, 1, \dots : w_i = u, x y^i z u_p \notin L$$

قضیه $|u| - |v| = m$: نامنظم است اگر و تنها اگر یک مجموعه نامتناهی $K \subseteq \Sigma^+$ وجود دارد که
 این مجموعه در دو کلاس هم ارزی قرار گیرد در اینصورت تقارن کلاس های هم ارزی نامتناهی است
 و گویا زبان منظم نیست

معادلسازی یک زبان با زبان دیگر

$$ex \rightarrow \{ u w w^R v \mid u, w, v \in \Sigma^+, |u| \geq 1, |v| \geq 1 \}$$

$$w \in \Sigma^+ \rightarrow w'a \rightarrow w w^R = w'a a w'$$

$$w' \in \Sigma^+ \rightarrow w'b \rightarrow w w^R = w'b b w' \rightarrow |\Sigma|^n \Sigma^* \Sigma^* (aa+bb) \Sigma^* \Sigma^+$$

$$v = \Sigma^+ \text{ و } u = |\Sigma|^n \Sigma^* \quad \text{نمونه خاص}$$

$$L = \{ w \in \{a, b, c\}^* \mid p n_a(w) + q n_b(w) + r n_c(w) + s = 0 \}$$

$$L = \{ a^m b^n c^k \mid p m + n k + r q + s = 0 \}$$

$\alpha \leftarrow \neq, =, \leq, <, >, \geq$ و α یک عدد صحیح
 اگر p, q, r, s هم علامت باشد لزوماً منظم است وگرنه نامنظم است

توابع خاص که مت آنرا گریز زبان منظم باشد منظم می ماند.

- $\text{Delet first}(L) = \{w \in \Sigma^* : aw \in L, \exists a \in \Sigma\} \sim L \setminus \epsilon$
 $\text{Delet last}(L) = \{w \in \Sigma^* : wa \in L, \exists a \in \Sigma\} \sim L \setminus \epsilon$
- $\text{insert}(L) = \{uav : a \in \Sigma, uv \in L, u, v \in \Sigma^*\}$
 $\text{drop}(L) = \{uv : uav \in L, a \in \Sigma, u, v \in \Sigma^*\}$
- $\text{tail}(L) = \{y : xy \in L \text{ for some } x \in \Sigma^*\}$
 $\text{head}(L) = \{x : xy \in L \text{ for some } y \in \Sigma^*\}$
- $\text{truncate}(L) \rightarrow$ removes the right most symbol from any string
 if L is a Regular language not containing ϵ , then $\text{truncate}(L)$ is also regular.
- $\text{even}(L)$ and $\text{odd}(L)$
- $\text{chop left}(L) \rightarrow$ remove the left most symbol of every string
 $\text{chop right}(L)$
- $\text{erase}(L) = \{a_1 a_2 \dots a_{k-1} a_{k+1} \dots : a_1 a_2 \dots a_{k-1} a_k a_{k+1} \dots \in L\}$
- $\text{perfect shuffle} = \{w = a_1 b_1 a_2 b_2 \dots a_k b_k : a_1 \dots a_k \in A, b_1 \dots b_k \in B, a_i, b_i \in \Sigma\}$
 $\text{shuffle} = \{w : w = a_1 b_1 a_2 b_2 \dots a_k b_k : a_1 \dots a_k \in A, b_1 \dots b_k \in B, a_i, b_i \in \Sigma^*\}$
- $\{u v : u \in L, v \in L^R\}$
- $\text{third}(L)$
- $\{a_1 a_2 a_3 a_4 \dots a_n : a_1 a_2 \dots a_n \in L\}$

نکته: مت عمل $\text{Perm}(L)$ مجموعه رشته های که شامل تمام جایگشت های رشته زبان هستند بسته نیستند

- $\text{shift}(L) = \{v : v = \text{shift}(w) \text{ for some } w \in L\}$
- $\text{exchange}(L) = \{v : v = \text{exchange}(w) \text{ for some } w \in L\}$
- $\text{min}(L) = \{w \in L : \text{there is no } u \in L, v \in \Sigma^+, \text{ such that } w = uv\}$

ساخت پیشوند پیوند subst یک زبان از روی DFA آن زبان

پیشوندها تمام حالتی رونمایی کن که از آن میر از حالت اولیه به نهایی هستند

تمام اون حالت که از میر حالت اولیه به نهایی اند باید بایه حرکت لا بیکر جدید بنویسیم به به حالت نهایی جدید با هم که نداریم

پیوندها تمام حالتی که از میر اولیه به نهایی مت روی به حالت آغازین تبدیل کن با trap که نداریم

subst به حالت اولیه و نهایی جدید تکون و از حالت اولیه جدید به تمام حالت با گذر لا به طرز تمام حالت با گذر لا به حالت نهایی جدید و و با trap که نداشته باش

$$- L = \{w^n \mid n \geq 1, w \in \{a, b\}^+\} \sim \Sigma^*$$

$$- L = \{w^n \mid n \geq 1, w \in \{a, b\}^+\} \sim \Sigma^+$$

$$- L = \{w^n \mid n \geq 2, w \in \{a, b\}^+\} \text{ not Reg}$$

$$- L = \{uww^Rv, w, v, u \in \Sigma^+\} \text{ Reg}$$

$$- L = \{uww^Rv, w, v, u \in \Sigma^+, |u| \geq |v|\} \text{ not Reg}$$

$$- L = \{uww^Rv, w, v, u \in \Sigma^+, |u| \geq^* |v|\} \text{ Reg}$$

$$- L = \{uww^Rv, v, u \in \Sigma^+, w \in \Sigma^+\} \text{ Reg}$$

$$- L = \{u^Rwv^Rw^Ru : u, v, w \in \{a, b\}^+\} \text{ Reg}$$

$$- L = \{vww^Rv^R : v, w \in \{a, b\}^+\} \text{ not Reg}$$

$$- L = \{vwv^Rw^R : v, w \in \{a, b\}^+\} \text{ not Reg}$$

$$- L = \{a^n \mid n \text{ is product of two number}\}$$

$$- L = \{a^n \mid n \text{ is either prime or the product of two or more prime number}\}$$

$$- L = \{w_1c w_2 : w_1, w_2 \in \{a, b\}^+, w_1 \neq w_2\} \text{ isn't Reg}$$

$$- L_1, L_2 \text{ is Reg } L = \{w : w \in L_1, w^R \in L_2\} \text{ Reg}$$

$$- L = \{w \in \{a, b, c\}^* : |w| = 3n_a(w)\}$$

$$- L = \{uvu : u \in \Sigma^+, v \in \Sigma^+\} \text{ isn't Reg}$$

$$- L = \{a^n b^m \mid n > m \text{ OR } n < m\} \text{ قابل نیست}$$

$$L = \{a^n b^m \mid n > m \text{ and } n < m\} \text{ قابل نیست}$$

$$L = \{a^n b^m \mid n \geq m \text{ OR } n \leq m\} \text{ قابل نیست}$$

$$L = \{a^n b^m \mid n \geq m \text{ and } n \leq m\} \text{ قابل نیست}$$

$$L = E \cup \Sigma^* H, E, H \subseteq \Sigma^+, H, E \rightarrow \text{متناهی}$$

زبان definite (زبان مشخص)

$$0/1/1-1-1 \rightarrow \text{ب}$$

$$*/R/0 \rightarrow \text{بسته}$$

نکته یک درخت اشتقاق خود یک درخت اشتقاق جزئی در عکس آن لزوما درخت نیست

نکته اگر در تمام قواعد $\alpha \rightarrow \beta$ و $K = |\alpha| > 1$ اندازه ارتفاع درخت اشتقاق $\leq \frac{|\omega| - 1}{K - 1} \leq h \leq \frac{|\omega|}{K}$

مسئله عضویت

نکته در الگوریتم جستجو کامل برای $\omega \in L$ لزوما مشخص می‌کند که رشته ω متعلق به L است یا نه متوقف می‌شود و اگر رشته متعلق به L نباشد الگوریتم جستجو کامل ممکن است متوقف نشود (در حلقه یا محدود قرار بگیرد)

نکته اگر گرامر مستقل از متن G فاقد قواعد $A \rightarrow A$ و قواعد $A \rightarrow B$ باشد الگوریتم جستجو کامل برای $\omega \in T^+$ مشخص می‌کند که $\omega \in L$ یا $\omega \notin L$ به تعبیری برای هر $\omega \in T^+$ متوقف می‌شود

نکته اگر گرامر افزایش باشد $O(2^{|w|})$ صوره اشتقاق لازم داریم (برای اینکه به ترتیب بررسی دوره‌ها نیاز داریم)

عمل زبانی الگوریتم جستجو کامل

$|P|$ تعداد قواعد \rightarrow این فرمول در واقع تعداد SF هام مشخص می‌کند

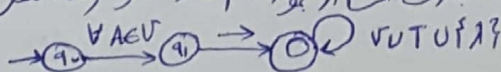
$$1 + |P| + |P|^2 + \dots + |P|^{2|w|} = O(|P|^{2|w|+1})$$

نکته اگر فرم گرامر جامعی باشد الگوریتم عضویت CFL در زمان $O(|w|^3)$ عضویت یا عدم عضویت رشته را مشخص می‌کند

نکته در گرامر ساده مرتبه زمانی تشخیص عضویت $O(|w|)$ است
 و هر گرامر ساده‌تر از این تغییر مفهوم است و در عکس آن لزوما برقرار نیست

چند نکته - مجموعه تمام عبارات منظم (وس a, b, \dots) یک زبان CF است
 $R \rightarrow R + R \mid RR \mid (R) \mid a \mid b \mid \phi \mid \lambda$

- مجموعه متغیر V و پایانه T مفروض و زبان L عبارت است از مجموعه تمام قواعد گرامر مستقل از متن که روی T و V تعریف شده است



وقواعد گرامر حساس به متن باشد \rightarrow یک زبان مستقل از متن است
 و زبان حاصل عام قواعد گرامر بدون محدودیت باشد \rightarrow یک مجموعه منظم است

گرامرهای مبهم

- گرامر G را مبهم گویند هرگاه برای حداقل یک اشتقاق بیش از یک درخت اشتقاق داشته باشد

- گرامر G مبهم

- ۱ اگر یک متغیر، جمله ای بکشد T^* به درختی تولید
- ۲ اگر در گرامر حریف باشد $(A \rightarrow A)$
- ۳ وجود همزمان $X \rightarrow Y$ و $X \rightarrow Z$ درست است یک متغیر

$A \rightarrow A \alpha \mid B A$
 $A \rightarrow A \alpha A$
 $(\alpha \in \{V, T\})^+$ مفید و A

- تشخیص مبهم بودن یک زبان

- ۱ L را بتوان به صورت اجتماع دو یا چند زبان نوشت
- ۲ در این زبان ها لازم است اشتراک غیر متناهی باشد
- ۳ اشتراک را نتوان در یک حذف و در دیگری قرار داد

نکته: - مسئله مبهم بودن یا نبودن گرامرها مستقل از متن یک مسئله تصمیم ناپذیر است

- مسئله ابعاد فقط در مورد گرامر زبان مستقل از متن یا متن نیست

- در صورتی که گرامر G غیر مبهم باشد برای $\{L(G) \mid L(G) \text{ تعداد گام ها حبت تولید در } MD \text{ و } RMD \text{ لزوماً برابر است}\}$

مولف بسیاری زبان ها مبهم و غیر مبهم

- خانواده زبان ها طبق مرتبه n و $n+1$ بسته است

- وارون هر زبان غیر مبهم لزوماً غیر مبهم است و خانواده زبان ها غیر مبهم تحت عمل R بسته اند

- زبان L ذاتاً مبهم اگر و تنها اگر L^R ذاتاً مبهم باشد و خانواده زبان ها ذاتاً مبهم تحت عمل مکملوس بسته اند

نکته: - $CFL = \{ \text{زبان غیر مبهم} \mid \text{زبان ذاتاً مبهم} \}$
 $\cap = \emptyset$

CFL [ذاتاً مبهم / غیر مبهم]
 CFG [غیر مبهم / مبهم]

- برای هر زبان غیر مبهم لزوماً یک گرامر غیر مبهم مولد وجود دارد

برای هر زبان غیر مبهم لزوماً یک گرامر مبهم مولد وجود ندارد

برای هر زبان ناقص غیر مبهم لزوماً یک گرامر مبهم مولد وجود دارد

- تعداد رشته طول n و با تعداد مساوی a و b $\left(\frac{n}{2}\right)$

- $w w^R$ ذاتاً مبهم نیست

$L = \{ a^n b^n \mid n \geq 1 \}$
 $S \rightarrow a S b \mid a S a$

- ساده سازی گرامر CF ← کاربرد ← یافتن زبان گرامر
 ← ساخت فرم نرمال یارس بتر
- تریب اعمال
- ① حذف قاعده $A \rightarrow \epsilon$ اگر $A \rightarrow S$ به این فرم ضمیمه
 ② حذف قاعده واحد
 ③ حذف غیر مفید \rightarrow به تریمینال فرم نمیشد
 غیر قابل دسترسی
- صغیر ارزیابی پیچیدگی گرامر \rightarrow $Complexity(G) = \sum_{A \rightarrow \alpha} (|\alpha| + 1) = \sum_{A \rightarrow \alpha} |\alpha| + \sum_{A \rightarrow \alpha} 1 = \sum_{A \rightarrow \alpha} |\alpha| + n$ که $n = |A|$

نکته: با حذف ϵ پیچیدگی گرامر حاصل ممکن است افزایش یابد.
 - با ϵ واحد $\sim \sim \sim \sim \sim \sim$
 - $\sim \sim$ ناصفید $\sim \sim \sim \sim$ کاهش $\sim \sim$
 - با این اعمال حذف قواعد گرامر کمینه نمی شود لزوماً.

نکته: حذف قواعد ϵ میتواند باعث ایجاد قواعد تولید واحد و ناصفید باشد که قبلاً نبوده است.
 - حذف قواعد تولید واحد میتواند باعث ایجاد قواعد تولید ϵ و ناصفیدی باشد که قبلاً نبوده است.
 - حذف قواعد ناصفید باعث ایجاد ϵ یا واحد نمیشود.

- اگر گرامری فاقد قواعد تولید ϵ باشد، حذف قواعد واحد میتواند فقط باعث ایجاد قواعد ناصفیدی باشد که قبلاً نبوده است و تولید نمیشد.
 - با حذف قواعد یک سرعت اشتقاق میتواند افزایش یابد.
 - اگر درست است یک قاعده تولید n متغیر بوج شونده باشد تعداد قواعد تولید که از این قاعده تولید با بوج شدن یا نشدن متغیرها بوج شونده تولید میشود حداکثر 2^n می باشد.

نکته: اگر G مبهم باشد با اعمال ساده سازی های فوق لزوماً باعث رفع ابهام گرامر نمیشود.

$$G: A \rightarrow x_1 | A x_1 | \dots | A x_n \quad \rightarrow \quad G': A \rightarrow x_1 | x_2 | \dots | x_n \quad \leftarrow \text{حذف خود بازگشتی}$$

$$A \rightarrow x_1 | x_2 | \dots | x_m \quad \rightarrow \quad A' \rightarrow x_1 | x_2 | \dots | x_m | A' x_1 | A' x_2 | \dots | A' x_m$$

- به ازای هر گرامر مستقل از متن دهانش که $\neq \epsilon$ باشد یک گرامر معادل بشکل نرمال چامسکی یا گریباخ موجود است.

- $A \rightarrow \alpha x \quad \alpha \in T, A \in V, x \in V^+$ **گریباخ**
- حذف لوغ + بازگشتی چیا هر گرامر ساده یک گرامر بشکل نرمال گریباخ دارند ولی عکس آن لزوماً درست نیست.
 - $\max(|V_T|) \leq |P| \leq \Pi(|V|)$
 - تعداد مراحل اشتقاق برای رشته طول n به برابر n است.
 - اگر G یک گرامر بشکل نرمال چامسکی با n متغیر باشد اگر G بتواند رشته ای بیش از n مرحله اشتقاق را تولید کند (یعنی نامتناهی است).
 - اگر گرامر بفرم چامسکی مبهم باشد برای هر رشته که بیش از یک درخت اشتقاق دارد، تعداد مراحل اشتقاق لزوماً برابر است.
 - $A \rightarrow BC \quad A, B, C \in V$
 $A \rightarrow \alpha \quad \alpha \in T$ **چامسکی**
 - حذف یونیکوین
 - اگر G یک گرامر k مرحله و k طول بزرگترین درخت قواعد تولید $|P| \leq |P'| \leq \Pi + (k-1)|P|$
 - برای طول رشته طول n مرحله اشتقاق نیاز داریم.
 - درخت اشتقاق تولید شده برای رشته n با استفاده از این گرامر با اعمال n طول رشته n $\leq |P| \leq \Pi + (k-1)|P|$
 - درخت اشتقاق تولید شده با این گرامر برای رشته طول n دارای $n \leq |P| \leq \Pi + (k-1)|P|$
 - درخت اشتقاق تولید شده برای رشته n با استفاده از این گرامر درخت اشتقاق دارد، تعداد مراحل اشتقاق لزوماً برابر است.

$$\delta: Q \times \{ \epsilon, 1, 2 \} \times \Gamma^* \rightarrow Q \times \Gamma^*$$

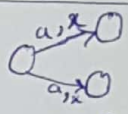
- هر ماشین پشته‌ای حرکت وابسته به بخار ورودی و بخار پشته می‌باشد

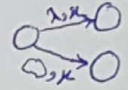
پذیرش توسط PDA \rightarrow قدر گرفتن درجات فضای و تمام شدن رشته ورودی.

$L(n) = \{ w \in \Sigma^* \mid (q_0, w, 2) = (q_f, 1, \alpha) \}$
 $\alpha \in \Gamma^*$ و $q_f \in F$

خاتمه شدن پشته و تمام شدن رشته ورودی $\{ q_0, w, 2 \} = (q_f, 1, \alpha) \}$
 $L(n) = \{ w \in \Sigma^* \mid (q_0, w, 2) = (q_f, 1, \alpha) \}$

عدم قطعیت در PDA

۱)  ۱

۲)  ۲

حرکت $(q, 1, \alpha)$ معاد x و نقل می‌کند این متمایز هم بخوند و هم بدون بخوندن علامتی از ورودی حرکت کنیم

نکته: $DPDA \neq NPDA$

$D = \{ L \subseteq \Sigma^* \mid \text{ماشین پشته‌ای غیر قطعی وجود دارد} \}$
 $C = \{ L \subseteq \Sigma^* \mid \text{برای تمام ماشین پشته‌ای قطعی وجود دارد} \}$

$\rightarrow C \subset D$

- برای هر زبان مستقل از متن قطعی لزوماً یک گرامر غیر مبهم وجود دارد، لذا هر زبان مستقل از متن قطعی لزوماً غیر مبهم و هر عکس آن درست نیست

- ماشین پشته‌ای $NPDA \equiv$ زبان‌ها مستقل از متن CFL

نکته: در باره پشته خالی و زبان قطعی

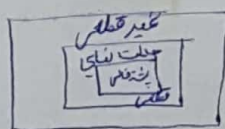
- معادل هر $NPDA$ باید پشته توسط حالت نهایی یک PDA به پشته خالی وجود دارد و معادل $NPDA$ باید پشته توسط پشته خالی $NPDA$ باید پشته توسط حالت نهایی وجود دارد. $NPDA$ حالت خالی \leftrightarrow $NPDA$ پشته خالی

- معادل هر $DPDA$ باید پشته توسط حالت نهایی لزوماً یک $DPDA$ توسط پشته خالی وجود ندارد و هر عکس آن برقرار است.

$PF \leftarrow$ پشته خالی / $NPFA \leftarrow$ پشته خالی x

$DPDA$ پشته خالی \leftrightarrow $DPDA$ با حالت خالی

$DPDA$ پشته \neq $DPDA$ با حالت نهایی

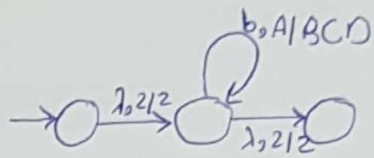


- اگر $DPDA$ وجود داشته باشد که پذیرنده زبان L باشد و L جزو الفبا زبان نباشد که $DPDA$ وجود دارد که $L \neq \{ \}$ را با پشته خالی پذیرش کند.

اسم PDA

① به هم گریختگی تبدیل کند

② $A \rightarrow bBCD$



نکته: هر زبان CF قابل ارائه با یک NPDA با پذیرش توسط حالت نهایی با 3 وضعیت قابل نمایش و صحنه 2 وضعیت.

- برای هر CF یک NPDA با یک وضعیت با شرط خالی شدن رشته وجود دارد.

- برای هر PDA یک NPDA معادل است که دارای دو ویژگی زمره

① فقط یک حالت نهایی q_f دارد که اگر انتها اگر رشته خالی شود وارد آن شود

② همه قوانین انتقال آن به شکل $\delta(q, a, A) = \{ (p, \alpha, c) \}$ هستند در حالیکه هر c_i برابر باشد با (q_i, B_i) یا (q_i, B_i) یا (q_i, B_i) به این معنی که هر حرکت یا محتوای رابه اندازه به واحد کم یا زیاد می کند

- هر زبان مستقل از متن L دارای یک PDA متناظر که برای L در 1000 نامتناهی صاف است

- هر PDA دارای یک PDA معادل بدون انتقال است

- ساده قلق رشته به زبان گرامر مستقل از متن به یکباره ماشین پشته اس T یک مسئله تصمیم پذیر است.

- در PDA پذیرنده δ ماکزیم ارتفاع رشته برای رشته داخل زبان است و با هر ورودی میزان حافظه مصرفی

زبان خطی $One\text{-}tum\text{-}PDA \equiv$ وسیله خوبی برای تشخیص خطی بودن زبان
بعد از اولین Pop دیگر Push نداریم
 $A \rightarrow xBy$
 $x, y \in T^*$
 $A, B \in V$

امثال محدودیت حافظه در پشته

- اگر طول رشته را محدود کنیم، چون حافظه را محدود کردیم، باین محدودیت هم توان با ماشین k و زبان حاصل منظم را می پذیرد

- اگر هیچ گاه عمل Pop را انجام ندهد آنگاه زبان که می پذیرد منظم است

- اگر در PDA با هر رشته ورودی ارتفاع پشته حافظه در حال افزایش باشد زبان منظم است

$LL(K)$ زبان هر گرامر $LL(K)$ یک زبان مستقل از متن قطعی و برای هر زبان مستقل از متن قطعی لزوماً گرامر $LL(K)$ دارد

$LR(1)$ زبان هر گرامر $LR(1)$ یک زبان مستقل از متن قطعی و برای هر زبان مستقل از متن قطعی لزوماً گرامر $LR(1)$ وجود دارد

$L_1 = \{ a^n b^m \mid m \neq n \text{ and } m \neq 2n \}$
CFL قطعی و غیر قطعی

$L_2 = \{ a^n b^m \mid m \neq n \text{ or } m \neq 2n \}$
منظم است و فقط 1 ورودی دارد

خاصیت ها خاص از Σ^+ و Σ^*

$\{ w \in \{a, b\}^* : w = uv, n_a(u) = n_b(v) \} \in \Sigma^*$

$\{ w^* : w = x \text{ and } x \in \Sigma^+ \} \in \Sigma^*$

$\{ uv : w = xy \text{ and } x, y \in \Sigma^+ \}$ از Σ^*

$\{ w : w = xy, x, y \in \{a, b\}^+ \text{ and } x \neq y \} \in \Sigma^+$

$\{ xy : x, y \in \{a, b\}^+ \text{ and } |x| \neq |y| \} \in \Sigma^+$

$\exists n \geq 0$

$\forall w \in L, |w| \geq n$

$\exists x, y, z, u, v \in \Sigma^* \quad w = uvxyz \quad |vxy| \leq n \quad |xy| \geq 1$

$\forall i \geq 0 \quad w_i = uv^i x y^i z \in L$

لیم‌تدریس مستقل از متن

برای اثبات مستقل از متن این لیم شرط کافی و برای
 ← خطی نیست
 ← هائیکس هم نیست
 ← منظم معریت
 اگر مستقل از متن باشد این نام نیست قطعا.

$\exists n \geq 0$

$\forall w \in L, |w| \geq n$

$\exists x, y, z, u, v \in \Sigma^* \quad w = uvxyz \quad |uvyz| \leq n \quad |vy| \geq 1$

$\forall i \geq 0 \quad w_i = uv^i x y^i z \in L$

لیم‌تدریس زبان خطی

شرط کافی برای اینکه زبان خطی نیست ← منظم نبودن هم این شرط کافی

۱. تقریباً لیم (و پرو غیر مفید بودت‌ات)

تقسیم پذیری در زبان‌های مستقل از متن

۲. عفتویت ← لیم بعد چاکس ← بعد CYK در زمان $O(n^3)$

۳. متناهی بودن (همه $|L(G)|$) ← لیم بعد رسم گراف اگر دور بودن متناهی

۴. وجود رشته‌ای بطول کم‌تر از n ← لیم چاکس $n-1$ گام

۵. گرامر شامل رشته‌های بطول زوج‌ات

۱. عفتویت

۲. منظم بودن

۳. تفسیر بودن

۴. حاصل بودن

۵. $L_1 = L_2$

۶. متناهی بودن

۱. ابعاد

۲. $L_1 = L_2$

۳. $L_1 \cap L_2 = \emptyset$

۴. $\bar{L} = \emptyset$

۵. کامل بودن

۶. منظم بودن

تقسیم نا پذیر در زبان‌های مستقل از متن

نکته ۵: ثابت تدریس در زبان‌های مستقل از متن بر تعداد غیر یابانه‌هاست و به $n = t + n + 1$ طول جملات زبان‌های مستقل از متن گویند

$S(L_1 \cup L_2) = \{ (uv)^* \mid u \in L_1, v \in L_2 \}$

زبان‌های مستقل از متن تحت عمل یزدن بسته است
 اگر (در هر زبان) تحت عمل مکمل به باشد (نشان دهد) مکمل آن (در زبان) تحت مکمل به است (نشان دهد)
 منظم تحت مکمل به ← نا منظم تحت مکمل به
 مستقل از متن تحت مکمل به ← نا مستقل از متن تحت مکمل به

Reg $\subseteq L_1 - L_2$ ← لزوماً مستقل از متن نیست و اگر L_1 مستقل از متن باشد L_2 نیز مستقل از متن باشد

فصل نهم ماشین های تورینگ

زبان یک ماشین تورینگ بدون محدودیت (ت) $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$
 $\Sigma \subseteq \Gamma - \{ \square \}$ $\delta: Q \times \Gamma_{read} \rightarrow Q \times \Gamma_{write} \times \{L, R\}$
 $\alpha \in (\cup U T)^+ \quad B \in (\cup U T)^+$

مسئله توقف در ماشین های تورینگ در این مدل حساب TM بر خلاف مدل های قبلی معروف شد که در مدل ها معتاد ندارد

پذیرنده $L(M) = \{ w \in \Sigma^+ : q_0 \vdash x_1 q_1 x_2 \text{ for some } q_1 \in F, x_1, x_2 \in \Gamma^+ \}$

پذیرش: ماشین M روی یک حالت عضو K توقف یا halt کند
 عدم پذیرش: (1) ماشین روی حالتی که عضو K نیست توقف کند
 (2) ماشین در یک حلقه نامتناهی بماند $x_1 q_1 x_2 \vdash \infty$

انواع ماشین تورینگ

حسابگر: بعد از این که ماشین تورینگ (TM) را حساب کرد و در حالت نهایی متوقف شد، هندوار باید روی او را کاراکتر رشته خروجی باشد.

توسعه اعداد بران شیراماسکت اصل بسوز

- 1) حالت stop داشته باشد
- 2) Multi track (چند شاره) $\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}$
- 3) Multi tape (چند نوار) $\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k$
- 4) Multithread $\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k$
- 5) نوار نیمه متناهی
- 6) non-deterministic $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ (در اصل یک ماشین می باشد که ماشین در حالت نهایی متوقف نشود)
- 7) offline $\delta: Q \times \Gamma \times \Gamma' \rightarrow Q \times \Gamma \times \Gamma' \times \{L, R\}^2$
- 8) Multidirectional (چند جهته)
- 9) رشته ای تقویت شده $\delta: Q \times (\Sigma \cup \{ \square \})^k \times \Gamma \times \Gamma' \rightarrow Q \times \Gamma^k \times \Gamma'^k \times \{L, R\}^k$ $TM \equiv K-PDA \quad K \geq 2$
- 10) توانایی تبدیل غار غیر خالص به خالص روزه
- 11) توانایی تبدیل شمار خالص به غیر خالص روزه

سایر مدل های ماشین تورینگ

محدود

- 1) هدف فقط بخوند
- 2) فقط روی بخش خاصی از نوار امکان نوشته شدن باشد
- 3) هدف فقط بشت چپ یا راست دور
- 4) در محدوده ورودی امکان نوشتن نیست

همه انواع FSA است و می کشد

نکته 8 - برای هر TM است ندارد با چند حالت نهایی لزوماً یک TM است ندارد باید حالت نهایی وجود دارد
 - برای هر TM لزوماً یک TM است ندارد معادل با حرکت ها وجود دارد که این را می کشد که حالت
 - اگر برای زبان L ماشین تورینگ وجود داشته باشد حروف الفبای M توان هر نگار ماشین تورینگ معادل برایش طراحی کرد

clock

- ضیق ان کا نقطہ صفر فی نواریں TM از $n=0$ است در حالی که $n=1$ است

- زبان L حساس به متن است اگر گویا LBA پذیرنده بر هر Γ وجود داشته باشد

- قدرت LBA از PDA بیشتر و از TM کمتر است

$$\alpha \rightarrow \beta$$
$$B, \alpha \in (V \cup T)^+$$
$$|\alpha| \leq |\beta|$$

در این ماستن اگر ارشد داخل زمان باشد با آن متوقف می شود
و مقدار بیکر بندس ماسی آن غلغل و کولان با آن زمان ابراز می شود و این غلغل است

تولید - هر $T_{\text{دک}}$ رشته با سبزی خواهد داشت و هر $T_{\text{دک}}$ را می‌توان با صفوی گذراند

- کدھر T_{∞} کے عدد پائیری اس وقت ولسر عدد پائیری لزوماً کہیں T_{∞} ہے۔

• ~ ~ ~ ~ ~ طبعی ~ ~ ~ ~ ~

- مجموعہ کے اس تمام ۸۸۸ فائیک مجموعہ نامتصر وزیر مجموعہ اسد اعلیٰ ہے جس میں مجموعہ ہندو ماشین ہاں
تو کئی شہادت .
اور اس کے بعد

شماره شمارا - اگر بین اعضای این مجموعه و هر ذکر مجموعه دلخواه از اعداد طبیعی به تناظر باشد، مجموعه شمارا

x, x^+, u, Q, Z مجموعه‌های حائزین هارنایک و مجموعه تمام زبان‌ها (مقطع u و CF)

DCFL شماریات و مرزبان لخواہ (عمر الفبا) شماریات

مجموعه اعداد حقیقی لزومان شمارا است

- تمام زبانی روس (الفبا^{*} ← 2^x نشمار است (اگر A شماران تمام باشد 2^A روزان شمار است)

- مجموعه همه زبان‌ها نامنتظم لزوماً نامشمار است. $2^{\mathbb{N}} =$ مجموعه نامنتظمی از زبان‌ها

هر مجموعه محدود از شمارات و اگر A شمارات باشد و B شمارات و C شمارات

- اگر A یک مجموعه شمارناقص باشد هر زیر مجموعه (صفاها یا نامتناهی) آن لزوماً شمارناقص است.

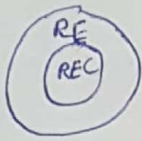
و چون x مع ثمرات m^* و x و m^* هم ثمرات

اگر A و B دو ... شمار باشد $A \cup B$ ، $A \cap B$ ، $A \setminus B$ نیز شمار است

زبان ها تقسیم پذیر RE بازگشت پذیر Decider
 زبان ها تقسیم پذیر RE بازگشت پذیر (Recursive Enumerable) Recognize
 همه تقسیم پذیر همه حل پذیر
 بر هر ورودی لزوما متوقف می شود و تصمیم گیری در کلاس زبان ها بازگشتی است

$w \in \begin{matrix} yes \\ no \\ pop \end{matrix}$

- نکته: - اگر L و \bar{L} هر دو RE باشند RE است.
- مکمل هر زبان RE تقسیم ناپذیر لزوما غیر فرمال و تشخیص ناپذیر است.
 - مکمل هر زبان CFL، CSL است چون CSL تحت عمل مکملیت است و هر CFL در واقع یک CSL است.
 - مکمل هر زبان حساس به متن نامستقل از متن لزوما نامستقل از متن نیست.
 - زبان ها تشخیص پذیر همه گرامر بدون محدودیت و TMs تشخیص دهند.
 - زبان تقسیم پذیر همه گرامر ندارد و TMs تقسیم گیرنده



نکته: $L = \{a^n b^n, n \geq 1\}$

- یک ماشین اتانوار دارد که برای مطابقت هر a با b باید $2n$ حرکت ببرد (مبارزه با پمپ)
- پیدا کردن حالت ماشین $O(n)$ است
- پیچیدگی زمانی $O(n \log n)$ است
- پیچیدگی فضای $O(n)$ است

- هر ماشین تورینگ غیر قطعی درباره درخت محاسبه یک ماشین تورینگ دارد که بر روی یک ورودی می تواند محاسبه این درخت را می توان با یک ماشین تورینگ قطعی با جستجو و سطح اول تولید کرد در صورت $NDTM$ رفت ها به صورت BFA و در DTM به صورت DF
- اگر ماشین تورینگ پذیرنده زبان بازگشتی داشته باشد با جایابی acc و rej به آن می رسیم و اگر تقسیم پذیر باشد با جایابی acc و rej به آن می رسیم.
- هر زبان بازگشتی لزوما ماشین صلی بی شمار از مجموعه بازگشتی نیست چونکه زبان محدود و مجموعه زبان ها بازگشتی است.
- ارتباط هر که بین میزان حافظه مصرفی بین LBA و PDA وجود دارد این است که هر دو از مرتبه $log n$ اند.
- پیچیدگی فضای در صورتی سئو نه مورد توجه و تحلیل قرار بگیرد که ماشین تورینگ به ازای هر رشته ورودی در هر شافه حاضریت محاسبه اش لزوما متوقف شود و پس این جمله درست است که اگر زبان L توسط یک ماشین تورینگ پذیرفته شود که مقدار حافظه مصرفی آن برای هر ورودی از محدودیت $log n$ تجاوز نکند آنگاه L تقسیم پذیر است.
- اگر یک DFA با n حالت تمام کلمات یا کمتر از n را در n^* را پذیرد آنگاه زبان آن بازگشتی چون باید تمام حالت های این ماشین شای باشد پس هر رشته ورودی که به این ماشین داده شود آن رشته را می پذیرد که
- یک ماشین تورینگ غیر قطعی که در واقع کنترل آن انتقال ها مستقل از محتوا و $transmit$ - مهم وجود دارد امکان حذف $transmit$ - در چنین ماشین یک مثال تقسیم پذیر است.

- سوال ۲۴ و ۳۰

فصل دوم تقسیم ناپذیری و کلمات پذیر

- تقسیم گیری در کلاس زبان ها بازگشت است چون می توانیم برای هر زبان بازگشت یک استواریم عضویت وجود دارد
- اگر $A \subseteq B$ به B که می پذیرد $(A \subseteq B)$ حل مسئله B به حل A کمک می کند و اگر راه حل برای حل مسئله B باشد A نیز قابل حل
- اگر $A \subseteq B$ و B تقسیم پذیری باشد A نیز تقسیم پذیر است و اگر B تشخیص پذیر باشد A نیز تشخیص پذیر است
- اگر $A \subseteq B$ و A تقسیم ناپذیر باشد B نیز تقسیم ناپذیر است و اگر A تشخیص پذیر باشد B نیز تشخیص پذیر است
- مسئله اینکه آیا یک صریحیت دارد یا براساس اولیه و راس آخر وجود دارد یک مسئله تقسیم پذیری است
- می توان برای مجموعه کراف ها یک درجه راس ها آنها کمتر یا مساوی 3 است یک DFA طراحی کرد

خواص بسیاری تمام زبانها

	انتقال	اشتراک	مکمل	تغییر	انسان	بازگشت	واحد	صورتی	واردن	امکان	اشتراک	انتقال	مکمل	اشتراک	انتقال	تقسیم
زبان	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
reg	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CFL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DCFL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CCFL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CSL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
rec	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
re	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

تقسیم پذیری

عقود	تقسیم پذیری	کامل بودن	مکمل بودن	اشتراک	تقسیم پذیری	مکمل بودن	عقود
✓	✓	✓	✓	✓	✓	✓	زبان ها منتظم
✓	✓	✓	✓	✓	✓	✓	زبان ها مستقل از متن
✓	✓	✓	✓	✓	✓	✓	زبان ها مستقل از متن صوری
✓	✓	✓	✓	✓	✓	✓	زبان ها صوری
✓	✓	✓	✓	✓	✓	✓	زبان ها بازگشت
✓	✓	✓	✓	✓	✓	✓	زبان ها بازگشت محدود