# نظریه‌ی زبان‌ها و ماشین‌ها

# درس‌نامه‌ی ۱۲

کاظم فولادی
http://kazim.fouladi.ir

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

# فهرست مطالب

# ۱۲ حدود محاسبات الگوریتمی

LIMITS OF ALGORITHMIC COMPUTATION

می‌خواهیم ببینیم که ماشین‌های تورینگ چه کارهایی را نمی‌توانند انجام دهند.
مسائل زیادی وجود دارد که الگوریتمی برای آنها وجود ندارد.
**برای زبان‌های غیر بازگشتی الگوریتم عضویتی وجود ندارد.**
زبان‌های غیر بازگشتی کاربرد عملی بسیارکمی دارند، اما مسئله به اینجا ختم نمی‌شود. برای مثال **برای تعیین اینکه آیا یک گرامر مستقل از متن مبهم است یا خیر الگوریتمی وجود ندارد** که کاربرد آن در زبان‌های برنامه‌سازی است.

## ۱-۱۲ مسائلی که توسط ماشین تورینگ قابل حل نیستند

مسائلی هستند که به روشنی و سادگی بیان می‌شوند و به نظر می‌رسد که ممکن است راه حل الگوریتمی برای آنها موجود باشد، اما غیر قابل حل توسط کامپیوتر هستند.

### ۱-۱-۱۲ تصمیم‌پذیری و محاسبه‌پذیری

**محاسبه‌پذیری (computability).** یک تابع مانند $f$ بر روی یک دامنه‌ی خاص محاسبه‌پذیر است، هرگاه یک ماشین تورینگ برای محاسبه‌ی $f$ برای تمام مقادیر دامنه موجود باشد،
در غیر این صورت $f$ محاسبه‌ناپذیر است.
ممکن است $f$ برای بخشی از دامنه‌ی خود محاسبه‌پذیر باشد، اما تنها در صورتی $f$ را محاسبه‌پذیر می‌گوییم که روی تمام دامنه محاسبه‌پذیر باشد (توجه به دامنه‌ی تابع در بحث محاسبه‌پذیری مهم است).

**تصمیم‌پذیری (decidability).** مسئله‌ی تصمیم‌گیری مسئله‌ای است که پاسخ آن بله / خیر باشد.
مسئله‌ای تصمیم‌پذیر است که یک ماشین تورینگ موجود باشد که به جملات مرتبط با دامنه‌ی مسئله پاسخ صحیح می‌دهد.

### ۲-۱-۱۲ مسئله‌ی توقف در ماشین تورینگ

ماشین تورینگ $M$ و ورودی $w$ داده شده‌اند، پرسش این است که:
اگر پیکربندی اولیه با $w$ توسط $M$ آغاز شود، آیا پس از انجام محاسبات توقف می‌کند؟
به عبارت دیگر **آیا ماشین $M$ با ورودی $w$ توقف می‌کند یا خیر؟**
و یا آیا $(M, w)$ **توقف می‌کند یا خیر؟**

در اینجا دامنه‌ی مسئله تمام ماشین‌های تورینگ و تمامی رشته‌های $w$ است.
الگوریتمی برای این مسئله وجود ندارد. زیرا ماشین تورینگ مکانیزمی برای تمایز محاسبات طولانی از محاسبات نامتناهی ندارد.

**تعریف**

فرض کنید که $w_M$ رشته‌ای باشد که ماشین $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ را توصیف می‌کند و $w$ رشته‌ای از الفبای $M$ باشد.
فرض می‌کنیم که $w_M$ و $w$ رشته‌هایی از ۰ و ۱ باشند.
یک راه حل مسئله‌ی توقف، یک ماشین تورینگ به نام $H$ است که برای **هر** $w_M$ و $w$ محاسبه‌ی زیر را انجام می‌دهد:

$$q_0 w_M w \vdash_H^* x_1 q_y x_2$$

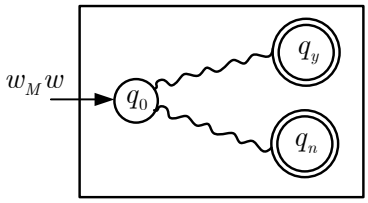
به شرطی که $M$ با عمل روی $w$ متوقف شود و همچنین در حالت دیگر

$$q_0 w_M w \vdash_H^* y_1 q_n y_2$$

به شرطی که $M$ با عمل روی $w$ توقف نکند.
در اینجا $q_y$ و $q_n$ هر دو حالت‌های نهایی ماشین $H$ هستند.

**قضیه** ماشین تورینگ $H$ که مطابق تعریف فوق عمل کند وجود ندارد و بنابراین مسئله‌ی توقف تصمیم‌ناپذیر است.

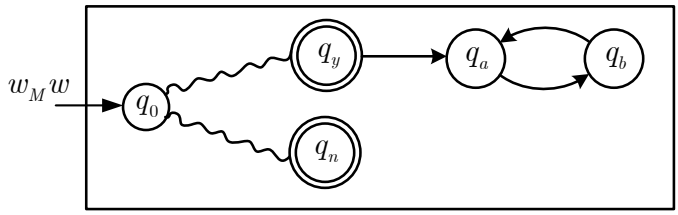
◄ **اثبات.**
برهان خلف: فرض می‌کنیم چنین الگوریتمی موجود باشد و در نتیجه ماشین تورینگ $H$ وجود دارد که مسئله‌ی توقف را حل می‌کند.
ورودی $H$ رشته‌ی $w_M w$ خواهد بود.
لازم است که با هر $w_m w$ داده شده، ماشین تورینگ $H$ با پاسخ yes یا no توقف نماید.
پاسخ این پرسش بر اساس اینکه $H$ درکدام‌یک از حالات نهایی $q_y$ یا $q_n$ توقف می‌کند، به دست می‌آید.




می‌خواهیم $H$ مطابق تعریف زیر عمل کند:

اگر $M$ بر روی $w$ توقف کند: $q_0 w_M w \vdash_H^* x_1 q_y x_2$
اگر $M$ بر روی $w$ توقف نکند: $q_0 w_M w \vdash_H^* y_1 q_n y_2$

حال $H$ را به گونه‌ای تغییر می‌دهیم که ماشین تورینگ $H'$ با ساختار زیر ایجاد شود:




با حالات اضافه شده در شکل می‌خواهیم گذرهایی که باید بین حالت $q_y$ و حالات جدید $q_a$ و $q_b$ انجام شوند، را بدون توجه به نماد روی نوار دنبال کنیم، به گونه‌ای که محتوای نوار بدون تغییر باقی بماند.
روش انجام این کار سرراست است.
با مقایسه‌ی $H$ و $H'$ می‌بینیم که در وضعیت‌هایی که $H$ به $q_y$ می‌رسد و توقف می‌کند، ماشین تغییر یافته‌ی $H'$ وارد حلقه‌ی نامتناهی می‌شود.
به طور رسمی، عمل $H'$ به صورت زیر توصیف می‌شود:

$$\text{اگر } M \text{ بر روی } w \text{ توقف کند:} \quad q_0 w_M w \vdash^*_{H'} \infty$$
$$\text{اگر } M \text{ بر روی } w \text{ توقف نکند:} \quad q_0 w_M w \vdash^*_{H'} y_1 q_n y_2$$

از روی $H'$ ماشین تورینگ دیگری مانند $\hat{H}$ می‌سازیم: $\hat{H}$ ورودی $w_M$ را می‌گیرد، آن را کپی می‌کند . در حالت اولیه‌ی خود $q_0$ کار را به پایان می‌رساند. پس از آن دقیقاً مانند $H'$ رفتار می‌کند.
عمل $\hat{H}$ به صورت زیر خواهد بود:

$$\text{اگر } M \text{ بر روی } w_M \text{ توقف کند:} \quad q_0 w_M \vdash^*_{\hat{H}} q_0 w_M w_M \vdash^*_{\hat{H}} \infty$$
$$\text{اگر } M \text{ بر روی } w_M \text{ توقف نکند:} \quad q_0 w_M \vdash^*_{\hat{H}} q_0 w_M w_M \vdash^*_{\hat{H}} y_1 q_n y_2$$

اکنون $\hat{H}$ یک ماشین تورینگ است، پس دارای توصیفی در $\{0,1\}^*$ می‌باشد: این رشته را با $\hat{w}$ نشان می‌دهیم.
این رشته علاوه بر اینکه توصیف $\hat{H}$ است، می‌تواند به عنوان ورودی هم استفاده شود.
بنابراین می‌توانیم بپرسیم که در صورت اعمال $\hat{w}$ روی $\hat{H}$ چه رخ می‌دهد؟
بر اساس بحث فوق با شناخت $M$ و $\hat{H}$ داریم:

$$\text{اگر } \hat{H} \text{ بر روی } \hat{w} \text{ توقف کند:} \quad q_0 \hat{w} \vdash^*_{\hat{H}} \infty$$
$$\text{اگر } \hat{H} \text{ بر روی } \hat{w} \text{ توقف نکند:} \quad q_0 \hat{w} \vdash^*_{\hat{H}} y_1 q_n y_2$$

واضح است که این بی‌مفهوم است.
این تناقض به ما نشان می‌دهد که فرض وجود $H$ و در نتیجه تصمیم‌پذیری مسئله‌ی توقف باید نادرست باشد.

**قضیه** اگر مسئله‌ی توقف تصمیم‌پذیر بود، آن‌گاه هر زبان شمارش‌پذیر بازگشتی، بازگشتی می‌بود. در نتیجه مسئله‌ی توقف تصمیم‌ناپذیر است.

◄ اثبات.

فرض می‌کنیم $L$ یک زبان شمارش‌پذیر بازگشتی بر روی $\Sigma$ و $M$ ماشین تورینگی باشد که $L$ را می‌پذیرد.
فرض می‌کنیم $H$ ماشین تورینگی باشد که مسئله‌ی توقف را حل می‌کند.
بر اساس این، روال زیر را می‌سازیم:
$H$ را روی $w_M w$ به کار می‌گیریم:
اگر $H$ پاسخ «خیر» را برگرداند، طبق تعریف $w \notin L$
اگر $H$ پاسخ «بله» را برگرداند، $M$ را روی $w$ به کار می‌گیریم،
اما $M$ باید توقف نماید، بنابراین سرانجام می‌فهمیم که $w \in L$ یا $w \notin L$.
این یک الگوریتم عضویت می‌سازد که موجب می‌شود $L$ بازگشتی باشد،
ولی ما می‌دانیم که زبان‌های شمارش‌پذیر بازگشتی‌ای وجود دارند که بازگشتی نیستند.
این تناقض نشان می‌دهد که $H$ نمی‌تواند وجود داشته باشد، یعنی مسئله‌ی توقف تصمیم‌ناپذیر است.

در واقع رابطه‌ی مستقیمی بین مسئله‌ی توقف و الگوریتم عضویت برای زبان‌های شمارش‌پذیر بازگشتی وجود دارد.

### ۳-۱-۱۲ کاهش یک مسئله‌ی تصمیم‌ناپذیر به یک مسئله‌ی دیگر

می‌گوییم مسئله‌ی $A$ به مسئله‌ی $B$ **کاهش** داده می‌شود و می‌نویسیم

$$A \subseteq_R B$$

اگر بتوانیم مسئله‌ی $B$ را حل کنیم، آنگاه بتوانیم مسئله‌ی $A$ را حل نماییم.
در این صورت تصمیم‌پذیری مسئله‌ی $A$ از تصمیم‌پذیری مسئله‌ی $B$ نتیجه شود و به طور معادل، اگر بدانیم که $A$ تصمیم‌ناپذیر است، می‌توانیم نتیجه بگیریم که $B$ نیز تصمیم‌ناپذیر است.

**مثال**

**مسئله‌ی ورود به حالت (state-entry problem)**

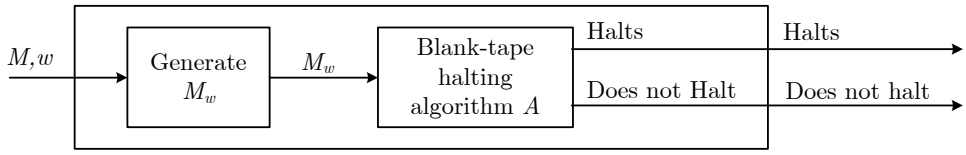
با داشتن هر ماشین تورینگ $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ و هر $q \in Q$ و $w \in \Sigma^+$، تصمیم بگیرید که آیا $M$ با ورودی $w$ وارد حالت $q$ می‌شود یا خیر؟
این مسئله تصمیم‌ناپذیر است.
برای کاهش مسئله‌ی توقف به مسئله‌ی ورود به حالت، فرض کنید که الگوریتم $A$ را داشته باشیم که این مسئله را حل می‌کند. در این صورت می‌توانیم از آن برای حل مسئله‌ی توقف استفاده کنیم.
برای مثال، با داشتن $M$ و $w$ ابتدا $M$ را تغییر می‌دهیم تا $\hat{M}$ را به دست آوریم به گونه‌ای که $\hat{M}$ در $q$ توقف کند اگر و فقط اگر $M$ توقف نماید.
این کار را می‌توانیم به سادگی با نگاه کردن به تابع گذر $\delta$ مربوط به $M$ انجام دهیم. اگر $M$ توقف کند، بدین خاطر است که برخی از $\delta(q_i, a)$ها تعریف نشده است.
برای به دست آوردن $\hat{M}$ هر $\delta$ی تعریف نشده را به صورت $\delta(q_i, a) = (q, a, R)$ تغییر می‌دهیم که در آن $q$ یک حالت نهایی است.

الگوریتم «ورود به حالت» $A$ را بر روی $(\hat{M}, q, w)$ به کار می‌گیریم.
اگر $A$ پاسخ yes بدهد، یعنی وارد حالت $q$ شده‌ایم و آنگاه $(M, w)$ توقف می‌کند.
اگر $A$ پاسخ no بدهد، آنگاه $(M, w)$ توقف نمی‌کند.
بنابراین فرض تصمیم‌پذیر بودن مسئله‌ی «ورود به حالت» الگوریتمی برای مسئله‌ی توقف به دست می‌دهد.
چون مسئله‌ی توقف تصمیم‌ناپذیر است، مسئله‌ی «ورود به حالت» نیز باید تصمیم‌ناپذیر باشد.

**مثال**

**مسئله‌ی توقف با نوار خالی (Blank-tape halting problem)**

یک مسئله‌ی دیگر که مسئله‌ی توقف می‌تواند بدان کاهش یابد:
با داشتن ماشین تورینگ $M$ تعیین کنید که آیا $M$ با شروع از نوار خالی توقف می‌کند یا خیر؟
این مسئله تصمیم‌ناپذیر است.
برای نشان دادن چگونگی کاهش، فرض کنید که یک $M$ و یک $w$ را داشته باشیم.
ابتدا از روی $M$ ماشین جدید $M_w$ را می‌سازیم که با نوار خالی شروع می‌کند و $w$ را بر روی آن می‌نویسد و خود را در وضعیت پیکربندی $q_0 w$ قرار می‌دهد.
پس از آن $M_w$ مطابق $M$ عمل می‌نماید.
واضح است که $M_w$ بر روی نوار خالی توقف می‌کند، اگر و فقط اگر $M$ بر روی $w$ توقف نماید.
اکنون، فرض کنید که مسئله‌ی توقف با نوار خالی تصمیم‌پذیر باشد. با $(M, w)$ داده شده، ابتدا $M_w$ را می‌سازیم و سپس الگوریتم توقف با نوار خالی را بر روی آن اعمال می‌کنیم.
بحث انجام شده به ما می‌گوید که آیا $M$ به کارگرفته شده روی $w$ توقف می‌کند یا خیر.
از آنجا که این کار می‌تواند برای هر $M$ و هر $w$ انجام شود، الگوریتم برای مسئله‌ی توقف با نوار خالی می‌تواند به الگوریتمی برای مسئله‌ی توقف تبدیل شود.
از آنجا که می‌دانیم مسئله‌ی دوم تصمیم‌ناپذیر است، این وضعیت باید برای مسئله‌ی توقف با نوار خالی هم برقرار باشد.




**توابع محاسبه‌ناپذیر** بیشتر مثال‌های توابع محاسبه‌ناپذیر، آنهایی هستند که سعی می‌کنند رفتار ماشین‌های تورینگ را به نوعی پیش‌بینی نمایند.

**مثال**

با فرض $\Gamma = \{0, 1, \square\}$ تابع $f$ را به صورت زیر تعریف می‌کنیم:

$f(n) =$ ماکزیمم تعداد حرکات انجام شده توسط یک ماشین تورینگ $n$ حالته که در صورت شروع با نوار خالی توقف می‌کند.

این تابع به صورتی که تعریف شده است، محاسبه‌پذیر نیست.

ابتدا باید مطمئن شویم که $f(n)$ برای همه‌ی $n$ها تعریف شده است.

نخست، توجه می‌کنیم که تعدادی متناهی ماشین تورینگ با $n$ حالت وجود دارد.

زیرا $Q$ و $\Gamma$ متناهی هستند، بنابراین $\delta$ دارای دامنه و برد متناهی می‌باشد.

این موجب می‌شود که تنها تعدادی متناهی $\delta$ی مختلف و درنتیجه تعدادی متناهی ماشین تورینگ $n$ حالته وجود داشته باشد.

از میان همه‌ی ماشین‌های $n$ حالته، برخی از آنها همیشه توقف می‌کنند (برای مثال، ماشین‌هایی که تنها دارای حالت‌های نهایی هستند و بنابراین هیچ حرکتی انجام نمی‌دهند).

برخی از این ماشین‌های تورینگ $n$ حالته، در صورت شروع با نوار خالی هرگز توقف نمی‌کنند؛ اما این ماشین‌ها در تعریف $f$ وارد نمی‌شوند.

هر ماشینی که توقف می‌کند، تعداد مشخصی حرکت اجرا می‌کند که بزرگترین آنها را به عنوان $f(n)$ در نظر می‌گیریم.

یک ماشین تورینگ دلخواه $M$ و عدد مثبت $m$ را در نظر می‌گیریم.

تغییر $M$ برای تولید $\hat{M}$ به گونه‌ای که همیشه با یکی از دو پاسخ زیر توقف کند، ساده است:

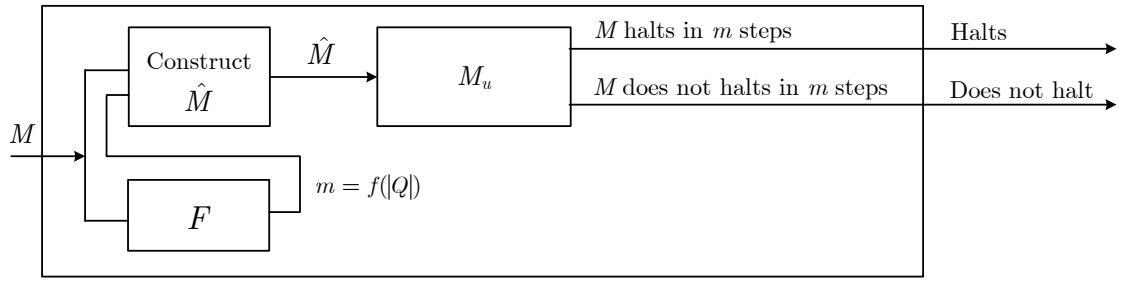
$M$ به کارگرفته شده بر روی نوار خالی حداکثر در $m$ حرکت توقف می‌کند.

$M$ به کارگرفته شده بر روی نوار خالی، بیش از $m$ حرکت انجام می‌دهد.

تمام کاری که برای این باید انجام شود، این است که $M$ را داشته باشیم، تعداد حرکات آن را بشماریم و این کار را در صورتی که شمارش از $m$ تجاوز کرد متوقف نماییم.

فرض می‌کنیم که $f(n)$ توسط یک ماشین تورینگ $F$ محاسبه‌پذیر باشد.

می‌توانیم $F$ و $\hat{M}$ را در کنار هم بگذاریم:




ابتدا $f(|Q|)$ را محاسبه می‌کنیم ($Q$ مجموعه‌ی حالات $M$ است).

این مقدار، حداکثر تعداد حرکاتی که $M$ می‌تواند انجام دهد تا توقف نماید را نشان می‌دهد.

مقداری که به دست می‌آوریم، به عنوان $m$ استفاده می‌شود تا $\hat{M}$ را به صورت طرح شده بسازیم و توصیفی از $\hat{M}$ به ماشین تورینگ جامع داده می‌شود تا آن را اجرا نماید.

این طرح به ما می‌گوید که آیا $M$ به کارگرفته شده روی یک نوار خالی درکمتر از $f(|Q|)$ حالت توقف می‌کند یا خیر؟

اگر دریابیم که $M$ به کارگرفته شده روی نوار خالی بیش از $f(|Q|)$ حرکت انجام می‌دهد، بر اساس تعریف $f$ نتیجه این است که $M$ هرگز توقف نمی‌کند.

بنابراین راه حلی برای مسئله‌ی توقف با نوار خالی داریم.
عدم امکان این نتیجه ما را وادار می‌کند که بپذیریم $f$ محاسبه‌پذیر نیست.

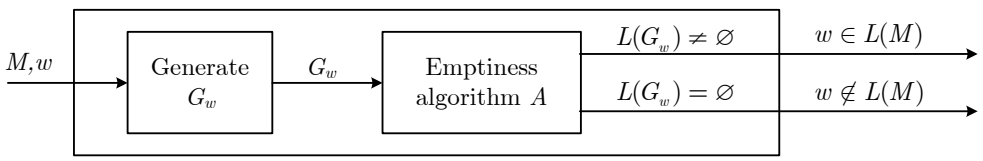
▲

## ۲-۱۲ مسائل تصمیم‌ناپذیر برای زبان‌های شمارش‌پذیر بازگشتی

برای زبان‌های شمارش‌پذیر بازگشتی، الگوریتم عضویت وجود ندارد.
زبان‌های شمارش‌پذیر بازگشتی آن‌قدر کلی هستند که هر پرسشی در مورد آنها تصمیم‌ناپذیر است.
در واقع، وقتی یک پرسش در مورد زبان‌های شمارش‌پذیر بازگشتی می‌پرسیم، متوجه می‌شویم که راهی برای کاهش مسئله‌ی توقف به این پرسش وجود دارد.

**قضیه** فرض کنید که $G$ یک گرامر نامقید باشد. در این صورت مسئله‌ی تعیین اینکه آیا $L(G)$ تهی است یا خیر، تصمیم‌ناپذیر است.

◄ **اثبات.**
مسئله‌ی عضویت برای زبان‌های شمارش‌پذیر بازگشتی را به این مسئله کاهش می‌دهیم.
فرض می‌کنیم که ماشین تورینگ $M$ و رشته‌ی $w$ را داریم. می‌توانیم $M$ را به صورت زیر تغییر دهیم:
ابتدا $M$ ورودی خود را در قسمت خاصی از نوار ذخیره می‌کند،
سپس هرگاه آماده‌ی ورود به یک حالت نهایی شد، آن را بررسی می‌کند و رشته‌ی ورودی را می‌پذیرد اگر و فقط اگر آن رشته $w$ باشد.
این کار را می‌توان به سادگی با تغییر $\delta$ و ایجاد ماشین $M_w$ انجام داد به گونه‌ای که $L(M_w) = L(M) \cap \{w\}$ باشد. حال گرامر متناظر $G_w$ را می‌سازیم.
مشخص است که $L(G_w)$ ناتهی است اگر و فقط اگر $w \in L(M)$ باشد.
حال فرض می‌کنیم که الگوریتمی مانند $A$ برای تصمیم‌گیری در مورد تهی بودن $L(G)$ داریم.
اگر $T$ الگوریتمی باشد که با آن $G_w$ را تولید می‌کنیم، آنگاه می‌توانیم $T$ و $A$ را مطابق شکل روی هم بگذاریم که حاصل مانند ماشین تورینگی است که برای هر $M$ و $w$ به ما می‌گوید $w \in L(M)$ است یا خیر.
اگر چنین ماشین تورینگی داشته باشیم، یک الگوریتم عضویت برای هر زبان شمارش‌پذیر بازگشتی داریم.
این مغایر با نتیجه‌ای است که قبلاً به دست آوردیم. پس نتیجه می‌گیریم که مسئله‌ی $L(G) = \varnothing$? تصمیم‌ناپذیر است.




■

**قضیه** فرض کنید که $M$ یک ماشین تورینگ باشد. در این صورت مسئله‌ی اینکه آیا $L(M)$ متناهی است یا خیر، تصمیم‌ناپذیر است.

◄ **اثبات.**

مسئله‌ی توقف $(M, w)$ را در نظر بگیرید.

از روی ماشین تورینگ $M$ ماشین دیگری مانند $\hat{M}$ می‌سازیم که عمل زیر را انجام دهد:

ابتدا همه‌ی حالات توقف $M$ به گونه‌ای تغییر داده می‌شود که اگر به هر یک از آنها برسیم، آنگاه کل ورودی توسط $\hat{M}$ پذیرفته می‌شود.

این کار را به این صورت انجام می‌دهیم که به هر پیکربندی توقف اجازه می‌دهیم وارد یک حالت نهایی شود.

سپس ماشین اولیه به گونه‌ای تغییر داده می‌شود که ماشین جدید $\hat{M}$ ابتدا $w$ را روی نوار خود تولید کند و بعد همان محاسباتی که توسط $M$ انجام می‌شود، با استفاده از $w$ که به تازگی ایجاد شده است، انجام دهد.

به عبارت دیگر، حرکت‌های انجام شده توسط $\hat{M}$ پس از اینکه $w$ را روی نوار خود نوشت، همان حرکاتی است که $M$ انجام می‌دهد اگر در پیکربندی $q_\circ w$ توقف نماید.

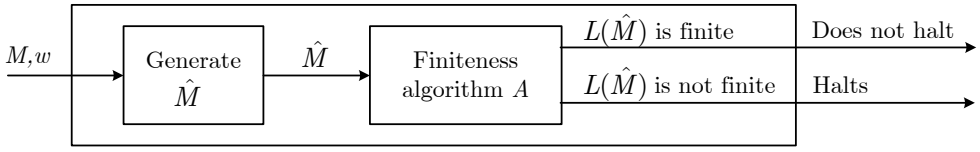
اگر $M$ توقف نماید، آنگاه $\hat{M}$ در یک حالت نهایی توقف می‌نماید.

بنابراین اگر $(M, w)$ توقف کند، آنگاه $\hat{M}$ برای تمامی ورودیها به یک حالت نهایی می‌رسد.

و اگر $(M, w)$ توقف ننماید، $\hat{M}$ هم توقف نمی‌کند و بنابراین چیزی را نمی‌پذیرد.

به عبارت دیگر، $\hat{M}$ یا زبان نامتناهی $\Sigma^+$ و یا زبان متناهی $\varnothing$ را می‌پذیرد.

حال اگر فرض کنیم که الگوریتم $A$ وجود دارد که به ما می‌گوید $L(\hat{M})$ متناهی است یا خیر، می‌توانیم راهی برای مسئله‌ی توقف مطابق شکل زیر بیابیم و بنابراین الگوریتمی برای تصمیم‌گیری در مورد اینکه آیا $L(M)$ متناهی است یا خیر نمی‌تواند وجود داشته باشد.




در اثبات این قضیه، طبیعت مسئله، یعنی اینکه «آیا $L(M)$ متناهی است؟»، مهم نیست.

پس می‌توان طبیعت مسئله را بدون تاثیر عمده بر روی بحث، تغییر داد.

**مثال**

نشان دهید که برای هر ماشین تورینگ دلخواه $M$ با $\Sigma = \{a, b\}$، مسئله‌ی «$L(M)$ شامل دو رشته‌ی متمایز با طول یکسان است یا خیر؟» تصمیم‌ناپذیر است.

از همان روش اثبات قضیه‌ی قبل استفاده می‌کنیم،

با این تفاوت که وقتی که $\hat{M}$ به پیکربندی توقف می‌رسد، به گونه‌ای تغییر می‌یابد که دو رشته‌ی $a$ و $b$ را

بپذیرد.
برای این کار، ورودی اولیه حفظ می‌شود و در انتهای محاسبه با $a$ و $b$ مقایسه می‌شود و تنها این دو رشته پذیرفته می‌شوند.
بنابراین اگر $(M, w)$ توقف کند، $\hat{M}$ دو رشته به طول یکسان را می‌پذیرد.
در غیر این صورت $\hat{M}$ چیزی را نخواهد پذیرفت،
مابقی بحث مشابه اثبات قضیه‌ی قبل است.

▲

◄ **نکته** می‌توان در مثال فوق پرسش‌هایی مانند «آیا $L(M)$ حاوی رشته‌ای به طول ۵ است» یا «آیا $L(M)$ منظم است» را جایگزین کرد، بدون اینکه بر اساس بحث خللی وارد شود.
همه‌ی این پرسش‌ها و پرسش‌های مشابه تصمیم‌ناپذیر هستند.

**قضیه** **قضیه‌ی رایس (Rice theorem).** هر خصوصیت غیربدیهی از یک زبان شمارش‌پذیر بازگشتی تصمیم‌ناپذیر است.

منظور از خصوصیت بدیهی (nontrivial) خصوصیتی است که در برخی و نه همه‌ی زبان‌های شمارش‌پذیر بازگشتی وجود دارد.

## ۳-۱۲ مسئله‌ی تناظر پست

کارکردن با مسئله‌ی توقف به طور مستقیم در بسیاری از موارد مشکل است.
مناسب است که نتایجی میانی ساخته شوند که از تصمیم‌ناپذیری مسئله‌ی توقف نتیجه شوند اما ارتباط نزدیک‌تری با مسئله‌ی مورد نظر ما داشته باشند و از طرفی بحث را ساده‌تر نمایند.
یکی از این نتایج میانی **مسئله‌ی تناظر پست (Post Correspondence Problem)** است.
که به صورت زیر بیان می‌شود:
دو دنباله از $n$ رشته بر روی الفبای $\Sigma$ داده شده است:

$$\begin{aligned} A &= w_1, w_2, \ldots, w_n \\ B &= v_1, v_2, \ldots, v_n \end{aligned}$$

گفته می‌شود که یک راه‌حل تناظر پست (PC-solution) برای زوج $(A, B)$ وجود دارد،
اگر دنباله‌ی ناتهی $i_1, i_2, \ldots, i_k$ از اعداد صحیح موجود باشد که

$$w_{i_1} w_{i_2} \ldots w_{i_k} = v_{i_1} v_{i_2} \ldots v_{i_k}$$

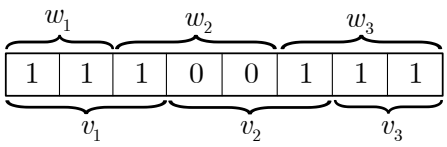
مسئله‌ی تناظر پست، طراحی الگوریتمی است که به ما می‌گوید برای هر زوج $(A, B)$ آیا راه حل PC وجود دارد یا خیر؟

**مثال**

فرض می‌کنیم $\Sigma = \{۰, ۱\}$ و

$$A = w_۱, w_۲, w_۳ \quad w_۱ = ۱۱, w_۲ = ۱۰۰, w_۳ = ۱۱۱$$
$$B = v_۱, v_۲, v_۳ \quad w_۱ = ۱۱۱, w_۲ = ۰۰۱, w_۳ = ۱۱$$

در این حالت مطابق شکل راه حل PC وجود دارد:




اما اگر

$$A = w_۱, w_۲, w_۳ \quad w_۱ = ۰۰, w_۲ = ۰۰۱, w_۳ = ۱۰۰۰$$
$$B = v_۱, v_۲, v_۳ \quad w_۱ = ۰, w_۲ = ۱۱, w_۳ = ۰۱۱$$

راه حل PC نمی‌تواند وجود داشته باشد، زیرا هر رشته‌ی متشکل از عناصر $A$ طولانی‌تر از رشته‌ی متشکل از عناصر $B$ است.

برای نمونه‌های خاص ممکن است بتوانیم راه حلی برای مسئله‌ی PC ارائه کنیم و یا اینکه نشان دهیم چنین راه حلی وجود ندارد، اما در حالت کلی الگوریتمی برای تصمیم‌گیری در مورد این مسئله تحت همه‌ی شرایط وجود ندارد. بنابراین مسئله‌ی تناظر پست تصمیم‌ناپذیر است.

**قضیه** مسئله‌ی تناظر پست تصمیم‌ناپذیر است.

## ۴-۱۲ مسائل تصمیم‌ناپذیر برای زبان‌های مستقل از متن

مسئله‌ی تناظر پست یک ابزار متداول برای مطالعه‌ی پرسش‌های تصمیم‌ناپذیر برای زبان‌های مستقل از متن است.

**قضیه** الگوریتمی وجود ندارد که تصمیم بگیرد آیا یک گرامر مستقل از متن داده شده مبهم است یا خیر.

◄ اثبات.

دو دنباله از رشته‌ها را به صورت

$$A = (w_۱, w_۲, \ldots, w_n), \qquad B = (v_۱, v_۲, \ldots, v_n)$$

روی الفبای $\Sigma$ در نظر بگیرید. یک مجموعه‌ی جدید از نمادهای متمایز $a_۱$، $a_۲$، ...، $a_n$ را انتخاب کنید به گونه‌ای که

$$\{a_۱, a_۲, \ldots, a_n\} \cap \Sigma = \varnothing$$

باشد و دو زبان

$$\begin{aligned} L_A &= \{w_i w_j \ldots w_l w_k a_k a_l \ldots a_j a_i\} \\ L_B &= \{v_i v_j \ldots v_l v_k a_k a_l \ldots a_j a_i\} \end{aligned}$$

را در نظر بگیرید. اکنون به گرامر مستقل از متن زیر توجه کنید:

$$G = (\{S, S_A, S_B\}, \Sigma \cup \{a_۱, a_۲, \ldots, a_n\}, S, P)$$

که در آن مجموعه‌ی قواعد $P$ اجتماع دو مجموعه‌ی زیر است:
اولین مجموعه $P_A$ متشکل است از:

$$\begin{aligned} S &\rightarrow S_A \\ S_A &\rightarrow w_i S_A a_i \mid w_i a_i \qquad i = ۱, ۲, \ldots, n \end{aligned}$$

و دومین مجموعه $P_B$ متشکل است از:

$$\begin{aligned} S &\rightarrow S_B \\ S_B &\rightarrow v_i S_B a_i \mid v_i a_i \qquad i = ۱, ۲, \ldots, n \end{aligned}$$

حال،

$$G_A = (\{S, S_A\}, \Sigma \cup \{a_۱, a_۲, \ldots, a_n\}, S, P_A)$$

و

$$G_B = (\{S, S_B\}, \Sigma \cup \{a_۱, a_۲, \ldots, a_n\}, S, P_B)$$

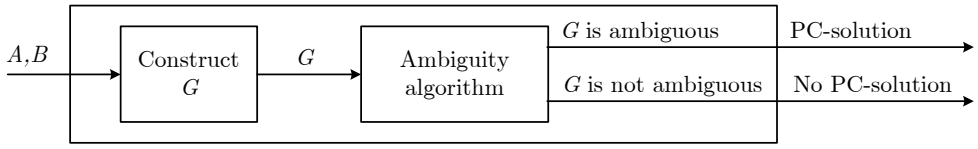
واضح است که در این صورت

$$\begin{aligned} L_A &= L(G_A) \\ L_B &= L(G_B) \\ L(G) &= L_A \cup L_B \end{aligned}$$

به سادگی می‌توان دید که $G_A$ و $G_B$ به خودی خود غیر مبهم هستند.
اگر یک رشته در $L(G)$ با $a_i$ پایان یابد، در این صورت اشتقاق آن با گرامر $G_A$ باید با $S \Rightarrow w_i S a_i$ شروع شود. به طور مشابه می‌توان در هر مرحله گفت که بایستی از چه قاعده‌ای استفاده گردد.
بنابراین اگر $G$ مبهم باشد، باید بدین دلیل باشد که یک $w$ وجود دارد که برای آن دو اشتقاق موجود است:

$$\begin{aligned} S &\Rightarrow S_A \Rightarrow w_i S_A a_i \Rightarrow^* w_i w_j \ldots w_k a_k \ldots a_j a_i = w, \\ S &\Rightarrow S_B \Rightarrow v_i S_B a_i \Rightarrow^* v_i v_j \ldots v_k a_k \ldots a_j a_i = w \end{aligned}$$

در نتیجه اگر $G$ مبهم باشد، آنگاه مسئله‌ی تناظر پست با زوج $(A, B)$ دارای یک راه حل است. بر عکس اگر $G$ نامبهم باشد، آنگاه مسئله‌ی تناظر پست نمی‌تواند راه حلی داشته باشد.
اگر الگوریتمی برای حل مسئله‌ی ابهام وجود داشته باشد، می‌توانیم آن را تغییر دهیم تا مسئله‌ی تناظر پست را مطابق شکل زیر حل کند. اما از آنجا که الگوریتمی برای مسئله‌ی تناظر پست وجود ندارد، نتیجه می‌گیریم که مسئله‌ی ابهام تصمیم‌ناپذیر است.




**قضیه** الگوریتمی برای تصمیم‌گیری در مورد اینکه آیا $L(G_1) \cap L(G_2) = \varnothing$ است یا خیر، برای گرامرهای مستقل از متن دلخواه $G_1$ و $G_2$، وجود ندارد.

◄ اثبات.
گرامر $G_A$ را به عنوان $G_1$ و $G_B$ را به عنوان $G_2$ در نظر می‌گیریم که در اثبات قضیه‌ی قبلی استفاده شده‌اند.
فرض می‌کنیم که $L(G_A)$ و $L(G_B)$ یک عضو مشترک داشته باشند، یعنی:

$$\begin{aligned} S_A &\Rightarrow^* w_i w_j \dots w_k a_k \dots a_j a_i, \\ S_B &\Rightarrow^* v_i v_j \dots v_k a_k \dots a_j a_i \end{aligned}$$

در این صورت زوج $(A, B)$ دارای یک راه حل PC است. برعکس اگر این زوج راه حل PC نداشته باشد، در این صورت $L(G_A)$ و $L(G_B)$ نمی‌توانند دارای عضو مشترک باشند.
نتیجه می‌گیریم که $L(G_A) \cap L(G_B)$ ناتهی است اگر و فقط اگر $(A, B)$ دارای یک راه حل PC باشد.
این کاهش قضیه را اثبات می‌کند.

# 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.