به نام خدا

نظریهٔ زبان‌ها و ماشین‌ها

آرش شفیعی

کتاب‌های مرجع

- مقدمه‌ای بر نظریهٔ زبان‌ها (زبان‌های صوری) و ماشین‌ها از پیتر لینز [1]
- مقدمه‌ای بر نظریهٔ محاسبات از مایکل سیپسر [2]
- مقدمه‌ای بر نظریهٔ ماشین‌ها، زبان‌ها، و محاسبات از جان هاپکرافت و همکاران [3]

---

[1] Introduction to Formal Languages and Automata، by Peter Linz

[2] Introduction to the Theory of Computation، by Michael Sipser

[3] Introduction to Automata Theory، Languages، and Computation، by John Hopcroft et al.

ارزشیابی

- تمرینات $\approx$ ۸ نمره
- آزمون میان‌دوره $\approx$ ۶ نمره
- آزمون پایان‌دوره $\approx$ ۶ نمره

مقدمه

مباحث اصلی

- **زبان‌های صوری** [1] (فرمال یا قراردادی) و **گرامر** [2] (دستور زبان) آنها
- **ماشین‌ها** [3] (اتوماتا) برای شناسایی (پذیرفتن) زبان‌ها
- **محاسبه‌پذیری** [4]
- **پیچیدگی** [5]

[1] formal language
[2] grammar
[3] automata
[4] computability
[5] complexity

مباحث اصلی

- **زبان‌های صوری و گرامرها و ماشین‌ها** : چگونه برای شناسایی یک زبان، یک ماشین طراحی می‌کنیم؟

  به عبارت دیگر محاسبات را چگونه توسط کامپیوترها انجام می‌دهیم؟

- **محاسبه‌پذیری** : محدویت ماشین‌ها شناسایی زبان‌ها چیست؟

  به عبارت دیگر چه نوع محاسباتی را با کامپیوترها می‌توانیم انجام دهیم و چه نوع محاسباتی را نمی‌توانیم؟

- **پیچیدگی** : چگونه زبان‌ها را از لحاظ پیچیدگی (درجهٔ سختی) آنها برای شناسایی توسط ماشین‌ها به دسته‌های متفاوت تقسیم‌بندی می‌کنیم؟

  به عبارت دیگر با چه مقدار و چه نوع حافظه‌ای و با چه سرعتی می‌توانیم محاسبات را انجام دهیم؟

مباحث اصلی

- ساختن یک نظریهٔ انتزاعی برای فهم اصول محاسبات و کامپیوترها
- پیدا کردن بینشی کلی در مورد چیستی و توانایی کامپیوترها
- کمک کردن به استفاده از کامپیوترها و زبان‌های برنامه‌نویسی برای حل مسائل
- به طور خلاصه: مدلسازی محاسبات برای استفاده در حل مسئله‌های محاسباتی

مباحث اصلی

- برای مدلسازی محاسبات از مفهوم **ماشین (اتوماتون)** استفاده می‌کنیم.
- یک ماشین یک ورودی را دریافت می‌کند، و تصمیم می‌گیرد که چگونه ورودی را به خروجی تبدیل کند.
- چگونگی حافظه و کنترل کنندهٔ (تصمیم‌گیرنده) هر ماشین را بیان می‌کنیم.
- پس یک ماشین انتزاعی مفهومی است برای تعریف یک محاسبه‌گر (کامپیوتر).

مباحث اصلی

- **زبان صوری** تشکیل شده است از تعدادی جملهٔ قابل قبول در آن زبان.
- برای ساختن یک جمله از تعدادی نماد (سمبول) و قوانینی برای ترکیب نمادها استفاده می‌کنیم.
- پس یک زبان صوری مفهومی انتزاعی از زبان‌های برنامه‌نویسی است.

پیش‌نیاز

- آشنایی با نظریهٔ مجموعه‌ها
- آشنایی با توابع و روابط
- آشنایی با گراف‌ها و درخت‌ها
- آشنایی با اثبات ریاضی

پیش‌نیاز

آشنایی با نظریهٔ مجموعه‌ها:

- اشتراک، اجتماع، تفاضل، متمم
- مجموعهٔ مرجع، مجموعهٔ تهی
- قوانین دمورگان
- زیر مجموعه، زیرمجموعهٔ محض، مجموعه‌های مجزا
- مجموعهٔ متناهی، مجموعهٔ نامتناهی
- مجموعهٔ توانی
- ضرب دکارتی
- افراز یک مجموعه

پیش‌نیاز

آشنایی با توابع و روابط:

- تابع و دامنه و برد
- تابع کامل و تابع جزئی
- مرتبهٔ بزرگی توابع، حداکثر از مرتبه (کران پایین حدی)، حداقل از مرتبه (کران بالای حدی)، هم‌مرتبه
- رابطه
- رابطهٔ هم‌ارزی، بازتابی، متقارن، ترایا

پیش‌نیاز

آشنایی با گراف‌ها و درخت‌ها

- گراف و رأس و یال
- گراف جهت‌دار و گراف معمولی (بدون جهت)
- گشت، مسیر، مسیر ساده، دور، حلقه
- درخت، ریشه، برگ، والد، فرزند
- سطح رأس درخت، ارتفاع درخت

پیش‌نیاز

آشنایی با اثبات ریاضی

- استدلال استقرایی
- برهان خلف

زبان‌ها

- زبان‌های طبیعی (مانند فارسی و انگلیسی و فرانسوی) مجموعه‌ای از نمادها (سمبول‌ها) و قوانین گرامری هستند که برای بیان مفاهیم و حقایق و ارتباط انسان‌ها به کار می‌روند.
- در این مبحث، برای مطالعهٔ علمی زبان‌های صوری (فرمال یا قراردادی) باید تعریف دقیق‌تری از زبان ارائه کنیم.

زبان‌ها

- یک زبان مجموعه ای است از **رشته‌ها**[1].
- یک رشته دنباله‌ای محدود است از **نمادها** (سمبول‌ها)[2].
- به یک مجموعه Σ از سمبول‌ها یک **الفبا**[3] می‌گوییم.

[1] string
[2] symbol
[3] alphabet

زبان‌ها

- برای مثال اگر الفبای $\Sigma = \{a, b\}$ را داشته باشیم،
- رشته‌های $abab$ و $aaabbba$ رشته‌هایی از الفبای Σ هستند.
- برای نمایش یک رشته با نام w و مقدار $abaaa$ می‌نویسیم : $w = abaaa$.

- **الحاق** [1] رشتهٔ w و رشتهٔ v با افزودن نمادهای رشتهٔ v به سمت راست نمادهای رشتهٔ w به دست می‌آید.
$$w = a_1 \cdots a_n, \quad v = b_1 \cdots b_m, \quad wv = a_1 \cdots a_n b_1 \cdots b_m$$
- **معکوس** [2] یک رشته با نوشتن نمادهای آن با ترتیب معکوس (از آخر به اول) به دست می‌آید.
$$w^R = a_n \cdots a_1$$
- **طول** [3] رشتهٔ w با $|w|$ نشان داده می‌شود که تعداد نمادهای آن رشته است.
- **رشتهٔ تهی** [4] رشته‌ای است که هیچ نمادی در آن نیست و با λ نشان داده می‌شود.
$$|\lambda| = 0, \quad \lambda w = w\lambda = w$$

[1] concatenation
[2] reverse
[3] length
[4] empty string

زبان‌ها

- یک **زیررشته** [1] از رشتهٔ w دنباله‌ای از نمادهای متوالی در w است.
- اگر $w = vu$ باشد، v و u به ترتیب **پیشوند** [2] و **پسوند** [3] نامیده می‌شوند.
- اگر $w = abbab$ باشد، $\{\lambda, a, ab, abb, abba, abbab\}$ مجموعهٔ همه پیشوندهای w است و $\{\lambda, b, ab, bab, bbab, abbab\}$ مجموعهٔ همه پسوندها.
- برای طول رشته‌ها این رابطه برقرار است : $|uv| = |u| + |v|$

[1] substring
[2] prefix
[3] suffix

زبان‌ها

- رشتهٔ w^n رشته‌ای است که از تکرار رشتهٔ w به تعداد n بار به دست می‌آید.
- تعریف می‌کنیم : $w^{\circ} = \lambda$
- مجموعهٔ همهٔ رشته‌هایی را که با الفبای Σ به دست می‌آید با Σ^* نشان می‌دهیم.
- Σ^* همیشه رشتهٔ λ را نیز در بر می‌گیرد.
- تعریف می‌کنیم : $\Sigma^+ = \Sigma^* - \{\lambda\}$
- در حالی که Σ طبق تعریف یک مجموعهٔ محدود است، Σ^* و Σ^+ همیشه نامحدود هستند.
- یک زبان L زیرمجموعه‌ای از Σ^* است.
- هر رشته از زبان L را یک **جمله** از زبان L می‌نامیم.

زبان‌ها

- اگر $\Sigma = \{a, b\}$ باشد، داریم : $\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \cdots\}$
- مجموعهٔ $\{a, aa, aab\}$ یک زبان بر روی الفبای Σ است. این زبان یک زبان محدود است.
- مجموعهٔ $L = \{a^n b^n : n \geq 0\}$ نیز یک زبان بر روی الفبای Σ است. اما این زبان یک زبان نامحدود است. بعضی جملات آن برابرند با aabb و aaaabbbb . اما جملهٔ abb عضو این زبان نیست.

زبان‌ها

- از آنجایی که یک زبان، یک مجموعه است، همهٔ عملیات اجتماع و اشتراک و تفاضل بر روی آنها تعریف می‌شوند.
- با توجه به این که مجموعه مرجع، مجموعه‌ای شامل همه جملات بر روی یک الفبا است، متمم یک زبان بدین صورت تعریف می‌شود: $\overline{L} = \Sigma^* - L$
- معکوس یک زبان، معکوس همهٔ جملات آن است: $L^R = \{w^R : w \in L\}$
- الحاق دو زبان، الحاق همهٔ جملات آن دو زبان است: $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$
- الحاق یک زبان به خودش را به تعداد n بار به صورت L^n تعریف می‌کنیم.
- اگر $L = \{a, ab\}$ باشد، داریم $L^2 = \{aa, aab, aba, abab\}$
- تعریف می‌کنیم : $L^0 = \{\lambda\}$

- **بستار-ستاره** [1] بر روی زبان L را بدین صورت تعریف می‌کنیم: $L^* = L^\circ \cup L^۱ \cup L^۲ \cup \cdots$
- **بستار-مثبت** [2] بر روی زبان L را بدین صورت تعریف می‌کنیم: $L^+ = L^۱ \cup L^۲ \cup \cdots$
- اگر $L = \{a^n b^n : n \geq ۰\}$ باشد، داریم $L^۲ = \{a^n b^n a^m b^m : n \geq ۰, m \geq ۰\}$.
- همچنین داریم : $L^R = \{b^n a^n : n \geq ۰\}$

[1] star-closure
[2] positive closure

گرامرها

- برای تعریف یک زبان صوری با استفاده از زبان ریاضی از نظریهٔ مجموعه‌ها استفاده کردیم.
- ولی مجموعه‌ها پاسخگوی همهٔ نیازهای ما نیستند و محدودیت‌هایی دارند.
- اکنون از گرامرها برای تعریف یک زبان استفاده می‌کنیم.

گرامرها

- گرامر در یک زبان طبیعی به ما می‌گوید که آیا ساختار یک جمله درست است یا غلط.
- به عبارت دیگر گرامر چگونگی ساختار جمله (نحوهٔ کنار هم قرار گرفتن واژه‌ها و تکواژها) را توصیف می‌کند.
- مثلا در زبان فارسی می‌توانیم گرامری به این صورت تعریف کنیم: <نهاد> <گزاره> → <جمله>
- حالا باید تعریف کنیم که نهاد و گزاره چه هستند.
- می‌توانیم تعریف کنیم: <گروه اسمی> → <نهاد> و <مفعول><فعل> → <گزاره> و <متمم><فعل> → <گزاره> و <مسند><فعل> → <گزاره> و ...
- مثلا در جملهٔ «خدا مهربان است»، «خدا» نهاد، «مهربان» مسند، و «است» فعل است. همچنین «مهربان است» یک گزاره است.

گرامرها

- پس در یک گرامر با یک جمله شروع می‌کنیم و اجزای آن را مشخص می‌کنیم.
- یک زبان تشکیل شده است از جملاتی که با آن گرامر ساختار بندی شده اند.
- برای زبان‌های صوری نیز به همین ترتیب عمل می‌کنیم.

گرامرها

- یک گرامر G به صورت یک چهارتایی $G = (V, T, S, P)$ تعریف می‌شود به طوری که:
- V مجموعه‌ای متناهی از **متغیر** [1] هاست.
- T مجموعه‌ای متناهی از **نماد های پایانی (پایانه)** [2] است.
- $S \in V$ نمادی است به نام **متغیر آغازی** [3] .
- P مجموعه‌ای متناهی از **قوانین تولید** [4] است.
- مجموعه‌های V و T غیرتهی و مجموعه‌هایی مجزا هستند.

[1] variable
[2] terminal symbols
[3] start variable
[4] production rules

گرامرها

- قوانین تولید تعیین می‌کنند چگونه گرامر یک رشته را به یک رشتهٔ دیگر تبدیل می‌کند.
- قوانین تولید را بدین صورت نمایش می‌دهیم : $x \to y$
- به طوری که : $x \in (V \cup T)^+$ و $y \in (V \cup T)^*$
- برای مثال رشتهٔ $w = uxv$ با استفاده از قانون تولید $x \to y$ به رشتهٔ $z = uyv$ تبدیل می‌شود.
- در این صورت می‌نویسیم $w \Rightarrow z$ و می‌خوانیم w نتیجه (به دست) می‌دهد z و یا z از w مشتق می‌شود.
- با اعمال قوانین تولید با ترتیب دلخواه، رشته‌های پی‌درپی مشتق می‌شوند.

گرامرها

- اگر داشته باشیم : $w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_n$ می‌گوییم w_1 در چند گام نتیجه می‌دهد w_n و می‌نویسیم

$$w_1 \overset{*}{\Rightarrow} w_n$$

- علامت $\overset{*}{\Rightarrow}$ به معنای نتیجه دادن با تعداد نامشخصی از گام‌ها است.

- مجموعهٔ همه رشته‌هایی که از یک گرامر به دست می‌آیند، و فقط از نمادهای پایانی تشکیل شده باشند (رشته‌های پایانی)، زبان مرتبط با آن گرامر را تعریف می‌کنند.

گرامرها

- فرض کنید $G = (V, T, S, P)$ یک گرامر باشد.
- آنگاه مجموعهٔ $L(G) = \{w \in T^* : S \overset{*}{\Rightarrow} w\}$ زبان تولید شده با استفاده از گرامر G است.
- اگر $w \in L(G)$ باشد، آنگاه دنبالهٔ $S \Rightarrow w_۱ \Rightarrow w_۲ \Rightarrow \cdots \Rightarrow w_n \Rightarrow w$ یک **نتیجه‌گیری (اشتقاق)**[1] از جملهٔ w است.
- همهٔ رشته‌های $S, w_۱, \cdots w_n$ که از متغیرها و نمادهای پایانی تشکیل شده‌اند، **صورت‌های جمله‌ای**[2] از این نتیجه‌گیری هستند.

---

[1] derivation
[2] sentential form

گرامرها

- گرامر $G = (\{S\}, \{a, b\}, S, P)$ با این قوانین تولید را در نظر بگیرید: $S \to \lambda \quad , \quad S \to aSb$
- بنابراین می‌توانیم داشته باشیم : $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$
- می‌توانیم بنویسیم : $S \overset{*}{\Rightarrow} aabb$
- رشتهٔ aabb یک جملهٔ تولید شده توسط این گرامر است و رشتهٔ aaSbb یک صورت جمله‌ای است.
- می‌توانیم حدس بزنیم که گرامر G زبان $L(G)$ را بدین صورت تعریف می‌کند : $L(G) = \{a^n b^n : n \geq 0\}$
- می‌توانیم این حدس را با استفاده از اثبات استقرایی اثبات کنیم.

گرامرها

- گرامر $G = (\{S\}, \{a, b\}, S, P)$ با این قوانین تولید را در نظر بگیرید: $S \rightarrow \lambda \quad , \quad S \rightarrow aSb$
- می‌توانیم حدس بزنیم که گرامر G زبان $L(G)$ را بدین صورت تعریف می‌کند : $L(G) = \{a^n b^n : n \geq 0\}$
- اول اینکه باید نشان دهیم جملهٔ $a^n b^n$ از این گرامر به دست می‌آید.
- کافی است قانون $S \rightarrow aSb$ را n بار و قانون $S \rightarrow \lambda$ را یک بار اعمال کنیم.

- دوم اینکه باید نشان دهیم که گرامر G فقط زبان $L(G)$ را تولید می‌کند.
- پس باید نشان دهیم که همهٔ صورت‌های جمله‌ای بدین شکل هستند : $(۱) \quad w_i = a^i S b^i$
- فرض استقرا این است که تساوی (۱) برای $i = ۱$ برقرار است.
- حال اگر تساوی (۱) برای هر n $\circ \leq i \leq n$ برقرار باشد، باید نشان دهیم برای $i = n + ۱$ نیز برقرار است.
- برای به دست آوردن $w_{n+۱}$ فقط می‌توانیم یک قانون را اعمال کنیم و آن هم قانون $S \rightarrow aSb$ است. پس الزاما داریم: $w_{n+۱} = a^{n+۱} S b^{n+۱}$
- نتیجه می‌گیریم همهٔ صورت های جمله‌ای این گرامر به شکل تساوی (۱) هستند، پس گرامر G فقط زبان $L(G)$ را تولید می‌کند.

- گرامری را بیابید که زبان L را تولید می‌کند : $L = \{a^n b^{n+1} : n \geq 0\}$

- گرامری را بیابید که زبان L را تولید می‌کند : $L = \{a^n b^{n+1} : n \geq 0\}$
- گرامر $G = (\{S, A\}, \{a, b\}, S, P\}$
- با قوانین تولید : $S \to Ab \;\; , \;\; A \to aAb \;\; , \;\; A \to \lambda$

گرامرها

- می‌توانیم در بسیاری مواقع جواب را به طور شهودی حدس بزنیم.
- برای اینکه نشان دهیم زبان L که حدس زده‌ایم توسط گرامر G تولید می‌شود، باید:
- (۱) نشان دهیم که همهٔ $w \in L$ توسط گرامر G با شروع صورت جمله‌ای S به دست می‌آیند،
- (۲) هر جملهٔ به دست آمده‌ای از گرامر G در زبان L است.

- الفبای $\Sigma = \{a, b\}$ و گرامر G با قوانین زیر را در نظر بگیرید:
$$S \rightarrow SS \quad , \quad S \rightarrow \lambda \quad , \quad S \rightarrow aSb \quad , \quad S \rightarrow bSa$$
- زبان $L(G)$ را تعیین کنید.

گرامرها

- الفبای $\Sigma = \{a, b\}$ و گرامر G با قوانین زیر را در نظر بگیرید:
$$S \rightarrow SS \quad , \quad S \rightarrow \lambda \quad , \quad S \rightarrow aSb \quad , \quad S \rightarrow bSa$$
- فرض کنید $n_a(w)$ و $n_b(w)$ تعداد تکرارهای a و b را در رشتهٔ w را نشان می‌دهند.
- در اینصورت داریم: $L = \{w : n_a(w) = n_b(w)\}$

گرامرها

- $S \to SS \;,\; S \to \lambda \;,\; S \to aSb \;,\; S \to bSa$
- $L = \{w : n_a(w) = n_b(w)\}$
- اول اینکه باید اثبات کنیم هر جمله‌ای توسط گرامر G تولید می‌شود، در زبان L وجو دارد.
- همهٔ صورت‌های جمله‌ای که در گرامر G تولید می‌شوند، تعداد a و b برابر دارند، چرا که قوانینی که یک a به صورت جمله‌ای اضافه می‌کنند، یک b نیز اضافه می‌کنند.
- بنابراین هر جمله‌ای که در $L(G)$ است، در L نیز وجود دارد.

گرامرها

- $S \to SS \;\; , \;\; S \to \lambda \;\; , \;\; S \to aSb \;\; , \;\; S \to bSa$
- $L = \{w : n_a(w) = n_b(w)\}$
- دوم اینکه باید اثبات کنیم همهٔ جمله‌هایی که در L وجود دارند، توسط گرامر G به دست می‌آیند.
- حالت‌های مختلف را در نظر می‌گیریم:
- اگر $w = aw_1b$ باشد، از آنجایی که تعداد a و b های w برابر است، تعداد a و b های w_1 هم برابر است. پس w_1 هم در L است. بنابراین w از این قانون به وجود آمده است : $S \to aSb$
- اگر $w = bw_1a$ باشد، به طور مشابه با حالت اول استدلال می‌کنیم.

گرامرها

- اگر $w = a \cdots a$ باشد، به صورت زیر استدلال می‌کنیم. فرض کنید به ازای مشاهدهٔ هر a یک واحد به یک شمارنده اضافه کنیم و به ازای مشاهدهٔ هر b یک واحد از شمارنده کم کنیم. در پایان این شمارش باید شمارنده به صفر برسد (چون تعداد a و b برابر است).
- پس در این حالت سوم که w با a شروع می‌شود و پایان می‌یابد، شمارنده باید قبل از اتمام رشته، یک بار به صفر رسیده باشد. در آن نقطه که شمارنده به صفر رسیده است، رشته‌ای به نام w_1 به دست آورده‌ایم که تعداد a و b در آن برابر است. دوباره از نقطه صفر در وسط رشته شروع می‌کنیم و در پایان شمارنده به صفر می‌رسد. پس در قسمت دوم رشته‌ای به نام w_2 به دست آورده‌ایم که تعداد a و b در آن برابر است.
- بنابراین در این حالت سوم $w = w_1 w_2$ جایی که هر دو رشتهٔ w_1 و w_2 در زبان هستند. پس برای این رشته می‌توانیم از قانون $S \rightarrow SS$ استفاده کنیم.
- در حالت چهارم $w = b \cdots b$ مشابه حالت سوم استدلال می‌کنیم.

گرامرها

- گرامر $G_1 = (\{A, S\}, \{a, b\}, S, P_1)$ با قوانین تولید P_1 به صورت زیر را در نظر بگیرید.
- $S \rightarrow aAb|\lambda \quad , \quad A \rightarrow aAb|\lambda$
- علامت خط عمودی | به معنی «فصل منطقی» (یا ی منطقی) است. بدین معنی که سمت چپ یک قانون می‌تواند عبارت اول در سمت راست قانون «یا» عبارت دوم در سمت راست قانون را نتیجه دهد.
- پس در این گرامر چهار قانون تولید داریم.
- زبان تولید شده توسط این گرامر برابر است با: $L(G_1) = \{a^n b^n : n \geq 0\}$

گرامرها

- گرامر $S \to aS|\lambda$ چه زبانی را تولید می‌کند؟
- گرامر $S \to aS|a$ چه زبانی را تولید می‌کند؟
- گرامر $S \to aS$ چه زبانی را تولید می‌کند؟

گرامرها

- گرامر $S \rightarrow aS|\lambda$ چه زبانی را تولید می‌کند؟ $L_{1} = \{\lambda, a, aa, \cdots\}$
- گرامر $S \rightarrow aS|a$ چه زبانی را تولید می‌کند؟ $L_{2} = \{a, aa, \cdots\}$
- گرامر $S \rightarrow aS$ چه زبانی را تولید می‌کند؟ $L_{3} = \{\} = \emptyset$

ماشین‌ها

- یک **ماشین** [1] (اتوماتون) مدلی انتزاعی از یک کامپیوتر (محاسبه‌گر) دیجیتال است.
- یک ماشین سازوکاری (مکانیزم) [2] برای خواندن رشتهٔ ورودی دارد (که این رشته‌ها بر روی الفبایی تعریف شده اند).
- رشتهٔ ورودی بر روی یک **فایل ورودی** [3] نوشته شده است که ماشین آن را می‌خواند.
- یک فایل ورودی از سلول [4] (خانه) هایی تشکیل شده است به طوری که هر نماد از رشتهٔ ورودی در یک سلول نوشته می‌شوند.

[1] automaton
[2] mechanism
[3] input file
[4] cell

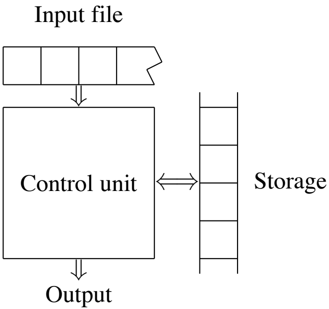
ماشین‌ها

- سازوکار خواندن رشتهٔ ورودی قادر به تشخیص انتهای رشته است.
- یک ماشین می‌تواند یک خروجی نیز تولید کند.
- یک ماشین می‌تواند یک **حافظه** [1] موقت نیز داشته باشد، که تشکیل شده است از تعداد نامحدودی سلول برای نگهداری نمادهایی از یک الفبا (لزوما این الفبا با الفبای رشتهٔ ورودی یکسان نیست).
- ماشین می‌تواند محتوای حافظه را بخواند و تغییر دهد.
- ماشین همچنین یک **واحد کنترل** [2] دارد که می‌تواند در یکی از **حالت‌های داخلی** [3] باشد، به طوری که تعداد حالت‌ها محدود است.
- ماشین می‌تواند با یک روش تعیین‌شده از یک حالت به یک حالت دیگر برود.

---
[1] storage
[2] control unit
[3] internal state

ماشین‌ها

– شکل زیر، یک ماشین را در حالت کلی نشان می‌دهد.



ماشین‌ها

- یک ماشین در بازه‌های زمانی گسسته عمل می‌کند.
- در هر نقطهٔ زمانی، واحد کنترل در یک حالت داخلی است و سازوکار ورودی، یک نماد را از فایل ورودی می‌خواند.
- حالت داخلی واحد کنترل در نقطهٔ زمانی بعدی، با یک **تابع گذار** [1] تعیین می‌شود.
- تابع گذار (انتقال)، حالت بعدی را بر اساس حالت فعلی، نماد فعلی بر روی ورودی، و اطلاعات فعلی روی حافظه تعیین می‌کند.
- در هر گذار از یک حالت به حالت دیگر، می‌تواند خروجی نیز تولید شود و یا اطلاعات روی حافظه تغییر کند.
- یک حالت از واحد کنترل، فایل ورودی، و حافظه را یک **پیکربندی** [2] می‌گوییم.
- گذار از یک پیکربندی به پیکربندی دیگر را یک **حرکت** [3] نامیم.

[1] transition function
[2] configuration
[3] move

ماشین‌ها

- همهٔ ماشین‌هایی که بررسی می‌کنیم، از این مدل کلی پیروی می‌کنند.
- همهٔ ماشین‌ها یک واحد کنترل دارند، اما تفاوت آنها در تولید خروجی و نوع حافظهٔ آنها است.
- خواهیم دید که نوع حافظه، توانایی ماشین‌ها را برای شناسایی زبان‌ها تعیین می‌کند.

ماشین‌ها

- ماشین‌ها می‌توانند **قطعی** [1] و یا **غیر قطعی** [2] باشند.
- یک ماشین قطعی، ماشینی است که در آن هر حرکت به طور منحصر به فرد به ازای پیکربندی فعلی تعیین شده است.
- بدین معنی که در یک ماشین قطعی، اگر حالت داخلی، ورودی، و محتوای حافظه را بدانیم، می‌توانیم دقیقا رفتار ماشین را در آینده پیش‌بینی کنیم.
- در یک ماشین غیرقطعی، در هر نقطهٔ زمانی، با توجه به پیکربندی فعلی، چندین امکان برای حرکت وجود دارد، پس تنها می‌توانیم مجموعه‌ای از حرکت‌های ممکن را پیش‌بینی کنیم.

---

[1] deterministic
[2] nondeterministic

ماشین‌ها

- یک ماشین که خروجی آن فقط «بله» و «خیر» است را یک **پذیرنده** [1] می‌نامیم.
- به ازای یک رشتهٔ داده شده، یک پذیرنده رشته را یا قبول می‌کند و یا رد می‌کند.
- ماشینی که توانایی تولید رشته‌ها در خروجی را داشته باشد، **مبدل** [2] می‌نامیم.

[1] accepter
[2] transducer

کاربردها

- علاوه بر اینکه یادگیری مبحث زبان‌ها و ماشین‌ها به ما روش فکر کردن در مسائل محاسباتی را می‌آموزد، زبان‌های صوری و گرامرها به طور گسترده‌ای در طراحی زبان‌های برنامه‌نویسی و کامپایلرها کاربرد دارند.
- برای طراحی یک زبان برنامه‌نویسی و تولید یک کامپایلر ابتدا نیاز به تعریف گرامر آن زبان داریم.

کاربردها

- برای مثال، قوانین تعریف یک متغیر در زبان برنامه‌نویسی سی چنین است:
- نام متغیر (۱) دنباله‌ای از حروف انگلیسی (کوچک و بزرگ)، ارقام، و زیرخط [1] است و (۲) تنها می‌تواند با حروف و زیرخط آغاز شود.
- پس گرامر آن را می‌توانیم بدین صورت تعریف کنیم:

$$
\begin{aligned}
<\text{name}> &\rightarrow <\text{letter}><\text{rest}> \mid <\text{uscore}><\text{rest}> \\
<\text{rest}> &\rightarrow <\text{letter}><\text{rest}> \mid <\text{digit}><\text{rest}> \mid <\text{uscore}><\text{rest}> \mid \lambda \\
<\text{letter}> &\rightarrow \text{a}|\text{b}|\cdots|\text{z}|\text{A}|\text{B}|\cdots|\text{Z} \\
<\text{digit}> &\rightarrow \text{۰}|\text{۱}|\cdots|\text{۹} \\
<\text{uscore}> &\rightarrow \_
\end{aligned}
$$

[1] underscore

کاربردها

$$
\begin{aligned}
<\text{name}> &\rightarrow <\text{letter}><\text{rest}> \mid <\text{uscore}><\text{rest}> \\
<\text{rest}> &\rightarrow <\text{letter}><\text{rest}> \mid <\text{digit}><\text{rest}> \mid <\text{uscore}><\text{rest}> \mid \lambda \\
<\text{letter}> &\rightarrow \text{a}|\text{b}|\cdots|\text{z}|\text{A}|\text{B}|\cdots|\text{Z} \\
<\text{digit}> &\rightarrow \text{۰}|\text{۱}|\cdots|\text{۹} \\
<\text{uscore}> &\rightarrow \_
\end{aligned}
$$

- در مثال فوق مجموعهٔ متغیرها شامل $<\text{name}>, <\text{rest}>, <\text{letter}>, <\text{digit}>, <\text{uscore}>$ و مجموعهٔ پایانه‌ها شامل حروف و ارقام و زیرخط می‌شوند.

- نام متغیر a۰ را از این گرامر اینگونه به دست می‌آوریم:

$$<\text{name}> \Rightarrow <\text{letter}><\text{rest}> \Rightarrow \text{a} <\text{rest}> \Rightarrow \text{a} <\text{digit}><\text{rest}> \Rightarrow \text{a۰} <\text{rest}> \Rightarrow \text{a۰}$$

کاربردها

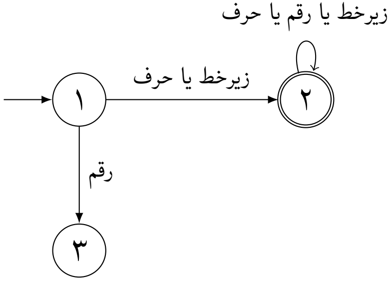
- همچنین می‌توانیم ماشینی طراحی کنیم که به ازای یک رشته داده شده پاسخ دهد که آیا رشتهٔ داده شده به عنوان یک نام متغیر مورد قبول است یا خیر.
- چنین ماشینی چنانکه اشاره شد یک پذیرنده است.

کاربردها

- واحد کنترل یک ماشین (پذیرنده و یا مبدل) معمولا به صورت یک گراف نمایش داده می‌شود.
- عملیات محاسبه (پذیرش یا تبدیل) از یکی از رأس‌ها (که با یک یال بدون مبدأ مشخص شده است) آغاز می‌شود.
- در هر واحد زمان، ماشین یک نماد را از ورودی می‌خواند.
- فرض کنید یالی با برچسب a از رأس x به رأس y وجود دارد. وجود این یال بدین معنی است که ماشین با خواندن نماد a از ورودی از حالت x به حالت y می‌رود.
- اگر خواندن یک رشته در حالتی به پایان برسد که با دو دایرهٔ تودرتو نشان داده شده است، رشته پذیرفته می‌شود.

کاربردها

- ماشین زیر نام یک متغیر را در زبان برنامه‌نویسی سی می‌پذیرد یا رد می‌کند.
- ماشین در حالت ۱ آغاز می‌شود. در هر گام، یک نماد از رشتهٔ ورودی خوانده می‌شود.
- اگر رشته‌ای در حالت ۲ پایان یابد، رشتهٔ مورد نظر پذیرفته می‌شود، در غیر اینصورت رشته رد می‌شود.



کاربردها

- در گراف واحد کنترل مربوط به یک مبدل، فرض کنید یالی با برچسب a/b از رأس x به y وجود دارد. وجود این یال بدین معنی است که با خواندن نماد a ماشین از حالت x به حالت y می‌رود و نماد b را در خروجی تولید می‌کند.
- برچسب a/λ بدین معنی است که با خواندن نماد a از ورودی، ماشین نماد λ را در خروجی تولید می‌کند و یا به عبارتی نمادی در خروجی نمی‌نویسد.

کاربردها

- یک مبدل طراحی کنید که یک عدد دودویی را به صورت یک رشته دریافت کند و معادل آن را در مبنای ۸ در خروجی بنویسد. برای مثال با دریافت رشتهٔ ۰۰۱۱۰۱۱۱۰ از ورودی، مبدل رشتهٔ ۱۵۶ را تولید می‌کند. فرض کنید طول رشتهٔ ورودی همیشه مضربی از ۳ است.

کاربردها

- یک مبدل طراحی کنید که یک عدد دودویی را به صورت یک رشته دریافت کند و معادل آن را در مبنای ۸ در خروجی بنویسد. برای مثال با دریافت رشتهٔ ۰۰۱۱۰۱۱۱۰ از ورودی، مبدل رشتهٔ ۱۵۶ را تولید می‌کند. فرض کنید طول رشتهٔ ورودی همیشه مضربی از ۳ است.