به نام خدا

# نظریهٔ زبان‌ها و ماشین‌ها

آرش شفیعی

# سلسله‌مراتب زبان‌ها

## سلسله‌مراتب زبان‌ها

- حال که با چندین دسته از زبان‌ها و ماشین‌ها آشنا شدیم به دسته‌بندی آنها می‌پردازیم.
- ماشین‌های تورینگ دسته‌ای از زبان‌ها را شناسایی می‌کنند که به آنها زبان‌های شمارش‌پذیر بازگشتی [1] می‌گوییم و ماشین‌های کراندار خطی زبان‌های حساس به متن [2] را شناسایی می‌کنند.
- بدین ترتیب چهار دسته از زبان‌ها را بررسی کردیم: زبان‌های منظم، مستقل از متن، حساس به متن، و شمارش‌پذیر بازگشتی.
- ماشین‌هایی که این چهار دسته از زبان‌ها را شناسایی می‌کنند به ترتیب ماشین‌های حالات متناهی، پشته‌ای، کراندار خطی، و تورینگ هستند.
- همچنین خواهیم دید که تعداد زبان‌ها از تعداد ماشین‌های تورینگ بیشتر است، پس باید زبان‌هایی وجود داشته باشند که هیچ ماشین تورینگی برای آنها وجود ندارد.

---
[1] recursively enumerable language
[2] context-sensitive language

# زبان‌های شمارش‌پذیر بازگشتی

- زبان L یک زبان شمارش‌پذیر بازگشتی [1] یا تشخیص پذیر [2] نامیده می‌شود، اگر یک ماشین تورینگ وجود داشته باشد که آن را بپذیرد.

- بنابراین اگر ماشین تورینگ M وجود داشته باشد به طوری که برای هر $w \in L$ داشته باشیم $q_0 w \overset{*}{\vdash}_M x_1 q_f x_2$ به طوری که $q_f$ یک حالت پایانی باشد، آنگاه زبان L را یک زبان شمارش‌پذیر بازگشتی می‌نامیم.

- برای رشته‌هایی که توسط ماشین M پذیرفته نمی‌شوند، ماشین ممکن است به حالت غیرپایانی برود یا در یک حلقهٔ بی‌پایان بیافتد.

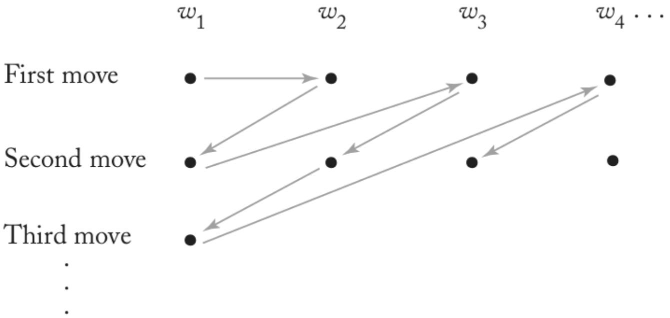
[1] recursively enumerable language
[2] recognizable or Turing-recongnizable

# زبان‌های شمارش‌پذیر بازگشتی

- زبان‌های شمارش‌پذیر بازگشتی به این دلیل شمارش‌پذیر خوانده می‌شوند که روشی برای شمارش جمله‌های آنها وجود دارد.
- برای شمارش جمله‌های آنها، همهٔ رشته‌های الفبا را مرتب می‌کنیم. به ترتیب رشتهٔ اول را به ماشین تورینگی که برای آن زبان طراحی شده برای اجرا می‌دهیم، ولی از آنجایی که ماشین ممکن است بر روی رشته هیچگاه توقف نکند، بنابراین فقط یک قدم را بر روی رشتهٔ اول اجرا می‌کنیم و سپس به رشتهٔ دوم می‌رویم، یک قدم را اجرا می‌کنیم و الی آخر. این روند را ادامه می‌دهیم و به ترتیب هر رشته‌ای را که توسط ماشین پذیرفته شد، شمارش می‌کنیم.

# زبان‌های شمارش‌پذیر بازگشتی

- زبان‌های شمارش‌پذیر بازگشتی به این دلیل شمارش‌پذیر خوانده می‌شوند که روشی برای شمارش جمله‌های آنها وجود دارد.



# زبان‌های شمارش‌پذیر بازگشتی

- زبان‌های شمارش‌پذیر بازگشتی به این دلیل بازگشتی خوانده می‌شوند که در گذشته مباحث نظریهٔ محاسبات در حوزه‌ای به نام نظریهٔ بازگشت [1] مطالعه می‌شدند و تعاریف بازگشتی در تعریف محاسبات نقش مهمی داشتند.

- در حال حاضر نظریهٔ محاسبه‌پذیری [2] و نظریهٔ بازگشت به یک معنی به کار می‌روند.

- نظریهٔ محاسبات شاخه‌ای از منطق و علوم کامپیوتر است که به بررسی محاسبه‌پذیری توابع می‌پردازد.
همچنین توابع محاسبه‌پذیر را در نظریهٔ محاسبات از حیث زمان و حافظهٔ مورد نیاز آنها دسته‌بندی می‌کنیم.

[1] recursion theory
[2] computability theory

# زبان‌های بازگشتی

- زبان $L$ بر روی الفبای $\Sigma$ یک زبان بازگشتی [1] یا تصمیم‌پذیر [2] نامیده می‌شود، اگر یک ماشین تورینگ وجود داشته باشد که $L$ را بپذیرد و برای هر رشتهٔ $w \in \Sigma^+$ توقف کند.
- به عبارت دیگر یک زبان بازگشتی است، اگر و تنها اگر برای آن الگوریتمی وجود داشته باشد که تعیین کند آیا هر رشتهٔ داده شده عضو آن زبان است یا خیر.

[1] recursive language
[2] decidable or Turing-decidable

# زبان‌های شمارش‌پذیر بازگشتی

- زبان‌های بازگشتی نیز شمارش‌پذیر هستند.
- برای شمارش جمله‌های آنها، همهٔ رشته‌های الفبا را مرتب می‌کنیم. به ترتیب همهٔ رشته‌ها را به ماشین تورینگی که برای آن زبان طراحی شده می‌دهیم. اگر رشته پذیرفته شد، آن را شمارش می‌کنیم و اگر پذیرفته نشد، به رشتهٔ بعدی می‌رویم و الی آخر.

# تشخیص‌پذیری و تصمیم‌پذیری

- برای مثال مسألهٔ دهم هیلبرت می‌پرسد: آیا الگوریتمی وجود دارد که بتواند به ازای هر چندجمله‌ای تعیین کند آیا آن چندجمله‌ای ریشه‌های صحیح دارد یا خیر؟
- نشان داده شده است که هیچ الگوریتم تصمیم‌پذیری (ماشین تورینگی که روی همهٔ ورودی‌ها متوقف شود) برای این مسأله وجود ندارد.
- به عبارت دیگر زبان (مجموعهٔ) همهٔ چندجمله‌ای ها با ریشهٔ صحیح، یک زبان تشخیص‌پذیر (شمارش‌پذیر بازگشتی) است و تصمیم‌پذیر (بازگشتی) نیست.

# تشخیص‌پذیری و تصمیم‌پذیری

- اگر زبان D را چنین تعریف کنیم:
$D = \{p :$ است p یک چندجمله‌ای با ریشه‌های صحیح$\}$
مسأله دهم هیلبرت می‌پرسد آیا D تصمیم‌پذیر است؟

- زبان D تصمیم‌پذیر نیست، ولی تشخیص‌پذیر است.

- به عبارت دیگر می‌توانیم به ازای یک چندجمله‌ای داده شدهٔ p ، همهٔ اعداد صحیح را بررسی کنیم. اگر به ازای تعدادی عدد صحیح ریشه‌ای برای چندجمله‌ای p پیدا شد، p پذیرفته می‌شود، ولی اگر ریشه‌ای برای p پیدا نشد، الگوریتم در حلقهٔ بی‌پایان می‌افتد و نمی‌توان تصمیم گرفت که ریشهٔ صحیح وجود ندارد.

# زبان‌های غیر بازگشتی

- حال می‌خواهیم نشان دهیم زبان‌هایی وجود دارند که تشخیص پذیر نیستند.
- قضیه: اگر $S$ یک مجموعهٔ نامتناهی شمارا (شمارش‌پذیر) باشد، آنگاه مجموعهٔ توانی $2^S$ ناشمارا (شمارش‌ناپذیر) است.

# زبان‌های غیر بازگشتی

- قضیه: اگر S یک مجموعهٔ نامتناهی شمارا (شمارش‌پذیر) باشد، آنگاه مجموعهٔ توانی ۲$^{S}$ ناشمارا (شمارش‌ناپذیر) است.

برهان خلف

- اثبات: فرض کنید مجموعهٔ $t_i$ یکی از اعضای مجموعهٔ توانی ۲$^{S}$ باشد. می‌نویسیم ...$t_i =$ ۱۱۰ اگر $t_i$ اولین عضو و دومین عضو S را در شامل شود اما سومین عضو S را شامل نشود، و الی آخر.

- حال فرض کنید مجموعهٔ همهٔ زیرمجموعه‌های S یعنی ۲$^{S}$ شمارا باشد. آنگاه می‌توانیم با استفاده از جدولی مشابه جدول زیر اعضای آن را بشماریم.

| | | | | | | |
|---|---|---|---|---|---|---|
| $t_1$ | 1 | 0 | 0 | 0 | 0 | ... |
| $t_2$ | 1 | 1 | 0 | 0 | 0 | ... |
| $t_3$ | 1 | 1 | 0 | 1 | 0 | ... |
| $t_4$ | 1 | 1 | 0 | 0 | 1 | ... |
| ⋮ | | | | | ⋱ | |

# زبان‌های غیر بازگشتی

- حال رشته‌ای که در جدول زیر روی قطر است را در نظر بگیرید. همهٔ صفرهای آن را به یک و همهٔ یک‌های آن را به صفر تبدیل کنید.
- اگر $2^S$ شمارا باشد، آنگاه باید متمم رشته‌ای که روی قطر قرار دارد نیز در این جدول شمارش شود و یکی از $t_i$ ها باشد. اما این رشته نمی‌تواند هیچ‌کدام از $t_i$ ها باشد، زیرا نه $t_1$ است (چون نماد اول آن با نماد اول $t_1$ متفاوت است) و نه $t_2$ است (چون نماد دوم آن با نماد دوم $t_2$ متفاوت است) و الی آخر.
- تناقض به وجود آمده نشان می‌دهد که فرض اولیه مبنی بر شمارا بودن $2^S$ نادرست بوده است.

| | | | | | | |
|---|---|---|---|---|---|---|
| $t_1$ | 1 | 0 | 0 | 0 | 0 | ... |
| $t_2$ | 1 | 1 | 0 | 0 | 0 | ... |
| $t_3$ | 1 | 1 | 0 | 1 | 0 | ... |
| $t_4$ | 1 | 1 | 0 | 0 | 1 | ... |
| ⋮ | | | | ⋱ | | |

# زبان‌های غیر بازگشتی

- قضیه: بر روی الفبای $\Sigma$ زبان‌هایی وجود دارند که شمارش‌پذیر بازگشتی (تشخیص‌پذیر) نیستند.
- اثبات: یک زبان بر روی الفبای $\Sigma$ یک زیرمجموعه است از مجموعهٔ $\Sigma^*$ . بنابراین مجموعهٔ همهٔ زبان‌ها برابر است با مجموعهٔ توانی $2^{\Sigma^*}$ . این مجموعه طبق قضیه‌ای که بیان شد، ناشمارا است. پس مجموعهٔ همهٔ زبان‌ها بر روی یک الفبای معین ناشمارا است.
از طرف دیگر اثبات کردیم که مجموعهٔ همهٔ ماشین‌های تورینگ شمارا است. پس تعداد زبان‌ها از تعداد ماشین‌های تورینگ بیشتر است و بنابراین زبان‌هایی وجود دارند که برای آنها هیچ ماشین تورینگی وجود ندارد.
- اکنون باید یک زبان پیدا کنیم که شمارش‌پذیر بازگشتی نباشد.

# زبان‌های غیر بازگشتی

- قضیه: یک زبان تشخیص‌پذیر (شمارش‌پذیر بازگشتی) وجود دارد که متمم آن تشخیص‌پذیر نیست.

- اثبات: مجموعهٔ همهٔ ماشین‌های تورینگ بر روی الفبای $\Sigma = \{a\}$ را در نظر بگیرید.

- این مجموعه شمارا است (قبلا روشی برای شمارش همهٔ ماشین‌های تورینگ بیان کردیم). پس مجموعهٔ شمارای $\{M_۱, M_۲, \cdots\}$ را در نظر می‌گیریم. برای هر یک از این ماشین‌های تورینگ $M_i$ یک زبان تشخیص‌پذیر (شمارش‌پذیر بازگشتی) $L(M_i)$ وجود دارد. همچنین برای هر زبان تشخیص‌پذیر بر روی $\Sigma$ یک ماشین تورینگ وجود دارد.

- حال زبان $L = \{a^i : a^i \in L(M_i)\}$ را در نظر بگیرید.
به ازای هر $i \geq ۱$، رشتهٔ $a^i$ عضو زبان $L$ است اگر و فقط اگر $a^i \in L(M_i)$ .

# زبان‌های غیر بازگشتی

- ابتدا باید نشان دهیم که $L$ یک زبان تشخیص‌پذیر است.
- می‌توانیم ماشین تورینگ جهانی $M_u$ را طراحی کنیم که همهٔ ماشین‌های تورینگ را شمارش می‌کند. ماشین $M_u$، رشتهٔ $a^i$ را به ماشین $M_i$ به عنوان ورودی می‌دهد. اگر رشته توسط ماشین $M_i$ پذیرفته شد، آنگاه توسط ماشین $M_u$ نیز پذیرفته می‌شود و بنابراین رشته عضو زبان $L$ است. پس برای زبان $L$ ماشین $M_u$ را پیدا کردیم، و در نتیجه زبان $L$ یک زبان تشخیص‌پذیر است.

# زبان‌های غیر بازگشتی

- حال متمم زبان L را در نظر بگیرید: $\overline{L} = \{a^i : a^i \notin L(M_i)\}$
- نشان می‌دهیم که زبان $\overline{L}$ شمارش‌پذیر بازگشتی (تشخیص‌پذیر) نیست.
- فرض کنید $\overline{L}$ تشخیص پذیر باشد. پس باید برای آن ماشین تورینگ $M_k$ وجود داشته به طوری که $\overline{L} = L(M_k)$ .
- حال رشتهٔ $a^k$ را در نظر بگیرید. این رشته عضو زبان L است یا زبان $\overline{L}$ ؟
- فرض کنید رشتهٔ $a^k \in \overline{L}$ آنگاه داریم $a^k \in L(M_k)$ زیرا فرض کردیم $\overline{L} = L(M_k)$ . از طرف دیگر طبق تعریف $\overline{L}$، رشتهٔ $a^k$ عضو زبان $\overline{L}$ است اگر $a^k \notin L(M_k)$. پس به تناقض می‌رسیم و بنابراین $a^k \notin \overline{L}$ .
- حال فرض کنید $a^k \in L$ پس $a^k \notin \overline{L}$ پس $a^k \notin L(M_k)$. اما در این صورت طبق تعریف $\overline{L}$، اگر $a^k \notin L(M_k)$ آنگاه داریم $a^k \in \overline{L}$ . پس به تناقض رسیدیم و فرض اولیه نادرست بوده و برای تشخیص $\overline{L}$ هیچ ماشینی وجود ندارد و بنابراین $\overline{L}$ نمی‌تواند تشخیص‌پذیر باشد.

# زبان‌های بازگشتی

- حال می‌خواهیم نشان دهیم زبان‌های بازگشتی شمارش پذیر (تشخیص‌پذیر) وجود دارند که بازگشتی (تصمیم‌پذیر) نیستند.

# زبان‌های بازگشتی

- قضیه: اگر L یک زبان تصمیم‌پذیر باشد، متمم آن نیز تصمیم‌پذیر است.
- اگر L تصمیم‌پذیر باشد پس به ازای هر رشته با استفاده از یک ماشین تورینگ می‌توان تعیین کرد که آیا رشته عضو L است یا خیر. همین ماشین تورینگ را برای متمم L نیز می‌توانیم به کار ببریم. اگر رشته‌ای توسط این ماشین پذیرفته شد، رشته عضو $\bar{L}$ نیست، در غیر اینصورت رشته عضو $\bar{L}$ است.

# زبان‌های بازگشتی

- قضیه: زبان‌های تشخیص‌پذیر وجود دارند که تصمیم‌پذیر نیستند. به عبارت دیگر خانوادهٔ زبان‌های تصمیم‌پذیر زیرمجموعهٔ خانوادهٔ زبان‌های تشخیص‌پذیر است.
- اثبات: زبان $L$ که قبلا به صورت $L = \{a^i : a^i \in L(M_i)\}$ توصیف کردیم در نظر بگیرید. ثابت کردیم این زبان تشخیص‌پذیر است. حال فرض کنید این زبان تصمیم‌پذیر باشد. پس متمم آن نیز باید تصمیم‌پذیر باشد. ولی ثابت کردیم متمم آن غیربازگشتی است. پس زبانی وجود دارد که تشخیص‌پذیر است ولی تصمیم‌پذیر نیست.

# زبان‌های بازگشتی

- قضیه: اگر $L$ و متمم آن $\overline{L}$ هر دو تشخیص‌پذیر باشند، آنگاه هر دو تصمیم‌پذیرند.
- اثبات: فرض کنیم ماشین تورینگ $M$ زبان $L$ و ماشین تورینگ $\widehat{M}$ زبان $\overline{L}$ را می‌پذیرد. به ازای هر رشتهٔ زبان $L$ می‌توانیم یک ماشین تورینگ طراحی کنیم که رشته را به ماشین $M$ و $\widehat{M}$ می‌دهد. اگر $M$ رشته را پذیرفت، رشته پذیرفته می‌شود و اگر $\widehat{M}$ رشته را پذیرفت، رشته رد می‌شود. پس $L$ یک زبان تصمیم‌پذیر است. به همین ترتیب $\overline{L}$ نیز تصمیم پذیر است.

# گرامرهای بدون محدودیت

- گرامر $G = (V, T, S, P)$ را یک گرامر بدون محدودیت [1] می‌گوییم اگر همهٔ قوانین تولید آن به صورت $u \to v$ باشد، به طوری که $u \in (V \cup T)^+$ و $v \in (V \cup T)^*$ .
- در یک گرامر بدون محدودیت، هیچ محدودیتی برای شکل قوانین در نظر گرفته نشده است. هر تعداد متغیر و نماد پایانی می‌توانند در سمت چپ و راست یک قانون قرار داشته باشند و تنها محدودیت این است که سمت چپ قانون تهی نباشد.

[1] unrestricted grammar

# گرامرهای بدون محدودیت

- هر زبان تولید شده توسط یک گرامر بدون محدودیت یک زبان شمارش‌پذیر بازگشتی (تشخیص‌پذیر) است.
- همچنین برای هر زبان تشخیص‌پذیر $L$ ، یک گرامر بدون محدودیت $G$ وجود دارد، به طوری که $L = L(G)$ .
- روشی برای شبیه‌سازی یک فرایند اشتقاق توسط یک ماشین تورینگ وجود دارد که در اینجا به آن نمی‌پردازیم.

# گرامرهای بدون محدودیت

- یک گرامر بدون محدودیت برای زبان $L = \{a^{2^k} : k \geq 0\}$ بنویسید.

# گرامرهای بدون محدودیت

- یک گرامر بدون محدودیت برای زبان $L = \{a^{2^k} : k \geq 0\}$ بنویسید.
- می‌توانیم جملات $L$ را به صورت بازگشتی تعریف کنیم. می‌دانیم $a \in L$ و به ازای هر $n \geq 1$ اگر $a^n \in L$ آنگاه $a^{2n} \in L$.
- پس باید گرامری بنویسیم که بتواند در هر مرحله تعداد نمادهای $a$ را دو برابر کند.

# گرامرهای بدون محدودیت

- پس باید گرامری بنویسیم که بتواند در هر مرحله تعداد نمادهای a را دو برابر کند.
- برای این کار از یک متغیر D استفاده می‌کنیم که می‌تواند هر نماد a در یک صورت جمله‌ای را با دو نماد a جایگزین کند.
- فرض کنید در یک صورت جمله‌ای تعدادی a تولید شده است، آنگاه متغیر D از چپ شروع می‌کند و هر a را با دو a جایگزین می‌کند.
- به عبارت دیگر ما به دنبال قانونی می‌گردیم که اشتقاق زیر را تولید کند:
  $\mathrm{Daaaa} \Rightarrow \mathrm{aaDaaa} \Rightarrow \mathrm{aaaaDaa} \Rightarrow \mathrm{aaaaaaDa} \Rightarrow \mathrm{aaaaaaaaD}$
- این قانون را به صورت زیر می‌نویسیم:
  $\mathrm{Da} \rightarrow \mathrm{aaD}$

# گرامرهای بدون محدودیت

- حال به دنبال یک قانون اولیه می‌گردیم که برای ما یک نماد a تولید کند و همچنین بتواند در هر بار یک متغیر D برای دو برابر کردن تعداد a ها تولید کند. این قوانین را بدین صورت می‌نویسیم:
$$S \rightarrow LaR$$
$$L \rightarrow LD$$
- سپس به دنبال قانونی می‌گردیم که متغیر D را پس از این که تعداد نمادهای a را دو برابر کرد، حذف کند. این قانون را بدین صورت می‌نویسیم:
$$DR \rightarrow R$$
- در پایان باید متغیرهای L و R را حذف کنیم:
$$L \rightarrow \lambda$$
$$R \rightarrow \lambda$$

# گرامرهای بدون محدودیت

- برای به دست آوردن جملهٔ aaaa فرایند اشتقاقی به صورت زیر داریم:
$$S \Rightarrow LaR \Rightarrow LDaR \Rightarrow LaaDR \Rightarrow LaaR \Rightarrow LDaaR \Rightarrow LaaDaR \Rightarrow LaaaaDR \Rightarrow LaaaaR \Rightarrow aaaaR \Rightarrow aaaa$$

# گرامرهای بدون محدودیت

- یک گرامر بدون محدودیت برای زبان $L = \{ww : w \in \{a, b\}^*\}$ بنویسید.

# گرامرهای بدون محدودیت

- یک گرامر بدون محدودیت برای زبان $L = \{ww : w \in \{a, b\}^*\}$ بنویسید.
- با تولید رشته‌های $ww^R$ آشنا هستیم، پس ابتدا گرامری می‌نویسیم که رشته‌هایی را تولید کند که نیمهٔ اول آنها $w$ و نیمهٔ دوم آنها معکوس $w$ باشد، با این تفاوت که در نیمهٔ دوم، به جای نمادهای پایانی، متغیر داریم.
متغیر $A$ به جای نماد $a$ و متغیر $B$ به جای نماد $b$ . بنابراین ابتدا یک صورت جمله‌ای به شکل $wW^R$ تولید می‌کنیم، به طوری که $W$ فقط از متغیرها تشکیل شده است.
- برای این کار قوانینی به صورت زیر می‌نویسیم:
$S \rightarrow T]$
$T \rightarrow aTA|bTB|[R$

# گرامرهای بدون محدودیت

- پس در یک دنباله از اشتقاق‌ها خواهیم داشت: $S \overset{*}{\Rightarrow} w[RW^R]$ .

- سپس باید صورت جمله‌ای $[RW^R]$ را تبدیل به $w$ کنیم.

- در هر بار با استفاده از یک متغیر به پایان رشتهٔ $W^R$ می‌رویم. آخرین متغیر این رشته را تشخیص می‌دهیم. اگر آخرین متغیر در رشته $A$ بود، آن را به $L_a$ تبدیل می‌کنیم و اگر $B$ بود، آن را به $L_b$ تبدیل می‌کنیم. سپس متغیر $L_a$ یا $L_b$ را به ابتدای $W^R$ می‌آوریم و آن را از براکت خارج کرده و به نماد پایانی تبدیل می‌کنیم.

- برای مثال برای به دست آوردن جملهٔ aabaaaba فرایند اشتقاق زیر را داریم:

$$S \Rightarrow T] \overset{*}{\Rightarrow} aabaTABAA] \Rightarrow aaba[RABAA]$$
$$\overset{*}{\Rightarrow} aaba[ABAAR] \Rightarrow aaba[ABAL_a] \overset{*}{\Rightarrow} aaba[L_aABA] \Rightarrow aabaa[RABA]$$
$$\overset{*}{\Rightarrow} aabaa[ABAR] \Rightarrow aabaa[ABL_a] \overset{*}{\Rightarrow} aabaa[L_aAB] \Rightarrow aabaaa[RAB]$$
$$\overset{*}{\Rightarrow} aabaaaba[R] \Rightarrow aabaaaba$$

- این گرامر را می‌توانیم به صورت زیر بنویسیم:
- $V = \{S, T, A, B, R, L_a, L_b, [, ]\}$

$$S \to T] \quad , \quad T \to aTA|bTB|[R$$
$$RA \to AR \quad , \quad RB \to BR$$
$$AR] \to L_a] \quad , \quad BR] \to L_b]$$
$$AL_a \to L_aA \quad , \quad AL_b \to L_bA \quad , \quad BL_a \to L_aB \quad , \quad BL_b \to L_bB$$
$$[L_a \to a[R \quad , \quad [L_b \to b[R$$
$$[R] \to \lambda$$

## گرامرهای حساس به متن

- گرامر $G = (V, T, S, P)$ حساس به متن گفته می‌شود، اگر همهٔ قوانین آن به صورت $x \to y$ باشند، به طوری که $x, y \in (V \cup T)^+$ و $|x| \leq |y|$ .

- این گرامرها حساس به متن نامیده می‌شوند، زیرا اثبات شده است که چنین گرامرهایی را می‌توان به یک فرم نرمال تبدیل کرد به طوری که همهٔ قوانین آن به شکل $xAy \to xvy$ باشند. بنابراین این قانون مانند قانون تولید $A \to v$ است با این تفاوت که متن (یعنی نمادهای اطراف $A$ در یک صورت جمله‌ای در فرایند اشتقاق) در اعمال قانون مؤثر است، بر خلاف گرامرهای مستقل از متن که بدون در نظر گرفتن متن، قوانین را اعمال می‌کند.

# زبان‌های حساس به متن

- زبان L حساس به متن گفته می‌شود اگر یک گرامر حساس به متن G برای آن وجود داشته باشد، به طوری که $L = L(G)$.

- از آنجایی که گرامر حساس به متن اجازهٔ تولید رشتهٔ تهی را نمی‌دهد، یک استثنا در نظر می‌گیریم برای حالتی که در زبان مورد نظر L نیاز به تولید رشتهٔ تهی داشته باشیم، بنابراین $L = L(G) \cup \{\lambda\}$.

# زبان‌های حساس به متن

- زبان $L = \{a^n b^n c^n : n \geq ۱\}$ یک زبان حساس به متن است. برای آن یک گرامر حساس به متن بنویسید.

## زبان‌های حساس به متن

- زبان $L = \{a^n b^n c^n : n \geq ۱\}$ یک زبان حساس به متن است. برای آن یک گرامر حساس به متن بنویسید.

- این گرامر را به دو روش می‌نویسیم.

- در روش اول ابتدا یک صورت جمله‌ای به شکل $a^n(BC)^n$ تولید می‌کنیم. برای این کار از قوانین تولید $S \to aSBC|aBC$ استفاده می‌کنیم.

- سپس همهٔ متغیرهای $B$ را به قبل از متغیرهای $C$ انتقال می‌دهیم. برای این کار از قانون زیر استفاده می‌کنیم:
$CB \to BC$

- در پایان باید همهٔ متغیرها را به نمادهای پایانی تبدیل کنیم. برای این کار از قوانین زیر استفاده می‌کنیم:
$aB \to ab \;\;,\;\; bB \to bb \;\;,\;\; bC \to bc \;\;,\;\; cC \to cc$

- دقت کنید که در اینجا از قوانین $B \to b$ و $C \to c$ استفاده نمی‌کنیم، زیرا می‌خواهیم در مرحلهٔ اول همهٔ متغیرهای $B$ را انتقال دهیم و سپس در مرحلهٔ بعد همهٔ متغیرها را به نمادهای پایانی تبدیل کنیم.

# زبان‌های حساس به متن

- برای به دست آوردن رشتهٔ aaabbbccc با استفاده از این گرامر فرایند اشتقاق زیر را داریم:

$$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC$$
$$\Rightarrow aaaBBCCBC \Rightarrow aaaBBCBCC \Rightarrow aaaBBBCCC$$
$$\Rightarrow aaabBBCCC \overset{*}{\Rightarrow} aaabbbCCC \Rightarrow aaabbbcCC \overset{*}{\Rightarrow} aaabbbccc$$

# زبان‌های حساس به متن

- در روش دوم از دو متغیر A و B استفاده می‌کنیم. متغیر A در سمت چپ رشته تولید می‌شود، سپس به سمت راست رشته حرکت می‌کند و به ازای هر a یک نماد b و یک نماد c می‌سازد. سپس این متغیر تبدیل به متغیر B می‌شود و به ابتدای رشته بازمی‌گردد. وقتی متغیر B به ابتدای رشته رسید، تبدیل به A می‌شود و این کار ادامه پیدا می‌کند تا رشتهٔ مورد نظر به دست آید.

- یک فرایند اشتقاق برای به دست آوردن رشتهٔ aaabbbcc در این روش به شکل زیر است:

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

# زبان‌های حساس به متن

- قوانین تولید را بدین صورت می‌نویسیم:

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

# زبان‌های حساس به متن

- برای هر زبان حساس به متن $L$ که شامل رشتهٔ تهی نیست، یک ماشین ماشین کراندار خطی $M$ وجود دارد به طوری که $L = L(M)$.
- اگر زبان $L$ توسط یک ماشین کراندار خطی پذیرفته شود، آنگاه یک گرامر حساس به متن وجود دارد که $L$ را تولید می‌کند.
- برای اثبات این دو قضیه می‌توان فرایند اشتقاق در گرامرهای مستقل از متن را با حرکت‌های یک ماشین کراندار خطی شبیه‌سازی کرد که در اینجا به آن نمی‌پردازیم.

# زبان‌های حساس به متن

- می‌توان نشان داد که هر زبان حساس به متن بازگشتی (تصمیم‌پذیر) است و همچنین می‌توان نشان داد که یک زبان بازگشتی وجود دارد که حساس به متن نیست.
- نتیجه می‌گیریم زبان‌های حساس به متن زیرمجموعهٔ زبان‌های بازگشتی هستند و بنابراین قدرت ماشین‌های کراندار خطی کمتر از ماشین‌های تورینگ است.

# سلسله‌مراتب زبان‌ها

- تا اینجا با چند دسته از زبان‌ها آشنا شدیم:
- $L_{RE}$ : زبان‌های شمارش‌پذیر بازگشتی (تشخیص‌پذیر)
- $L_{CS}$ : زبان‌های حساس به متن
- $L_{CF}$ : زبان‌های مستقل از متن
- $L_{REG}$ : زبان‌های منظم

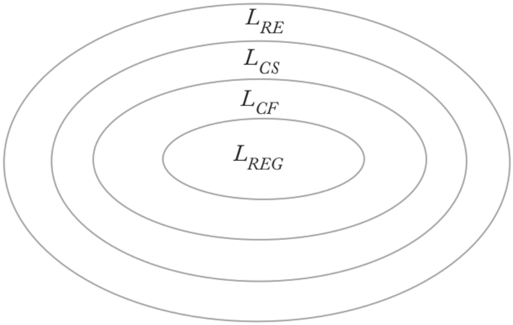
# سلسله‌مراتب زبان‌ها

- چامسکی، یکی از پایه‌گذاران زبان‌های صوری، همهٔ زبان‌های صوری را به چهار نوع (نوع ۰، نوع ۱، نوع ۲، و نوع ۳) [1] تقسیم کرده است.
- زبان‌های نوع صفر زبان‌هایی هستند که با گرامرهای بدون محدودیت تولید می‌شوند، یعنی زبان‌های شمارش‌پذیر بازگشتی. زبان‌های نوع یک زبان‌های حساس به متن، نوع دو زبان‌های مستقل از متن، و نوع سه زبان‌های منظم هستند.
- زبان‌های نوع $i$ در این طبقه‌بندی، زیرمجموعهٔ زبان‌های نوع $i - ۱$ هستند.

[1] type ۰ ، type ۱ ، type ۲ ، type ۳

# سلسله‌مراتب زبان‌ها

- بر اساس طبقه‌بندی چامسکی، سلسله‌مراتب زبان‌ها را می‌توان به صورت زیر نشان داد.

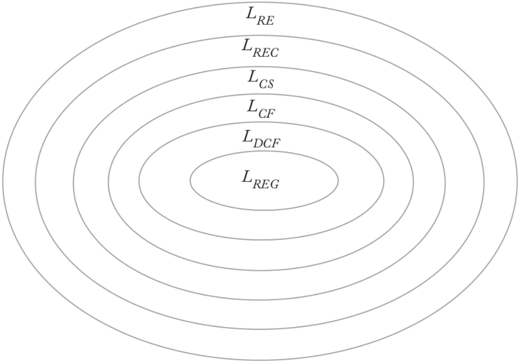


# سلسله‌مراتب زبان‌ها

- همچنین با دو طبقه دیگر از زبان‌ها آشنا شدیم.
- $L_{DCF}$ : زبان‌های مستقل از متن قطعی
- $L_{REC}$ : زبان‌های بازگشتی (تصمیم‌پذیر)

# سلسله‌مراتب زبان‌ها

- سلسله‌مراتب زبان‌ها را بر اساس آنچه مطالعه کردیم، می‌توان به صورت زیر نشان داد.



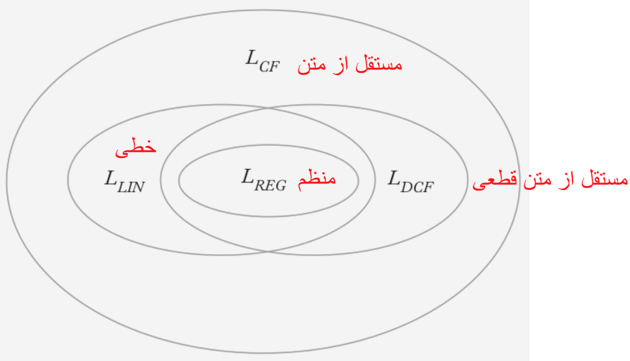

شمارش پذیر بازگشتی > بازگشتی > حساس به متن > مستقل از متن > مستقل از متن قطعی > منظم

# سلسله‌مراتب زبان‌ها

- همچنین با زبان‌های خطی $L_{LIN}$ آشنا شدیم که توسط گرامرهای خطی تولید می‌شوند. زبان‌های خطی زیرمجموعهٔ زبان‌های مستقل از متن هستند و هر زبان منظم یک زبان خطی است، پس این زبان‌ها ابرمجموعهٔ زبان‌های منظم هستند.
- از طرف دیگر زبان‌های خطی وجود دارند که زبان مستقل از متن قطعی نیستند و زبان‌های مستقل از متن قطعی وجود دارند که خطی نیستند، اما این دو زبان اشتراک‌هایی دارند.

# سلسله‌مراتب زبان‌ها

- جایگاه زبان‌های خطی را می‌توان به صورت زیر نشان داد.



# سلسله‌مراتب زبان‌ها

- همچنین ماشین‌های کراندار خطی را غیرقطعی تعریف کردیم. جایگاه زبان‌هایی که با ماشین‌های کراندار خطی قطعی پذیرفته می‌شوند تاکنون شناخته نشده است.
- به عبارت دیگر تاکنون اثباتی ارائه نشده است مبنی بر اینکه زبان‌های حساس به متن قطعی زیر مجموعهٔ زبان‌های حساس به متن هستند یا خیر.

# سلسله‌مراتب زبان‌ها

| مخفف | زبان | گرامر | ماشین |
| --- | --- | --- | --- |
| REG<br>نوع ۳ | منظم | $A \to aB, A \to Ba, A \to a$<br>$A, B \in V, a \in T^*$ | متناهی قطعی و متناهی غیرقطعی |
| CF<br>نوع ۲ | مستقل از متن | $A \to \alpha$<br>$A \in V, \alpha \in (V \cup T)^*$ | پشته‌ای |
| CS<br>نوع ۱ | حساس به متن | $x \to y$<br>$x, y \in (V \cup T)^+, \|y\| \geq \|x\|$ | کراندار خطی |
| RE<br>نوع ۰ | شمارش‌پذیر بازگشتی | $x \to y$<br>$x \in (V \cup T)^+, y \in (V \cup T)^*$ | تورینگ |