۹ ماشین‌های تورینگ

TURING MACHINES

- مفهوم آتوماتا تا کجا می‌تواند ادامه یابد؟
- قوی‌ترین آتوماتون چیست؟
- محدوده‌ی محاسباتی آن تا کجاست؟

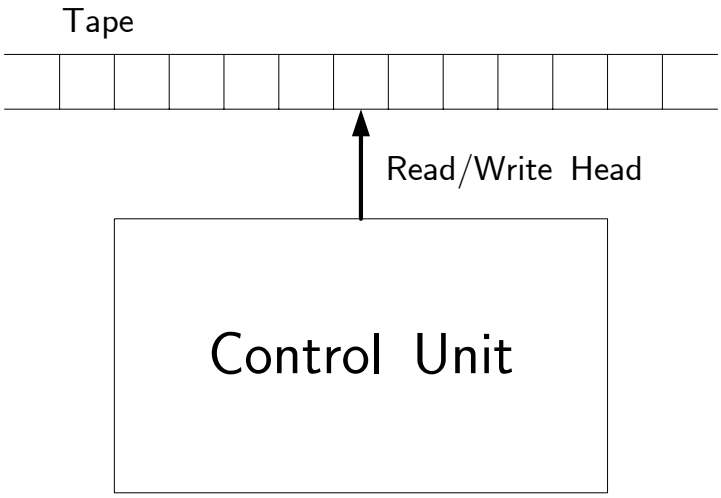
این پرسش‌ها ما را به بحث اساسی ماشین تورینگ و در نتیجه تعریف دقیق از ایده‌ی محاسبات الگوریتمی یا مکانیکی می‌رساند.

۹-۱ ماشین تورینگ استاندارد

حافظه‌ی ماشین تورینگ، یک آرایه‌ی یک بعدی از سلول‌هاست که هر سلول می‌تواند یک نماد را نگهداری کند. این آرایه از دو سر نامحدود است و می‌تواند بی‌شمار نماد را در خود ذخیره کند. اطلاعات می‌تواند به ترتیب دلخواه از این آرایه خوانده شده و نوشته شود. به این دستگاه ذخیره‌سازی اصطلاحاً **نوار** (tape) می‌گوییم.

۹-۱-۱ تعریف ماشین تورینگ

- ماشین تورینگ، یک آتوماتون است که حافظه‌ی موقت آن نوار است.
- این نوار به سلول‌هایی تقسیم شده است که هر سلول آن قادر به ذخیره کردن یک نماد است.
- متناظر با این نوار، یک هد خواندن / نوشتن وجود دارد که می‌تواند به راست و چپ حرکت کند و در هر حرکت یک نماد را بخواند / بنویسد.
- ماشین تورینگ فایل ورودی یا مکانیزم خروجی بخصوصی ندارد و هر نوع تغییر ورودی / خروجی به واسطه‌ی نوار انجام می‌شود (وجود فایل ورودی و خروجی تغییری در نتیجه به وجود نمی‌آورد).



تعریف **ماشین تورینگ** ماشین تورینگ به صورت زیر تعریف می‌شود:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$$

که در آن:

Q: مجموعه‌ی **حالات** داخلی

Σ: **الفبای ورودی**

Γ: **الفبای نوار** $\Sigma \subseteq \Gamma - \{\square\}$

δ: **تابع گذر حالت** که به صورت زیر تعریف می‌شود:

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$$

$\square$: **علامت فاصله‌ی خالی** *(blank)* که $\square \in \Gamma$

q_0: **حالت اولیه** $q_0 \in Q$

F: مجموعه‌ی **حالات نهایی** $F \subseteq Q$

◄ **تذکر** علامت فاصله‌ی خالی نمی‌تواند به عنوان ورودی استفاده شود.

◄ δ یک تابع جزیی روی $Q \times \Gamma$ است و تفسیر آن نحوه‌ی عملکرد ماشین تورینگ را روشن می‌کند (برنامه‌ی ماشین تورینگ).

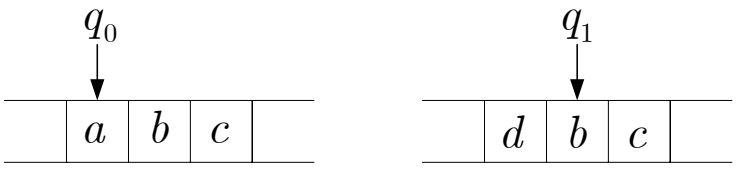
◄ آرگومان‌های δ عبارتند از: حالت فعلی واحد کنترل و حرف فعلی خوانده شده از روی نوار

◄ حاصل این تابع، یک حالت جدید در واحد کنترل، یک نماد جدید که جایگزین حرف خوانده شده از روی نوار می‌شود و یک علامت حرکت به سمت راست $(R, \rightarrow)$ یا چپ $(L, \leftarrow)$ می‌باشد.

◄ علامت حرکت تعیین می‌کند که بعد از جایگزینی حرف جدید بر روی نوار، هد خواندن و نوشتن باید یک سلول به سمت راست یا یک سلول به سمت چپ برود.

مثال

اگر گذر $\delta(q_0, a) = (q_1, d, R)$، محتوای به صورت زیر تغییر می‌کند:



- ماشین تورینگ یک کامپیوتر ساده است که دارای واحد پردازشی با حافظه‌ی محدود و نواری با حافظه‌ی نامحدود است.
- دستورات این کامپیوتر بسیار محدود، اما کافی است.
- تابع گذر حالت δ عملکرد ماشین را تعریف می‌کند و برنامه‌ی ماشین نام دارد.
- آتوماتون با یک حالت ابتدایی و مقداری اطلاعات بر روی نوار شروع می‌کند و سپس قدم‌هایی را که به واسطه‌ی تابع گذر حالت کنترل می‌شود، برمی‌دارد.
- در طول روال فوق، محتوای سلول‌ها می‌تواند آزمایش شود.
- ماشین تورینگ، زمانی به حالت توقف می‌رسد که وارد یک پیکربندی شود که در آن δ تعریف نشده باشد.
- در واقع فرض می‌کنیم که در حالات نهایی، δ تعریف نشده است و بنابراین ماشین تورینگ با ورود به حالت نهایی متوقف می‌شود.

مثال

ماشین تورینگ با

$$Q = \{q_0, q_1\}, \quad \Sigma = \{a, b\}, \quad \Gamma = \{a, b, \Box\}, \quad F = \{q_1\}$$

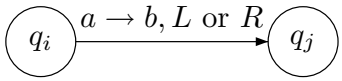
و تابع گذر

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

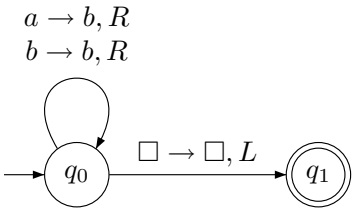
حرف‌های a را با b جایگزین می‌کند، اما bها را تغییر نمی‌دهد.
وقتی ماشین به اولین فاصله‌ی خالی برسد، یک سلول به عقب می‌رود و در حالت نهایی q_1 توقف می‌کند.

بازنمایی یک ماشین تورینگ با دیاگرام گذر حالت نیز انجام می‌شود:

$$\delta(q_i, a) = (q_j, b, L \text{ or } R)$$



برای مثال، دیاگرام گذر حالت برای ماشین تورینگ مثال فوق به صورت زیر خواهد بود:



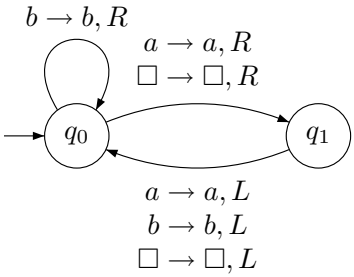
مثال

ماشین تورینگ با

$$Q = \{q_0, q_1\}, \quad \Sigma = \{a, b\}, \quad \Gamma = \{a, b, \square\}, \quad F = \varnothing$$

و تابع گذر

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



این ماشین هرگز توقف نخواهد کرد:
اگر ابتدای نوار $ab\ldots$ باشد و هد خواندن / نوشتن روی a قرار داشته باشد،
ماشین a را می‌خواند، اما آن را تغییر نمی‌دهد و به حالت q_1 می‌رود و هد روی b قرار می‌گیرد.
ماشین b را می‌خواند، اما آن را تغییر نمی‌دهد و به حالت q_0 بر می‌گردد و هد به سمت چپ می‌رود.
حال دقیقاً به وضعیت اولیه برگشته‌ایم و دنباله حرکت‌ها تکرار می‌شود.

واضح است که اطلاعات اولیه روی نوار هرچه باشد، ماشین هرگز توقف نمی‌کند و هد مرتباً به راست یا چپ می‌رود، بدون اینکه تغییری ایجاد شود.

$\Leftarrow$ ماشین تورینگ در یک حلقه‌ی نامتناهی گرفتار می‌شود.

▲

ماشین تورینگ استاندارد و خواص آن به صورت قراردادی:

۱) دارای نواری است که از دو سر نامحدود است، پس هر تعداد حرکت به سمت راست یا چپ امکان دارد.

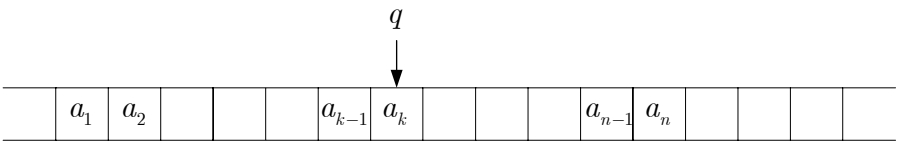
۲) ماشین تورینگ قطعی است (به ازای هر پیکربندی یک حرکت تعریف می‌شود).

۳) فایل ورودی خاصی وجود ندارد. فرض می‌کنیم در ابتدای کار نوار دارای محتوای مشخصی است و می‌توان بخشی از آن را به عنوان خروجی در نظر گرفت.
دستگاه خروجی خاصی هم وجود ندارد. وقتی ماشین توقف می‌کند، می‌توان خروجی را بخشی از محتوای نوار دانست.

توصیف بلافصل برای تعیین پیکربندی هر پیکربندی ماشین تورینگ با حالت فعلی واحد کنترل، محتوای نوار و موقعیت هد خواندن / نوشتن مشخص می‌شود. به قالب کلی

$$a_1 a_2 \dots a_{k-1}\ q\ a_k \dots a_{n-1} a_n$$

یک توصیف بلافصل از پیکربندی گفته می‌شود که بازنمایی وضعیت زیر در ماشین تورینگ است:



فرض می‌شود که قسمت قبل از a_1 و بعد از a_n حاوی فاصله‌ی خالی $\square$ است و تاثیری ندارد. اگر فاصله‌های خالی بر بحث تاثیر داشته باشند، آنگاه لازم است که آنها را نیز نشان بدهیم.

توصیف بلافصل $q\square w$ یعنی اینکه هد روی سلول سمت چپ نسبت به اولین حرف w است که یک $\square$ است.

همانند دیگر آتوماتا، حرکت از یک پیکربندی به پیکربندی دیگر با $\vdash$ نشان داده می‌شود.

مثال

▼

$$\delta(q_1, c) = (q_2, e, R) \quad \Rightarrow \quad ab\ q_1\ cd \vdash abe\ q_2\ d$$

▲

تعریف فرض کنید که $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ یک ماشین تورینگ باشد. در این صورت رشته‌ای مانند

$$a_1 a_2 \dots a_{k-1}\ \mathbf{q_1}\ a_k a_{k+1} \dots a_n$$

با $q_1 \in Q$ و $a_i \in \Gamma$، یک توصیف بلافصل از M است.
حرکتی مانند

$$a_1 a_2 \dots a_{k-1}\ \mathbf{q_1}\ a_k a_{k+1} \dots a_n \vdash a_1 a_2 \dots a_{k-1} b\ \mathbf{q_2}\ a_{k+1} \dots a_n$$

امکان‌پذیر است اگر و فقط اگر

$$\delta(q_1, a_k) = (q_2, b, R)$$

و حرکتی مانند

$$a_1 a_2 \dots a_{k-1}\ \mathbf{q_1}\ a_k a_{k+1} \dots a_n \vdash a_1 \dots\ \mathbf{q_2}\ a_{k-1} b a_{k+1} \dots a_n$$

امکان‌پذیر است اگر و فقط اگر

$$\delta(q_1, a_k) = (q_2, b, L)$$

گفته می‌شود ماشین M با شروع از پیکربندی آغازین مانند $x_1\ q_i\ x_2$ توقف *(halt)* می‌کند اگر

$$x_1\ q_i\ x_2 \vdash^* y_1\ q_j\ a y_2$$

برای هر a و q_j که برای آن $\delta(q_j, a)$ تعریف شده نباشد، صحت داشته باشد.
دنباله‌ای از پیکربندی‌ها که به یک حالت توقف منتهی می‌شود، یک **محاسبه** گفته می‌شود.
اگر از پیکربندی اولیه‌ی $x_1\ q\ x_2$ ماشین تورینگ هرگز توقف نکند، می‌نویسیم

$$x_1\ q\ x_2 \vdash^* \infty$$

که به معنی ورود ماشین به حلقه‌ی نامتناهی است.

۹-۱-۲ ماشین تورینگ به عنوان پذیرنده‌ی زبان

رشته‌ی w روی نوار نوشته می‌شود و فضاهای باقیمانده با نماد $\square$ پر می‌شود.
ماشین از حالت q_0 شروع می‌کند، در حالی که هد بر روی سمت چپ‌ترین نماد w است.
اگر پس از دنباله‌ای از حرکات، ماشین تورینگ وارد یک حالت نهایی شود و متوقف گردد، آنگاه رشته‌ی w پذیرفته می‌شود.

تعریف

اگر $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ یک ماشین تورینگ باشد، آنگاه زبان پذیرفته شده توسط M به صورت زیر است:

$$L(M) = \{w \in \Sigma^+ : q_0\ w \vdash^* x_1\ q_f\ x_2, x_1, x_2 \in \Gamma^*, q_f \in F\}$$

ورودی w با فضای خالی در دو طرف آن روی نوار نوشته می‌شود. علت اینکه $\square$ از ورودی کنار گذاشته شده است، محدود کردن ورودی به قسمتی از نوار است که با $\square$ احاطه شده است. بدون انجام این کار ماشین نمی‌تواند محل ورودی را بیابد.
اگر $w \notin L(M)$،

- ماشین در یک حالت غیرنهایی متوقف می‌شود، یا
- وارد یک حلقه‌ی بی‌نهایت می‌شود و هرگز توقف نمی‌کند.

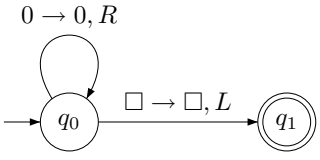
هر رشته‌ای که M با آن متوقف نمی‌شود، طبق تعریف در $L(M)$ نیست.

مثال

یک ماشین تورینگ برای پذیرش زبان $L(00^*)$ روی $\Sigma = \{0, 1\}$
از سمت چپ ورودی آغاز می‌کنیم، نمادها را یکی یکی می‌خوانیم و آن نماد را با 0 تطبیق می‌دهیم.
اگر نماد خوانده شده صفر بود، به سمت راست ادامه می‌دهیم، اگر تا رسیدن به فاصله‌ی خالی چیزی جز صفر ندیدیم، رشته پذیرفته می‌شود.
اگر در هر جایی از رشته 1 باشد، آن رشته در $L(00^*)$ نیست و در یک حالت غیرنهایی توقف می‌کنیم:

$$Q = \{q_0, q_1\}, \qquad F = \{q_1\}$$

$$\begin{aligned} \delta(q_0, 0) &= (q_0, 0, R) \\ \delta(q_0, \square) &= (q_1, \square, L) \end{aligned}$$



مثال

یک ماشین تورینگ برای پذیرش زبان $L = \{a^n b^n : n \geq 1\}$ روی $\Sigma = \{a, b\}$

روی اولین a شروع می‌کنیم و آن را با نمادی چون x جایگزین می‌کنیم.
سپس به سمت چپ‌ترین (اولین) نماد b می‌رویم و آن را با y جایگزین می‌نماییم.
دوباره به سمت‌چپ‌ترین a می‌رویم و آن را به x تبدیل می‌کنیم.
دوباره به سمت‌چپ‌ترین b می‌رویم و آن را به y تبدیل می‌کنیم.
در نهایت، اگر هیچ a یا b نمانده باشد، آنگاه رشته پذیرفته می‌شود.

$$Q = \{q_0, q_1, q_2, q_3, q_4\}, \quad F = \{q_4\}, \quad \Sigma = \{a, b\}, \quad \Gamma = \{a, b, x, y, \Box\}$$

$\delta(q_0, a)$	$=$	(q_1, x, R)	سمت‌چپ‌ترین a را به x تبدیل می‌کند و هد را به سمت راست می‌برد
$\delta(q_1, a)$	$=$	(q_1, a, R)	از aهای بعدی عبور می‌کند
$\delta(q_1, y)$	$=$	(q_1, y, R)	هد را بر روی اولین b قرار می‌دهد
$\delta(q_1, b)$	$=$	(q_2, y, L)	نماد b را به y تبدیل و به حالت q_2 می‌رود
$\delta(q_2, y)$	$=$	(q_2, y, L)	معکوس کردن گذرهای فوق تا رسیدن به x
$\delta(q_2, a)$	$=$	(q_2, a, L)	هد را بر روی سمت چپ‌ترین a قرار می‌دهد
$\delta(q_2, x)$	$=$	(q_0, x, R)	و به حالت اولیه برمی‌گردد

با یک دور اعمال قواعد فوق داریم

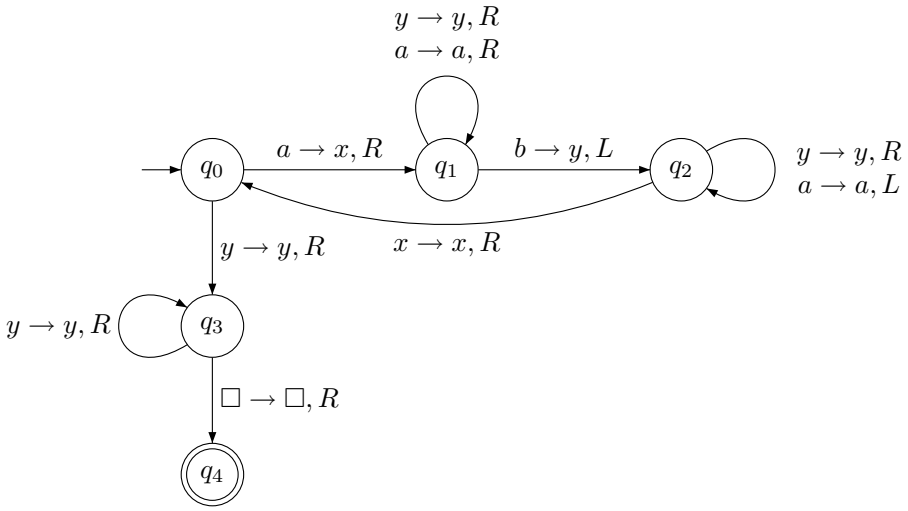
$$q_0\ aa \ldots abb \ldots b \vdash^* x\ q_0\ a \ldots ayb \ldots b$$

با دو دور اعمال قواعد فوق داریم

$$q_0\ aa \ldots abb \ldots b \vdash^* xx\ q_0\ \ldots ayy \ldots b$$

برای پایان دادن بررسی می‌کنیم که آیا همه‌ی aها و bها جایگزین شده‌اند یا خیر.

$$\begin{aligned} \delta(q_0, y) &= (q_3, y, R) \\ \delta(q_3, y) &= (q_3, y, R) \\ \delta(q_3, \Box) &= (q_4, \Box, R) \end{aligned}$$



اگر رشته‌ای که در زبان نیست وارد کنیم، محاسبات در یک حالت غیرنهایی متوقف می‌شود.

▲

مثال

▼

ماشین تورینگ برای پذیرش زبان $L = \{a^n b^n c^n : n \geq 1\}$.

هر a، b و c را به ترتیب با x، y و z جایگزین می‌کنیم.

در انتها بررسی می‌کنیم که آیا همه‌ی علایم اولیه جایگزین شده اند یا خیر.

▲

نتیجه ماشین تورینگ، زبان‌هایی که مستقل از متن نیستند را می‌پذیرد، یعنی ماشین تورینگ از PDA قوی‌تر است.

۹-۱-۳ ماشین تورینگ به عنوان تراگذر

ماشین تورینگ علاوه بر پذیرنده‌ی زبان، یک مدل ساده‌ی انتزاعی برای کامپیوترهای دیجیتال است.

هدف کامپیوتر، تبدیل ورودی به خروجی است، یعنی به صورت یک تراگذر عمل می‌کند.

ورودی یک محاسبه، تمامی نمادها بجز $\square$ را دربرمی‌گیرد.

خروجی چیزی است که در پایان محاسبه روی نوار باقی می‌ماند.

ماشین تورینگ تراگذر، پیاده‌سازی یک تابع f است که به صورت زیر تعریف می‌شود:

$$\hat{w} = f(w)$$

به شرط اینکه

$$q_0\ w \vdash_M^*\ q_f\ \hat{w}$$

برای حالت نهایی q_f برقرار باشد.

تعریف تابع f با دامنه‌ی D را محاسبه‌پذیر تورینگ *(turing computable)* یا قابل محاسبه می‌گوییم اگر یک ماشین تورینگ مانند $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ موجود باشد که برای هر رشته‌ی $w \in D$، M وارد یک حالت نهایی شود، یعنی:

$$\forall w \in D \qquad q_0\ w \vdash_M^* q_f\ f(w)$$

◄ **نکته** تمام توابع ریاضی معمول، حتی اگر بسیار پیچیده باشند، محاسبه‌پذیر تورینگ هستند.

مثال ▼

ماشین تورینگ جمع (جمع دو عدد صحیح x و y با حاصل $x + y$)

روش نمایش اعداد روی نوار
هر عدد صحیح مثبت مانند x را به صورت $w(x) \in \{1\}^+$ نمایش می‌دهیم به طوری که $|w(x)| = x$ باشد.

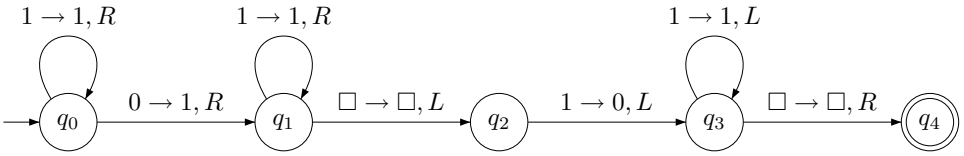
یعنی x به صورت x رقم ۱ و y به صورت y رقم ۱ نمایش داده می‌شود.
x و y روی نوار با یک صفر جدا می‌شوند.
پس از محاسبه، $x + y$ بر روی نوار باقی می‌ماند و به یک صفر ختم می‌شود.
هد در منتهی الیه سمت چپ این نتیجه قرار می‌گیرد، یعنی:

$$\mathbf{q_0}\ w(x)\mathbf{0}w(y) \vdash^* \mathbf{q_f}\ w(x+y)\mathbf{0}$$

کاری که باید انجام شود، بردن صفر بین دو عدد به انتهای سمت راست $w(y)$ است.
بنابراین جمع با الحاق دو رشته انجام می‌شود:

$$Q = \{q_0, q_1, q_2, q_3, q_4\}, \qquad F = \{q_4\}$$

$$\begin{aligned}
\delta(q_0, 1) &= (q_0, 1, R)\\
\delta(q_0, 0) &= (q_1, 1, R)\\
\delta(q_1, 1) &= (q_1, 1, R)\\
\delta(q_1, \square) &= (q_2, \square, L)\\
\delta(q_2, 1) &= (q_3, 0, L)\\
\delta(q_3, 1) &= (q_3, 1, L)\\
\delta(q_3, \square) &= (q_4, \square, R)
\end{aligned}$$



مثال

ماشین تورینگ کپی (کپی کردن رشته‌ای از ۱ها)

$$\forall w \in \{1\}^+ \qquad \mathbf{q_0}\ w \vdash^* \mathbf{q_f}\ ww$$

برای حل این مساله، روال زیر دنبال می‌شود:

(۱) هر ۱ را با x جایگزین می‌کنیم.

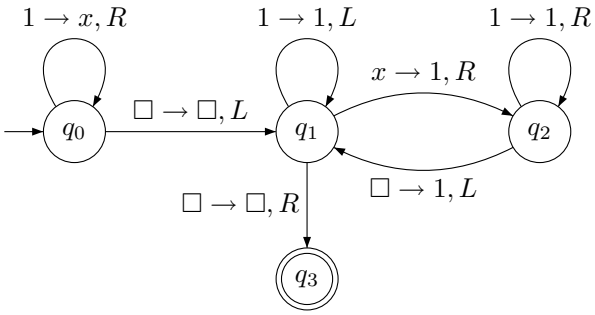
(۲) سمت راست‌ترین x را یافته، آن را با 1 جایگزین می‌کنیم.

(۳) به انتهای سمت راست ناحیه‌ی غیرخالی می‌رویم و در آنجا یک ۱ قرار می‌دهیم.

(۴) قدم‌های (۲) و (۳) را تکرار می‌کنیم تا دیگر xای باقی نماند.

$$\begin{aligned}
\delta(q_0, 1) &= (q_0, x, R)\\
\delta(q_0, \square) &= (q_1, \square, L)\\
\delta(q_1, x) &= (q_2, 1, R)\\
\delta(q_2, 1) &= (q_2, 1, R)\\
\delta(q_2, \square) &= (q_1, 1, L)\\
\delta(q_1, 1) &= (q_1, 1, L)\\
\delta(q_1, \square) &= (q_3, \square, R)
\end{aligned}$$

که در آن q_3 تنها حالت نهایی است.



۹-۲ ترکیب ماشین‌های تورینگ برای انجام کارهای پیچیده

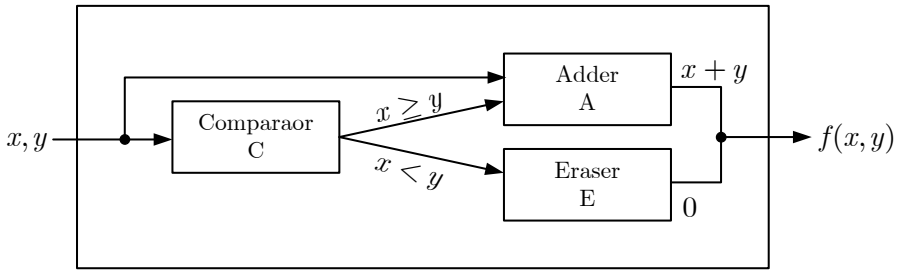
۹-۲-۱ توصیف ماشین تورینگ با دیاگرام بلوکی

مثال

یک ماشین تورینگ برای محاسبه‌ی تابع

$$f(x,y)=\begin{cases} x+y & ,x\geq y \\ 0 & ,x<y \end{cases}$$

x و y اعداد صحیح مثبت در نمایش 1^+
مقدار 0 با 0 و بقیه‌ی نوار به صورت فضای خالی نمایش داده می‌شود.
این ماشین تورینگ را به صورت دیاگرام بلوکی نمایش می‌دهیم:



ابتدا از ماشین مقایسه‌گر $\{a^nb^n\}$ استفاده می‌کنیم تا مشخص کند آیا $x\geq y$ است یا خیر.
اگر چنین بود، مقایسه‌گر یک دستور شروع را به جمع‌کننده می‌فرستد تا $x+y$ را محاسبه کند.
در غیر این صورت یک برنامه‌ی پاک‌کننده فراخوانی می‌شود که همه‌ی 1ها را به فضای خالی تبدیل کند.
ارسال دستور به چه صورتی انجام می‌شود؟
محاسبات قابل انجام توسط مقایسه‌کننده‌ی C:

$$q_{C,0}w(x)0w(y)\vdash^* q_{A,0}w(x)0w(y) \quad ,x\geq y$$
$$q_{C,0}w(x)0w(y)\vdash^* q_{E,0}w(x)0w(y) \quad ,x<y$$

اگر $q_{A,0}$ و $q_{E,0}$ حالات شروع A و E باشند، می‌بینیم که C، A یا E را به کار می‌اندازد.

محاسبات انجام شده توسط جمع‌کننده: $q_{A,0}w(x)0w(y)\vdash^* q_{A,f}w(x+y)0$
محاسبات انجام‌شده توسط پاک‌کننده: $q_{E,0}w(x)0w(y)\vdash^* q_{E,f}0$

نتیجه‌ی اعمال فوق، ماشین تورینگی است که اعمال C، A و E را مطابق شکل فوق ترکیب می‌کند.

۹-۲-۲ توصیف ماشین تورینگ با استفاده از شبه کد

مثال

if a **then** q_j **else** q_k

اگر ماشین تورینگ a را بخواند، بدون توجه به حالت فعلی و بدون تغییر دادن محتوای نوار یا حرکت دادن هد، به حالت q_j می‌رود.

اگر ورودی چیزی غیر از a باشد، ماشین بدون انجام هیچ تغییری به q_k می‌رود.

$$\begin{aligned}
&\forall q_i \in Q && \delta(q_i, a) = (q_{j0}, a, R)\\
&\forall q_i \in Q, \forall b \in \Gamma - \{a\} && \delta(q_i, b) = (q_{k0}, b, R)\\
&\forall c \in \Gamma && \delta(q_{j0}, c) = (q_j, c, L)\\
&\forall c \in \Gamma && \delta(q_{k0}, c) = (q_k, c, L)
\end{aligned}$$

q_{j0} و q_{k0} حالت‌های جدید هستند و برای این استفاده می‌شوند که هد ماشین تورینگ در هر حرکت به طور معمول جابجا می‌شود ولی در دستور فوق ما نمی‌خواهیم این جابجایی صورت گیرد. با این حالت‌های جدید اجازه می‌دهیم که هد حرکت کند، اما مجدداً برگردد.

می‌توان یک دستور سطح بالا را با یک زیربرنامه جایگزین نمود.

یعنی وقتی ماشین تورینگ به صورت یک زیربرنامه استفاده می‌شود، ماشین تورینگ دیگری می‌تواند به طور مکرر آن را فراخوانی کند:

این بدان معنی است که باید اطلاعاتی در مورد برنامه‌ی فراخوانی کننده ذخیره کنیم تا پس از بازگشت از زیربرنامه بتوان آن پیکربندی را ایجاد نمود.

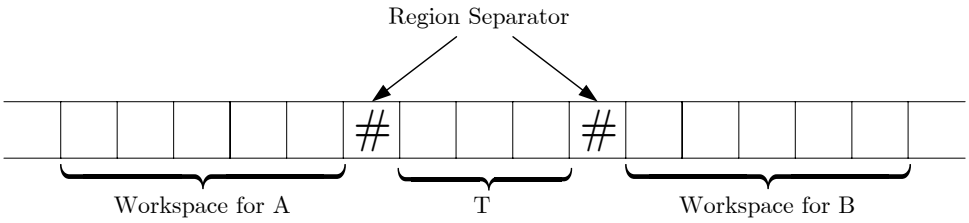
برای مثال،

ماشین A را در حالت q_i، restart می‌کنیم به طوری که هد در محل قبلی‌اش قرار گیرد (چون ممکن است در هنگام کار B حرکت کرده باشد).

در یک زمان دیگر، ممکن است که ماشین A ماشین B را از وضعیت q_j فراخوانی کند که در آن مورد، باید کنترل به این حالت برگردد.

* برای حل مساله‌ی انتقال کنترل باید بتوانیم از A به B و برعکس اطلاعاتی را رد و بدل نماییم تا بتوانیم پیکربندی A را وقتی که کنترل را از B پس می‌گیرد، بازسازی کنیم تا مطمئن شویم محاسباتی که موقتاً در A متوقف شده بود، از اجرای B متاثر نمی‌شود.

برای حل این مساله، می‌توان نوار را به چند قسمت تقسیم کرد:



قبل از اینکه ماشین A ماشین B را فراخوانی کند، اطلاعاتی که B نیاز دارد (مانند حالت A و آرگومان‌های مورد نیاز B) را بر روی قسمت T نوار می‌نویسد (اطلاعات موقتی).
سپس A با انجام تغییر حالت به حالت اولیه‌ی B، کنترل را به B می‌سپارد.
پس از این انتقال کنترل، B ورودی خود را از T می‌گیرد.
چون ناحیه‌ی کاری B از ناحیه‌ی کاری A و T جداست، بنابریان تداخلی رخ نمی‌دهد.
وقتی B کارش به پایان می‌رسد، نتایج مربوطه را در T می‌نویسد و انتظار دارد که A آن را پیدا کند.
به این روش دو ماشین می‌توانند به نحو لازم با هم ارتباط برقرار کنند.

◄ **تذکر** از این پس می‌توانیم برنامه‌های ماشین تورینگ را به شبه کد بنویسیم، به شرط آنکه بدانیم چگونه می‌توان این شبه کد را به برنامه‌های ماشین تورینگ تبدیل نماییم.

مثال

ماشین تورینگ ضرب (ضرب دو عدد صحیح مثبت با نمایش 1^+) با ترکیب ماشین‌های جمع و کپی

$$00\underbrace{1\,1\,1}_{y}0\underbrace{1\,1}_{x} \quad \vdash \quad 0\underbrace{1\,\ldots\,1}_{xy}0\underbrace{1\,1\,1}_{y}0\underbrace{1\,1}_{x}$$

۱) گام‌های زیر را آن‌قدر تکرار کنید تا رقم 1 دیگری در x باقی نماند.

- یک رقم 1 در x پیدا کنید و آن را با یک a جایگزین کنید.
- سمت چپ‌ترین 0 را با $0y$ جایگزین کنید.

۲) همه‌ی aها را با 1 جایگزین نمایید.

ماشین‌های تورنگ ابتدایی بسیار ساده هستند، اما در ترکیب با یکدیگر می‌توانند کارهای پیچیده‌ای انجام دهند.

۹-۳ تز تورینگ

می‌خواهیم به این پرسش پاسخ دهیم که آیا ماشین‌های تورینگ از نظر قدرت معادل با کامپیوترهای دیجیتال هستند یا خیر؟

می‌توان مجموعه دستورالعمل‌های یک کامپیوتر نمونه را با ماشین تورینگ پیاده‌سازی نمود، اما مشکل در اینجاست که معنی کامپیوتر نمونه مشخص نیست.

می‌توان سعی کرد روالی پیدا نمود که بتوان برای آن یک برنامه‌ی کامپیوتری نوشت، اما برای آن ماشین تورینگی یافت نشود. تاکنون کسی موفق به انجام چنین کاری نشده است.

همه‌ی شواهد نشان می‌دهد که ماشین تورینگ به اندازه‌ی کامپیوتر دیجیتال قدرتمند است.

تز تورینگ هر گونه محاسبه‌ای که به طور مکانیکی قابل انجام باشد، با ماشین تورینگ نیز قابل انجام است.

این تز قابل اثبات نیست. برای اثبات این تز لازم است که عبارت «به طور مکانیکی» را دقیقاً تعریف کرد که باعث می‌شود یک مدل انتزاعی جدید را تعریف کنیم که ما را به جایی فراتر از ماشین تورینگ نمی‌رساند، پس:

تعریف به یک محاسبه، مکانیکی گفته می‌شود اگر و فقط اگر بتواند با ماشین تورینگ انجام شود.

اگر ماشین تورینگ را یک تعریف در نظر بگیریم، آیا این تعریف آن‌قدر جامع است که تمام کارهای قابل انجام توسط کامپیوتر را پوشش دهد؟ پاسخ مثبتی که بر روی آن اجماع وجود داشته باشد وجود ندارد، اما شواهد در تایید آن بسیار قوی است:

۱) هر کاری که با یک کامپیوتر دیجیتال قابل انجام است، با ماشین تورینگ هم قابل انجام است.

۲) تا کنون کسی نتوانسته است مساله‌ای را پیدا کند که با الگوریتم قابل حل باشد، ولی نتوان برای آن ماشین تورینگ نوشت.

۳) مدل‌های دیگری برای محاسبه‌ی مکانیکی پیشنهاد شده است، اما هیچ‌یک از آنها قوی‌تر از ماشین تورینگ نیستند.

تز تورینگ در حد یک فرضیه باقی مانده است.

تز تورینگ همان نقشی را در علوم کامپیوتر ایفا می‌کند که قوانین اساسی در فیزیک و شیمی ایفا می‌کنند. برای مثال، فیزیک کلاسیک بر پایه‌ی قوانین نیوتون استوار است. گرچه به آنها قانون می‌گوییم، اما هرگز اثبات نشده‌اند و فقط چون بر تجربیات و مشاهدات منطبق هستند، پذیرفته شده‌اند. اگر یک نتیجه‌ی تجربی به دست آید که مغایر با این قوانین باشد، آنگاه این قوانین زیر سوال می‌روند. در مورد ماشین تورینگ هم وضع به همین شکل است، اگرچه احتمال وقوع چنین چیزی بسیار کم است.

با پذیرش تز تورینگ می‌توان تعریف دقیقی از الگوریتم ارائه داد:

تعریف

الگوریتم یک الگوریتم برای تابع $f : D \to R$ ماشین تورینگ M است که با هر ورودی $d \in D$ بر روی نوار آن، نهایتاً **متوقف** شده و پاسخ صحیحی به $f(d) \in R$ بر روی نوارش می‌دهد.

به طور مشخص باید به ازای هر $d \in D$ داشته باشیم:

$$q_0\ d \ \vdash^*\ q_f\ f(d) \quad , q_f \in F$$

با ارتباط دادن الگوریتم با ماشین تورینگ، می‌توان ادعاهایی مانند الگوریتمی وجود دارد یا هیچ الگوریتمی وجود ندارد را ثابت نمود.

از آنجا که نوشتن برنامه برای ماشین تورینگ حتی در مسایل ساده بسیار زمان‌بر است، به همین دلیل به تز تورینگ متوسل می‌شویم و ادعا می‌کنیم که هر کاری را بتوان با کامپیوتر انجام داد، با ماشین تورینگ هم قابل انجام است.

پس می‌توان در تعریف الگوریتم، ماشین تورینگ را با برنامه‌های پاسکال یا دیاگرام بلوکی و ... جایگزین کرد.