نظریه‌ی زبان‌ها و ماشین‌ها

درس‌نامه‌ی ۲

کاظم فولادی
http://kazim.fouladi.ir

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

فهرست مطالب

۲ آتوماتای متناهی

FINITE AUTOMATA

۲-۱ پذیرنده‌ی متناهی‌حالت

۲-۱-۱ تعریف

آتوماتون متناهی‌حالت، ساده‌ترین نوع آتوماتاست که حافظه‌ی موقت ندارد.
پذیرنده‌ی متناهی‌حالت به دو شکل موجود است:

- پذیرنده‌ی متناهی‌حالت قطعی (deterministic finite-state accepter: DFA)
- پذیرنده‌ی متناهی‌حالت غیرقطعی (non-deterministic finite-state accepter: NFA)

تعریف **آتوماتون متناهی‌حالت قطعی (DFA).** آتوماتون متناهی‌حالت قطعی، یک پنج‌تایی مرتب به صورت

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

است که در آن

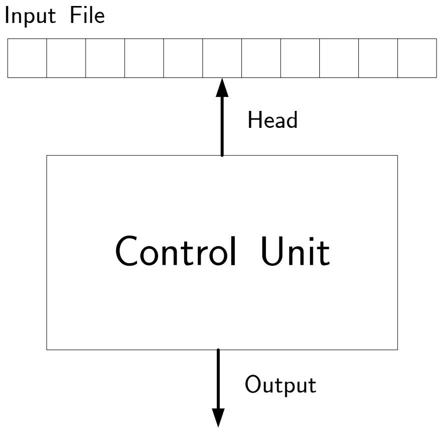
Q **مجموعه‌ی حالات**: مجموعه‌ی متناهی و ناتهی از حالت‌های داخلی *(internal states)*

Σ **مجموعه‌ی الفبای ورودی** *(alphabet)*

δ **تابع گذر حالت** به صورت $\delta : Q \times \Sigma \rightarrow Q$ که حالت بعدی آتوماتون را بر اساس حالت فعلی و نماد ورودی جاری تعیین می‌کند *(state transition function)*

q_0 **حالت اولیه** $q_0 \in Q$ *(initial state)*

F **مجموعه‌ی حالات نهایی** $F \subseteq Q$ *(final states)*



- یک DFA ابتدا در حالت q_0 است و نشانگر ورودی آن بر روی اولین نماد ورودی (سمت چپ‌ترین نماد) قرار دارد.
- با هر حرکت آتوماتون، نشانگر ورودی یک نماد به سمت راست می‌رود.
- یک رشته توسط آتوماتون وقتی پذیرفته می‌شود که در پایان رشته، آتوماتون در یکی از حالت‌های نهایی خود قرار گرفته باشد.
- برای مشخص کردن آتوماتون متناهی باید تابع گذر حالت δ را مشخص کرد.

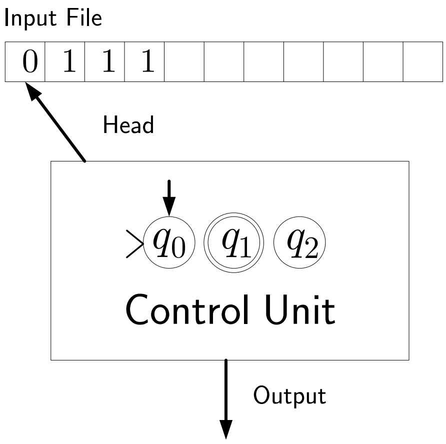
نمایش DFA برای نمایش DFA دو روش کلی وجود دارد:

- **گراف گذر حالت (state-transition graph)**
  برچسب‌گذاری رأس‌ها: با نام حالت‌ها؛
  برچسب‌گذاری یال‌ها: به ازای هر نماد الفبا یک یال از هر حالت خارج می‌شود.
- **جدول گذر حالت (state-transition table)**
  برای هر حالت، یک سطر و برای هر حرف الفبا یک ستون داریم.

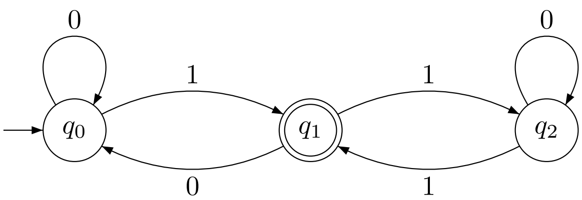
	a_1	a_2	$\dots$	a_k
q_0	q_1	q_2	$\dots$	q_k
q_1			$\dots$	
$\vdots$				
q_m			$\dots$	

مثال

شکل زیر، پیکربندی اولیه‌ی یک آتوماتون متناهی با سه حالت را نشان می‌دهد:



که گراف گذر حالت آن به صورت زیر است:



تعریف ریاضی این آتوماتون به صورت $M = (\{q_{۰}, q_{۱}, q_{۲}\}, \{۰, ۱\}, \delta, q_{۰}, \{q_{۱}\})$ می‌باشد که در آن تابع گذر حالت به صورت زیر تعریف شده است:

$$\delta(q_{۰}, ۰) = q_{۰}$$
$$\delta(q_{۰}, ۱) = q_{۱}$$
$$\delta(q_{۱}, ۰) = q_{۰}$$
$$\delta(q_{۱}, ۱) = q_{۲}$$
$$\delta(q_{۲}, ۰) = q_{۲}$$
$$\delta(q_{۲}, ۱) = q_{۱}$$

که نمایش آن با جدول گذر حالت به صورت زیر است:

	۰	۱
$q_{۰}$	$q_{۰}$	$q_{۱}$
$q_{۱}$	$q_{۰}$	$q_{۲}$
$q_{۲}$	$q_{۲}$	$q_{۱}$

با نمایش پیکربندی هرگام به صورت **(ادامه‌ی ورودی, حالت فعلی)**، آتوماتون فوق ورودی خود را که رشته‌ی $w = ۰۱۱۱$ است، به صورت زیر مصرف می‌کند:

$$(q_{۰}, ۰۱۱۱) \vdash (q_{۰}, ۱۱۱) \vdash (q_{۱}, ۱۱) \vdash (q_{۲}, ۱) \vdash (q_{۱}, \lambda)$$

و چون رشته‌ی ورودی تمام شد و حالت $q_{۱}$ یک حالت نهایی برای این آتوماتون است، رشته‌ی w توسط این آتوماتون پذیرفته می‌شود.

اما اگر رشته‌ی $w = ۱۰۰$ را به عنوان ورودی به آتوماتون می‌دادیم:

$$(q_۰, ۱۰۰) \vdash (q_۱, ۰۰) \vdash (q_۰, ۰) \vdash (q_۰, \lambda)$$

رشته‌ی ورودی تمام می‌شد اما ماشین در یک حالت غیرنهایی توقف می‌کرد؛ پس w پذیرفته نمی‌شد.

◄ **تذکر** به نمایش وضعیت آتوماتون در هرگام به صورت **(ادامه‌ی ورودی, حالت فعلی)**، یک پیکربندی گفته می‌شود. حرکت بین پیکربندیها با نماد $\vdash$ توصیف می‌شود.

تعریف **تابع گذر حالت گسترش‌یافته.** تابع گذر حالت گسترش‌یافته (δ^*) مشابه δ تعریف می‌شود، با این تفاوت که آرگومان دوم آن به جای یک نماد الفبا، یک رشته است:

$$\delta^* : Q \times \Sigma^* \rightarrow Q$$

و به صورت بازگشتی تعریف می‌گردد:

$$\begin{aligned} \delta^*(q, \lambda) &= q && , q \in Q,\ w \in \Sigma^* \\ \delta^*(q, wa) &= \delta(\delta^*(q, w), a)) && , q \in Q,\ w \in \Sigma^*,\ a \in \Sigma \end{aligned}$$

مثال

برای محاسبه‌ی $\delta^*(q_۱, ۰۱۱)$ در مثال قبلی به صورت زیر عمل می‌کنیم:

$$\begin{aligned} \delta^*(q_۱, ۰۱۱) &= \delta(\delta^*(q_۱, ۰۱), ۱) \\ \delta^*(q_۱, ۰۱) &= \delta(\delta^*(q_۱, ۰), ۱) \\ \delta^*(q_۱, ۰) &= \delta(\delta^*(q_۱, \lambda), ۰) \\ \delta^*(q_۱, \lambda) &= q_۱ \end{aligned}$$

پس

$$\begin{aligned} \delta^*(q_۱, ۰) &= \delta(q_۱, ۰) = q_۰ \\ \delta^*(q_۱, ۰۱) &= \delta(q_۰, ۱) = q_۱ \\ \delta^*(q_۱, ۰۱۱) &= \delta(q_۱, ۱) = q_۲ \end{aligned}$$

◄ آرگومان اول δ می‌تواند یک مجموعه از حالات $P \subseteq Q$ باشد که در این صورت به شکل زیر تعریف می‌شود:

$$\delta(P, a) = \bigcup_{p \in P} \delta(p, a) \qquad , a \in \Sigma$$

۲-۱-۲ زبان یک DFA

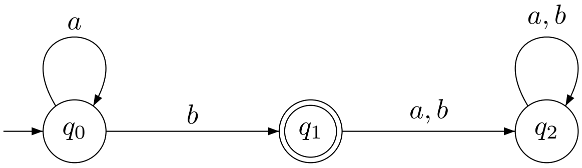
زبان یک DFA، مجموعه‌ی تمام رشته‌هایی است که توسط آتوماتون پذیرفته می‌شود.

تعریف زبان پذیرفته شده توسط آتوماتون متناهی $M = (Q, \Sigma, \delta, q, F)$ عبارت است از مجموعه‌ی

$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}$$

مثال

زبان پذیرفته شده توسط آتوماتون متناهی زیر، عبارت است از $L = \{a^n b : n \geq 0\}$.



◄ **تذکر** δ و δ^* توابع تام هستند (یعنی روی کل دامنه عمل می‌کنند). به عبارت دیگر، DFA تمام رشته‌های Σ^* را پردازش کرده، آن‌ها را قبول یا رد می‌کند.

تعریف **پذیرش یک رشته.** اگر پس از اتمام رشته، آتوماتون در یکی از حالات نهایی خود باشد، آن رشته پذیرفته می‌شود.

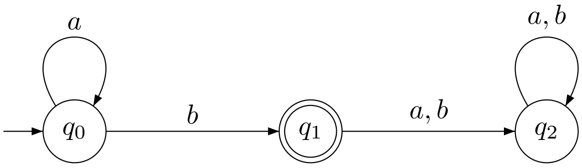
تعریف **عدم پذیرش یک رشته.** اگر پس از اتمام رشته، آتوماتون در یکی از حالات غیر نهایی خود باشد، آن رشته رد می‌شود.

تعریف **حالت تله (trap).** حالت تله یک حالت غیرنهایی است که آتوماتون با ورود به آن دیگر از آن خارج نمی‌شود. حالت تله دنباله‌ی رشته را مصرف می‌کند تا تمام شود.

◄ **تذکر** هرگاه در تعریف تابع گذر، گذر حالت با یکی از عناصر الفبا مشخص نباشد، با آن عنصر وارد حالت تله می‌شویم.

مثال

در آتوماتون متناهی زیر، حالت $q_۲$ یک حالت تله است:



عملگر $\vdash_M$ عملگر $\vdash_M$ برای توصیف دنباله‌ای از گذرهای یک آتوماتون M استفاده می‌شود:

$$\vdash_M : (Q \times \Sigma^*) \to (Q \times \Sigma^*)$$

قضیه اگر $M = (Q, \Sigma, \delta, q_\circ, F)$ یک DFA باشد G_M گراف گذر حالت مربوط به آن باشد، در این صورت

$$\forall q_i, q_j \in Q,\ \ w \in \Sigma^* \to \delta^*(q_i, w) = q_j$$

اگر و فقط اگر یک گشت $(walk)$ با برچسب w از q_i به q_j موجود باشد.

$$\boxed{(q_i, w) \vdash_M (q_j, w') \quad \textbf{iff} \quad w = \sigma w',\ \sigma \in \Sigma,\quad \delta(q_i, \sigma) = q_j}$$

۲-۱-۳ مکمل یک DFA

تعریف **مکمل یک DFA.** مکمل یک DFAی M، آتوماتونی است که زبان $\overline{L(M)}$ را می‌پذیرد.

$$\overline{L(M)} = \{w \in \Sigma^* : \delta^*(q_\circ, w) \notin F\}$$

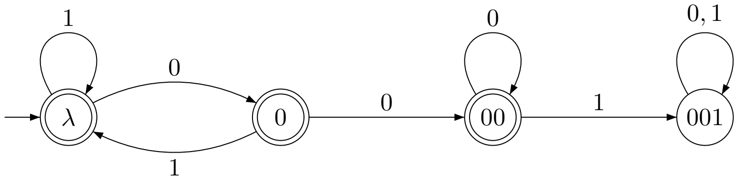
الگوریتم مکمل کردن یک DFA

۱) حالت تله را در صورت عدم وجود اضافه می‌کنیم.

۲) حالات نهایی را به حالات غیرنهایی و حالات غیرنهایی را به نهایی تبدیل می‌کنیم.

مثال

یک DFA برای پذیرش همه‌ی رشته‌های روی $\{0,1\}^*$ بجز آنهایی که شامل زیررشته‌ی ۰۰۱ هستند، به صورت زیر است:



◄ **نکته** DFA می‌تواند ورودی قبلی را به یاد آورد، اما نه از طریق حافظه، بلکه از طریق حالت فعلی آن.

۲-۱-۴ ترکیب DFAها

اگر $M_1 = (Q_1, \Sigma, \delta_1, q_{0_1}, F_1)$ یک DFA برای پذیرش زبان $L_1 = L(M_1)$ باشد و نیز $M_2 = (Q_2, \Sigma, \delta_2, q_{0_2}, F_2)$ یک DFA برای پذیرش زبان $L_2 = L(M_2)$ باشد، آنگاه برای اجتماع، اشتراک و تفاضل L_1 و L_2، DFAی $M = (Q, \Sigma, \delta, q_0, F)$ را می‌توان به صورت زیر به دست آورد:

$$Q = Q_1 \times Q_2$$

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)) \qquad , q_1 \in Q_1,\ q_2 \in Q_2,\ a \in \Sigma$$

$$q_0 = (q_{0_1}, q_{0_2})$$

$L(M)$	$(q_1, q_2) \in F$
$L(M_1) \cup L(M_2)$	$q_1 \in F_1 \vee q_2 \in F_2$
$L(M_1) \cap L(M_2)$	$q_1 \in F_1 \wedge q_2 \in F_2$
$L(M_1) - L(M_2)$	$q_1 \in F_1 \wedge q_2 \notin F_2$

۲-۱-۵ قضایایی در مورد DFA

- اگر یک DFA با k حالت رشته‌ی w با طول بیشتر از $k-1$ را بپذیرد، آنگاه زبان پذیرفته شده توسط آن نامتناهی است.
- اگر یک DFA با k حالت رشته‌ی w را بپذیرد، حتماً رشته‌ای با طول کمتر از k را خواهد پذیرفت.
- اگر یک DFA با k حالت، یک زبان نامتناهی را بپذیرد، رشته‌ای مانند w با طول $k \le |w| \le 2k$ را خواهد پذیرفت.
- شرط لازم و کافی برای آنکه یک DFA زبان Σ^* را بپذیرد، آن است که تمامی حالات دسترس‌پذیر آن، حالت نهایی باشد.

- شرط لازم و کافی برای آن‌که یک DFA زبان $\varnothing$ را بپذیرد، آن است که تمامی حالات دسترس‌پذیر آن، حالت غیرنهایی باشد.
- شرط لازم و کافی برای آن‌که یک DFA رشته‌ی λ را بپذیرد، آن است که حالت شروع آن حالت نهایی هم باشد.

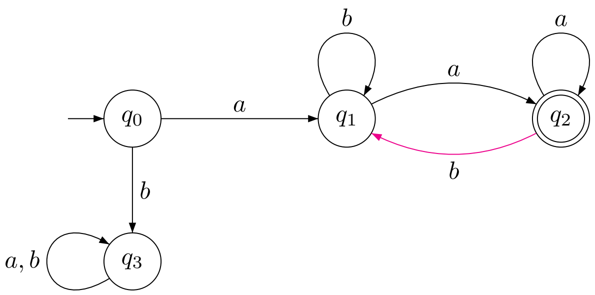
۲-۲ زبان منظم

تعریف **زبان منظم.** زبان L منظم نام دارد، اگر و فقط اگر یک *DFA* مانند M موجود باشد که $L = L(M)$

نتیجه هر زبان منظم توسط یک DFA پذیرفته می‌شود.

مثال

زبان $L = \{awa : w \in \{a,b\}^*\}$ یک زبان منظم است، زیرا برای آن DFAی زیر وجود دارد:



۳-۲ آتوماتون متناهی غیرقطعی

تعریف **آتوماتون متناهی حالت غیرقطعی (NFA).** یک *NFA* همانند *DFA* است، با این تفاوت که تابع گذر حالت آن به صورت زیر تعریف می‌شود:

$$\delta : Q \times \Sigma \to ۲^Q$$

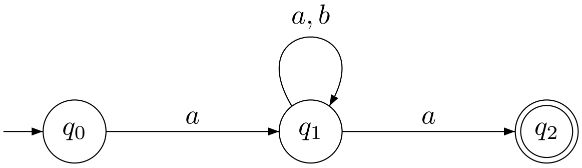
◄ **نکته** DFA خواناتر است، اما NFA معمولاً جمع و جورتر است.

◄ برد تابع δ مجموعه‌ی توانی ۲^Q است، بنابراین مقدار آن زیرمجموعه‌ای از Q است.

◄ ممکن است داشته باشیم $\delta(q_۱, a) = \varnothing$، یعنی با ورودی a هیچ تغییر حالتی نداریم ($\varnothing$ معادل حالت تله است).

مثال

شکل زیر یک نمونه آتوماتون متناهی حالت غیرقطعی را نشان می‌دهد:



تعریف **آتوماتون متناهی حالت غیرقطعی با حرکات λ (λ-NFA).** یک λ-NFA همانند NFA است، با این تفاوت که تابع گذر حالت آن به صورت زیر تعریف می‌شود:

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow ۲^Q$$

یعنی λ به عنوان ورودی قابل قبول است.

مفهوم حرکت با ورودی λ حرکت λ به معنی گذر حالت بدون مصرف ورودی (مصرف انرژی) است. به عبارت دیگر تغییر حالت انجام می‌شود ولی مکانیزم خواندن ورودی متوقف می‌ماند.

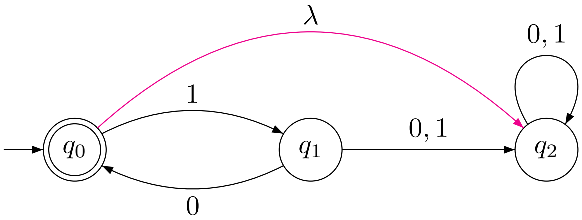
تابع گذر گسترش‌یافته برای NFA

$$\delta^* : Q \times \Sigma^* \rightarrow ۲^Q$$

$$\begin{aligned} \delta^*(q, \lambda) &= \{q\} \\ \delta^*(q, wa) &= \{p : \exists r \in \delta^*(q, w), p \in \delta(r, a)\} \end{aligned}$$

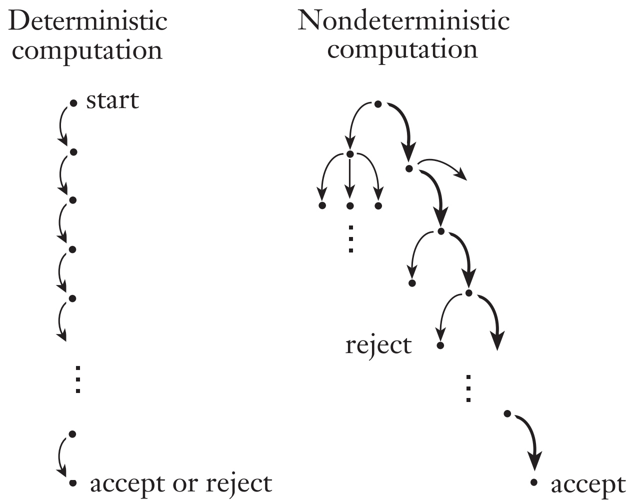
مثال

شکل زیر یک نمونه آتوماتون متناهی حالت غیرقطعی با حرکات λ را نشان می‌دهد:



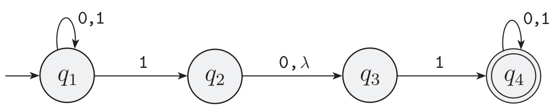
۲-۳-۱ معنای عدم قطعیت

ماشین غیرقطعی می‌تواند به صورت غیرقطعی تصمیم‌گیری صحیح را انجام دهد (بدون عقبگرد): می‌توان تصور کرد که تعداد نامتناهی پردازنده وجود دارد. هرگاه با یک حرف ورودی چند مسیر وجود داشت، یک پردازنده‌ی جدید فعال می‌شود و هر مسیر با ادامه‌ی ورودی توسط یک پردازنده دنبال می‌شود. هرگاه یکی از پردازنده‌ها به حالت نهایی رسید، به بقیه‌ی پردازنده‌ها سیگنال توقف می‌دهد. (به جای بی‌نهایت پردازنده، می‌توان فرض کرد که آتوماتون قابلیت تولید مثل دارد و هرگاه به چند گزینه می‌رسد، خودش را تکثیر می‌کند.)

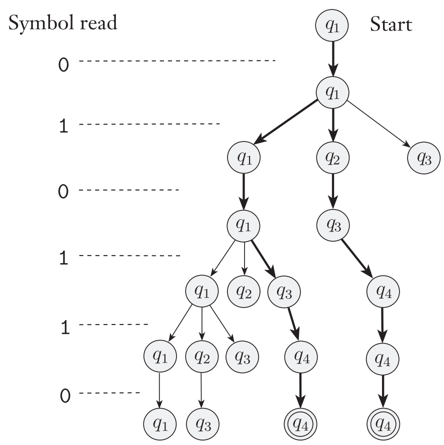


مثال

NFAی زیر را در نظر بگیرید:



حرکت‌های این آتوماتون غیرقطعی بر روی رشته‌ی ۰۱۰۱۱۰ با درخت زیر قابل نمایش است:



- عدم قطعیت مکانیزم مؤثری برای توصیف زبان‌های پیچیده است (مثلاً $\{a^{۳}\}\cup\{a^{۲n} : n \geq ۱\}$).
- از آتوماتای غیرقطعی می‌توان برای توصیف الگوریتم‌های جستجو با عقبگرد استفاده کرد.

۲-۳-۲ زبان یک NFA

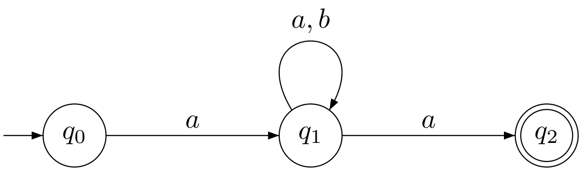
تعریف زبان یک NFA مانند M عبارت است از مجموعه‌ی:

$$L(M) = \{w \in \Sigma^* : \delta(q_۰, w) \cap F \neq \varnothing\}$$

یعنی ممکن است چند راه برای پذیرش یک رشته موجود باشد.

مثال

برای زبان $L = \{awa : w \in \{a, b\}^*\}$ می‌توان NFA زیر را ارائه کرد:



۴-۲ هم‌ارزی DFA و NFA

تعریف **دو آتوماتون هم‌ارز.** آتوماتون‌های $M_۱$ و $M_۲$ هم‌ارز هستند اگر و فقط اگر هر دو یک زبان را بپذیرند.

$$\boxed{M_۱ \equiv M_۲ \quad \textbf{iff} \quad L(M_۱) = L(M_۲)}$$

قضیه اگر L زبان پذیرفته شده توسط NFAی $M_N = (Q_N, \Sigma, \delta_N, q_۰, F_N)$ باشد، آن‌گاه یک DFA به صورت $M_D = (Q_D, \Sigma, \delta_D, \{q_۰\}, F_D)$ وجود دارد به طوری که $L = L(M_D)$.

- برای هر NFA، یک DFA وجود دارد که همان زبان را می‌پذیرد. به عبارت دیگر، قدرت NFA و DFA برابر است.

الگوریتم تبدیل NFA به DFA

- هر حالت DFA به صورت زیرمجموعه‌ای از حالات NFA است:

$$Q_D = ۲^{Q_N}$$

- تابع گذر حالت برای هر حالت $\{q_۰, q_۱, \ldots, q_n\}$ به صورت زیر تعیین می‌شود:

$$\delta_D(\{q_۰, q_۱, \ldots, q_n\}, a) = \bigcup_{i=۰}^{n} \delta_N(q_i, a) = \{p_۰, p_۱, \ldots, p_m\}$$

- مجموعه‌ی حالات نهایی F_D شامل حالت‌هایی از Q_D خواهد بود که حداقل یکی از عناصر F_N در آنها باشد.
- حالت اولیه‌ی DFA به صورت $q_{۰_D} = \{q_{۰_N}\}$ می‌باشد.

نتیجه زبان پذیرفته شده توسط NFA یک زبان منظم است.

قضیه اگر L زبان پذیرفته شده توسط λ-*NFA*ی $M_\lambda = (Q_\lambda, \Sigma, \delta_\lambda, q_{۰_\lambda}, F_\lambda)$ باشد، آن‌گاه یک *NFA* به صورت $M_D = (Q_N, \Sigma, \delta_N, q_{۰_N}, F_N)$ وجود دارد به طوری که $L = L(M_N)$.

◄ برای هر λ-NFA، یک NFA وجود دارد که همان زبان را می‌پذیرد. به عبارت دیگر، قدرت λ-NFA و NFA برابر است.

الگوریتم تبدیل λ-NFA به NFA

- مجموعه‌ی حالات λ-NFA و NFA با هم برابر است.

$$Q_N = Q_\lambda$$

- تابع گذر حالت برای NFA به صورت زیر تعریف می‌شود:

$$\delta_N(q, a) = \delta_\lambda(\delta^*_\lambda(q, \lambda), a)$$

که در آن $\delta^*_\lambda(q, \lambda)$ مجموعه‌ی تمام حالاتی است که از حالت q تنها با حرکات λ (بدون مصرف ورودی) می‌توان به آنها رسید:

$$\delta^*_\lambda(q, \lambda) = \lambda\text{−closure}(q)$$

و به معنی آن است که وقتی در حالت q هستیم، ممکن است در چه حالاتی باشیم و از حالت‌ها با همان ورودی به چه حالت‌هایی وارد می‌شویم.

- حالت اولیه‌ی λ-NFA و NFA یکسان است:

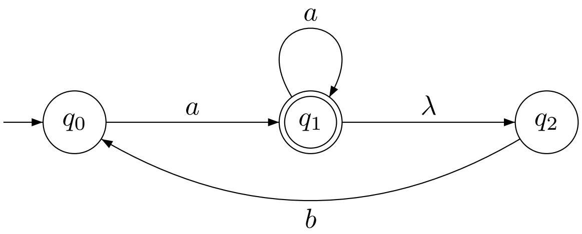
$$q_{۰_N} = q_{۰_\lambda}$$

- مجموعه‌ی حالات نهایی NFA به صورت زیر تعیین می‌شود:

$$F_N = F_\lambda \cup \{q_i : \delta^*_\lambda(q_i, \lambda) \cap F_\lambda \neq \varnothing\}$$

مثال

می‌خواهیم تابع δ_N را برای NFAی معادل با λ-NFA را به دست آوریم:

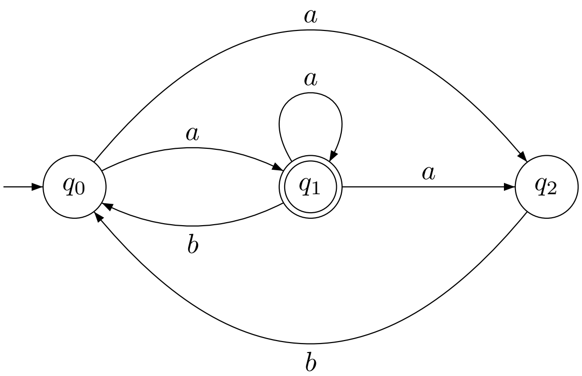


$$\lambda\text{−closure}(q_۰) = \{q_۰\}, \quad \lambda\text{−closure}(q_۱) = \{q_۱, q_۲\}, \quad \lambda\text{−closure}(q_۲) = \{q_۲\}$$

جدول گذر حالت برای δ_N به صورت زیر است:

	a	b
$q_۰$	$\{q_۱, q_۲\}$	$\{\}$
$q_۱$	$\{q_۱, q_۲\}$	$\{q_۰\}$
$q_۲$	$\{\}$	$\{q_۰\}$

گراف گذر حالت نیز به صورت زیر به دست می‌آید:



ملاحظه می‌شود که تعداد حالت‌های آتوماتون تغییری نکرده است، اما پس از حذف گذر λ تعدادی یال جدید به گراف اضافه شده است.

◄ **تذکر** تعداد گام‌های یک ماشین (DFA، NFA یا λ-NFA) برای پردازش رشته‌ی w برابر با $O(|w|)$ است. به عبارت دیگر، زمان پردازش در آتوماتون متناهی، خطی است.

۲-۵ کاهش تعداد حالات یک آتوماتون متناهی قطعی

یک DFA زبان منحصر به فردی را می‌پذیرد، اما عکس آن صادق نیست، یعنی DFAهای مختلفی برای پذیرش یک زبان وجود دارد. به عبارت دیگر، دو DFA ممکن است معادل باشند، اما تعداد حالات آنها متفاوت باشد.

برای کاهش تعداد حالات می‌توان از قواعد زیر استفاده کرد:

- حالت تله را می‌توان با تمام گذرهایش نادیده گرفت.
- حالت‌های دسترس‌ناپذیر از $q_۰$ را می‌توان با همه‌ی گذرهایش از DFA حذف کرد. با بررسی مسیرها از حالت شروع، حالاتی که در این مسیرها قرار ندارند، دسترس‌ناپذیرند.
- حالت‌های تکراری DFA (حالت‌های تمایزناپذیر/ معادل) را نیز می‌توان ادغام نمود.

تعریف **حالت‌های تمایزناپذیر/ معادل.** دو حالت p و q از یک DFA را تمایزناپذیر (یا معادل) می‌گوییم، اگر

$$\delta^*(p,w)\in F \Rightarrow \delta^*(q,w)\in F \qquad ,\forall w\in\Sigma^*$$

و

$$\delta^*(p,w)\notin F \Rightarrow \delta^*(q,w)\notin F \qquad ,\forall w\in\Sigma^*$$

تمایزناپذیری دارای خاصیت هم‌ارزی است.

تعریف **حالت‌های تمایزناپذیر/ معادل (تعریف بازگشتی).** دو حالت p و q را معادل مرتبه صفر گویند و می‌نویسند $p\equiv_۰ q$ هرگاه هر دوی این حالت‌ها نهایی یا هر دو غیرنهایی باشد.

$$p\equiv_۰ q \quad \textbf{iff} \quad p,q\in F \ \vee \ p,q\notin F$$

دو حالت p و q را معادل مرتبه k گویند و می‌نویسند $p\equiv_k q$ هرگاه p و q معادل مرتبه‌ی $k-۱$ بوده و به ازای هر $a\in\Sigma$، $\delta(p,a)$ و $\delta(q,a)$ هم معادل مرتبه‌ی $k-۱$ باشد.

$$p\equiv_k q \quad \textbf{iff} \quad p\equiv_{k-۱} q \ \wedge \ \forall a\in\Sigma(\delta(p,a)\equiv_{k-۱}\delta(q,a))$$

دو حالت p و q را معادل (تمایزناپذیر) گویند اگر و فقط اگر از همه‌ی مرتبه‌ها معادل باشند:

$$p\equiv q \quad \textbf{iff} \quad \forall k\geq ۰ \ \ p\equiv_k q$$

الگوریتم یافتن حالت‌های معادل برای می‌نیمم‌سازی یک DFA

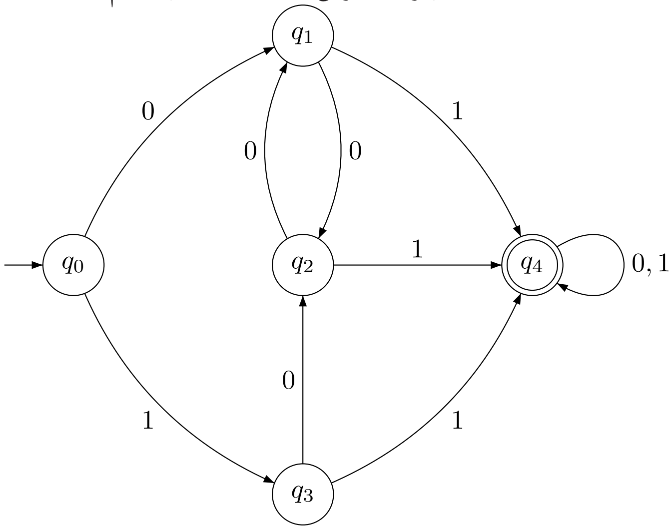
- حالت‌های دسترس‌ناپذیر را حذف کنید.
- از $k \leftarrow ۰$ شروع کنید.
- تاوقتی که دیگر هیچ دو حالت معادل جدیدی وجود ندارد:
  - حالت‌های معادل مرتبه‌ی k را بیابید و آنها را با یک نام مشترک مجدداً نام‌گذاری کنید.
  - تابع گذر حالت را با جایگذاری نام مشترک جدید برای حالت‌های معادل بازنویسی کنید.
  - $k \leftarrow k + ۱$

قضیه

اگر M یک DFA باشد، الگوریتم می‌نیمم‌سازی یک DFAی دیگر به نام $\hat{M}$ برمی‌گرداند که $L(M) = L(\hat{M})$. به علاوه $\hat{M}$ می‌نیمم است. به این معنا که هیچ DFAی دیگری با تعداد حالت کمتر که $L(M)$ را بپذیرد وجود ندارد.

مثال

می‌خواهیم DFAی معادل با DFAی زیر با کمترین تعداد حالات را بیابیم:



نخست، جدول گذر حالت را برای این DFA ایجاد می‌کنیم و در آن حالات نهایی و غیرنهایی را تفکیک می‌کنیم (تعیین حالات معادل مرتبه‌ی صفر):

	۰	۱	
$q_۰$	$q_۱$	$q_۳$	$g_۱^۰$
$q_۱$	$q_۲$	$q_۴$	
$q_۲$	$q_۱$	$q_۴$	
$q_۳$	$q_۲$	$q_۴$	
$q_۴$	$q_۴$	$q_۴$	$g_۲^۰$

یعنی،

	۰	۱	
q_0	g_1^0	g_1^0	
q_1	g_1^0	g_2^0	
q_2	g_1^0	g_2^0	g_1^0
q_3	g_1^0	g_2^0	
q_4	g_2^0	g_2^0	g_2^0

سپس از روی معادل‌های مرتبه‌ی صفر، معادل‌های مرتبه‌ی یک را تفکیک می‌کنیم:

	۰	۱	
q_0	g_2^1	g_2^1	g_1^1
q_1	g_2^1	g_3^1	
q_2	g_2^1	g_3^1	g_2^1
q_3	g_2^1	g_3^1	
q_4	g_3^1	g_3^1	g_3^1

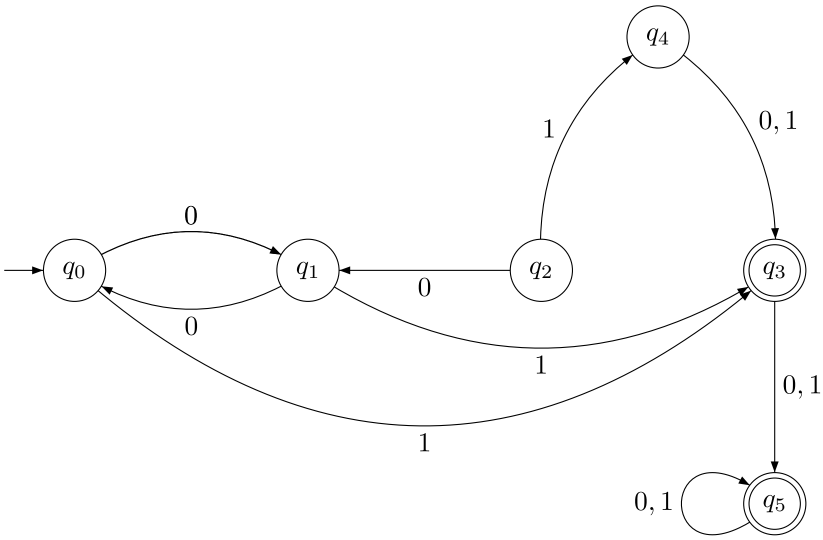
در اینجا کلیه‌ی حالت‌های تمایزناپذیر مشخص شده‌اند: حالت‌های q_1، q_2 و q_3 معادل هستند و در نهایت جدول گذر حالت برای DFAی می‌نیمال به صورت زیر خواهد بود:

	۰	۱
g_1	g_2	g_2
g_2	g_2	g_3
g_3	g_3	g_3

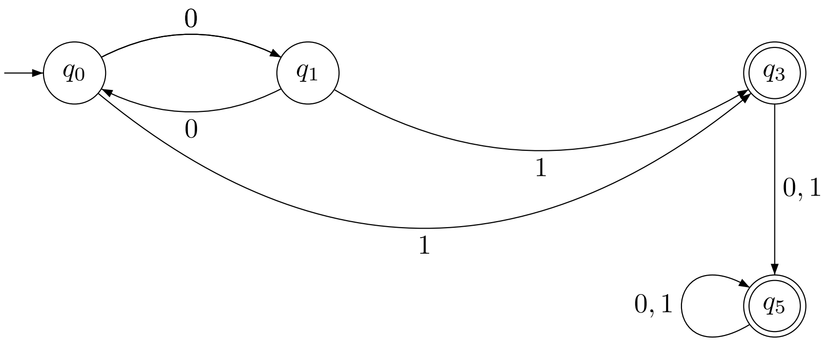
که در آن g_1 حالت آغازین و g_3 حالت نهایی است.

مثال

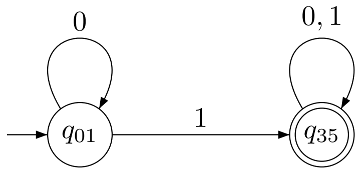
می‌خواهیم DFAی معادل با DFAی زیر با کمترین تعداد حالات را بیابیم:



ابتدا حالت‌های دسترس‌ناپذیر را کنار می‌گذاریم. حالت دسترس‌ناپذیر، حالتی است که هیچ مسیری با شروع از حالت آغازین بدان ختم نشود. در آتوماتون فوق، حالت q_2 و به تبع آن حالت q_4 دسترس‌ناپذیر هستند؛ بنابراین می‌توان آنها را از دیاگرام فوق حذف کرد:



حال مشاهده می‌شود که حالت‌های q_0 و q_1 با هم معادل هستند و همچنین حالت‌های q_3 و q_5 نیز معادل می‌باشند: هر دو حالت q_0 و q_1 با دیدن یک ۱ وارد حالت نهایی q_3 می‌شوند. حالت q_0 تعداد زوجی ۰ و حالت q_1 تعداد فردی ۰ صفر را در یک حلقه مصرف می‌کنند. در مجموع می‌توان هر دو را یک حالت q_{01} در نظر گرفت که تعدادی صفر را مصرف می‌کند و سپس با دیدن یک ۱ وارد یک حالت پایانی می‌شود. حالت q_3 یک حالت نهایی است که با دیدن ۰ یا ۱ وارد یک حالت نهایی دیگر q_5 می‌شود. در این حالت نیز ۰ یا ۱ دیده شده در همان حالت مصرف می‌شود. پس می‌توان هر دوی این حالات را به عنوان یک حالت نهایی q_{35} در نظر گرفت که هر نماد ورودی را در خودش مصرف می‌کند:



استفاده از روال کاهش حالات DFA نیز ما را به همین نتیجه می‌رساند.

▲

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.