Linguaggi regolari e automi a stati finiti

Automi a stati finiti

Gli automi a stati finiti sono usati come modello per

- Software per la progettazione di circuiti digitali.
- Analizzatori lessicali di un compilatore.
- Ricerca di parole chiave in un file o sul web.
- Software per verificare sistemi a stati finiti, come protocolli di comunicazione.

Concetti di base

- **Alfabeto**: Insieme fnito e non vuoto di simboli
  - Esempio: $\Sigma = \{0, 1\}$ alfabeto binario
  - Esempio: $\Sigma = \{a, b, c, \ldots, z\}$ insieme di tutte le lettere minuscole
  - Esempio: Insieme di tutti i caratteri ASCII
- **Stringa:** Sequenza finita di simboli da un alfabeto Σ, e.g. 0011001
- **Stringa vuota:** La stringa con zero occorrenze di simboli da Σ
  - La stringa vuota e' denotata con ϵ

Lunghezza di una stringa: Numero di posizioni per i simboli nella stringa.

$|w|$ denota la lunghezza della stringa w

$|0110| = 4, |\epsilon| = 0$

Potenze di un alfabeto: Σ^k = insieme delle stringhe di lunghezza k con simboli da Σ

Example: $\Sigma = \{0, 1\}$

$\Sigma^1 = \{0, 1\}$

$\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^0 = \{\epsilon\}$

L'insieme di tutte le stringhe su Σ e' denotato da Σ^*

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$

Anche:

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots$

$\Sigma^* = \Sigma^+ \cup \{\epsilon\}$

Concatenazione: Se x e y sono stringhe, allora xy e' la stringa ottenuta rimpiazzando una copia di y immediatamente dopo una copia di x

$x = a_1 a_2 \ldots a_i, y = b_1 b_2 \ldots b_j$

$xy = a_1 a_2 \ldots a_i b_1 b_2 \ldots b_j$

Esempio: $x = 01101, y = 110, xy = 01101110$

Nota: Per ogni stringa x

$$x\epsilon = \epsilon x = x$$

Linguaggi

Definizione: Se Σ e' un alfabeto, e $L \subseteq \Sigma^*$, allora L e' un linguaggio

Esempi di linguaggi:

- L'insieme delle parole italiane legali
- L'insieme dei programmi C legali
- L'insieme delle stringhe che consistono di n zeri seguiti da n uni

$$\{\epsilon, 01, 0011, 000111, \ldots\}$$

Altri esempi

- L'insieme delle stringhe con un numero uguale di zeri e di uni

$$\{\epsilon, 01, 10, 0011, 0101, 1001, \ldots\}$$

- L_P = insieme dei numeri binari il cui valore e' primo

$$\{10, 11, 101, 111, 1011, \ldots\}$$

- Il linguaggio vuoto $\emptyset$
- Il linguaggio $\{\epsilon\}$ consiste della stringa vuota

Nota: $\emptyset \neq \{\epsilon\}$

Nota: L'alfabeto Σ e' sempre finito

Cosa ci interessa?

- Studiare gli strumenti formali per definire (riconoscere linguaggi) linguaggi (partendo dagli automi a stati finiti)
- Problema: la stringa w e' un elemento di un linguaggio L?
- Che risorse computazionali sono necessarie per rispondere a questa domanda?
- Applicazione: Fare il parsing di un programma C = controllare se il programma e' corretto, e se lo e', produrre un albero di parsing.

Automi a stati finiti deterministici

Un DFA e' una quintupla

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

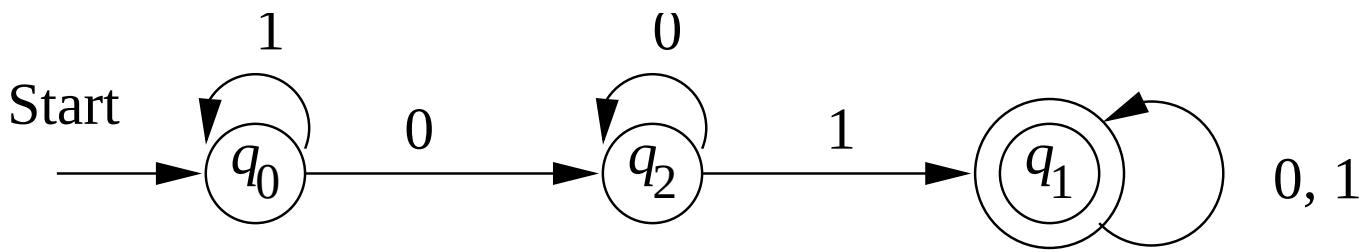
- Q e' un insieme finito di *stati*
- Σ e' un *alfabeto finito* ($=$ simboli in input)
- δ e' una *funzione di transizione* $\delta : Q \times \Sigma \mapsto Q$
- $q_0 \in Q$ e' lo *stato iniziale*
- $F \subseteq Q$ e' un insieme di *stati finali*

Diagramma di transizione

L'automa $A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$ come una *tabella di transizione*:

	0	1
$\rightarrow q_0$	q_2	q_0
$\star q_1$	q_1	q_1
q_2	q_2	q_1

L'automa come un *diagramma di transizione*:

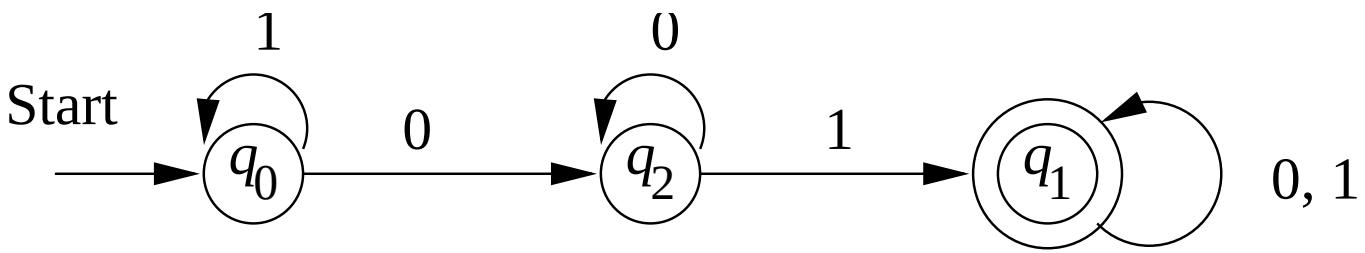


Accettazione

Un automa a stati finiti (FA) *accetta* una stringa $w = a_1 a_2 \cdots a_n$ se esiste un cammino nel diagramma di transizione che

1. Inizia nello stato iniziale
2. Finisce in uno stato finale (di accettazione)
3. Ha una sequenza di etichette $a_1 a_2 \cdots a_n$

Esempio: L'automa a stati finiti



accetta ad esempio la stringa 01101

Linguaggio accettato

- La funzione di transizione δ deve essere estesa alle stringhe
- introduciamo, $\hat{\delta} : Q \times \Sigma^* \mapsto Q$ che opera su stati e stringhe (invece che su stati e simboli)
- intuitivamente, $\hat{\delta}(p, w) = q$ significa che q e' lo stato raggiungibile da p avendo letto la stringa w

Formalmente

- La funzione di transizione estesa

$\hat{\delta} : Q \times \Sigma^* \mapsto Q$ si definisce induttivamente

Base: $$\hat{\delta}(q, \epsilon) = q$$

Induzione: $$\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$$

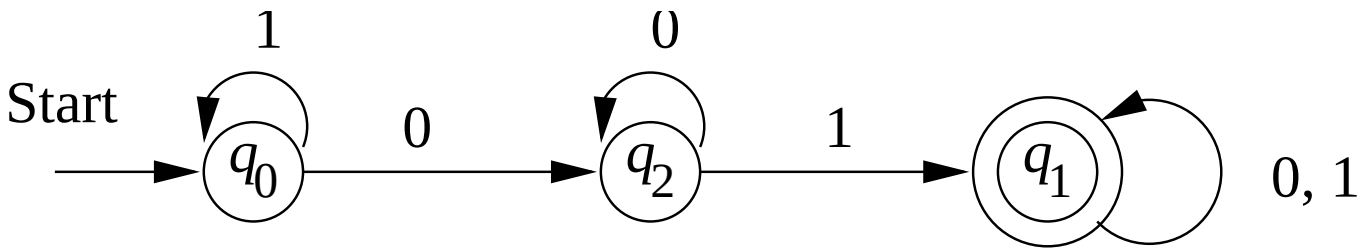
- Formalmente, il *linguaggio accettato da A* e'

$$L(A) = \{w : \hat{\delta}(q_0, w) \in F\}$$

- I linguaggi accettati da automi a stati finiti sono detti *linguaggi regolari*

Esempio

L'automa

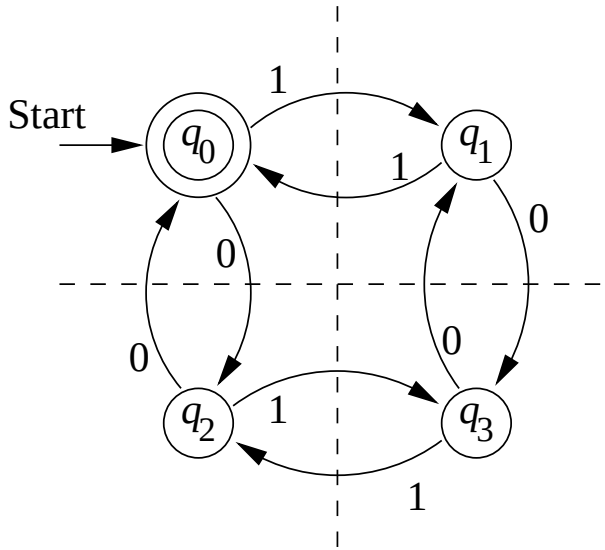


accetta il linguaggio

$$L = \{x01y : x, y \in \{0,1\}^*\}$$

Esempio

Esempio: DFA che accetta tutte e sole le stringhe con un numero pari di zeri e un numero pari di uni



Esempio

Rappresentazione tabulare dell'automa

		0	1
$\star \longrightarrow$	q_0	q_2	q_1
	q_1	q_3	q_0
	q_2	q_0	q_3
	q_3	q_1	q_2

Esercizi

- DFA per i seguenti linguaggi sull'alfabeto $\{0, 1\}$:
  - Insieme di tutte le strighe che finiscono con 00
  - Insieme di tutte le stringhe con tre zeri consecutivi
  - Insieme delle stringhe con 011 come sottostringa
  - Insieme delle stringhe che cominciano o finiscono (o entrambe le cose) con 01

Automi a stati finiti deterministici (DFA)

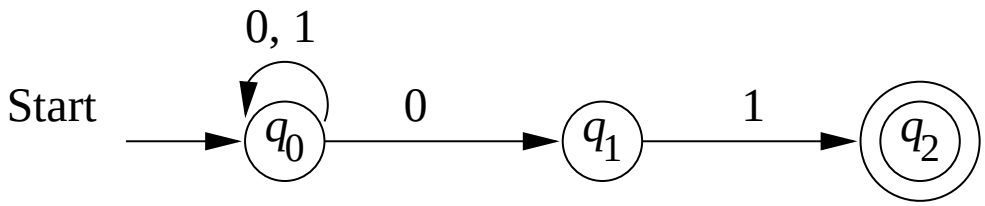
- un DFA puo' essere in un solo stato in ogni momento
- per ogni stringa di input w esiste *uno ed un solo* cammino nel diagramma di transizione, a partire dallo stato iniziale
- data una stringa w qual'e' il costo della decisione $w \in L(A)$?

Automi a stati finiti non deterministici (NFA)

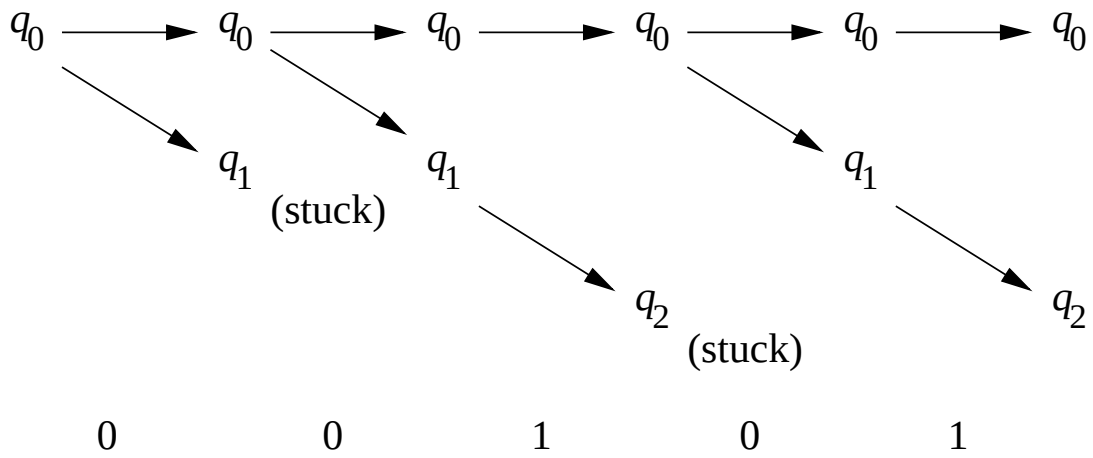
Un NFA puo' essere in vari stati nello stesso momento, oppure, visto in un altro modo, puo' "scommettere" su quale sara' il prossimo stato
Formalmente, quando l'automa e' in un certo stato e legge un certo simbolo puo' eseguire varie mosse

Esempio

Un automa che accetta tutte e solo le stringhe che finiscono in 01.



Nello stato q_0 leggendo 0 puo' muoversi in q_1 (e' arrivato al 01 finale) oppure rimanere in q_0
Ecco cosa succede quando l'automa elabora l'input 00101



Definizione formale di NFA

Formalmente, un NFA e' una quintupla

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

- Q e' un insieme finito di stati
- Σ e' un alfabeto finito
- δ e' una funzione di transizione $\delta : Q \times \Sigma \mapsto \wp(Q)$ (all'insieme dei sottoinsiemi di Q)
- $q_0 \in Q$ e' lo *stato iniziale*
- $F \subseteq Q$ e' un insieme di *stati finali*

Esempio

L' NFA di due pagine fa e'

$$(\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$$

dove δ e' la funzione di transizione

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$\star q_2$	$\emptyset$	$\emptyset$

Nota: dato un simbolo possono esserci anche zero mosse (tipo per q_1 e 0)

Linguaggio Accettato

- intuitivamente, una stringa w e' *accettata* da un NFA sse esiste un cammino dallo stato iniziale ad uno stato finale, etichettato con w (come abbiamo visto prima nell'esempio possono esistere vari di cammini)
- formalmente, dobbiamo definire la funzione di transizione estesa $\hat{\delta} : Q \times \Sigma^* \mapsto \wp(Q)$, che restituisce *un insieme di stati*
- $\hat{\delta}(p, w) = X$ significa che X e' l'insieme di stati raggiungibili da p avendo letto la stringa w

Funzione di Transizione Estesa

Definiamo $\hat{\delta} : Q \times \Sigma^* \mapsto \wp(Q)$, induttivamente

Base: $$\hat{\delta}(q, \epsilon) = \{q\}$$

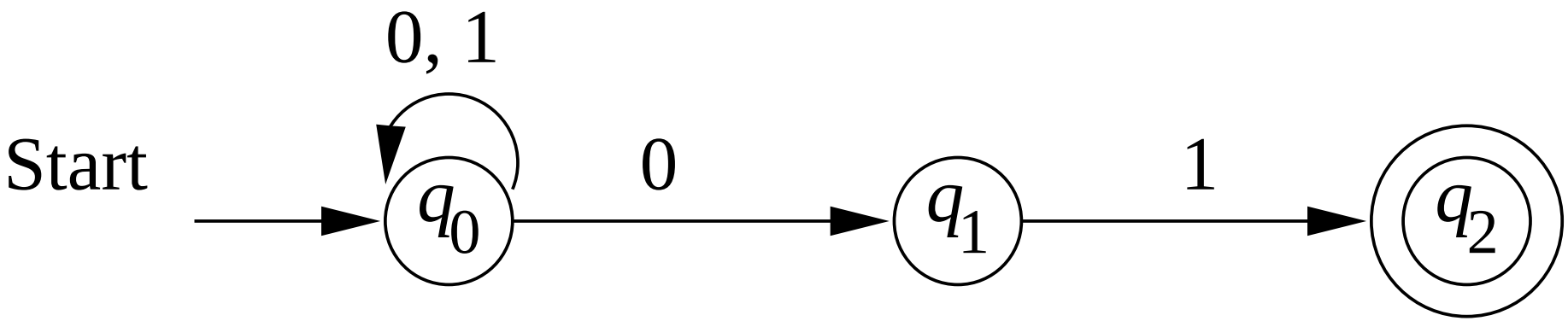
Induzione:

$$\hat{\delta}(q, xa) = \bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a)$$

- Il *linguaggio accettato da A* e'

$$L(A) = \{w : \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

Proviamo formalmente che l' NFA



accetta il linguaggio $\{x01 : x \in \Sigma^*\}$. Faremo una induzione mutua sui tre enunciati seguenti

0. $w \in \Sigma^* \Rightarrow q_0 \in \hat{\delta}(q_0, w)$
1. $q_1 \in \hat{\delta}(q_0, w) \Leftrightarrow w = x0$
2. $q_2 \in \hat{\delta}(q_0, w) \Leftrightarrow w = x01$

Base: Se $|w| = 0$ allora $w = \epsilon$. Allora l'enunciato (0) segue dalla defnizione Per (1) e (2) entrambi i lati sono falsi per ϵ
Induzione: Assumiamo che $w = xa$, dove $a \in \{0, 1\}$, $|x| = n$ e gli enunciati (0)–(2) valgono per x. Si mostra che gli enunciati valgono per xa.

Equivalenza di DFA e NFA

- Gli NFA sono di solito piu' facili da "programmare".
- per contro, qual'e' il costo della decisione $w \in L(A)$?
- Tuttavia, per ogni NFA N c'e' un DFA D, tale che $L(D) = L(N)$, e viceversa.
- DFA ed NFA hanno lo stesso potere espressivo

Equivalenza di DFA e NFA

- Sorprendentemente, per ogni NFA N c'e' un DFA D, tale che $L(D) = L(N)$, e viceversa.
- Questo comporta una *contruzione a sottonsiemi*, un esempio importante di come un automa B puo' essere costruito da un altro automa A.
- Dato un NFA

$$N = (Q_N, \Sigma, \delta_N, q_0, F_N)$$

costruiremo un DFA

$$D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$$

tali che

$$L(D) = L(N)$$

.

I dettagli della costruzione a sottoinsiemi:

- $Q_D = \{S : S \subseteq Q_N\}$
- $F_D = \{S \subseteq Q_N : S \cap F_N \neq \emptyset\}$
- Per ogni $S \subseteq Q_N$ e $a \in \Sigma$,

$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$$

Nota: $|Q_D| = 2^{|Q_N|}$, il numero degli stati cresce in modo esponenziale. Tuttavia, la maggior parte degli stati in Q_D sono "garbage", cioe' non raggiungibili dallo stato iniziale.

Costruiamo δ_D dall' NFA gia' visto:

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$\star\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\star\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\star\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$\star\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

Nota: Gli stati di D corrispondono a sottoinsiemi di stati di N, ma potevamo denotare gli stati di D in un altro modo, per esempio $A - F$.

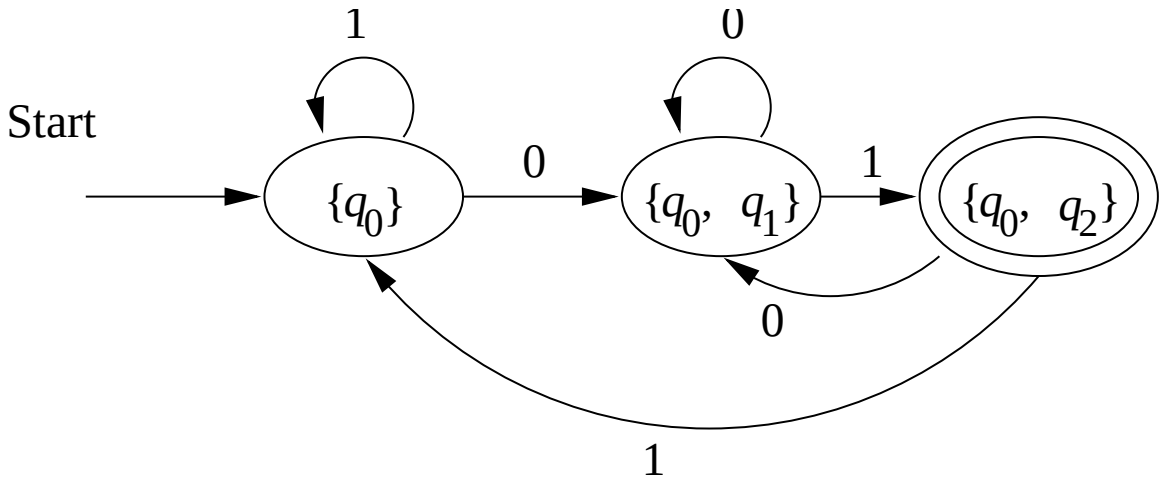
	0	1
A	A	A
$\rightarrow B$	E	B
C	A	D
$\star D$	A	A
E	E	F
$\star F$	E	B
$\star G$	A	D
$\star H$	E	F

Possiamo spesso evitare la crescita esponenziale degli stati costruendo la tabella di transizione per D solo per stati accessibili S come segue:

Base: $S = \{q_0\}$ e' accessibile in D

Induzione: Se lo stato S e' accessibile, lo sono anche gli stati in $\bigcup_{a \in \Sigma} \delta_D(S, a)$.

Esempio: Il "sottoinsieme" DFA con stati accessibili solamente.



Teorema 2.11: Sia D il DFA ottenuto da un NFA N con la costruzione a sottoinsiemi. Allora $L(D) = L(N)$.
Prova: Prima mostriamo per induzione su $|w|$ che

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$$

Base: $w = \epsilon$. L'enunciato segue dalla definizione.

Induzione:

$$\hat{\delta}_D(\{q_0\}, xa) \stackrel{\text{def}}{=} \delta_D(\hat{\delta}_D(\{q_0\}, x), a)$$

$$\stackrel{\text{i.h.}}{=} \delta_D(\hat{\delta}_N(q_0, x), a)$$

$$\stackrel{\text{cst}}{=} \bigcup_{p \in \hat{\delta}_N(q_0, x)} \delta_N(p, a)$$

$$\stackrel{\text{def}}{=} \hat{\delta}_N(q_0, xa)$$

Ora segue che $L(D) = L(N)$.

Teorema 2.12: Un linguaggio L e' accettato da un DFA se e solo se L e' accettato da un NFA.

Prova: La parte "se" e' il Teorema 2.11.

Per la parte "solo se" notiamo che un qualsiasi DFA puo' essere convertito in un NFA equivalente modificando la δ_D in δ_N secondo la regola seguente:

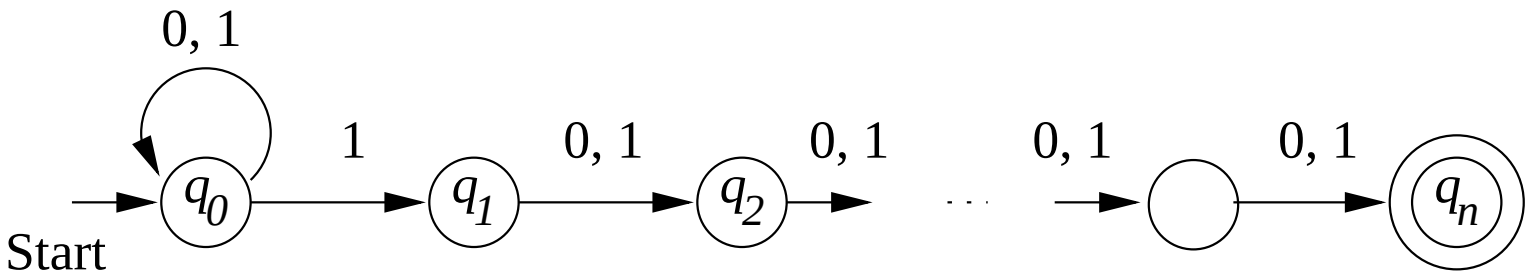
- Se $\delta_D(q, a) = p$, allora $\delta_N(q, a) = \{p\}$.

Per induzione su $|w|$ si puo' mostrare che se $\hat{\delta}_D(q_0, w) = p$, allora $\hat{\delta}_N(q_0, w) = \{p\}$.

L'enunciato del teorema segue.

Crescita esponenziale degli stati

Nella pratica il DFA ottenuto tramite la costruzione a sottoinsiemi ha approssimativamente lo stesso numero di stati del NFA equivalente.
Tuttavia, esiste la possibilita' che tutti i 2^n stati del DFA risultino raggiungibili dallo stato iniziale
Esempio: esiste un NFA N con $n+1$ stati che non ha nessun DFA D equivalente con meno di 2^n stati



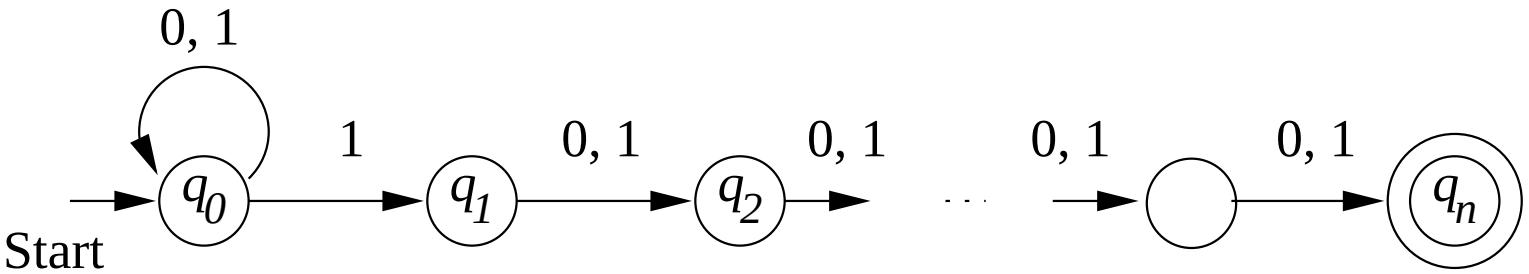
Linguaggio riconosciuto

- N si trova nello stato q_0 dopo avere letto una qualsiasi sequenza di input w
- N si trova nello stato q_i, per $i >= 1$, dopo avere letto $w1a_1...a_{i-1}$, dove a_j e' un simbolo di input
- il linguaggio riconsciuto da N sono le sequenze in cui l'ennesimo simbolo dalla fine e' 1,

$$L(N) = \{x1c_2c_3 \cdots c_n : x \in \{0,1\}^*, c_i \in \{0,1\}\}$$

Crescita esponenziale degli stati

Facciamo vedere che l'NFA N non ha nessun DFA D equivalente con meno di 2^n stati



Supponiamo che esista un DFA equivalente D con meno di 2^n stati. Intuitivamente, D deve ricordare gli ultimi n simboli che ha letto. Notiamo che ci sono 2^n sequenze di bit $a_1 a_2 \cdots a_n$. Quindi deve esserci uno stato raggiungibile dallo stato iniziale leggendo due stringhe diverse di lunghezza n.
Formalmente,
$\exists\ q, a_1 a_2 \cdots a_n, b_1 b_2 \cdots b_n : q \in \hat{\delta}_N(q_0, a_1 a_2 \cdots a_n), q \in \hat{\delta}_N(q_0, b_1 b_2 \cdots b_n), a_1 a_2 \cdots a_n \neq b_1 b_2 \cdots b_n$

Caso 1:

$1a_2 \cdots a_n$

$0b_2 \cdots b_n$

Allora q deve essere sia uno stato di accettazione che uno stato di non accettazione.

Caso 2:

$a_1 \cdots a_{i-1} 1 a_{i+1} \cdots a_n$

$b_1 \cdots b_{i-1} 0 b_{i+1} \cdots b_n$

Ora $\hat{\delta}_N(q_0, a_1 \cdots a_{i-1} 1 a_{i+1} \cdots a_n 0^{i-1}) =$
$\hat{\delta}_N(q_0, b_1 \cdots b_{i-1} 0 b_{i+1} \cdots b_n 0^{i-1})$

e $\hat{\delta}_N(q_0, a_1 \cdots a_{i-1} 1 a_{i+1} \cdots a_n 0^{i-1}) \in F_D$
$\hat{\delta}_N(q_0, b_1 \cdots b_{i-1} 0 b_{i+1} \cdots b_n 0^{i-1}) \notin F_D$

FA con transizioni epsilon

- presentiamo un altra estensione degli automi a stati finiti, gli *automi con ϵ-transizioni*
- intuitivamente, ammettiamo che esistano transizioni etichettate con la stringa vuota (l'automa esegue una mossa spontaneamente senza leggere input)
- questa estensione e' utile perche' facilita la specifica di linguaggi, ma non aggiunge potere espressivo

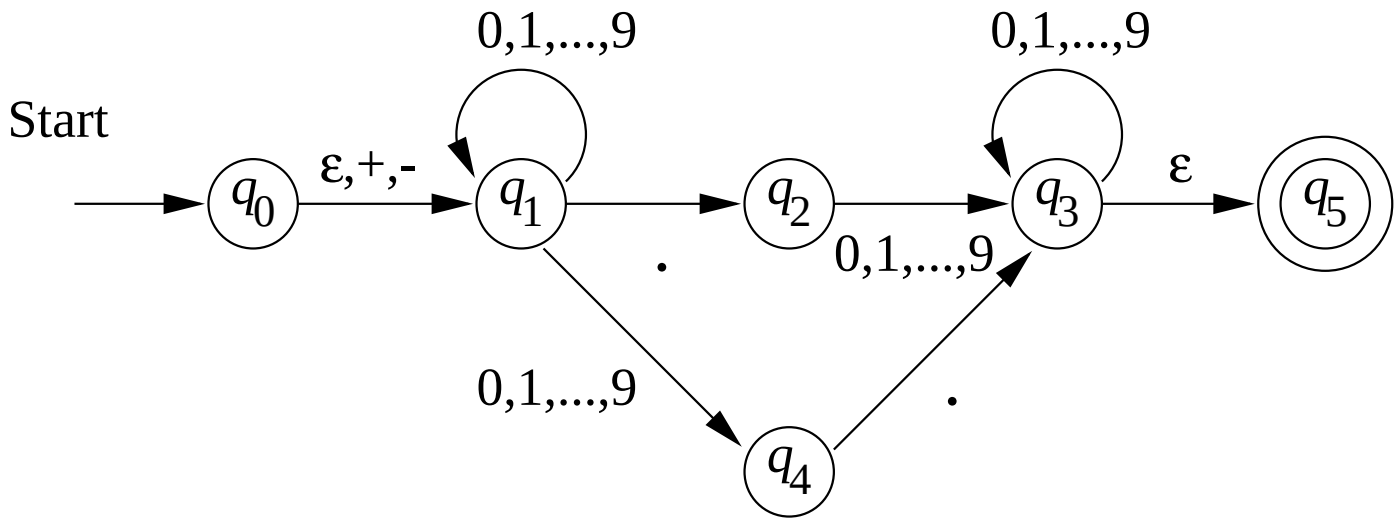
Esempio

Vogliamo definire un automa che accetta numeri decimali, ovvero stringhe definite come segue:

1. Un segno + o -, opzionale
2. Una stringa di cifre decimali
3. un punto decimale
4. un'altra stringa di cifre decimali

Una delle stringhe (2) e (4) sono opzionali (ma non entrambi)

FA con transizioni epsilon



- lo stato q_0 rappresenta la situazione in cui e' stato letto il segno (opzionale), ma non ancora il .
- lo stato q_2 rappresenta la situazione in cui e' stato letto il . ma non necessariamente cifre $0, \ldots, 9$
- lo stato q_4 rappresenta la situazione in cui e' stata letta almeno una cifra $0, \ldots, 9$, ma non ancora .
- lo stato q_3 rappresenta la situazione in cui e' stato letto il punto ., e almeno una cifra $0, \ldots, 9$, prima o dopo

Definizione Formale

Un ϵ-NFA e' una quintupla $(Q, \Sigma, \delta, q_0, F)$ dove

- Q, Σ, q_0, F come in NFA
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \mapsto \wp(Q)$ e' una funzione da $Q \times \Sigma \cup \{\epsilon\}$ all'insieme dei sottoinsiemi di Q.

Esempio

L' ϵ-NFA della pagina precedente

$$E = (\{q_0, q_1, \ldots, q_5\}, \{., +, -, 0, 1, \ldots, 9\}\ \delta, q_0, \{q_5\})$$

dove la tabella delle transizioni per δ e'

	ϵ	+,-	.	$0, \ldots, 9$
$\rightarrow q_0$	$\{q_1\}$	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$	$\{q_1, q_4\}$
q_2	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_3	$\{q_5\}$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_4	$\emptyset$	$\emptyset$	$\{q_3\}$	$\emptyset$
$\star q_5$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Linguaggio Accettato

- si definisce come per NFA, utilizzando pero' una nozione diversa di $\hat{\delta}$, funzione di transizione estesa
- $\hat{\delta}$ utilizza la nozione di ϵ-closure di uno stato

Epsilon-chiusura

Chiudiamo uno stato aggiungendo tutti gli stati raggiungibili da lui tramite una sequenza $\epsilon\epsilon\cdots\epsilon$

Definizione induttiva di ECLOSE(q)

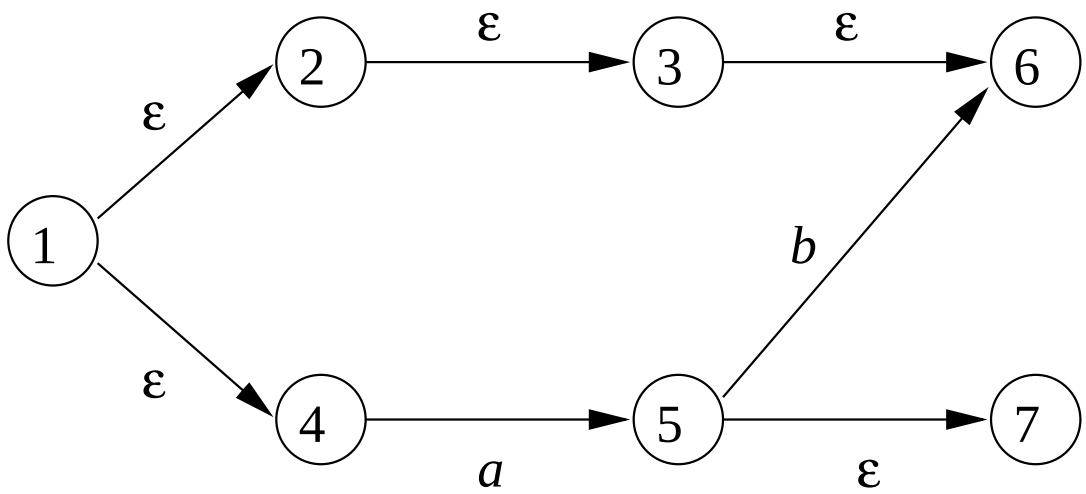
Base:

$q \in \text{ECLOSE}(q)$

Induzione:

$p \in \text{ECLOSE}(q)$ and $r \in \delta(p, \epsilon) \quad \Rightarrow r \in \text{ECLOSE}(q)$

Esempio di epsilon-chiusura



Per esempio,

$$\text{ECLOSE}(1) = \{1, 2, 3, 4, 6\}$$

Funzione di transizione estesa

- Definizione induttiva di $\hat{\delta}$ per automi ϵ-NFA

Base:

$$\hat{\delta}(q, \epsilon) = \text{ECLOSE}(q)$$

Tutti gli stati raggiungibili da q con ϵ-transizioni

Funzione di transizione estesa

- Definizione induttiva di $\hat{\delta}$ per automi ϵ-NFA

Induzione:

$$\hat{\delta}(q, xa) = \bigcup_{p \in \delta(\hat{\delta}(q,x),a)} \text{ECLOSE}(p)$$

Induttivamente, si considerano gli stati raggiungibili da q leggendo x, poi si fa una mossa in cui si legge a. Infine, si considerano tutti gli stati raggiungibili con ϵ-transizioni.

Esempio

Calcoliamo $\hat{\delta}(q_0, 5.6)$ per l'NFA dei numeri decimali

1. $\hat{\delta}(q_0, \epsilon) = \{q_0, q_1\}$;
2. calcoliamo $\hat{\delta}(q_0, 5)$. Abbiamo $\delta(q_0, 5) \cup \delta(q_1, 5) = \{q_1, q_4\}$, dove q_0 e q_1 sono calcolati nel passo precedente. Applicando la epsilon da q_1 e q_4 rimaniamo in q_1 e q_4 rispettivamente. Quindi $\hat{\delta}(q_0, 5) = \{q_1, q_4\}$.
3. Procedendo in modo analogo si trova $\hat{\delta}(q_0, 5.6) = ECLOSE(q_3) = \{q_3, q_5\}$. Siccome q_5 e' uno stato accettante, la stringa appartiene al linguaggio.

Da ϵ-NFA a DFA

Dato un ϵ-NFA

$$E = (Q_E, \Sigma, \delta_E, q_0, F_E)$$

costruiremo un DFA

$$D = (Q_D, \Sigma, \delta_D, q_D, F_D)$$

tale che

$$L(D) = L(E)$$

Differenza nella costruzione e' che e' sufficiente considerare come stati Q_D i sottoinsiemi di Q_E, che sono ϵ-chiusi chiusura. Formalemente, $S = \text{ECLOSE}(S)$ sta ad indicare che per ogni sttao $q \in S$, $ECLOSURE(q) \in S$. Nota che $E - CLOSE(\emptyset) = \emptyset$.

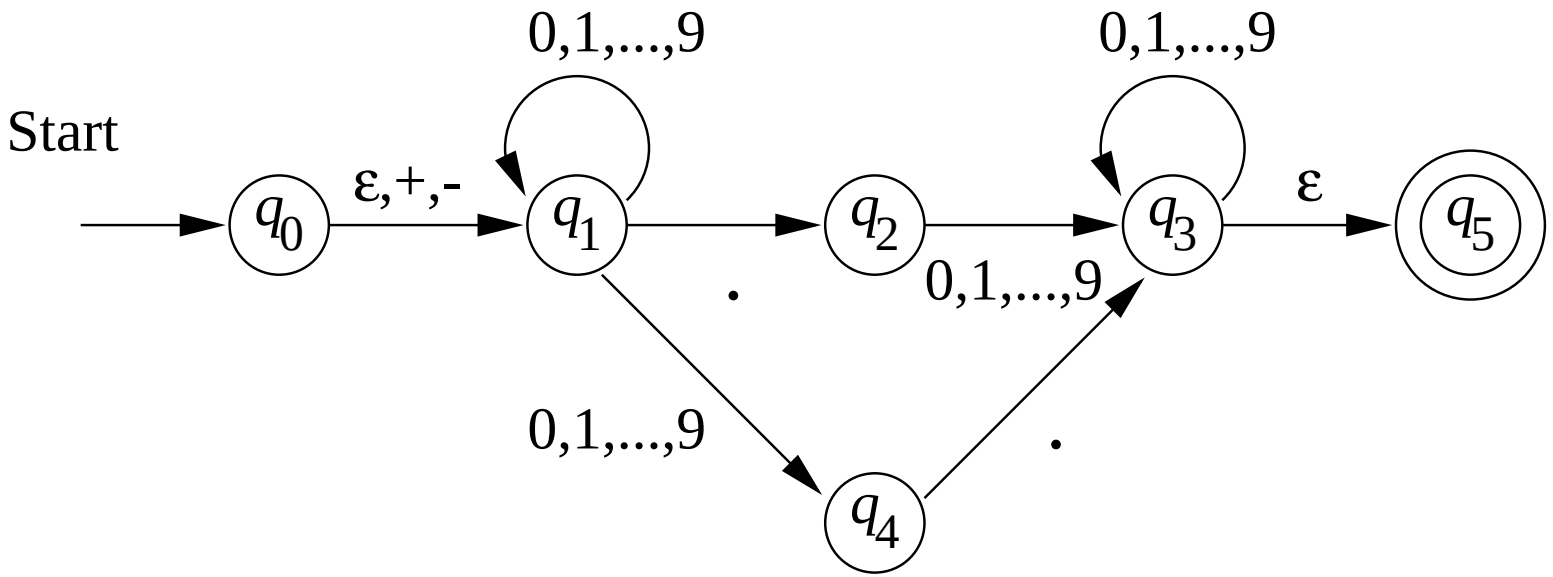
Dettagli della costruzione

- $Q_D = \{S : S \subseteq Q_E \text{ e } S = \text{ECLOSE}(S)\}$ i sottoinsiemi ϵ-chiusi
- $q_D = \text{ECLOSE}(q_0)$ gli stati raggiungibili da q_0 tramite ϵ-transizioni
- $F_D = \{S : S \in Q_D \text{ e } S \cap F_E \neq \emptyset\}$
- $\delta_D(S, a) =$

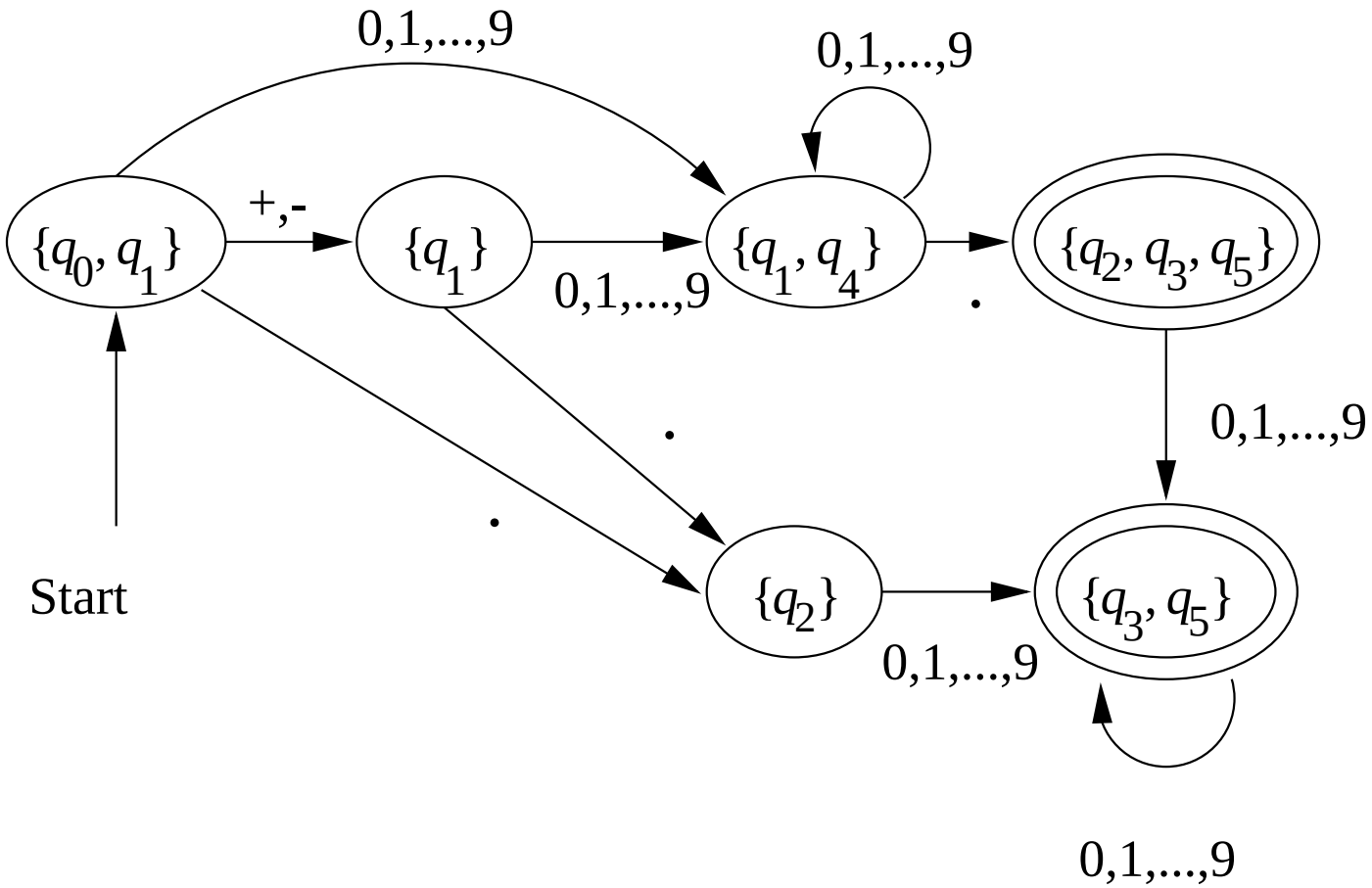
$$\bigcup \{\text{ECLOSE}(p) : p \in \delta(t, a) \text{ per alcuni } t \in S\}$$

Esempio

ϵ-NFA E



DFA D corrispondente ad E



Commenti Esempio

1. il DFA si costruisce a partire dallo stato iniziale, $\text{ECLOSE}(q_0) = \{q_0, q_1\}$;
2. poi si aggiungono gli insiemi di stati ϵ-chiusi, raggiungibili per ogni $a \in \Sigma$ (e cosi' via)
3. la figura non riporta lo stato trappola $\emptyset$ e le mosse verso di esso

Teorema 2.22: Un linguaggio L e' accettato da un ϵ-NFA E se e solo se L e' accettato da un DFA.
Prova: Usiamo D costruito come sopra e mostriamo per induzione che $\hat{\delta}_E(q_0, w) = \hat{\delta}_D(q_D, w)$

Base: $\hat{\delta}_E(q_0, \epsilon) = \text{ECLOSE}(q_0) = q_D = \hat{\delta}(q_D, \epsilon)$

Induzione:

$$\hat{\delta}_E(q_0, xa) = \bigcup_{p \in \delta_E(\hat{\delta}_E(q_0,x),a)} \text{ECLOSE}(p)$$

$$= \bigcup_{p \in \delta_D(\hat{\delta}_D(q_D,x),a)} \text{ECLOSE}(p)$$

$$= \bigcup_{p \in \hat{\delta}_D(q_D,xa)} \text{ECLOSE}(p)$$

$$= \hat{\delta}_D(q_D, xa)$$