نظریه‌ی زبان‌ها و ماشین‌ها

درس‌نامه‌ی ۱۱

کاظم فولادی
http://kazim.fouladi.ir

ویراست اول: ۱۳۸۵
ویراست دوم: ۱۳۸۷
ویراست سوم: ۱۳۹۳

فهرست مطالب

۱۱ سلسله‌مراتب زبان‌های رسمی و آتوماتا ۱

۱-۱۱ زبان‌های بازگشتی و زبان‌های شمارش‌پذیر بازگشتی ۱

۱-۱-۱۱ زبان‌هایی که شمارش‌پذیر بازگشتی نیستند ۳

۲-۱-۱۱ زبانی که شمارش‌پذیر بازگشتی نیست ۴

۳-۱-۱۱ زبانی که شمارش‌پذیر بازگشتی است، اما بازگشتی نیست ۵

۲-۱۱ گرامرهای نامقید ۶

۱-۲-۱۱ چگونگی پیروی ماشین تورینگ از یک گرامر نامقید ۷

۳-۱۱ گرامرهای حساس به متن ۱۰

۱-۳-۱۱ زبان‌های حساس به متن و آتوماتای کران‌دار خطی ۱۰

۲-۳-۱۱ ارتباط میان زبان‌های بازگشتی و حساس به متن ۱۲

۴-۱۱ خانواده‌ی زبان‌های نوع صفر: زبان، گرامر و ماشین ۱۴

۵-۱۱ سلسله مراتب چامسکی ۱۵

۱۱ سلسله‌مراتب زبان‌های رسمی و آتوماتا

A Hierarchy of Formal Languages and Automata

مطالعه‌ی زبان‌های رسمی (زبان‌های صوری):

- **هدف اول:** بررسی زبان‌های مرتبط با ماشین تورینگ و بررسی محدودیت‌های آنها
  آیا زبان‌هایی وجود دارند که با ماشین تورینگ پذیرفته نشوند؟
- **هدف دوم:** بررسی ارتباط میان ماشین تورینگ و انواع گرامرها
  $\Leftarrow$ دسته‌بندی گرامرها $\Leftarrow$ دسته‌بندی زبان‌ها

بیشتر بحث‌های این فصل در مورد زبان‌هایی صادق است که حاوی λ نباشند.
ماشین‌های تورینگ به گونه‌ای که تعریف شده‌اند نمی‌توانند رشته‌ی تهی را بپذیرند.

۱۱-۱ زبان‌های بازگشتی و زبان‌های شمارش‌پذیر بازگشتی

بین زبان‌هایی که برای آنها یک ماشین تورینگ پذیرنده وجود دارد و زبان‌هایی که برای آنها الگوریتم عضویت وجود دارد، تمایز قایل می‌شویم (چون ماشین تورینگ بر روی یک ورودی که آن را نمی‌پذیرد، لزوماً توقف نمی‌کند).

تعریف **زبان شمارش‌پذیر بازگشتی (recursively enumerable).** زبان L را **شمارش‌پذیر بازگشتی** می‌گوییم، اگر ماشین تورینگی وجود داشته باشد که آن را بپذیرد.
یعنی ماشین تورینگی مانند M وجود دارد که برای هر $w \in L$ با $q_f \in F$،

$$q_0\ w \vdash^* x_1\ q_f\ x_2$$

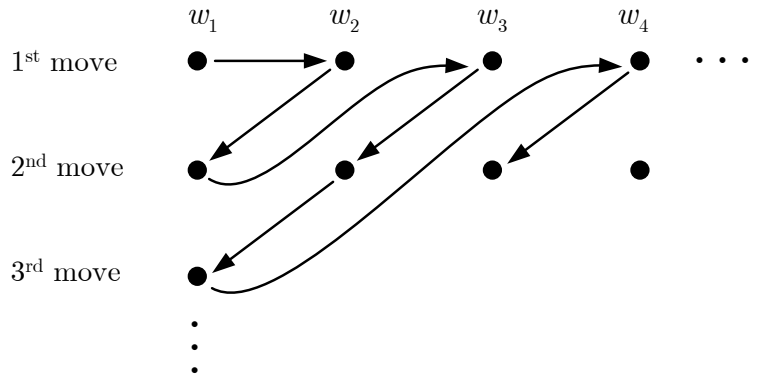
این تعریف چیزی در مورد اینکه اگر $w \notin L$ نمی‌گوید (توقف در حالت غیرنهایی یا حلقه‌ی نامتناهی).

تعریف **زبان بازگشتی (recursive).** زبان L بر روی Σ را **بازگشتی** می‌گوییم، اگر ماشین تورینگ M که L را می‌پذیرد، برای هر $w \in \Sigma^+$ توقف کند.
به عبارت دیگر، یک زبان بازگشتی است اگر و فقط اگر الگوریتم عضویتی برای آن موجود باشد.

اگر یک زبان بازگشتی باشد، آن‌گاه حتماً روال شمارش برای آن وجود دارد:
به فرض، M یک ماشین تورینگ باشد که عضویت در یک زبان بازگشتی L را تعیین می‌کند.
ابتدا ماشین تورینگ $\hat{M}$ را می‌سازیم که تمامی رشته‌های Σ^* را به ترتیب سره تولید می‌کند $(w_1, w_2, w_3, \ldots)$.
همین طور که رشته‌ها تولید می‌شوند، وارد M می‌شوند اما M تنها در صورتی که $w_i \in L$ باشد، آن را روی نوارش می‌نویسد.
اما اینکه برای هر زبان شمارش‌پذیر بازگشتی روال شمارش وجود دارد، به آسانی قابل ملاحظه نیست:
زیرا اگر $w_j \notin L$، ماشین M وقتی با w_j بر روی نوارش شروع شود، ممکن است هرگز متوقف نشود و بنابراین هرگز به رشته‌ی بعد از w_j نمی‌رسیم. برای اطمینان از عدم وقوع این حالت:

- ابتدا M را وادار به تولید w_1 می‌کنیم و اجازه می‌دهیم که M یک حرکت روی آن انجام دهد.
- آن‌گاه می‌گذاریم که $\hat{M}$، w_2 را تولید کند و M یک حرکت بر روی آن انجام دهد و پس از آن حرکت دوم روی w_1 را انجام دهد.
- پس از آن w_3 را تولید کرده و یک حرکت روی آن و حرکت دوم روی w_2 و سپس حرکت سوم روی w_1 را انجام می‌دهیم.
- ...

واضح است که M وارد حلقه‌ی نامتناهی نمی‌شود.
چون هر $w \in L$ توسط $\hat{M}$ تولید و توسط M در تعداد قدم‌های متناهی پذیرفته می‌شود، هر رشته‌ای که در L باشد، نهایتاً توسط M تولید می‌شود.



هر زبانی که برای آن یک روال شمارش موجود باشد، شمارش‌پذیر بازگشتی است

۱۱-۱-۱ زبان‌هایی که شمارش‌پذیر بازگشتی نیستند

قضیه فرض کنید که S یک مجموعه‌ی نامتناهی شمارا باشد. در این صورت مجموعه‌ی توانی آن یعنی 2^S ناشماراست.

◄ **اثبات.**
اگر $S = \{s_1, s_2, s_3, \ldots\}$ باشد، آنگاه هر $t \in 2^S$ می‌تواند به صورت رشته‌ای از ۰ و ۱ها نمایش داده شود که ۱ در موقعیت i قرار دارد اگر و فقط اگر s_i در t موجود باشد.
برای مثال $\{s_2, s_3, s_6\}$ به صورت ۰۱۱۰۰۱۰۰ و $\{s_1, s_3, s_5, \ldots\}$ به صورت ۱۰۱۰۱... نمایش داده می‌شود.
اگر 2^S شمارا باشد، آنگاه عناصر آن باید به ترتیبی مانند $t_1, t_2, t_3, \ldots$ باشد و همانند شکل زیر می‌توان آنها را در یک جدول بیاوریم:

$$\begin{array}{lcccccc} t_1 & \textcircled{1} & 0 & 0 & 0 & 0 & \cdots \\ t_2 & 1 & \textcircled{1} & 0 & 0 & 0 & \cdots \\ t_3 & 1 & 1 & \textcircled{0} & 1 & 0 & \cdots \\ t_4 & 1 & 1 & 0 & \textcircled{0} & 1 & \cdots \\ \vdots & & & & & \ddots & \end{array}$$

در این جدول، عناصر قطر اصلی را متمم می‌کنیم.
دنباله‌ی جدید (قطر مکمل شده)، عنصری از 2^S مثلاً t_i را نمایش می‌دهد، ولی نمی‌تواند t_1 باشد، به همین دلیل نمی‌تواند t_2 باشد و یا t_i دیگر (زیرا عنصر iام دنباله‌ی جدید، مکمل عنصر iام t_i است).
پس چاره‌ای نیست مگر اینکه 2^S ناشمارا باشد. ■

* این نوع بحث، **قطری کردن** (diagonalization) نام دارد (مکمل قطر با هر سطر دیگری متفاوت است) (منتسب به G. F. Cantor) وکاربرد دیگر آن اثبات ناشمارا بودن مجموعه‌ی اعداد حقیقی است.

اصل قطری‌سازی: اگر در یک ماتریس متشکل از صفرها و یک‌ها، قطر اصلی را مکمل کنیم، این قطر با هیچ یک از سطرهای ماتریس مساوی نخواهد بود.

نتیجه تعداد ماشین‌های تورینگ کمتر از تعداد زبان‌هاست ⇐ زبان‌هایی وجود دارند که شمارش‌پذیر بازگشتی نیستند.

قضیه به ازای هر الفبای Σ، زبان‌هایی وجود دارند که شمارش‌پذیر بازگشتی نیستند.

◄ اثبات.
از اینکه $L \subseteq \Sigma^*$ است داریم $L \in ۲^{\Sigma^*}$. چون Σ^* نامتناهی و شماراست، پس ۲^{Σ^*} ناشماراست. از طرفی مجموعه‌ی همه‌ی ماشین‌های تورینگ شماراست، پس زبان‌هایی وجود دارند که معادل با آنها ماشین تورینگی وجود ندارد، پس زبان‌هایی وجود دارند که شمارش‌پذیر بازگشتی نیستند.

۱۱-۱-۲ زبانی که شمارش‌پذیر بازگشتی نیست

چون هر زبانی که با الگوریتم توصیف شود، توسط یک ماشین تورینگ پذیرفته می‌شود، در نتیجه شمارش‌پذیر بازگشتی است. پس توصیف زبانی که شمارش‌پذیر بازگشتی نیست باید به صورت غیر مستقیم باشد.

قضیه زبان شمارش‌پذیر بازگشتی‌ای وجود دارد که متمم آن شمارش‌پذیر بازگشتی نیست.

◄ اثبات.
فرض می‌کنیم $\Sigma = \{a\}$ باشد.
مجموعه‌ی همه‌ی ماشین‌های تورینگ روی این الفبای ورودی را در نظر می‌گیریم که مجموعه‌ای شماراست.
بنابراین می‌توان ترتیب $M_۱, M_۲, M_۳, \ldots$ را برای عناصر آن در نظر گرفت.
برای هر ماشین تورینگ M_i، زبان شمارش‌پذیر بازگشتی $L(M_i)$ وجود دارد.
برعکس برای هر زبان شمارش‌پذیر بازگشتی بر روی Σ یک ماشین تورینگ وجود دارد که آن را می‌پذیرد.
زبان جدید L را به صورت زیر در نظر می‌گیریم:

$$\forall i \geq ۱ (a^i \in L \Leftrightarrow a^i \in L(M_i))$$

زبان L خوش‌تعریف است: چون $a^i \in L(M_i)$ ، در نتیجه $a^i \in L$ باید درست یا نادرست باشد.
متمم L به صورت

$$\bar{L} = \{a^i : a^i \notin L(M_i)\} \qquad (۱-۱۱)$$

خوش‌تعریف است، اما نشان می‌دهیم که شمارش‌پذیر بازگشتی نیست.
برهان خلف: فرض می‌کنیم $\bar{L}$ شمارش‌پذیر بازگشتی باشد. در این صورت باید ماشین تورینگ M_k موجود باشد که مطابق فرض $\bar{L} = L(M_k)$. رشته‌ی a^k را در نظر می‌گیریم: $a^k \in L$ یا $a^k \in \bar{L}$؟
اگر $a_k \in \bar{L}$ آنگاه طبق فرض باید $a^k \in L(M_k)$ باشد، اما به موجب (۱-۱۱) داریم $a_k \notin \bar{L}$.
برعکس اگر $a_k \in L$ آنگاه $a_k \notin \bar{L}$ و طبق فرض $a_k \notin L(M_k)$ و به موجب (۱-۱۱) داریم $a^k \in \bar{L}$.
تناقض اجتناب‌ناپذیر است، پس فرض اینکه $\bar{L}$ شمارش‌پذیر بازگشتی است، نادرست است.
برای تکمیل اثبات باید ثابت کنیم که L شمارش‌پذیر بازگشتی است (با استفاده از روال شمارش ماشین تورینگ): اگر a^i را داشته باشیم، ابتدا i را با شمارش تعداد aها پیدا می‌کنیم، سپس با استفاده از روال

شمارش برای ماشین‌های تورینگ M_i را می‌یابیم. نهایتاً توصیف آن را همراه با a^i به ماشین تورینگ جامع M_u که اعمال M را روی a^i شبیه‌سازی می‌کند، می‌دهیم. اگر $a^i \in L$ باشد، محاسبات M_u نهایتاً پایان می‌یابد. نتیجه، ماشین تورینگی است که هر $a_i \in L$ را می‌پذیرد و بر اساس تعریف، L شمارش‌پذیر بازگشتی است.

۳-۱-۱۱ زبانی که شمارش‌پذیر بازگشتی است، اما بازگشتی نیست

قضیه اگر زبان L و متمم آن $\bar{L}$ هر دو شمارش‌پذیر بازگشتی باشند، آنگاه هر دو زبان بازگشتی هستند. اگر L بازگشتی باشد، آنگاه $\bar{L}$ هم بازگشتی است و در نتیجه هر دو شمارش‌پذیر بازگشتی هستند.

◄ **اثبات.**

اگر L و $\bar{L}$ هر دو شمارش‌پذیر بازگشتی باشند، آنگاه ماشین‌های تورینگ M و $\hat{M}$ به عنوان روال‌های شمارش L و $\bar{L}$ وجود دارند.

M رشته‌های $w_1, w_2, w_3, \ldots$ در L را تولید می‌کند و

$\hat{M}$ رشته‌های $\hat{w}_1, \hat{w}_2, \hat{w}_3, \ldots$ در $\bar{L}$ را تولید می‌کند.

ابتدا اجازه می‌دهیم که M رشته‌ی w_1 را تولید کرده و آن را با w مقایسه کند:

اگر $w_1 \neq w$ بود، اجازه می‌دهیم که $\hat{w}_1$ توسط $\hat{M}_1$ تولید شود و مجدداً مقایسه را انجام می‌دهیم. اگر نیاز به ادامه دادن داشته باشیم، اجازه می‌دهیم که M رشته‌ی w_2 و $\hat{M}$ رشته‌ی $\hat{w}_2$ را تولید کرده و به همین ترتیب ادامه می‌دهیم.

هر $w \in \Sigma^*$ توسط M یا $\hat{M}$ تولید می‌شود، بنابراین در نهایت تطابقی خواهیم داشت. اگر رشته‌ی مذکور توسط M تولید شود، w به L تعلق دارد $(w \in L)$ وگرنه $w \in \bar{L}$. این فرآیند یک الگوریتم عضویت برای هر دوی L و $\bar{L}$ است و در نتیجه هر دوی آنها بازگشتی هستند.

برعکس اگر L بازگشتی باشد، آنگاه الگوریتم عضویتی برای آن وجود دارد، اما همین الگوریتم، الگوریتم عضویتی برای $\bar{L}$ می‌شود، اگر تنها نتیجه‌ی آن را عکس کنیم. بنابراین $\bar{L}$ بازگشتی است.

چون هر زبان بازگشتی، شمارش‌پذیر بازگشتی است، اثبات کامل است.

قضیه زبان شمارش‌پذیر بازگشتی‌ای وجود دارد که بازگشتی نیست؛ یعنی خانواده‌ی زبان‌های بازگشتی زیرمجموعه‌ای محض از خانواده‌ی زبان‌های شمارش‌پذیر بازگشتی است.

◄ **اثبات.**

زبان L در دو قضیه‌ی قبل این مطلب را نشان می‌دهد.

نتیجه زبان‌های خوش‌تعریفی وجود دارند که نمی‌توانیم برای آنها الگوریتم عضویت بسازیم.

۱۱-۲ گرامرهای نامقید

تعریف **گرامر نامقید (unrestricted).** گرامر $G = (V, T, S, P)$ را **نامقید** می‌گویند، اگر تمامی قواعد آن به صورت $u \to v$ باشد که در آن

$$u \in (V \cup T)^+, \qquad v \in (V \cup T)^*$$

- در یک گرامر نامقید هیچ شرطی بر روی قواعد اعمال نمی‌شود.
- هر تعداد متغیر و ترمینال می‌تواند در سمت چپ یا راست و به هر ترتیبی قرار گیرد.
- تنها محدودیت این است که λ نمی‌تواند سمت چپ یک قاعده قرار داشته باشد.

گرامرهای نامقید بزرگ‌ترین خانواده‌ی زبان‌ها را که می‌خواهیم آنها را به صورت مکانیکی تشخیص دهیم، تولید می‌کند.

قضیه هر زبانی که توسط یک گرامر نامقید تولید شود، شمارش‌پذیر بازگشتی است.

◄ **اثبات.**
گرامر مذکور روال شمارشی را برای شمردن همه‌ی رشته‌های زبان به طور سیستماتیک تعریف می‌کند.
برای مثال، می‌توانیم فهرستی از همه‌ی رشته‌های $w \in L$ تهیه کنیم که $S \Rightarrow w$ (در یک قدم).
چون تعداد قواعد گرامر متناهی است، تعداد این رشته‌ها متناهی خواهد بود.
سپس فهرستی از همه‌ی رشته‌های w که $w \in L$ است و در دو قدم تولید می‌شوند تهیه می‌کنیم
$S \Rightarrow x \Rightarrow w$
و به همین ترتیب ادامه می‌دهیم.
این اشتقاق‌ها را می‌توان با یک ماشین تورینگ شبیه‌سازی نمود،
در نتیجه روال شمارش برای زبان وجود دارد و این زبان شمارش‌پذیر بازگشتی است.

۱۱-۲-۱ چگونگی پیروی ماشین تورینگ از یک گرامر نامقید

ماشین تورینگ $M = (Q, \Sigma, \Gamma, \delta, q_\circ, \square, F)$ را داریم و می‌خواهیم گرامر G را بسازیم به طوری که $L(G) = L(M)$.

چون محاسبه‌ی ماشین تورینگ را می‌توان با دنباله‌ای از توصیفات بلافصل نوشت، کاری می‌کنیم که گرامر مورد نظر به گونه‌ای باشد که

$$q_\circ w \Rightarrow^* xq_f y \qquad \text{iff} \qquad q_\circ w \vdash^* xq_f y$$

مشکل در چگونگی ایجاد ارتباط بین اشتقاق $S \Rightarrow^* w$ و $q_\circ w \Rightarrow^* xq_f y$ برای هر $w \in L(G)$ است.

برای حل این مساله، گرامری می‌سازیم که دارای خواص زیر باشد:

۱) به ازای هر $w \in \Sigma^*$ از S بتوان $q_\circ w$ را مشتق کرد.

۲) $q_\circ w \Rightarrow^* xq_f y$ ممکن باشد اگر و فقط اگر $q_\circ w \vdash^* xq_f y$ صادق باشد.

۳) وقتی که رشته‌ی $xq_f y$ با $q_f \in F$ تولید می‌شود، گرامر این رشته را به w اولیه تبدیل کند.

دنباله‌ی کامل اشتقاق‌ها به صورت زیر است:

$$S \Rightarrow^* q_\circ w \Rightarrow^* xq_f y \Rightarrow^* w \qquad (۱۱-۲)$$

قدم سوم در اشتقاق فوق مشکل است: **گرامر چگونه می‌تواند w را به خاطر بیاورد، اگر در قدم دوم تغییر کرده باشد؟**

برای حل این مشکل:

رشته‌ها را کدگذاری می‌کنیم، به گونه‌ای که نسخه‌ی کدگذاری شده، شامل دو کپی از w باشد.

کپی اول ذخیره می‌شود، در حالی که کپی دوم در قدم‌های اشتقاق استفاده می‌شود.

وقتی وارد پیکربندی نهایی می‌شویم، گرامر همه‌چیز را بجز w پاک می‌کند.

برای تولید دو کپی از w و رسیدگی به نماد حالت M (که نهایتاً باید توسط گرامر حذف شود)، متغیرهای V_{ab} و V_{aib} را برای هر $a \in \Sigma \cup \{\square\}$ و $b \in \Gamma$ و تمام iهایی که $q_i \in Q$ معرفی می‌کنیم:

متغیر V_{ab} دو نماد a و b را کدگذاری می‌کند، و

متغیر V_{aib} نمادهای a و b و نیز حالت q_i را کدگذاری می‌کند.

اولین گام در (۱۱-۲) در شکل کدگذاری شده به صورت زیر خواهد بود:

$$\begin{aligned} S &\rightarrow V_{\square\square} S \mid S V_{\square\square} \mid T \\ T &\rightarrow T V_{aa} \mid V_{a^\circ a} \quad \forall a \in \Sigma \end{aligned}$$

قواعد فوق گرامر را قادر می‌کند نسخه‌ی کدگذاری شده‌ی هر رشته‌ی $q_\circ w$ با هر تعداد فضای خالی را در ابتدا و انتها بسازد.

در گام دوم (۱۱-۲)،

به ازای هر گذر حالت $\delta(q_i, c) = (q_j, d, R)$ در M، قاعده‌ی

$$V_{aic} V_{pq} \rightarrow V_{ad} V_{pjq}$$

را به ازای هر $a, p \in \Sigma \cup \{\square\}$ و $q \in \Gamma$ را به گرامر می‌افزاییم.

به ازای هر گذر حالت $\delta(q_i, c) = (q_j, d, L)$ در M، قاعده‌ی

$$V_{pq}V_{aic} \rightarrow V_{pjq}V_{ad}$$

را به ازای هر $a, p \in \Sigma \cup \{\square\}$ و $q \in \Gamma$ را به گرامر می‌افزاییم.
اگر در گام دوم (۱۱-۲) M وارد یک حالت نهایی شود، آن‌گاه باید
گرامر همه چیز را بجز w که در اولین شاخص V ذخیره شده است، حذف کند، بنابراین قاعده‌ی

$$V_{ajb} \rightarrow a$$

را برای هر $q_j \in F$، $a \in \Sigma \cup \{\square\}$ و $b \in \Gamma$ به گرامر می‌افزاییم.
این کار اولین پایانه‌ی رشته را تولید می‌کند که باعث می‌شود

$$\begin{aligned} cV_{ab} &\rightarrow ca \\ V_{ab}c &\rightarrow ac \end{aligned}$$

برای هر $a, c \in \Sigma \cup \{\square\}$ و $b \in \Gamma$ قاعده‌ی

$$\square \rightarrow \lambda$$

به G اضافه می‌شود.
این قاعده در مواردی به کار می‌رود که M به خارج قسمتی از نوار که توسط w اشغال شده است، می‌رود.
برای این کار ابتدا باید از دو قاعده‌ی اول برای تولید $\square \ldots \square q_0 w \square \ldots \square$ استفاده کرد که نمایانگر قسمت استفاده شده‌ی نوار است.
فضاهای خالی در انتها توسط قاعده‌ی آخر حذف می‌شود.

مثال

فرض کنید $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ یک ماشین تورینگ باشد با

$$\begin{aligned} Q &= \{q_0, q_1\} \\ \Gamma &= \{a, b, \square\} \\ \Sigma &= \{a, b\} \\ F &= \{q_1\} \end{aligned}$$

و

$$\begin{aligned} \delta(q_0, a) &= (q_0, q, R) \\ \delta(q_0, \square) &= (q_1, \square, L) \end{aligned}$$

این ماشین زبان $L(a^+)$ را می‌پذیرد.
حال محاسبه‌ی زیر را در نظر بگیرید:

$$q_0 aa \vdash aq_0 a \vdash aaq_0 \square \vdash aq_1 a\square$$

که رشته‌ی aa را می‌پذیرد. برای اشتقاق این رشته با G ابتدا دو قاعده‌ی اول را برای به دست آوردن رشته‌ی شروع مناسب به کار می‌گیریم:

$$S \Rightarrow SV_{\square\square} \Rightarrow TV_{\square\square} \Rightarrow TV_{aa}V_{\square\square} \Rightarrow V_{a0a}V_{aa}V_{\square\square}$$

آخرین شکل جمله‌ای، نقطه‌ی شروع خوبی برای آن قسمت از اشتقاقی است که محاسبات ماشین تورینگ را تقلید می‌کند که حاوی ورودی aa در دنباله‌ی اولین شاخص‌ها و توصیف بلافصل اولیه‌ی $q_\circ aa$ در شاخص‌های باقیمانده است. سپس

$$\begin{aligned} V_{a\circ a}V_{aa} &\rightarrow V_{aa}V_{a\circ a} \\ V_{a\circ a}V_{\square\square} &\rightarrow V_{aa}V_{\square\circ\square} \end{aligned}$$

و

$$V_{aa}V_{\square\circ\square} \rightarrow V_{a\backslash a}V_{\square\square}$$

را اعمال می‌کنیم. قدم‌های بعدی اشتقاق به صورت

$$V_{a\circ a}V_{aa}V_{\square\square} \Rightarrow V_{aa}V_{a\circ a}V_{\square\square} \Rightarrow V_{aa}V_{aa}V_{\square\circ\square} \Rightarrow V_{aa}V_{a\backslash a}V_{\square\square}$$

هستند. دنباله‌ی اولین شاخص‌ها یکسان باقی می‌ماند و همواره ورودی اولیه را به خاطر می‌سپارد. دنباله‌ی سایر شاخص‌ها به صورت

$$\circ aa\square, a\circ a\square, aa\circ\square, a\backslash a\square$$

خواهد بود که معادل با توصیف‌های بلافصل ذکر شده برای رشته‌ی aa است. سرانجام داریم:

$$V_{aa}V_{a\backslash a}V_{\square\square} \Rightarrow V_{aa}aV_{\square\square} \Rightarrow V_{aa}a\square \Rightarrow aa\square \Rightarrow aa$$

▲

به طور خلاصه، برای یافتن گرامر نامقید G معادل با یک ماشین تورینگ M بر اساس تابع گذر δ متعلق به M بر اساس جدول زیر عمل می‌کنیم:

$M = (Q, \Sigma, \Gamma, \delta, q_\circ, \square, F)$	$G = (V, \Sigma, S, P)$	
	$S \rightarrow V_{\square\square}S \mid SV_{\square\square} \mid T$ $T \rightarrow TV_{aa} \mid V_{a\circ a}$	$\forall a \in \Sigma$
$\delta(q_i, c) = (q_j, d, R)$	$V_{aic}V_{fg} \rightarrow V_{ad}V_{fjg}$	$\forall a, f \in \Sigma \cup \{\square\}, \forall g \in \Gamma$
$\delta(q_i, c) = (q_j, d, L)$	$V_{fg}V_{aic} \rightarrow V_{fjg}V_{ad}$	$\forall a, f \in \Sigma \cup \{\square\}, \forall g \in \Gamma$
$q_j \in F$	$V_{ajb} \rightarrow a$	$\forall a \in \Sigma \cup \{\square\}, \forall b \in \Gamma$
	$cV_{ab} \rightarrow ca$	$\forall a, c \in \Sigma \cup \{\square\}, \forall b \in \Gamma$
	$V_{ab}c \rightarrow ac$	$\forall a, c \in \Sigma \cup \{\square\}, \forall b \in \Gamma$
	$\square \rightarrow \lambda$	

قضیه به ازای هر زبان شمارش‌پذیر بازگشتی L گرامر نامقیدی مانند G وجود دارد که $L = L(G)$ باشد.

◄ **اثبات.**

ساختار توضیح داده شده اطمینان می‌دهد که اگر $x \vdash y$، آنگاه $e(x) \Rightarrow^* e(y)$، که در آن $e(x)$ کدگذاری شده‌ی رشته‌ی x بر اساس قرارداد ذکر شده است.

با استفاده از استقرا روی تعداد قدم‌ها نشان می‌دهیم که $e(q_\circ w) \Rightarrow^* e(y)$ اگر و فقط اگر $q_\circ w \vdash^* y$.
همچنین باید نشان دهیم که می‌توان هر پیکربندی اولیه را تولید نمود و اینکه می‌توان w را بازسازی کرد، اگر و فقط اگر M وارد پیکربندی نهایی شود.

۱۱-۳ گرامرهای حساس به متن

بین گرامرهای مستقل از متن و گرامرهای نامقید، گرامرهای متعددی را می‌توان یافت، اما اغلب آنها مفید نیستند.
در بین آنهایی که مهم هستند، گرامرهای حساس به متن توجه زیادی را به خود جلب کرده است.
این گرامرها با یک کلاس از ماشین‌های تورینگ با نام آتوماتای کران‌دار خطی مرتبط هستند.

تعریف **گرامر حساس به متن (context-sensitive grammar).** گرامر $G = (V, T, S, P)$ را در صورتی حساس به متن می‌گویند که تمامی قواعد آن به صورت

$$x \to y$$

باشد که در آن $x, y \in (V \cup T)^+$ و $|x| \leq |y|$ باشد.

این تعریف، غیر انقباضی بودن (non-contracting) گرامرهای حساس به متن را نشان می‌دهد:
یعنی طول شکل‌های جمله‌ای متوالی در یک اشتقاق نمی‌تواند کوتاه‌تر شود.
(توجه کنید که در این نوع گرامر قاعده‌ی λ نداریم.)

◄ **نکته** می‌توان نشان داد که گرامرهای حساس به متن را می‌توان به شکلی تبدیل کرد که در آن همه‌ی قواعد به شکل

$$xAy \to xvy$$

باشد. این معادل است با آن که بگوییم $A \to v$ در صورتی قابل اعمال است که A بین x و y قرارگیرد.

۱-۳-۱۱ زبان‌های حساس به متن و آتوماتای کران‌دار خطی

تعریف **زبان حساس به متن (context-sensitive language).** زبان L را در صورتی حساس به متن گویند که گرامر حساس به متنی مانند G موجود باشد که

$$L = L(G) \cup \{\lambda\} \quad \text{یا} \quad L = L(G)$$

در تعریف این نوع گرامر، $x \to \lambda$ مجاز نیست، بنابراین یک گرامر حساس به متن هرگز نمی‌تواند رشته‌ی تهی را تولید کند.
با افزودن رشته‌ی تهی به تعریف کلی زبان‌های حساس به متن (نه گرامر) می‌توان ادعا کرد که **خانواده‌ی زبان‌های مستقل از متن، زیرمجموعه‌ای از خانواده‌ی زبان‌های حساس به متن است.**

◄ **تذکر** مطابق تعریف دیگری از گرامر حساس به متن، λ تنها می‌تواند در قاعده‌ی $S \to \lambda$ ظاهر شود، به شرطی که در سایر قواعد گرامر، S سمت چپ یا سمت راست هیچ قاعده‌ی دیگری ظاهر نشود (از این تعریف در این درس استفاده نخواهیم کرد).

مثال

زبان $L = \{a^n b^n c^n : n \geq ۱\}$ یک زبان حساس به متن است، زیرا گرامر حساس به متن زیر برای آن وجود دارد:

$$\begin{aligned} S &\to abc \mid aAbc \\ Ab &\to bA \\ Ac &\to Bbcc \\ bB &\to Bb \\ aB &\to aa \mid aaA \end{aligned}$$

برای رشته‌ی $w = a^۲b^۲c^۲$ داریم:

$$S \Rightarrow a\underline{Ab}c \Rightarrow ab\underline{Ac} \Rightarrow ab\underline{B}bcc \Rightarrow \underline{aB}bbcc \Rightarrow aabbcc = a^۲b^۲c^۲$$

و برای رشته‌ی $w = a^۳b^۳c^۳$ داریم:

$$\begin{aligned} S &\Rightarrow aAbc \Rightarrow abAc \Rightarrow abBbcc \Rightarrow aBbbcc \\ &\Rightarrow aaAbbcc \Rightarrow aabAbcc \Rightarrow aabbAcc \\ &\Rightarrow aabbBbccc \Rightarrow aabBbbccc \Rightarrow aaBbbbccc \Rightarrow aaabbbccc \end{aligned}$$

در این مثال، متغیرهای A و B به عنوان «پیام‌رسان» استفاده شده‌اند. یک A که در سمت چپ ایجاد شده است، به سمت راست اولین c حرکت می‌کند و در آنجا یک b و یک c دیگر ایجاد می‌کند. سپس پیام‌رسان B را به سمت چپ عقب می‌فرستد تا a متناظر را ایجاد کند. این فرآیند بسیار شبیه به برنامه‌ای است که یک نفر می‌تواند برای ماشین تورینگ پذیرنده‌ی این زبان بنویسد.

پیدا کردن گرامر حساس به متن برای یک زبان ساده نیست. آسان‌ترین راه برای انجام این کار استفاده از برنامه‌ی ماشین تورینگ معادل و تبدیل آن به گرامر است.

قضیه به ازای هر زبان حساس به متن L بدون λ، یک آتوماتون کران‌دار خطی مانند M هست که $L = L(M)$.

◄ **اثبات.**
اگر L حساس به متن باشد، آنگاه گرامر حساس به متنی برای $L - \{\lambda\}$ وجود دارد.
نشان می‌دهیم که اشتقاق‌های این گرامر می‌تواند توسط یک LBA شبیه‌سازی شود.
این LBA دارای دو شیار خواهد بود: یکی حاوی رشته‌ی w و دیگری حاوی شکل‌های جمله‌ای G.
نکته‌ی کلیدی این است که: هیچ شکل جمله‌ای نمی‌تواند دارای طولی بیشتر از $|w|$ باشد،
نکته‌ی دیگر اینکه: آتوماتای کران‌دار خطی طبق تعریف غیرقطعی است.
پس محاسبات توضیح داده شده در دو قضیه‌ی قبل، می‌تواند با استفاده از فضایی که ابتدا توسط w اشغال شده است، انجام شود؛ یعنی می‌تواند به وسیله‌ی LBA صورت گیرد.

■

قضیه اگر زبان L توسط آتوماتون کران‌دار خطی M پذیرفته شود، آنگاه گرامر حساس به متنی وجود دارد که L را تولید می‌کند.

◄ **اثبات.**
این اثبات مشابه اثبات قضیه‌ی مربوط به زبان‌های شمارش‌پذیر بازگشتی است. در آنجا تمام قواعد تولید غیرانقباضی هستند، بجز قاعده‌ی آخر ($\lambda \rightarrow \square$) که می‌توان آن را حذف کرد (زیرا تنها زمانی استفاده می‌شود که ماشین تورینگ از فضای ورودی اولیه خارج شود که در اینجا پیش نمی‌آید). گرامر به دست آمده با آن ساختار بدون استفاده از این قاعده‌ی غیرضروری غیرانقباضی خواهد بود و بنابراین اثبات کامل می‌شود.

■

۱۱-۳-۲ ارتباط میان زبان‌های بازگشتی و حساس به متن

قضیه هر زبان حساس به متن L بازگشتی است.

◄ **اثبات.**
زبان حساس به متن L را همراه با گرامر حساس به متن مرتبط با آن در نظر می‌گیریم و به یک اشتقاق w نگاه می‌کنیم:

$$S \Rightarrow x_1 \Rightarrow x_2 \Rightarrow \ldots \Rightarrow x_n \Rightarrow w$$

می‌توان فرض کرد که همه‌ی شکل‌های جمله‌ای در یک اشتقاق متفاوت هستند، یعنی $\forall i \neq j \; x_i \neq x_j$.
تعداد گام‌های اشتقاق، تابع کران‌داری از $|w|$ است.
چون G غیرانقباضی است، $|x_j| \leq |x_{j+1}|$ است؛

تنها مورد لازم این است که یک m وابسته به G و w وجود دارد که $|x_j| \leq |x_{j+m}|$ برای هر j، $m = m(|w|)$ تابع کران‌داری از $|V \cup T|$ و $|w|$ است.
این درست است، زیرا متناهی بودن $|V \cup T|$ به معنای این است که فقط تعدادی متناهی رشته با طول مشخص داریم. بنابراین طول یک اشتقاق برای رشته‌ی $w \in L$، حداکثر $|w|m(|w|)$ است.
این در واقع یک الگوریتم عضویت برای L به ما می‌دهد:
تمامی اشتقاق‌ها تا طول $|w|m(|w|)$ را ملاحظه می‌کنیم.
چون مجموعه‌ی قواعد G متناهی است، تعدادی متناهی از این اشتقاق‌ها وجود دارد.
اگر یکی از این اشتقاق‌ها w را بدهد، آن‌گاه $w \in L$ وگرنه $w \notin L$

قضیه زبان بازگشتی‌ای وجود دارد که حساس به متن نیست.

◄ **اثبات.**
مجموعه‌ی همه‌ی گرامرهای حساس به متن را روی $T = \{a, b\}$ در نظر می‌گیریم.
قرارداد می‌کنیم که مجموعه‌ی متغیرها به صورت $V = \{V_۰, V_۱, V_۲, \ldots\}$ باشد.
هر گرامر حساس به متن با مجموعه‌ی قواعدش به طور کامل مشخص می‌شود که این مجموعه را می‌توان با رشته‌ی زیر نمایش داد:

$$x_۱ \to y_۱; x_۲ \to y_۲; \ldots; x_m \to y_m$$

بر روی این رشته همریختی زیر را اعمال می‌کنیم:

$$\begin{aligned} h(a) &= ۰۱۰ \\ h(b) &= ۰۱^۲۰ \\ h(\to) &= ۰۱^۳۰ \\ h(;) &= ۰۱^۴۰ \\ h(V_i) &= ۰۱^{i+۵}۰ \end{aligned}$$

بنابراین هر گرامر حساس به متن می‌تواند به طور یکتا با رشته‌ای از $L((۰۱^+۰)^*)$ بازنمایی شود.
به ازای هر یک از این رشته‌ها، حداکثر یک گرامر حساس به متن وجود دارد.
فرض می‌کنیم یک مرتب‌سازی سره بر روی $\{۰, ۱\}^+$ معرفی نماییم تا بتوان این رشته‌ها را به صورت

$$w_۱, w_۲, w_۳, \ldots$$

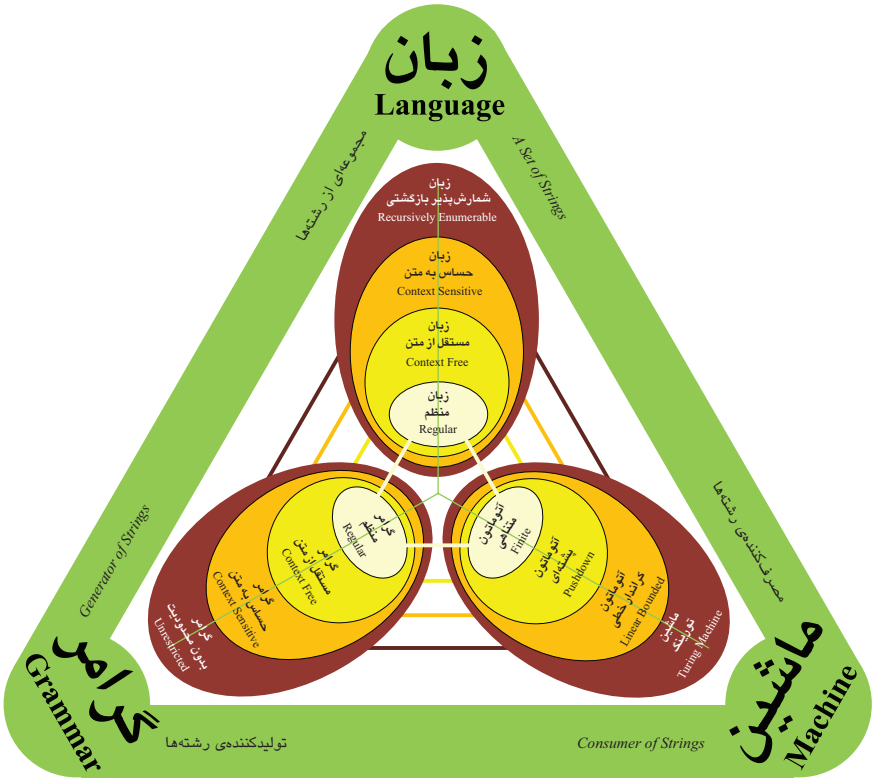
بنویسیم. رشته‌ای مانند w_j ممکن است که یک گرامر حساس به متن را تعریف نکند. اگر این رشته گرامر حساس به متن G را تعریف کند، آن را G_j می‌نامیم و زبان L را به صورت زیر تعریف می‌کنیم:

$$L = \{w_j : w_j \text{ defines context-sensitive grammar } G_j \text{ and } w \notin L(G_j)\}$$

زبان L خوش‌تعریف و بازگشتی است. برای اثبات این موضوع یک الگوریتم عضویت می‌سازیم:
اگر w_i گرامر G_i را تعریف نکند، آنگاه $w_i \notin L$.
اگر این رشته گرامری را تعریف نکند، آنگاه $L(G_i)$ بازگشتی است و می‌توانیم از الگوریتم عضویت قضیه‌ی قبلی استفاده کنیم تا دریابیم که آیا $w_i \in L(G)$ هست یا خیر. اگر $w_i \in L(G)$ نباشد، آنگاه $w_i \in L$ است.
اما L حساس به متن نیست، زیرا اگر این‌گونه بود، آنگاه w_j وجود داشت به طوری که $L = L(G)$ باشد. در این صورت می‌توانیم بپرسیم که آیا $w_j \in L(G_j)$ هست یا خیر. اگر فرض کنیم که $w_j \in L(G_j)$ است، آنگاه طبق تعریف $w_j \notin L$، اما $L = L(G)$ است که یک تناقض را نشان می‌دهد.
برعکس اگر فرض کنیم که $w_j \notin L(G_j)$ باشد، آنگاه طبق تعریف $w_j \in L$ است و تناقض دیگری پیش می‌آید.
بنابراین L حساس به متن نیست.

نتیجه LBA ضعیف‌تر از ماشین تورینگ و قوی‌تر از PDA است.

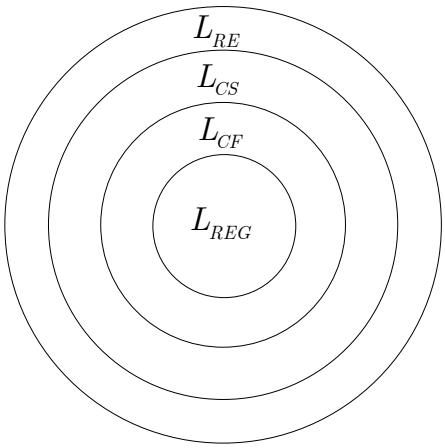
۴-۱۱ خانواده‌ی زبان‌های نوع صفر: زبان، گرامر و ماشین



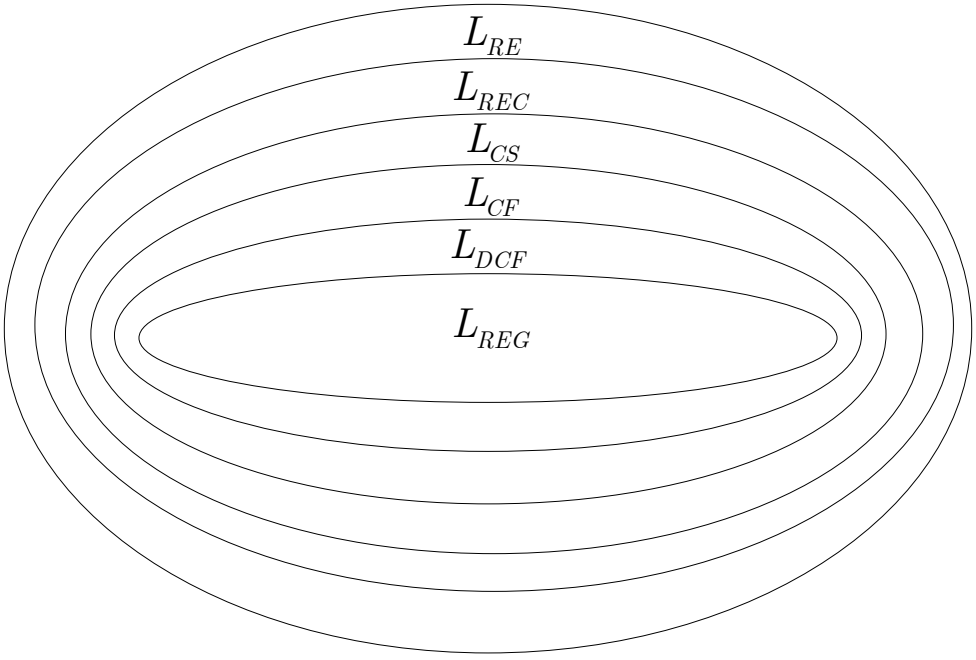
۵-۱۱ سلسله مراتب چامسکی

زبان‌های شمارش‌پذیر بازگشتی	L_{RE}	نوع صفر	$\mathcal{L}_0$
زبان‌های حساس به متن	L_{CS}	نوع اول	$\mathcal{L}_1$
زبان‌های مستقل از متن	L_{CF}	نوع دوم	$\mathcal{L}_2$
زبان‌های منظم	L_{REG}	نوع سوم	$\mathcal{L}_3$

$$\mathcal{L}_{i+1} \subset \mathcal{L}_i \quad i = 0, 1, 2$$



زبان‌های مستقل از متن قطعی	L_{DCF}
زبان‌های مستقل از متن خطی	L_{LIN}
زبان‌های بازگشتی	L_{REC}



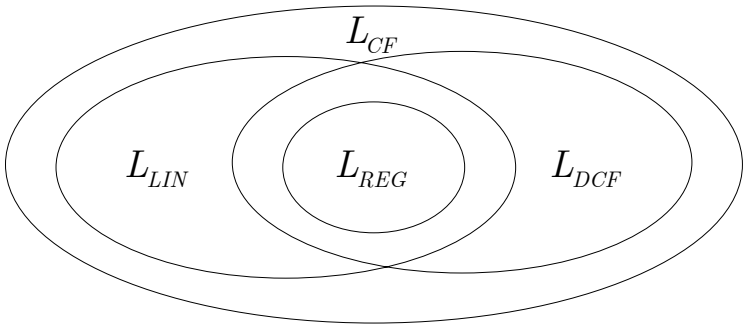
مثال

زبان مستقل از متن $L = \{w : n_a(w) = n_b(w)\}$ قطعی است، اما خطی نیست.

مثال

زبان مستقل از متن $L = \{a^n b^n\} \cup \{a^n b^{2n}\}$ خطی است، اما قطعی نیست.

نتیجه



مسائل حل نشده

ارتباط آتوماتای کران‌دار خطی قطعی DLBA با سایر آتوماتا
و اینکه عدم قطعیت در اینجا چه نقشی دارد؟
نمی‌دانیم که آیا خانواده‌ی زبان‌های پذیرفته شده توسط یک DLBA زیرمجموعه‌ی محضی از زبان‌های حساس به متن است یا خیر؟

زبان	گرامر	ماشین	سطح چامسکی	ماشین قطعی و غیر قطعی
زبان‌های شمارش‌پذیر بازگشتی	گرامرهای نامقید	ماشین‌های تورینگ	نوع صفر	هم‌ارز
زبان‌های حساس به متن	گرامرهای حساس به متن	آتوماتای کران‌دار خطی	نوع اول	نمی‌دانیم
زبان‌های مستقل از متن	گرامرهای مستقل از متن	آتوماتای پشته‌ای	نوع دوم	ناهم‌ارز
زبان‌های منظم	گرامرهای منظم	آتوماتای متناهی	نوع سوم	هم‌ارز

مراجع

[1] P. Linz, **An Introduction to Formal Languages and Automata**, 5th Ed., Jones and Bartletts, 2012.

[2] M. Sipser, **Introduction to the Theory of Computation**, 3rd Ed., Cengage Learning, 2013.