۱ مقدمه‌ای بر نظریه‌ی محاسبات

An Introduction to the Theory of Computation

۱-۱ مقدمه

۱-۱-۱ محاسبات چیست؟

محاسبات: پردازش اطلاعات بر اساس مجموعه‌ای متناهی از عملیات یا قواعد.

مثال‌هایی از محاسبات:

- حساب قلم و کاغذ
- چرتکه
- خط‌کش و بکارگیری ساختارهای هندسی
- کامپیوترهای دیجیتال

دیگر دستگاه‌های محاسباتی

- برنامه‌های C، JAVA، ...
- اینترنت و دیگر سیستم‌های توزیع‌شده
- سلول‌ها / DNA ؟
- مغز انسان؟
- کامپیوترهای کوانتومی؟

۱-۱-۲ از نظریه چه می‌خواهیم؟

- **عمومیت**
  - مستقل از فناوری
  - انتزاع: نادیده گرفتن جزئیات غیرضروری
- **دقت**
  - ریاضی بودن، رسمی بودن
  - امکان اثبات قضایا در مورد محاسبات (مثبت و منفی):
    چه چیزی می‌تواند محاسبه شود؟
    چه چیزی نمی‌تواند محاسبه شود؟

۱-۱-۳ مشخصه‌های مسایل محاسباتی

- **گسسته بودن (Discreteness)**: امکان بیان حالت سیستم با تعداد محدودی از اطلاعات
- **انتزاع (Abstraction)**: امکان نادیده گرفتن جزئیات
- **عمومیت (Generality)**: امکان اعمال یک مدل ریاضی واحد به تعداد زیادی از فناوری‌ها

۱-۱-۴ حوزه‌های نظریه‌ی محاسبات

نظریه‌ی محاسبات:

آتوماتا، محاسبه‌پذیری و پیچیدگی

پرسش اساسی: **قابلیت‌های پایه و محدودیت‌های کامپیوتر چیست؟**

- **نظریه‌ی پیچیدگی**

  پرسش اساسی: **آیا مسایلی وجود دارند که به طور کارامد قابل حل نباشند؟**

  چه چیزی باعث می‌شود که برخی از مسایل از نظر محاسباتی دشوار و برخی دیگر آسان باشند؟

  نظریه‌ی پیچیدگی ⇐ دسته‌بندی مسایل محاسباتی به آسان و دشوار

  چگونه می‌توانیم مسایل دشوار را حل کنیم؟

- **نظریه‌ی محاسبه‌پذیری**

  پرسش اساسی: **آیا مسایلی وجود دارند که با کامپیوتر قابل حل نباشند؟**

  نظریه‌ی محاسبه‌پذیری ⇐ دسته‌بندی مسایل محاسباتی به قابل حل و غیر قابل حل

- **نظریه‌ی آتوماتا**

  پرسش اساسی: **کامپیوتر چیست؟**

  نظریه‌ی آتوماتا ⇐ مدل‌های مختلف برای محاسبات

  نقطه‌ی شروع ما برای مطالعه‌ی نظریه‌ی محاسبات: نظریه‌ی آتوماتا و زبان‌ها

۱-۲ مقدمات ریاضی

۱-۲-۱ مجموعه‌ها

- **عضویت** در مجموعه‌ها
- روش‌های **نمایش مجموعه‌ها** (نمایش با اعضا، نمایش با نماد ریاضی)
- **مجموعه‌ی جهانی** (Universal set) و **مجموعه‌ی تهی** (Empty set) و خواص آن‌ها
- **عملگرهای مجموعه‌ای**: اجتماع، اشتراک، تفاضل، متمم
- **زیرمجموعه** بودن و **تساوی** مجموعه‌ها، زیرمجموعه‌ی محض، مجموعه‌های مجزا (disjoint)
- **مجموعه‌ی توانی** یک مجموعه (۲^S)
- مجموعه‌های **متناهی** (finite) و **نامتناهی** (infinite)

۱-۲-۲ رابطه و تابع

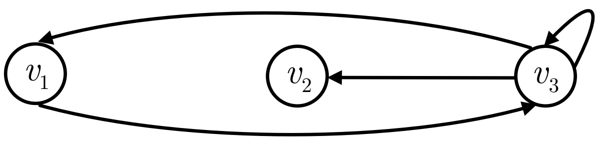
- **ضرب دکارتی** دو مجموعه
- **تابع** $(f : A \to B)$: توابع تام (total) $(D_f = A)$ و توابع جزیی (partial) $(D_f \subset A)$
- توابع یک به یک، توابع پوشا
- **رابطه**: انواع روابط (هم‌ارزی، ترتیب)

گراف‌ها و درخت‌ها

- **گراف** (راس‌ها و یال‌ها)
- گراف ساده، گراف جهت‌دار، برچسب یال، برچسب راس
- **گشت (walk)**: دنباله‌ای از یال‌های متصل به هم
- طول یک گشت: تعداد یال‌های گشت از مبدا تا مقصد
- **مسیر (path)**: راهی که یال تکراری ندارد
- **مسیر ساده (simple path)**: مسیری که راس تکراری ندارد
- **دور (چرخه) (cycle)**: یک گشت از راس v به خودش بدون یال‌های تکراری (دور با پایه‌ی v)
- **دور ساده (simple cycle)**: دوری که در آن بجز راس پایه، هیچ راس تکراری ندارد
- **درخت جهت‌دار (درخت)**: گراف جهت‌داری که دور ندارد و دارای یک گره‌ی خاص به نام ریشه است

مثال

گراف زیر را در نظر می‌گیریم:



یک گشت: $(v_۱, v_۳), (v_۳, v_۳), (v_۳, v_۱), (v_۱, v_۳), (v_۳, v_۲)$

یک مسیر: $(v_۱, v_۳), (v_۳, v_۲)$

یک دور: $(v_۱, v_۳), (v_۳, v_۳), (v_۳, v_۱)$

۱-۲-۳ روش‌های اثبات

روش‌های متداول برای اثبات قضایای ریاضی:

- **اثبات با استقرای ریاضی**

- **اثبات با برهان خلف**

  برای اثبات $\alpha \Rightarrow \beta$ نشان می‌دهیم که $\alpha \wedge \neg\beta$ نادرست است.

- **اثبات با ساختن (proof by construction)**

مثال

می‌خواهیم ثابت کنیم برای هر $n \geq ۳$، یک گراف ساده با n راس وجود دارد که درجه‌ی تمام رئوس آن برابر با ۲ است.

اثبات: گراف G را به صورت زیر می‌سازیم و حکم ثابت است:

$$\begin{aligned} G &= (V, E) \\ V &= \{۱, ۲, ۳, \ldots, n\} \\ E &= \{(i, i+۱) : i = ۱, ۲, \ldots, n-۱\} \cup \{(n, ۱)\} \end{aligned}$$

۱-۳ مفاهیم بنیادی نظریه‌ی محاسبات

۱-۳-۱ مفهوم اول: زبان

الفبا

تعریف **الفبا** یک مجموعه‌ی متناهی و ناتهی از نمادهای تجزیه‌ناپذیر

الفبا را معمولاً با Σ نمایش می‌دهیم. به هر عضو الفبا $a \in \Sigma$ یک نماد می‌گوییم.

مثال

هر یک از مجموعه‌های زیر یک الفبا را نشان می‌دهد:

$$\begin{aligned}\Sigma_۱ &= \{۱, ۲, ۳, ۴, \dots, ۹\}\\ \Sigma_۲ &= \{a, b, c, \dots, z, A, B, C, \dots, Z\}\\ \Sigma_۳ &= \{*, +, /, =\}\end{aligned}$$

رشته‌ها

تعریف **رشته** دنباله‌ای متناهی از نمادهای الفبا

مثال

هر یک از موارد زیر به ترتیب یک رشته روی الفبای $\Sigma_۱$، $\Sigma_۲$ و $\Sigma_۳$ است:

$$\begin{aligned}w_۱ &= ۱۲۳۴۵\\ w_۲ &= aabbaaab\\ w_۳ &= +/*+**\end{aligned}$$

- **الحاق دو رشته (concatenation)**: اگر u و v دو رشته باشند، حاصل الحاق u و v رشته‌ای است که از قرار دادن v به دنبال u به صورت uv به دست می‌آید.

- **توان‌های یک رشته:**

$$w^{1} = w, \qquad w^{n+1} = w^n w$$

- **طول یک رشته:** طول رشته‌ی w که با $|w|$ نشان داده می‌شود، برابر با تعداد نمادهای آن تعریف می‌شود.

$$w = a_1 a_2 \cdots a_n \qquad \Rightarrow \qquad |w| = n$$

- **رشته‌ی تهی:** رشته‌ای که طول آن صفر است. رشته‌ی تهی با λ نشان داده می‌شود.

$$|\lambda| = 0$$

رشته‌ی تهی عضو خنثای عمل الحاق است:

$$\forall w \in \Sigma^* \qquad \Rightarrow \qquad \lambda w = w\lambda = w$$

بنا بر تعریف برای هر رشته‌ی w داریم:

$$w^{0} = \lambda$$

رشته‌ی تهی عضو هیچ الفبایی نیست:

$$\forall \Sigma \ \lambda \notin \Sigma$$

◄ **تذکر** رشته‌ی تهی با فاصله‌ی خالی ($\square$) متفاوت است ($|\square| = 1$).

- **معکوس یک رشته:** معکوس یک رشته‌ای است که اگر از انتها خوانده شود برابر با آن رشته شود.

$$w = a_1 a_2 \cdots a_n \qquad \Rightarrow \qquad w^R = a_n \cdots a_2 a_1$$

تعریف بازگشتی:

$$w = av, \qquad a \in \Sigma, v \in \Sigma^* \qquad \Rightarrow \qquad w^R = v^R a$$

- **زیررشته:** بخشی از نمادهای پی در پی یک رشته، یک زیررشته‌ی آن رشته نام دارد.
می‌گوییم u زیررشته‌ی w است و می‌نویسیم $u \sqsubseteq w$ اگر و فقط اگر رشته‌های α و β وجود داشته باشند که برای آن‌ها داشته باشیم:

$$w = \alpha u \beta$$

تعداد زیررشته‌های یک رشته w: $|\text{SUBSTRINGS}(w)| = \frac{1}{2}(|w|(|w|+1)) + 1$

  - **پیشوند:** زیررشته‌ای که از ابتدای رشته شروع شود ($\alpha = \lambda$)
تعداد پیشوندهای یک رشته w: $|\text{PREFIXES}(w)| = |w| + 1$
  - **پسوند:** زیررشته‌ای که به انتهای رشته ختم شود. ($\beta = \lambda$)
تعداد پسوندهای یک رشته w: $|\text{SUFFIXES}(w)| = |w| + 1$

- **زیردنباله:** رشته‌ی حاصل از حذف صفر نماد یا بیشتر از رشته.
حداکثر تعداد زیردنباله‌های یک رشته w: $|\text{SUBSEQUENCES}(w)| \leq 2^{|w|}$
رشته‌ی تهی، زیررشته، پیشوند، پسوند و زیردنباله‌ی هر رشته‌ای است.
هر رشته‌ای زیررشته و زیردنباله‌ی خودش است.

چند رابطه در مورد رشته‌ها. برای $u, v, w \in \Sigma^*$ و $n \geq ۰$ داریم:

$$|\lambda| = ۰$$
$$|a| = ۱, \qquad a \in \Sigma$$
$$|uv| = |u| + |v|$$
$$|w^n| = n|w|$$
$$(uv)^R = v^R u^R$$
$$(w^n)^R = (w^R)^n$$

مجموعه‌ی کلیه‌ی رشته‌های قابل تولید از الفبای Σ را می‌توان از اتصال صفر یا تعدادی از نمادهای Σ به دست آورد. این مجموعه را با Σ^* نشان می‌دهیم:

$$\Sigma^* = \Sigma^۰ \cup \Sigma^۱ \cup \Sigma^۲ \cup \cdots = \bigcup_{i=۰}^{\infty} \Sigma^i$$

که در آن توان‌های Σ به صورت زیر تعریف می‌شود:

$$\Sigma^۰ = \{\lambda\}, \qquad \Sigma^{i+۱} = \{aw : a \in \Sigma, w \in \Sigma^i\}$$

اگر رشته‌ی تهی را از Σ^* کنار بگذاریم، Σ^+ را خواهیم داشت:

$$\Sigma^+ = \Sigma^۱ \cup \Sigma^۲ \cup \Sigma^۳ \cup \cdots = \bigcup_{i=۱}^{\infty} \Sigma^i$$

بدیهی است که

$$\Sigma^+ = \Sigma^* - \{\lambda\}$$
$$\Sigma^* = \Sigma^+ \cup \{\lambda\}$$

اگرچه Σ متناهی است، اما Σ^* و Σ^+ همیشه نامتناهی هستند.

مشخص است که Σ^i حاوی همه‌ی رشته‌های به طول i بر روی الفبای Σ است و برای هر i داریم:
$$|\Sigma^i| = |\Sigma|^i$$

مثال

برای یک الفبای دو حرفی Σ داریم:

$$\begin{aligned} \Sigma &= \{a, b\} \\ \Sigma^* &= \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\} \\ \Sigma^+ &= \{a, b, aa, ab, ba, bb, aaa, aab, \ldots\} \end{aligned}$$

زبان

تعریف **زبان** یک زبان مجموعه‌ای از رشته‌های یک الفباست، یعنی $L \subseteq \Sigma^*$

اگر مجموعه‌ی L متناهی باشد، L یک زبان متناهی است و در غیر این صورت نامتناهی است.
به هر عضو مجموعه‌ی L یک رشته یا **جمله** گفته می‌شود.

مثال

L_1 و L_2 هر دو زبان‌هایی بر روی الفبای $\Sigma = \{a, b\}$ هستند:

$$\begin{aligned} L_1 &= \{a, aa, aab\} \\ L_2 &= \{a^n b^n : n \geq 0\} \end{aligned}$$

عملیات روی زبان‌ها: فرض می‌کنیم $L, L_1, L_2 \in \Sigma^*$

- اجتماع دو زبان:

$$L_1 \cup L_2 = \{x : x \in L_1 \vee x \in L_2\}$$

- اشتراک دو زبان:

$$L_1 \cap L_2 = \{x : x \in L_1 \wedge x \in L_2\}$$

- تفاضل دو زبان:

$$L_1 - L_2 = \{x : x \in L_1 \wedge x \notin L_2\}$$

- متمم یک زبان:

$$\bar{L} = \Sigma^* - L$$

- عکس یک زبان:

$$L^R = \{x^R : x \in L\}$$

- الحاق دو زبان:

$$L_1 L_2 = \{x_1 x_2 : x_1 \in L_1 \wedge x_2 \in L_2\}$$

- توان یک زبان:

$$L^0 = \{\lambda\}, \qquad L^{n+1} = L^n L = L L^n$$

• بستار ستاره‌ای:

$$L^* = L^{۰} \cup L^{۱} \cup L^{۲} \cup \cdots = \bigcup_{i=۰}^{\infty} L^i$$

• بستار مثبت:

$$L^+ = L^{۱} \cup L^{۲} \cup L^{۳} \cup \cdots = \bigcup_{i=۱}^{\infty} L^i$$

مطابق این دو تعریف داریم:

$$L^* = L^+ \cup \{\lambda\}$$
$$L^+ = LL^*$$

مثال

اگر $L = \{a^n b^n : n \geq ۰\}$ باشد در این صورت داریم:

$$\begin{aligned} L^R &= \{a^n b^n : n \geq ۰\} \\ L^{۲} &= LL = \{a^n b^n : n \geq ۰\}\{a^m b^m : m \geq ۰\} = \{a^n b^n a^m b^m : n, m \geq ۰\} \end{aligned}$$

زبان تهی: زبان تهی، زبانی است که هیچ رشته‌ای ندارد و با $\varnothing$ یا $\{\}$ نشان داده می‌شود.
بدیهی است که $\lambda \neq \{\}$
زبان تهی عضو صفر عمل الحاق زبان‌هاست:

$$L\varnothing = \varnothing L = \varnothing$$

در حالی که $\{\lambda\}$ عضو خنثای عمل الحاق زبان‌هاست.

$$L\{\lambda\} = \{\lambda\}L = L$$

به علاوه داریم:

$$\varnothing^* = \{\lambda\}, \qquad \varnothing^+ = \varnothing$$

چند رابطه در مورد عملیات روی زبان‌ها. اگر L، $L_۱$، $L_۲$ و $L_۳$ زبان‌هایی روی Σ^* باشند، داریم:

$L_1 L_2 \neq L_2 L_1$
$|L_1 L_2| \neq |L_2 L_1|$
$|L_1 L_2| \leq |L_1||L_2|$
$(L_1 L_2) L_3 = L_1 (L_2 L_3)$
$L_1 (L_2 \cup L_3) = L_1 L_2 \cup L_1 L_3$
$(L_2 \cup L_3) L_1 = L_2 L_1 \cup L_3 L_1$
$L_1 (L_2 \cap L_3) \subseteq L_1 L_2 \cap L_1 L_3$
$(L_2 \cap L_3) L_1 \subseteq L_2 L_1 \cap L_3 L_1$
$L_1 \subseteq L_2 \Rightarrow L_1^n \subseteq L_2^n$
$L_1 \subseteq L_1 L_2^*$
$L_1 \subseteq L_2^* L_1$
$L_1 \subseteq L_2 \Rightarrow L_1^* \subseteq L_2^*$
$L_1 \subseteq L_2 \Rightarrow L_1^+ \subseteq L_2^+$
$L^* L = L L^* = L^+$
$L^* L^* = L^* = (L^*)^* = (L^*)^+ = (L^+)^*$
$(L_1 \cup L_2)^* = (L_1^* \cup L_2^*)^* = (L_1^* L_2^*)^*$
$L_1 (L_2 L_1)^* = (L_1 L_2)^* L_1$

الفبای Σ را در نظر بگیرید:

مجموعه‌ی همه‌ی رشته‌ها با طول زوج	$(\Sigma^2)^*$
مجموعه‌ی همه‌ی رشته‌ها با طول فرد	$(\Sigma^2)^*\Sigma$
مجموعه‌ی همه‌ی رشته‌ها به طول n	Σ^n
مجموعه‌ی همه‌ی رشته‌ها با طول مضرب k	$(\Sigma^k)^*$

۱-۳-۲ مفهوم دوم: گرامر

تعریف

گرامر گرامر G یک چهارتایی مرتب به صورت

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

است که در آن

V مجموعه‌ای متناهی از ناپایانه‌ها *(non-terminal)* یا متغیرها *(variable)*

T مجموعه‌ای از پایانه‌ها *(terminal)* (الفبای گرامر)

S یک عنصر خاص از V با نام نماد شروع *(start symbol)*

P مجموعه‌ای متناهی از قواعد تولید *(production rule)*

با فرض اینکه V و T ناتهی و مجزا هستند $(V \cap T = \varnothing)$

قواعد تولید قواعد تولید اساس تعریف یک گرامر است. هر قاعده‌ی تولید (قاعده) یک زوج مرتب به صورت $(x, y) \in P$ است که به شکل

$$x \rightarrow y$$

نمایش داده می‌شود که در آن

$x \in (V \cup T)^+$ (هر ترکیبی از پایانه‌ها و ناپایانه‌های گرامر بجز رشته‌ی تهی)

$x \in (V \cup T)^*$ (هر ترکیبی از پایانه‌ها و ناپایانه‌های گرامر)

از قواعد تولید در عمل اشتقاق (derivation) استفاده می‌شود.

اشتقاق منظور از اشتقاق، اعمال یک قاعده‌ی تولید بر روی یک رشته است.

اگر رشته‌ی w به صورت $w = uxv$ باشد و قاعده‌ی تولید $x \rightarrow y$ را داشته باشیم، می‌توانیم آن را بر روی این رشته اعمال کنیم و رشته‌ی z به صورت $z = uyv$ را به دست آوریم. به این عمل اشتقاق می‌گوییم و آن را به صورت زیر نشان می‌دهیم

$$w \Rightarrow z$$

و می‌گوییم w، z را مشتق می‌کند یا z از w مشتق می‌شود.

می‌توان قواعد تولید را به طور پی در پی و به تعداد دلخواه استفاده کرد تا رشته‌ی مورد نظر به دست آید:

$$w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n$$

و گفته می‌شود که w_n از w_1 مشتق می‌شود. می‌نویسیم

$$w_1 \Rightarrow^* w_n$$

که به معنی آن است که w_n طی صفر یا چند مرحله تولید می‌شود.

در فرایند اشتقاق، هرگاه سمت چپ یک قاعده با یک زیررشته از یک رشته تطابق پیدا کند، می‌توانیم سمت راست آن قاعده را با آن زیررشته جایگزین کنیم.

$$w \Rightarrow z \quad \textbf{iff} \quad w = w_1uw_2,\ z = w_1vw_2,\ w_1,\ w_2 \in (V \cup T)^*,\ u \to v \in P$$

برای بیان جزئیات عمل اشتقاق از نمادگذاری‌های زیر استفاده می‌شود:

$\Rightarrow_G$	اشتقاق توسط گرامر G	
$\Rightarrow$	اشتقاق در یک مرحله G	(دارای خاصیت تراگذری)
$\Rightarrow^*$	اشتقاق در صفر مرحله یا بیشتر G	(دارای خاصیت بازتابی و تراگذری)
$\Rightarrow^+$	اشتقاق در یک مرحله یا بیشتر G	(دارای خاصیت تراگذری)
$\Rightarrow^n$	اشتقاق در n مرحله	(دارای خاصیت تراگذری)
$\overset{(r)}{\Rightarrow}$	اشتقاق با قاعده‌ی r	

مثال

گرامر $G = (\{S\}, \{a, b\}, S, P)$ با مجموعه قواعد $P = \{S \to aSb, S \to \lambda\}$ را در نظر می‌گیریم. برای این گرامر می‌توان اشتقاق زیر را به دست آورد:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

که می‌توان به جای آن نوشت:

$$S \Rightarrow^* aabb$$

زبان تولید شده توسط یک گرامر

تعریف **زبان تولید شده توسط یک گرامر** اگر $G = (V, T, S, P)$ یک گرامر باشد، آن‌گاه مجموعه‌ی

$$L(G) = \{w \in T^* : S \Rightarrow^* w\}$$

زبان تولید شده توسط G نام دارد.

مثال

زبان تولید شده توسط گرامر $G = (\{S\}, \{a, b\}, S, P)$ با مجموعه قواعد $P = \{S \to aSb, S \to \lambda\}$ عبارت است از:

$$L = \{a^n b^n : n \geq \circ\}$$

اگر $w \in L$ باشد، آن‌گاه

$$S \Rightarrow w_\text{۱} \Rightarrow w_\text{۲} \Rightarrow \ldots \Rightarrow w_n \Rightarrow w$$

را یک اشتقاق از جمله‌ی w می‌گویند. رشته‌های S، $w_\text{۱}$، $w_\text{۲}$، ...، w_n که حاوی پایانه‌ها و ناپایانه‌ها هستند به **فرم جمله‌ای** (sentential form) موسوم هستند.

مجموعه‌ی همه‌ی فرم‌های جمله‌ای قابل اشتقاق از یک گرامر را با $SF(G)$ نمایش می‌دهیم و به صورت

$$SF(G) = \{W \in (T \cup V)^* : S \Rightarrow^* W\}$$

تعریف می‌کنیم. بدیهی است که

$$L(G) \subset SF(G)$$

◄ **نکته** برای این که ثابت کنیم یک زبان توسط یک گرامر تولید می‌شود، باید ثابت کنیم که

۱) گرامر تمام رشته‌های زبان را می‌سازد (هر رشته‌ی $w \in L$ می‌تواند توسط گرامر تولید شود).

۲) گرامر هیچ جمله‌ای را خارج از آن زبان تولید نمی‌کند (هر جمله‌ی تولید شده توسط گرامر در L است).

برای اثبات اینکه $L = L(G)$ است، باید ثابت کنیم که $L \subseteq L(G)$ و $L(G) \subseteq L$.

تعریف **هم‌ارزی گرامرها** دو گرامر $G_\text{۱}$ و $G_\text{۲}$ معادل هستند، اگر و فقط اگر هر دو یک زبان واحد را تولید کنند.

$$G_\text{۱} \equiv G_\text{۲} \quad \text{iff} \quad L(G_\text{۱}) = L(G_\text{۲})$$

◄ **نکته** **یک شرط کافی برای هم‌ارزی دو گرامر G_1 و G_2:**
اگر قاعده‌ی $x \to y$ را در G_1 داشته باشیم، باید این قاعده در G_2 نیز موجود باشد و یا اشتقاق $x \Rightarrow^*_{G_2} y$ را بتوان تولید نمود.

◄ **تذکر** برای بررسی هم‌ارزی گرامرها، ابتدا بررسی کنید که آیا هر دو گرامر رشته‌ی λ را تولید می‌کنند یا خیر.

چند زبان معروف و گرامر متناظر با آنها. برای $\Sigma = \{a, b\}$ داریم:

$L(G)$	$P(G)$
$\{a^n : n \geq 0\}$	$S \to aS \mid \lambda$
$\{w : n_a(w) = n_b(w)\}$	$S \to aSb \mid bSa \mid SS \mid \lambda$
$\{w : w \in \{a, b\}^*\}$	$S \to aS \mid bS \mid \lambda$
$\{ww^R : w \in \{a, b\}^*\}$	$S \to aSa \mid bSb \mid \lambda$
$\{a^n b^n : n \geq 0\}$	$S \to aSb \mid \lambda$
$\{a^n b^m : n \geq 0, m > n\}$	$S \to aSb \mid B,\ B \to bB \mid b$

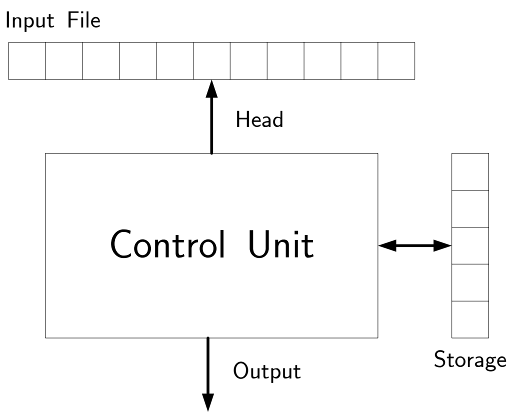
گرامر برای ترکیب زبان‌ها

$$L_1 = L(G_1), \qquad G_1 = (V_1, T, S_1, P_1)$$
$$L_2 = L(G_2), \qquad G_2 = (V_2, T, S_2, P_2)$$
$$V_1 \cap V_2 = \varnothing, \qquad S \notin V_1 \cup V_2$$

$L_1 \cup L_2 = L(G)$	$G = (V_1 \cup V_2 \cup \{S\}, T, S, P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\})$
$L_1 L_2 = L(G)$	$G = (V_1 \cup V_2 \cup \{S\}, T, S, P_1 \cup P_2 \cup \{S \to S_1 S_2\})$
$L_1^* = L(G)$	$G = (V_1 \cup \{S\}, T, S, P_1 \cup \{S \to SS_1 \mid \lambda\}) \ \equiv$ $G = (V_1 \cup \{S\}, T, S, P_1 \cup \{S \to S_1 S \mid \lambda\})$

۱-۳-۳ مفهوم سوم: ماشین (آتوماتون)

آتوماتون (ج. آتوماتا) ، یک مدل انتزاعی از کامپیوتر است.



- آتوماتون مکانیزمی برای خواندن **ورودی** دارد.
ورودی آتوماتون یک رشته‌ی الفبایی داده شده است که بر روی نوار ورودی نوشته شده است.
آتوماتون می‌تواند فایل ورودی را بخواند، اما نمی‌تواند آن را تغییر دهد.
نوار ورودی به چند سلول تقسیم شده و هر سلول یک نشانه از الفباست.
نوار ورودی از چپ به راست خوانده می‌شود (هر بار یک نشانه).
مکانیزم خواندن می‌تواند انتهای رشته‌ی ورودی را تشخیص دهد.
- آتوماتون می‌تواند **خروجی** تولید کند.
- آتوماتون می‌تواند دارای **حافظه‌ی موقت** باشد.
حافظه دارای تعداد نامتناهی سلول است که هر سلول آن یک نماد الفبا را در خود جای می‌دهد.
آتوماتون می‌تواند محتوای حافظه‌ی موقت را تغییر دهد.
- آتوماتون دارای یک **واحد کنترل** است که هر زمان می‌تواند در یکی از حالات داخلی خود باشد.
تعداد حالات آتوماتون متناهی است.
آتوماتون می‌تواند با ترتیب مشخص تغییر حالت دهد.
آتوماتون در هر زمان در یک حالت خاص قرار دارد و نماد مشخصی را از ورودی می‌خواند.
حالت بعدی آتوماتون توسط تابع گذر (transition function) مشخص می‌شود.
تابع گذر بر اساس **نماد ورودی**، **حالت فعلی** و **محتوای حافظه** حالت بعدی را تعیین می‌کند.
به **حالت کنترل**، **باقیمانده‌ی ورودی** و **محتوای حافظه** یک پیکربندی (configuration) می‌گویند.
تغییر حالت می‌تواند موجب تولید خروجی یا تغییر در حافظه شود.
تغییر وضعیت از یک حالت به حالت دیگر، حرکت (move) نام دارد.

حافظه و **نوع خروجی** بر نوع آتوماتون تاثیر دارد.

آتوماتای پذیرنده / تراگذر

- **پذیرنده (accepter)**: آتوماتونی که خروجی آن بلی / خیر باشد (ورودی خود را قبول یا رد کند).
- **تراگذر (transducer)**: آتوماتونی که خروجی آن در قالب یک رشته است.

آتوماتای قطعی / غیرقطعی

- **آتوماتون قطعی (deterministic)**: آتوماتونی که در آن با دانستن **حالت فعلی**، **ورودی** و **محتوای حافظه** می‌توان رفتار بعدی آتوماتون را تعیین کرد.
- **آتوماتون غیرقطعی (non-deterministic)**: آتوماتونی که قطعی نباشد: در هر پیکربندی می‌توان چند حرکت مختلف انجام داد.

۱-۴ رابطه‌ی میان زبان، گرامر و ماشین

زبان، گرامر و ماشین، هر سه با مفهوم رشته سر و کار دارند:

- **زبان** مجموعه‌ای از رشته‌هاست،
- **گرامر** رشته‌های زبان را تولید می‌کند، و
- **ماشین** رشته‌های زبان را مصرف می‌کند.

متناظر با هر گرامر، یک زبان و حداقل یک ماشین وجود دارد.

این درس به مطالعه‌ی خانواده‌های مختلف زبان‌ها، گرامرها و ماشین‌ها و رابطه‌ی آنها می‌پردازد.