Pattern Matching Mediante Automi

Aho-Corasick
Espressioni Regolari

Addea Simonetta
Camaroto Alessio
Tani Alice

Sommario

- Introduzione
- Nozioni Preliminari
  - Automi
- Pattern Matching Multiplo
  - Aho - Corasick
    - Bidimensionale
- Espressioni Regolari
- Conclusioni

Introduzione

- Introduzione
- Nozioni Preliminari
  - Automi
- Pattern Matching Multiplo
  - Aho - Corasick
    - Bidimensionale
- Espressioni Regolari
- Conclusioni

Introduzione

- La biologia computazionale ha come tema centrale la ricerca di sequenze molecolari all'interno di sequenze più lunghe, allo scopo di
  - ricostruire una sequenza di DNA dai singoli pezzi ottenuti da esperimenti
  - ricercare delle specifiche caratteristiche nelle sequenze di DNA
  - determinare quanto sono differenti due sequenze di codice genetico
  - ricercare motivi, strutturati e non, che occorrono frequentemente nel DNA

Il problema ...

- **Quando si ottiene una nuova sequenza, le possibili domande da porsi sono**
  1. **la sequenza presenta somiglianze con qualche altra sequenza già memorizzata in una banca dati ?**
  2. **la sequenza è già presente nella banca dati ?**

… Pattern Matching

1. **La sequenza presenta somiglianze con qualche altra sequenza già memorizzata in una banca dati ?**
   - **Bisogna risolvere un problema di pattern matching approssimato**
     - **Data una stringa pattern P e una stringa testo T, vogliamo cercare occorrenze di sequenze "simili" a P in T**

… Pattern Matching

2. **La sequenza è già presente nella banca dati ?**
   - **Bisogna risolvere un problema di pattern matching esatto**
     - **Data una stringa pattern P di lunghezza m e una stringa testo T di lunghezza n, entrambe definite sull'alfabeto Σ, vogliamo stabilire se P occorre in T**

Approcci Algoritmici

- **Abbiamo visto vari approcci algoritmici al problema**
  - **Forza Bruta**
  - **Morris - Pratt**
  - **Knuth - Morris - Pratt**
  - **Karp - Rabin**
  - **Suffix Tree**
  - **...**

Approcci Algoritmici

- **Noi vedremo un approccio basato su automi per risolvere il problema del pattern matching**
  - **esatto nel caso multiplo: Aho-Corasick**
  - **esatto: Espressioni Regolari**

Nozioni Preliminari: Automi

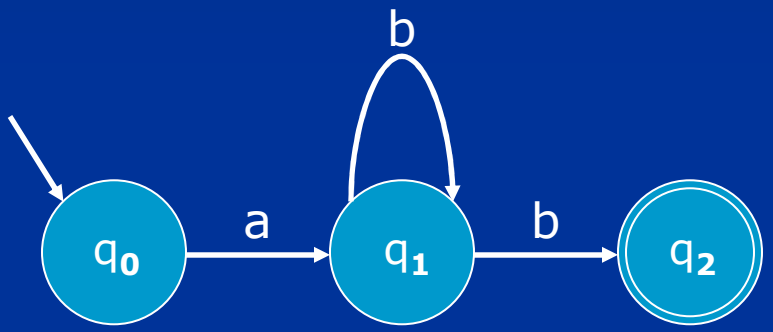
- Introduzione
- Nozioni Preliminari
  - Automi
- Pattern Matching Multiplo
  - Aho - Corasick
    - Bidimensionale
- Espressioni Regolari
- Conclusioni

Automi

- Un automa a stati finiti è una quintupla $(Q, \Sigma, \delta, q_0, F)$
  - Q è un insieme finito di stati
  - Σ è un alfabeto finito di caratteri di input
  - $q_0 \in Q$ è lo stato iniziale
  - $F \subseteq Q$ è l'insieme degli stati finali
  - δ è la funzione di transizione $Q \times \Sigma \rightarrow Q$
    - ad ogni coppia (stato, carattere) associa un insieme di stati successivi, Q

δ: Esempio

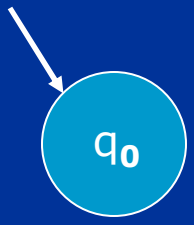
- **Due modalità di rappresentazione**



	a	b
q_0	$\{q_1\}$	/
q_1	/	$\{q_1, q_2\}$
q_2	/	/

Rappresentazione Grafica

Rappresentazione Tabellare

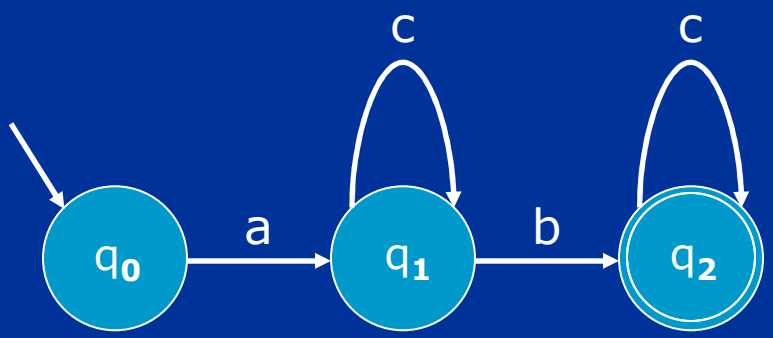


Stato Iniziale

Stato Finale

Automi Deterministici (DFA)

- **Un DFA è un automa tale che**
  - **per ogni stato s e ogni carattere di ingresso x, c'è al più una transizione dallo stato s la cui etichetta contiene x**



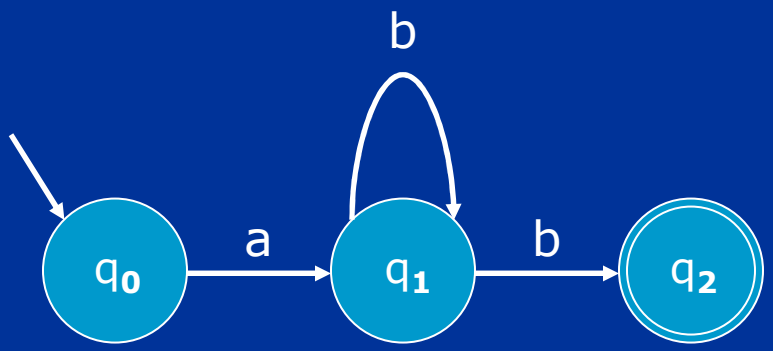
	a	b	c
q_0	$\{q_1\}$	/	/
q_1	/	$\{q_2\}$	$\{q_1\}$
q_2	/	/	$\{q_2\}$

Rappresentazione Grafica

Rappresentazione Tabellare

Automi Non Deterministici (NFA)

- **Un NFA è un automa tale che**
  - **può avere (ma non necessariamente) due o più transizioni, a partire dallo stesso stato, che contengono lo stesso simbolo**



Rappresentazione Grafica

	a	b
q_0	$\{q_1\}$	/
q_1	/	$\{q_1, q_2\}$
q_2	/	/

Rappresentazione Tabellare

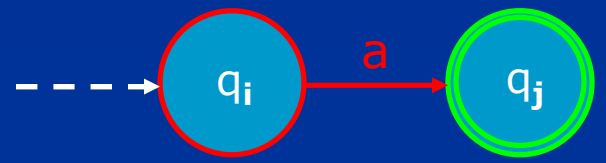
A cosa servono gli automi ?

- **Un automa**
  - **definisce un linguaggio L, ossia un insieme di stringhe su un alfabeto Σ**
  - **riconosce se una data stringa, detta input, appartiene ad L**

Riconoscimento Stringhe

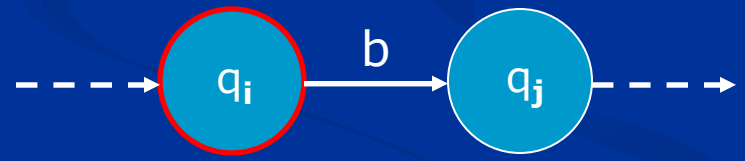
- **Riconoscimento**

$T = \ldots a \quad \delta(q_i, a) = q_j$

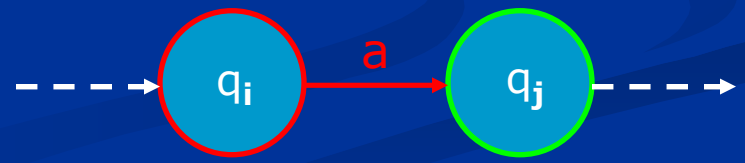


- **Non Riconoscimento**

$T = \ldots a \ldots \quad \delta(q_i, a) = \{\}$



$T = \ldots a \quad \delta(q_i, a) = q_j$



Pattern Matching Multiplo

- Introduzione
- Nozioni Preliminari
  - Automi
- **Pattern Matching Multiplo**
  - Aho - Corasick
    - Bidimensionale
- Espressioni Regolari
- Conclusioni

Pattern Matching Multiplo

- Ricerca simultanea di più pattern $P_1,\ldots, P_k$ in un testo T
- Approcci possibili
  - Eseguire k ricerche indipendenti, una per ciascun pattern P_i da trovare nel testo T => O(k*costo singola ricerca)
  - Aho-Corasick: utilizza la similarità tra i pattern da ricercare => $O(|P_1|+\ldots+|P_k|+|T|)$

Aho-Corasick

- Introduzione
- Nozioni Preliminari
  - Automi
- Pattern Matching Multiplo
  - Aho - Corasick
    - Bidimensionale
- Espressioni Regolari
- Conclusioni

Definizione

- **L'automa di Aho-Corasick è un automa deterministico esteso, definito da due funzioni**
  1. **la funzione di transizione** $\delta: Q \times \Sigma \rightarrow Q$
  2. **la funzione di fallimento** $\varphi: Q \rightarrow Q$

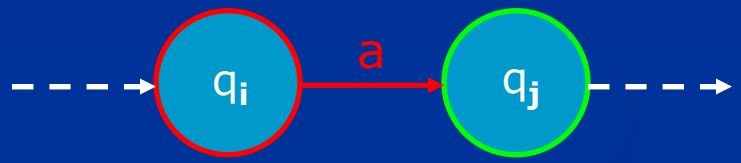
Funzione di transizione δ

1. **La funzione di transizione δ**
   - **dato uno stato i e il carattere del testo c, δ(i,c) restituisce**
     - **il prossimo stato dell'automa se esiste (match)**
     - **un valore vuoto (mismatch)**

Match vs. Mismatch

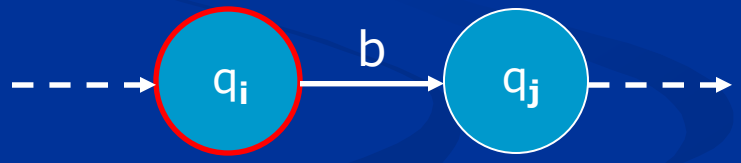
- **Match**

$T = \ldots a \ldots \quad \delta(q_i, a) = q_j$



- **Mismatch**

$T = \ldots a \ldots \quad \delta(q_i, a) = \{\}$



Funzione di fallimento φ

2. **La funzione di fallimento φ**
   - **φ(i) restituisce il prossimo stato dell'automa, in caso di mismatch con il carattere corrente c nello stato i, ossia nel caso in cui δ(i,c) fornisca un valore vuoto**
   - **il valore di φ(i) è determinato sfruttando una variante del concetto di bordo di una stringa già visto in KMP**

Modifica del concetto di Bordo

- In KMP, il bordo di una stringa s è definito come
  - bordo(s) = la più lunga sottostringa propria di s che ne è sia suffisso che prefisso
- In Aho-Corasick, il bordo di una stringa s è definito come
  - bordo(s) = il più lungo suffisso proprio di s che è anche prefisso di almeno uno dei pattern $P_1,\ldots,P_k$ da ricercare nel testo T

Costruzione Automa

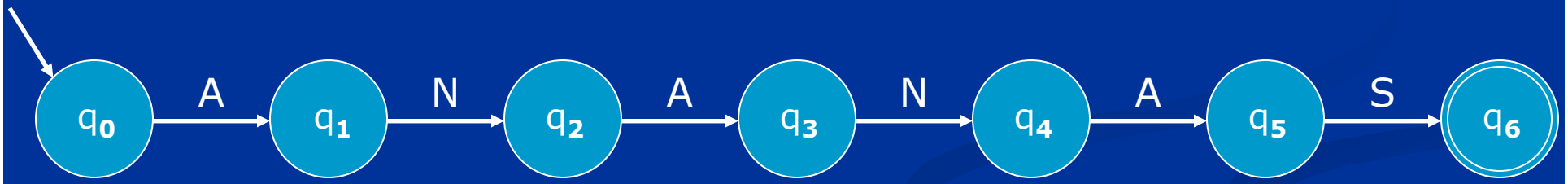
- **Complessità: $O(|P_1|+...+|P_k|)$**
- **La costruzione dell'automa avviene in 2 fasi**
  - **fase 1: costruzione dello scheletro dell'automa che riconosce i pattern $P_1...P_k$**
    - **definizione della funzione δ**
  - **fase 2: definizione della funzione φ**

Costruzione Automa

- **Fase 1: costruzione dello scheletro dell'automa per i pattern $P_1,\ldots,P_k$**
  1. **Costruiamo l'automa che riconosce il pattern P_1**

Esempio

- P_1 = ANANAS



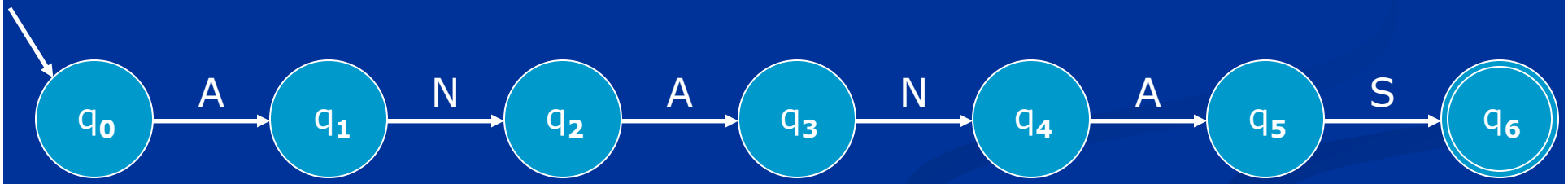
Costruzione Automa

2. **Per $1 \leq i < k$**
   a. **calcoliamo il più lungo prefisso in comune, lcp, tra P_{i+1} e tutti i pattern già inseriti nell'automa**

Esempio

- P_1 = ANANAS, P_2 = ANATRA

  lcp = ANA

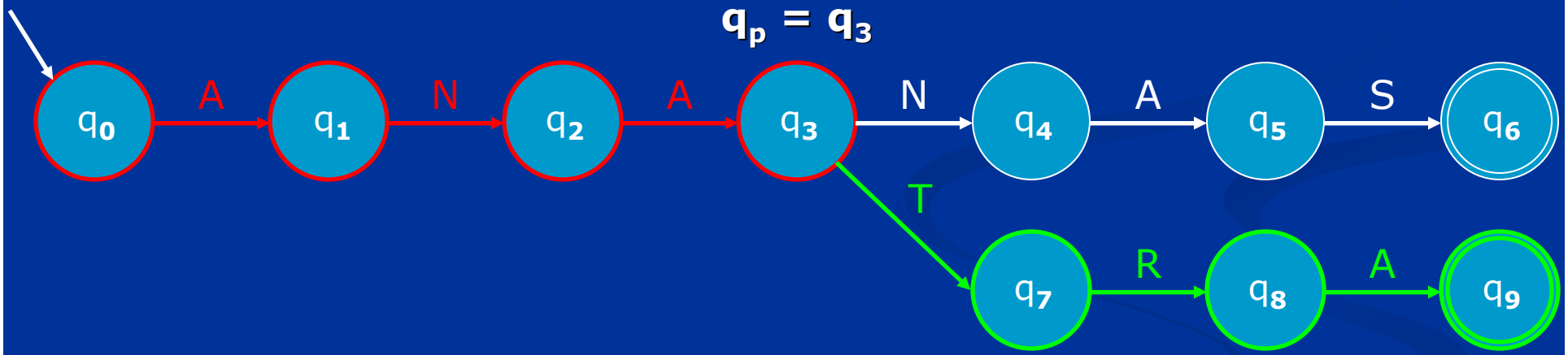


Costruzione Automa

b. **Usiamo la funzione δ per visitare l'automa a partire dallo stato iniziale fino allo stato q_p in cui termina l'lcp calcolato al punto a. Se l'lcp è**
   - **la stringa vuota, \$, si collega l'automa che riconosce l'intero pattern P_{i+1} allo stato iniziale dell'automa di Aho-Corasick**
   - **diverso dalla stringa vuota, si collega l'automa che riconosce i rimanenti $\{P_{i+1} - lcp\}$ caratteri del pattern P_{i+1}, allo stato q_p**

Esempio

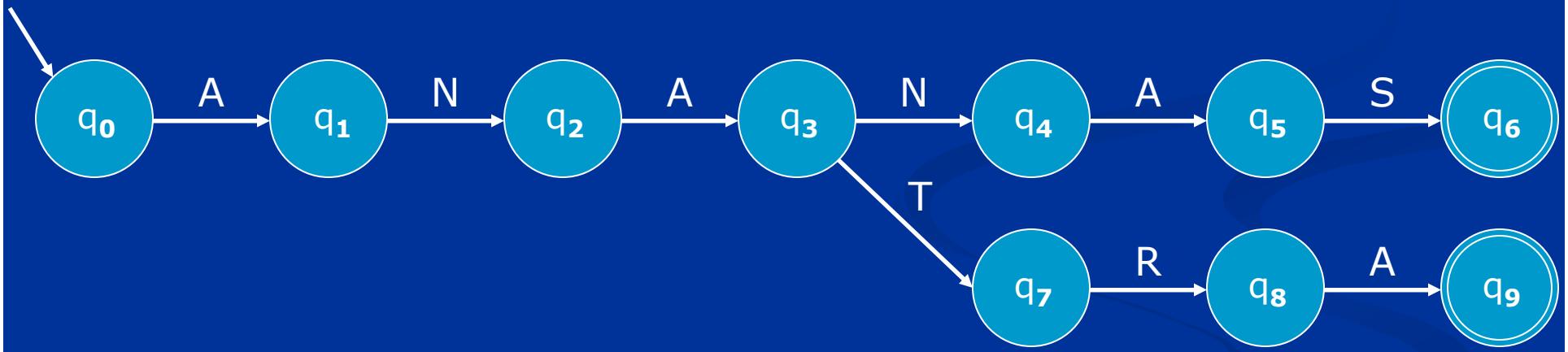
- P_1 = ANANAS, P_2 = ANATRA



$\{P_2 - lcp\}$ = TRA

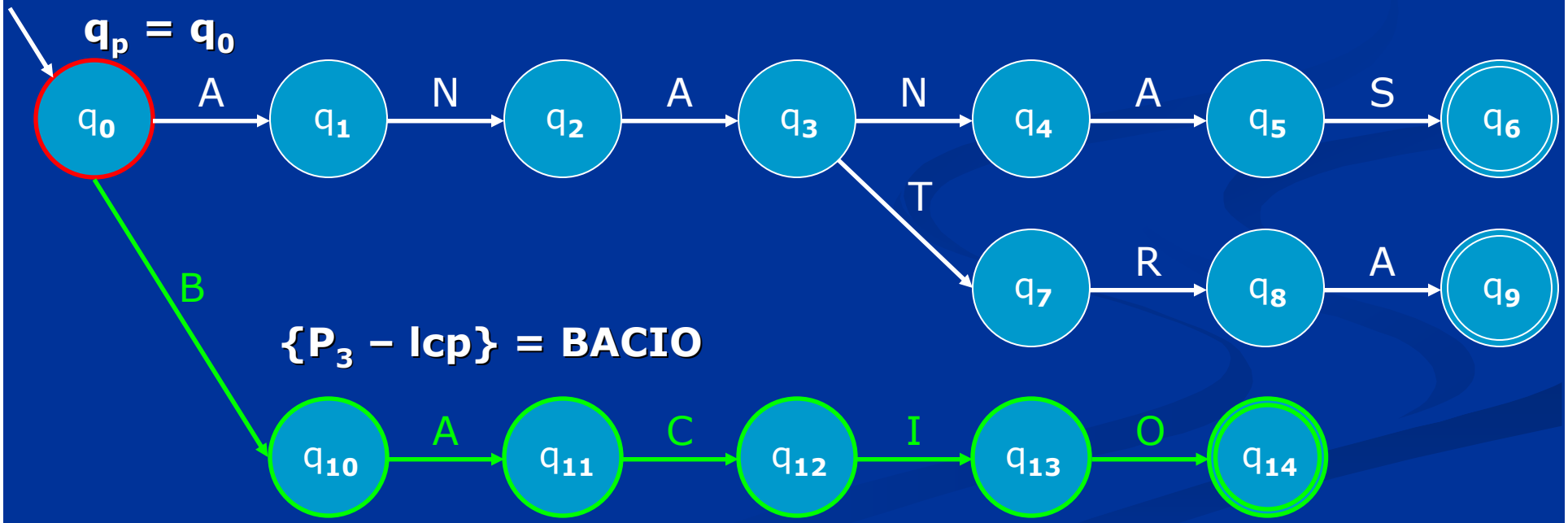
Esempio

- P_1 = $ANANAS, P_2 = $ANATRA
- P_3 = $BACIO lcp = $



Esempio

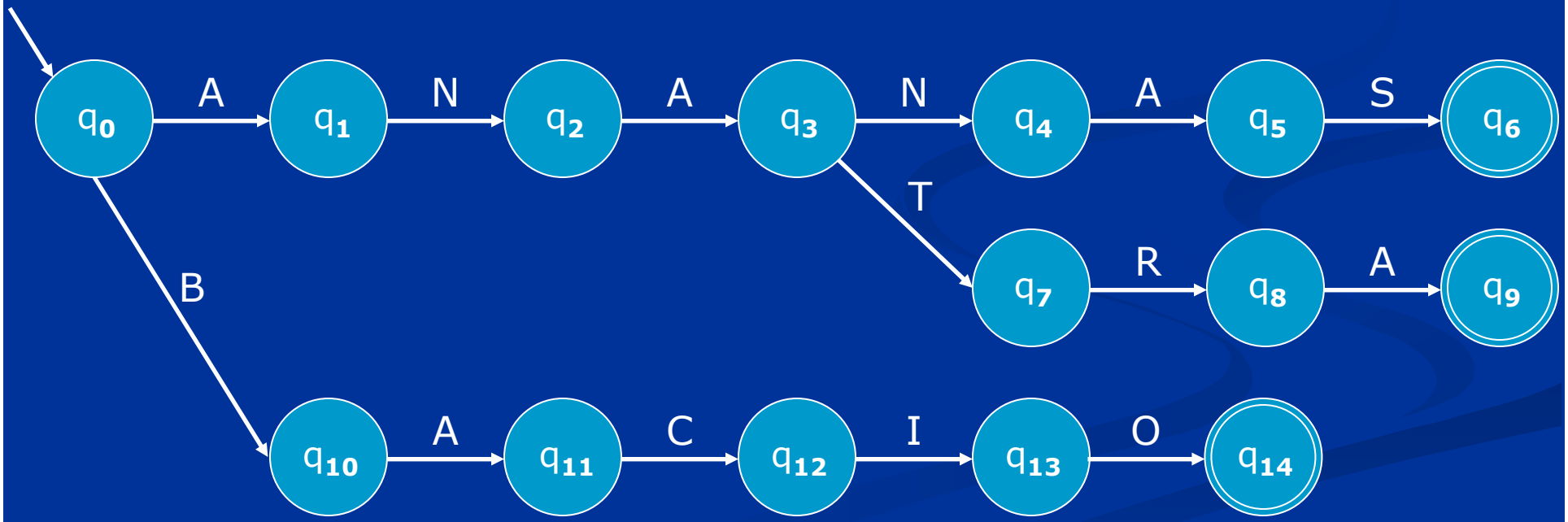
- P_1 = $ANANAS, P_2 = $ANATRA
- P_3 = $BACIO



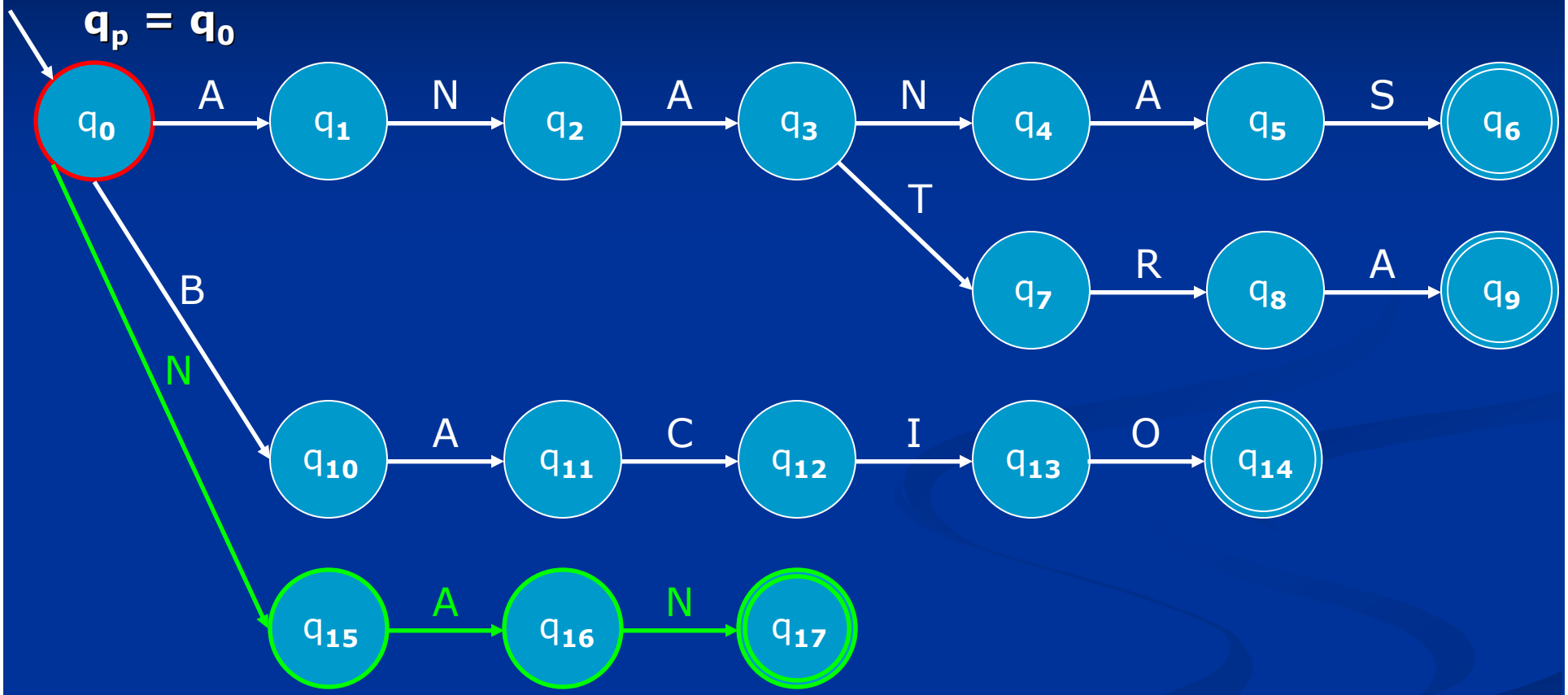
Esempio

- P_1 = $ANANAS, P_2 = $ANATRA
- P_3 = $BACIO, P_4 = $NAN

lcp = $

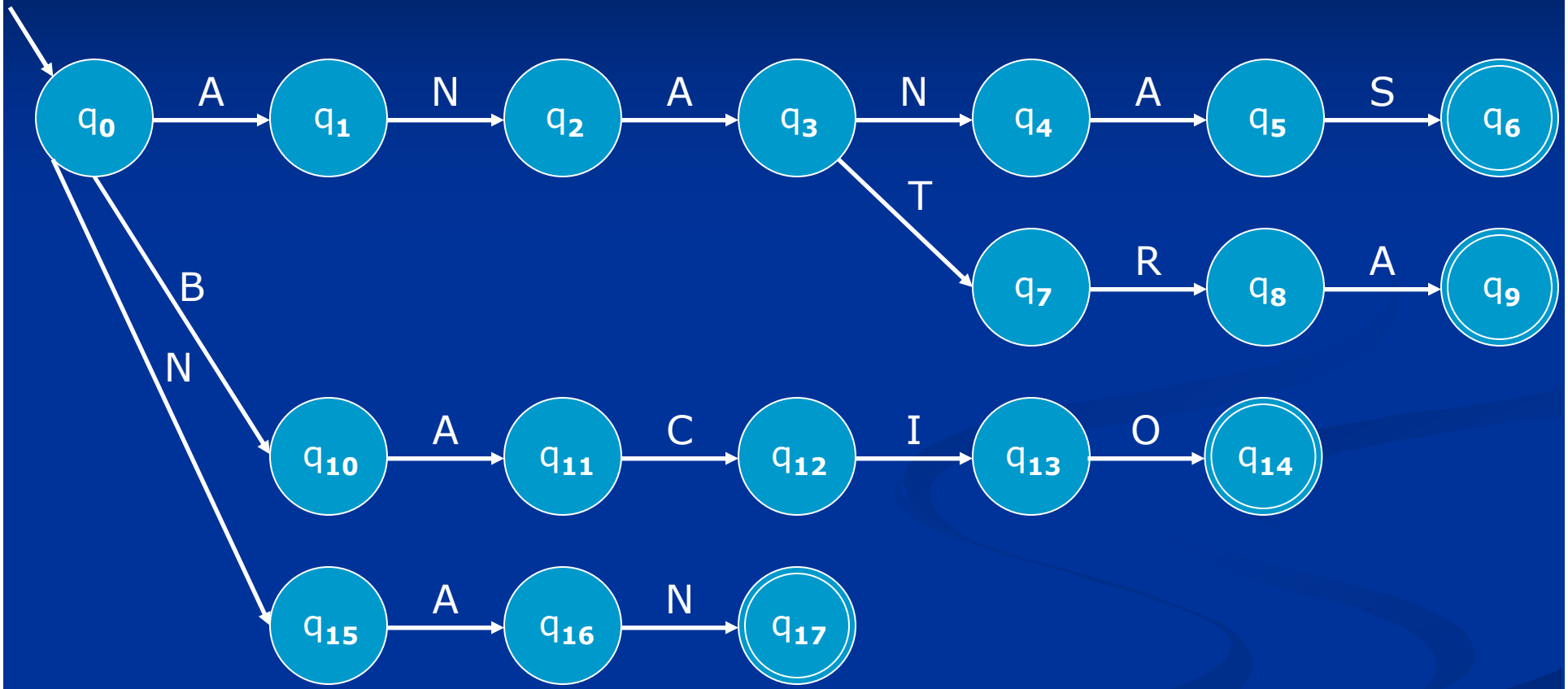


Esempio



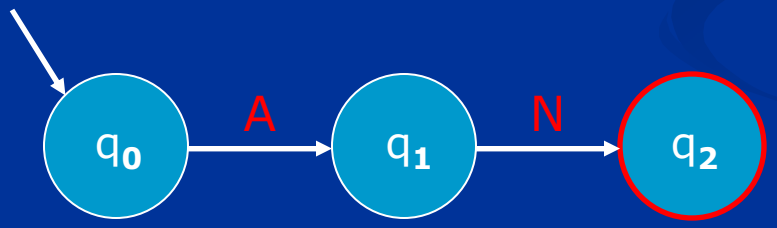
$\{P_4 - lcp\} = NAN$

Esempio


q0
A
q1
N
q2
A
q3
N
q4
A
q5
S
q6
T
q7
R
q8
A
q9
B
N
q10
A
q11
C
q12
I
q13
O
q14
q15
A
q16
N
q17

Costruzione Automa

- **Fase 2: definizione della funzione φ**
  - **per ogni stato i dell'automa**
    - **Sia L(qi) il prefisso associato allo stato qi, ossia la stringa ottenuta visitando l'automa a partire dallo stato iniziale q_0 fino a q_i**



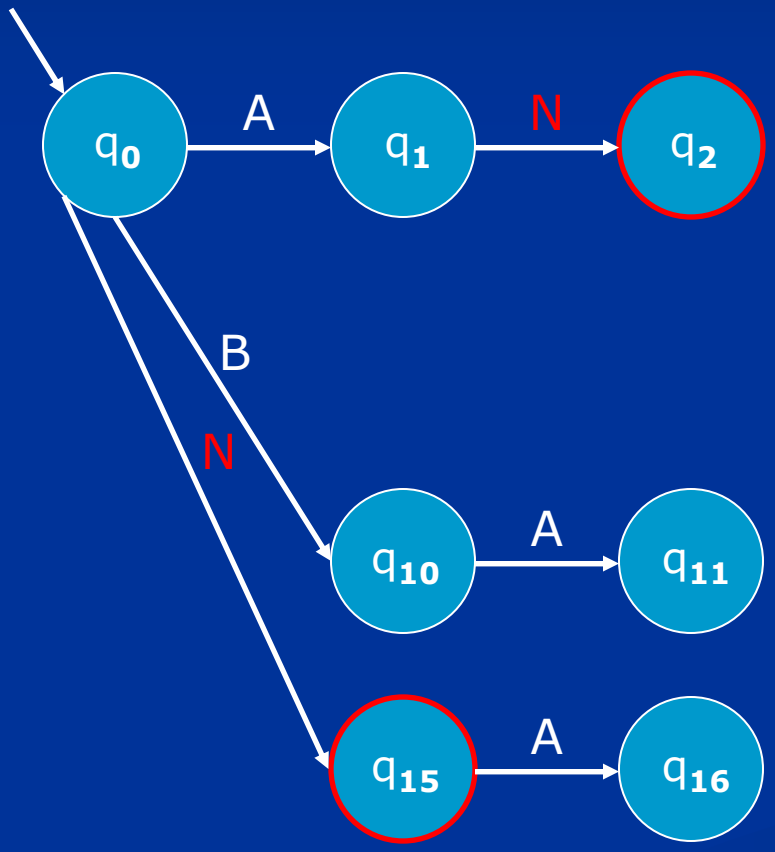
$L(q_2)$= AN

Costruzione Automa

- φ(i) è calcolata nel seguente modo
  - $\varphi(q_0) = -1$
  - per i > 0,
    1. si calcola il bordo di $L(q_i)$ come definito in precedenza
    2. si pone $\varphi(q_i)$ = stato in cui termina tale bordo

φ: Esempio

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
φ(i)	-1	0	15	16	17	3	0	0	0	1	0	1	0	0	0	0	1	2



$L(q_2)$ = AN

Bordo($L(q_2)$) = N

P1 = ANANAS

P2 = ANATRA

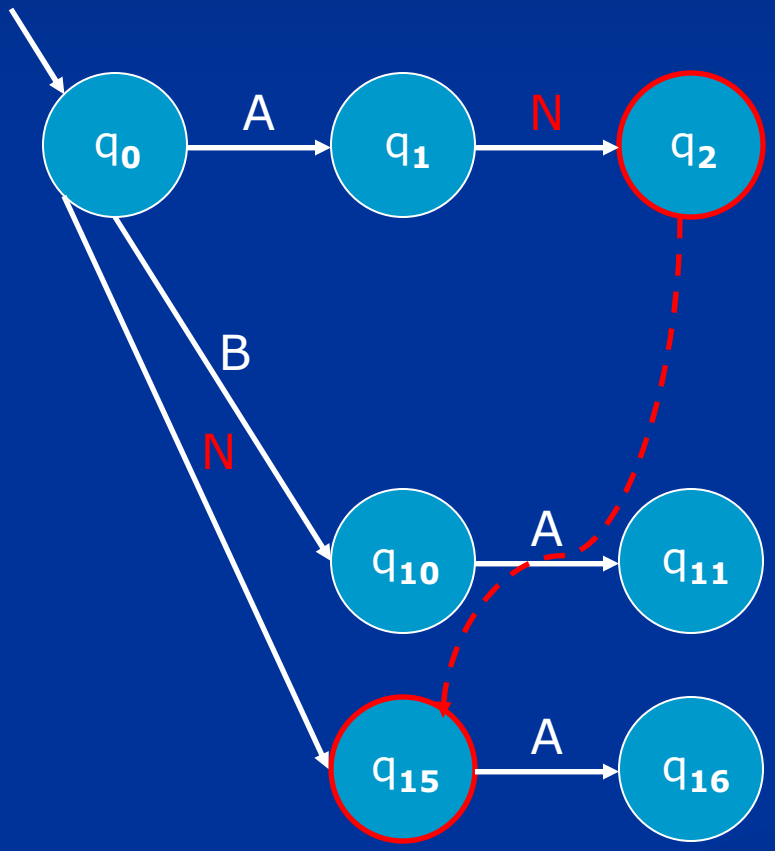
P3 = BACIO

P4 = NAN

$\varphi(q_2) = q_{15}$

φ: Esempio

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
φ(i)	-1	0	15	16	17	3	0	0	0	1	0	1	0	0	0	0	1	2



$L(q_2)$ = AN

Bordo($L(q_2)$) = N

P1 = ANANAS

P2 = ANATRA

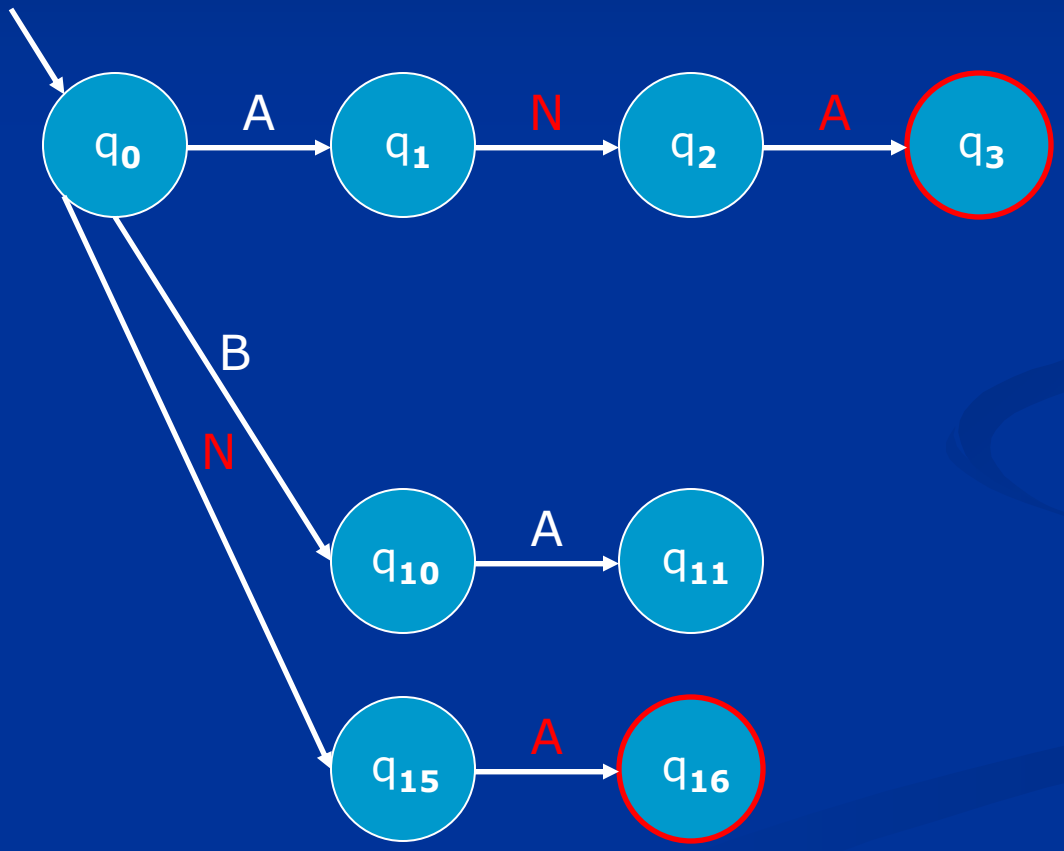
P3 = BACIO

P4 = NAN

$\varphi(q_2) = q_{15}$

φ: Esempio

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
φ(i)	-1	0	15	16	17	3	0	0	0	1	0	1	0	0	0	0	1	2



$L(q_3)$ = ANA

Bordo($L(q_3)$) = NA

P1 = ANANAS

P2 = ANATRA

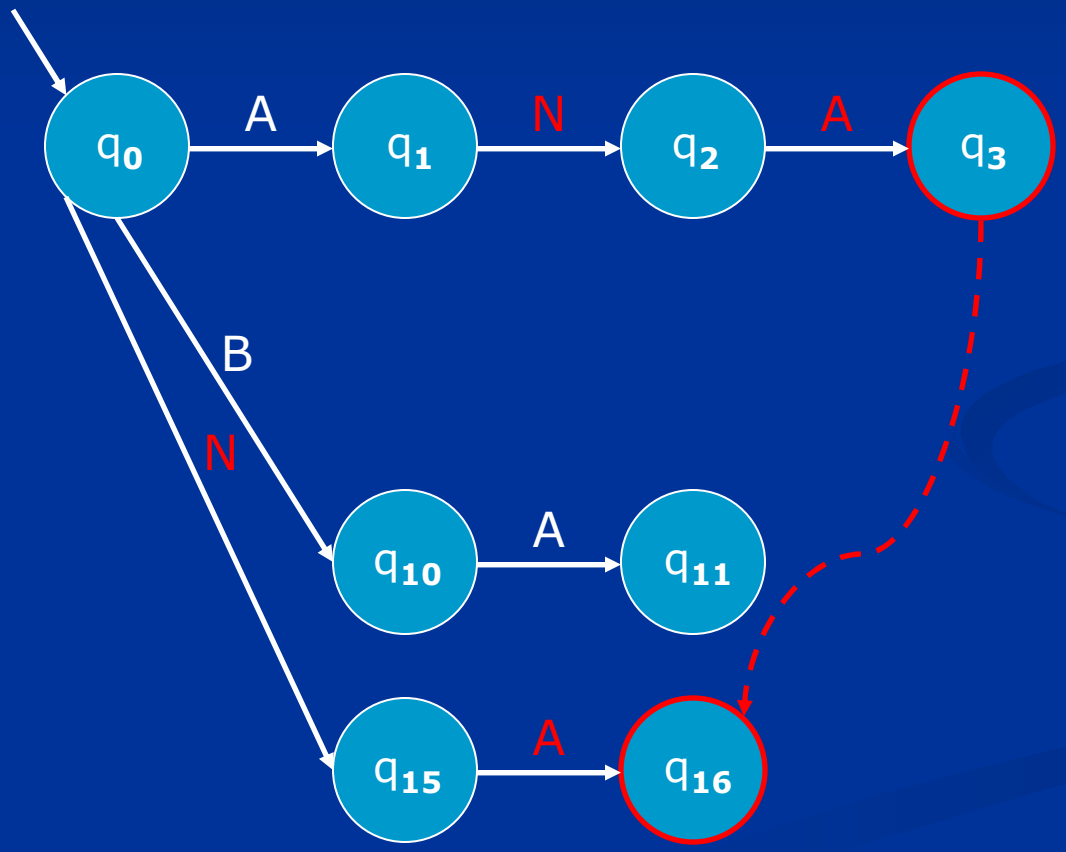
P3 = BACIO

P4 = NAN

$\varphi(q_3) = q_{16}$

φ: Esempio

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
φ(i)	-1	0	15	16	17	3	0	0	0	1	0	1	0	0	0	0	1	2



$L(q_3)$ = ANA

Bordo($L(q_3)$) = NA

P1 = ANANAS

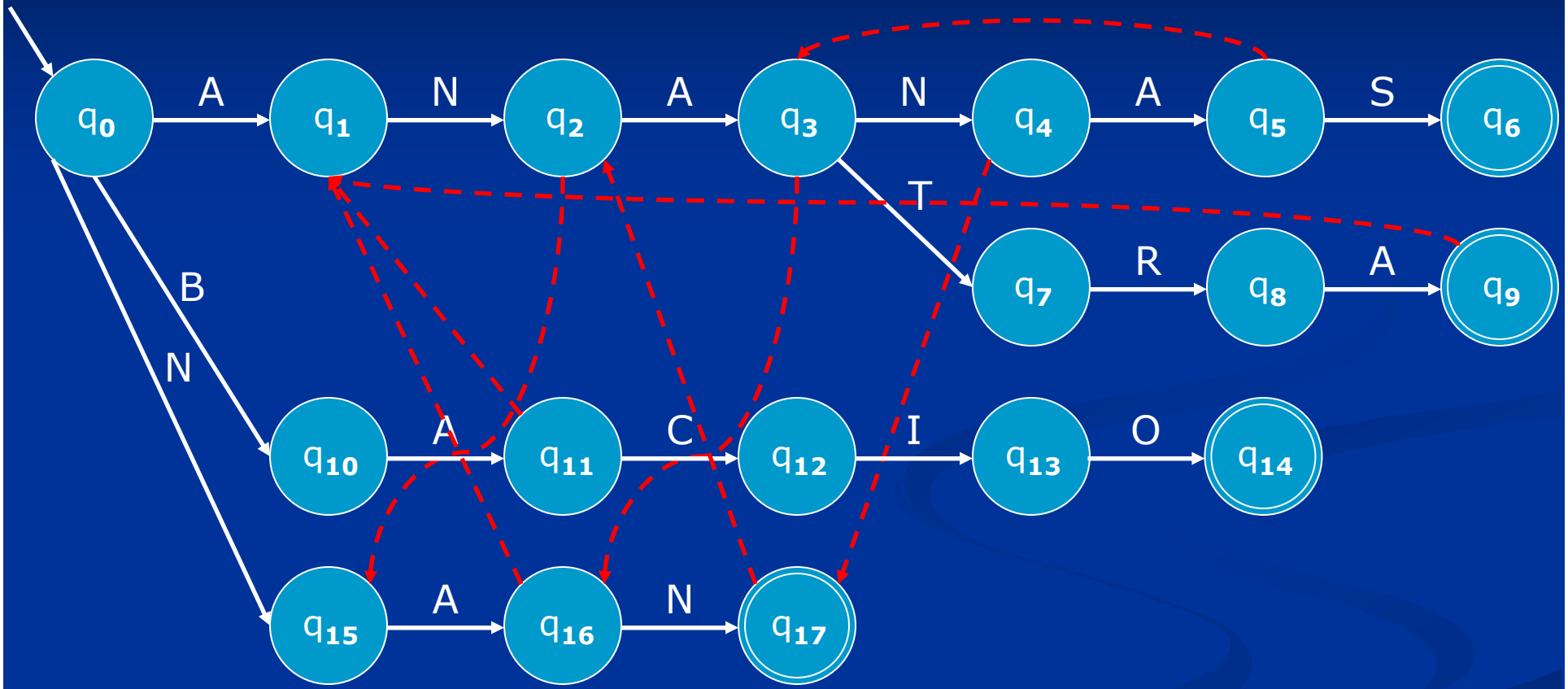
P2 = ANATRA

P3 = BACIO

P4 = NAN

$\varphi(q_3) = q_{16}$

Esempio

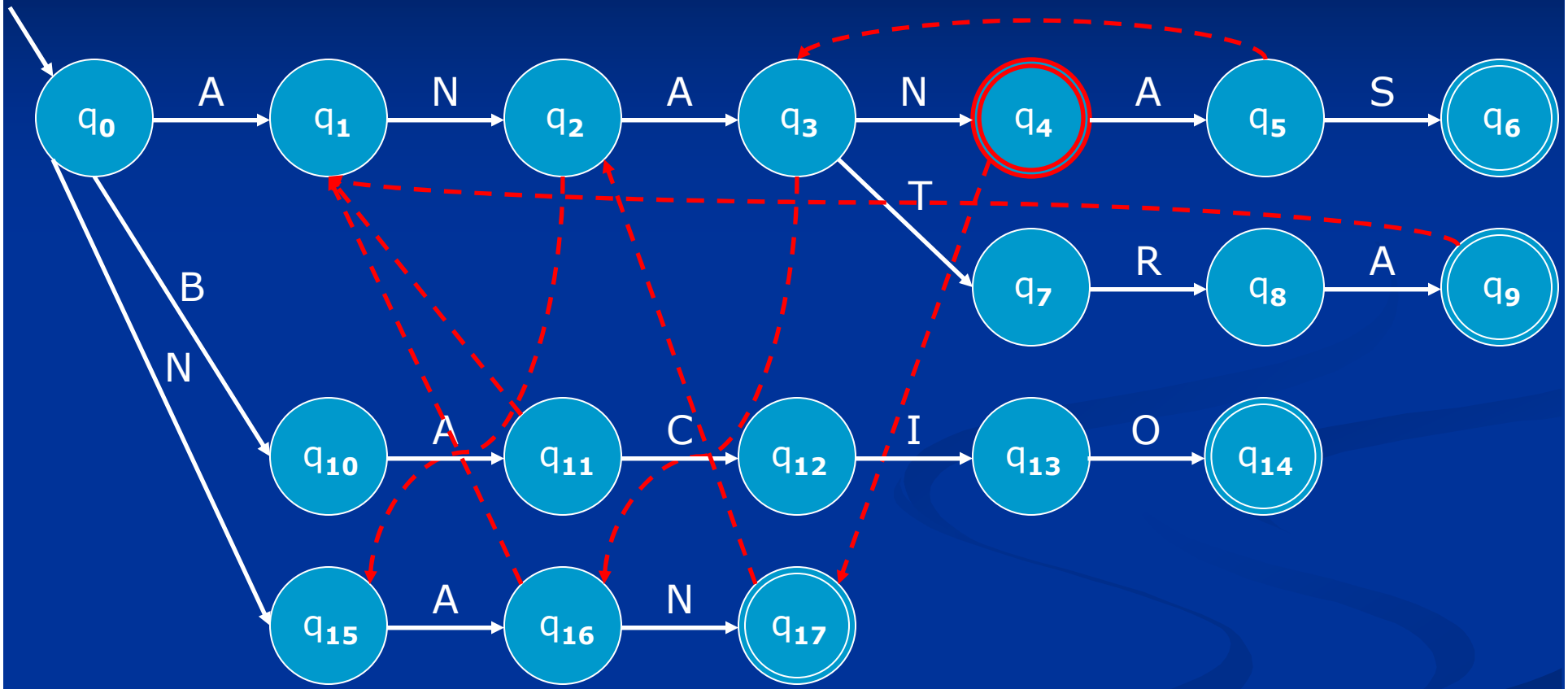


Nota: per questioni di leggibilità, abbiamo omesso gli archi uscenti dagli stati q_i con Bordo($L(q_i)$) = vuoto che puntano allo stato q_0

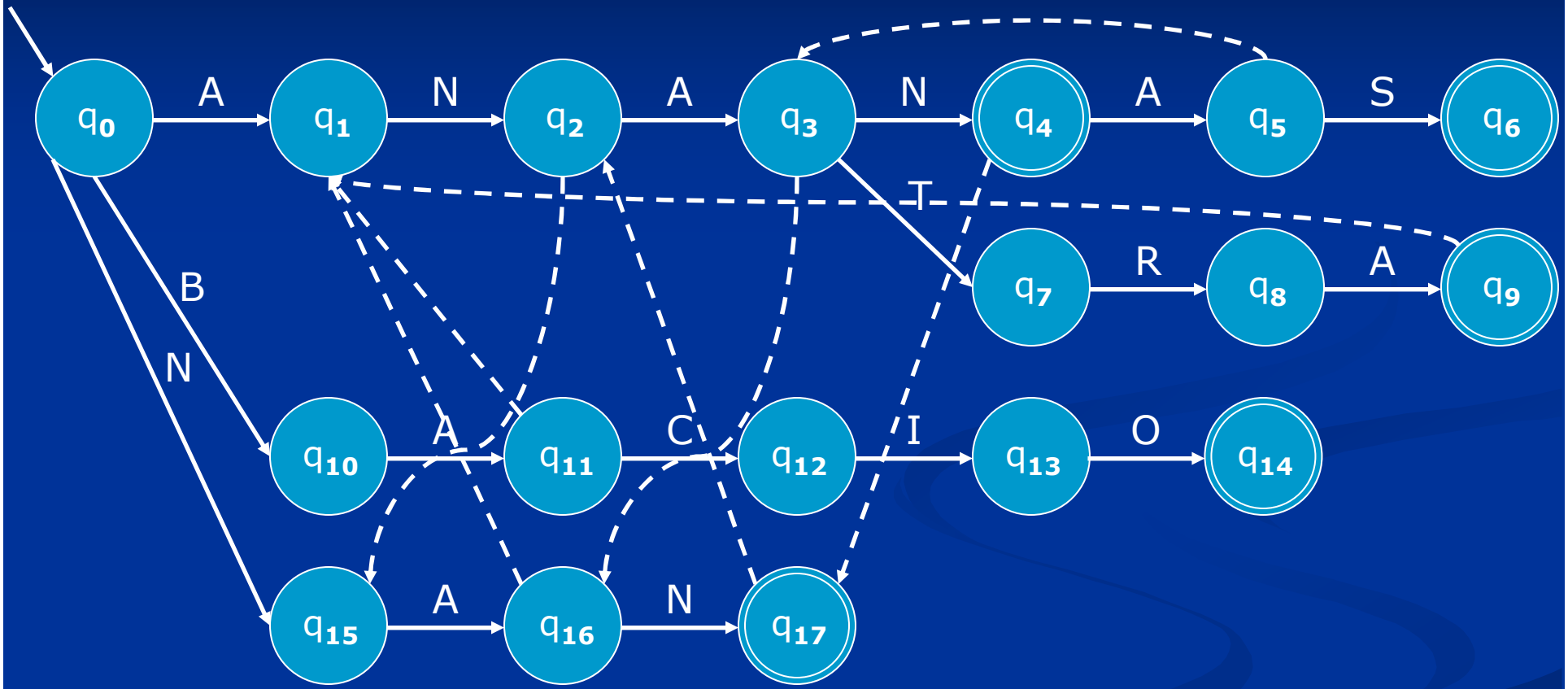
Costruzione Automa

- Se lo stato φ(i) è uno stato finale allora lo è anche lo stato i
  - Se un pattern viene riconosciuto nello stato φ(i), allora lo stesso pattern viene riconosciuto anche nello stato i

Esempio


q0
A
q1
N
q2
A
q3
N
q4
A
q5
S
q6
T
B
N
q7
R
q8
A
q9
q10
A
q11
C
q12
I
q13
O
q14
q15
A
q16
N
q17

Esempio Finale



Ricerca nell'Automa

- **Complessità: O(|T|)**
- **Si utilizzano le funzioni δ e φ per la ricerca dei pattern $P_1,\ldots,P_k$ all'interno del testo T**
- **Ci sono due possibili varianti**
  1. **Ricerca della prima occorrenza di un P_i**
  2. **Ricerca di tutte le occorrenze dei pattern $P_1,\ldots,P_k$**

Ricerca nell'Automa

- La ricerca parte dallo stato iniziale dell'automa
- Ad ogni passo
  1. nello stato corrente q_i, si legge il prossimo carattere c del testo
  2. si calcola $q_j = \delta(q_i,c)$
     - se q_j diverso da vuoto (match), si passa nello stato q_j e si torna al punto 1
     - se q_j = vuoto (mismatch), si passa nello stato $q_i = \varphi(q_i)$ e si torna al punto 2
  3. se lo stato corrente q_i è uno stato finale, si restituisce l'occorrenza del pattern trovata

Ricerca nell'Automa

- Se si verifica un mismatch nello stato iniziale q_0 ?
  - Nessuno dei pattern inizia con il carattere corrente c
  - Si forza la lettura del carattere successivo del testo T

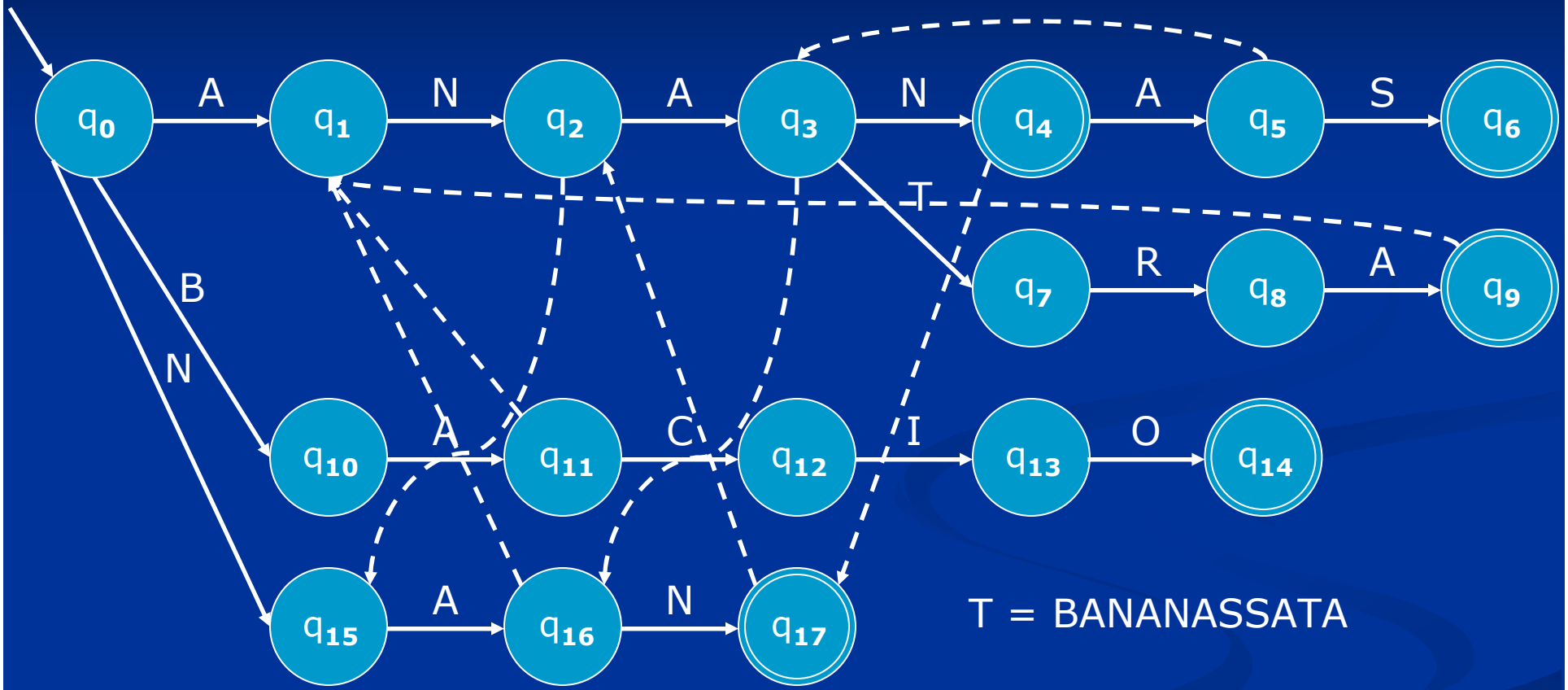
Qual è il significato della funzione φ ?

- **Se durante il riconoscimento di un pattern P_i si verifica un mismatch con il carattere corrente c del testo nello stato i, è possibile che alcuni caratteri riconosciuti fino a quel momento corrispondano al prefisso di un altro pattern P_j**

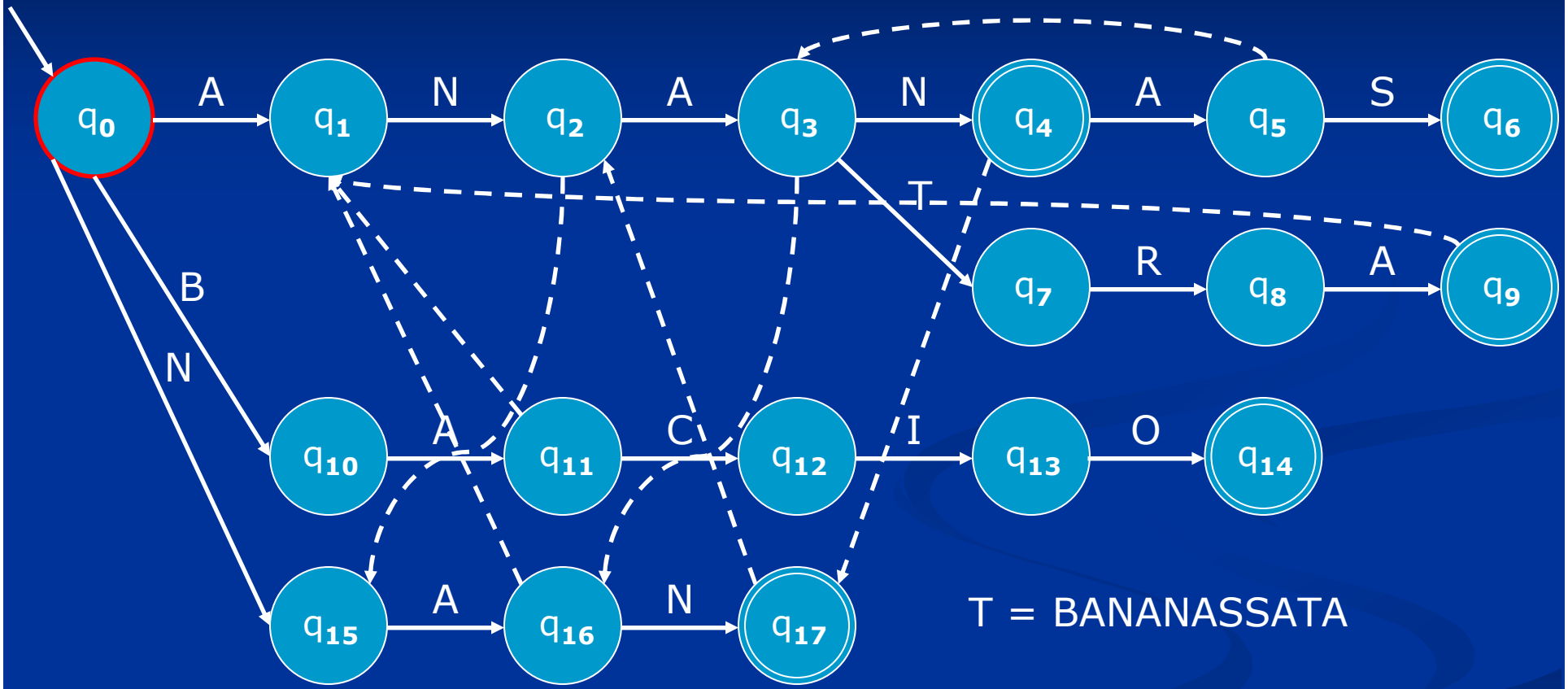
Qual è il significato della funzione φ ?

- **Il valore di φ(i) indica proprio lo stato da cui è possibile continuare la ricerca di tale pattern P_j a partire dal carattere immediatamente successivo a quello del prefisso trovato, senza dover ricominciare il riconoscimento dall'inizio**

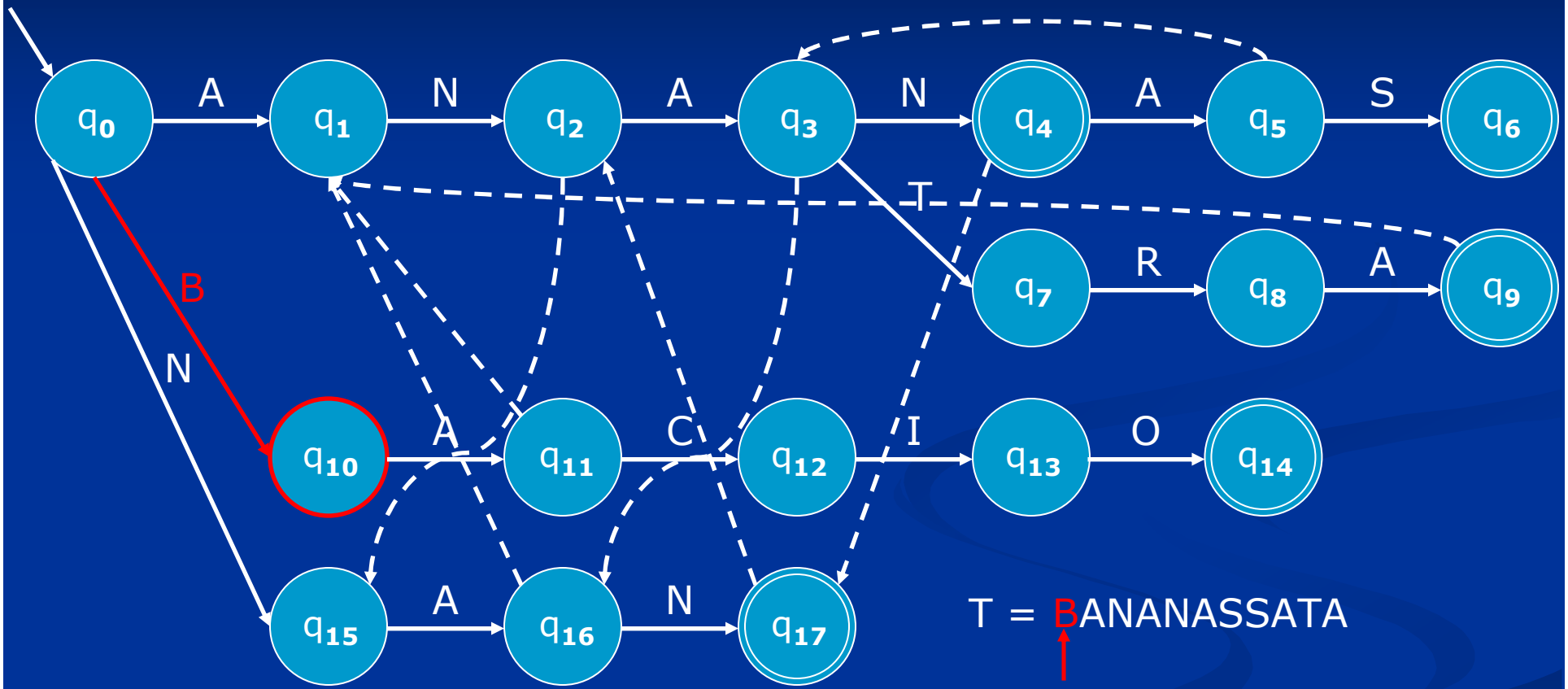
Ricerca: esempio



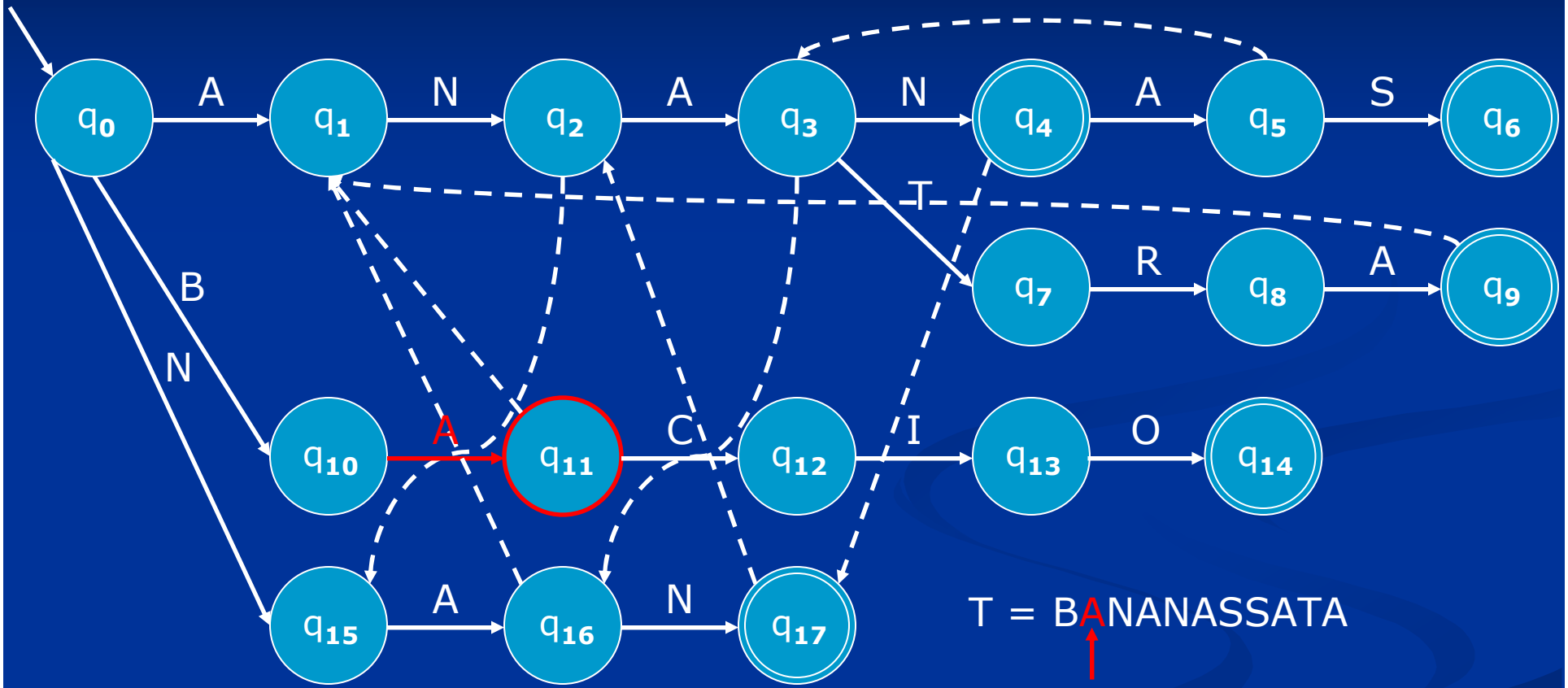
Ricerca: esempio



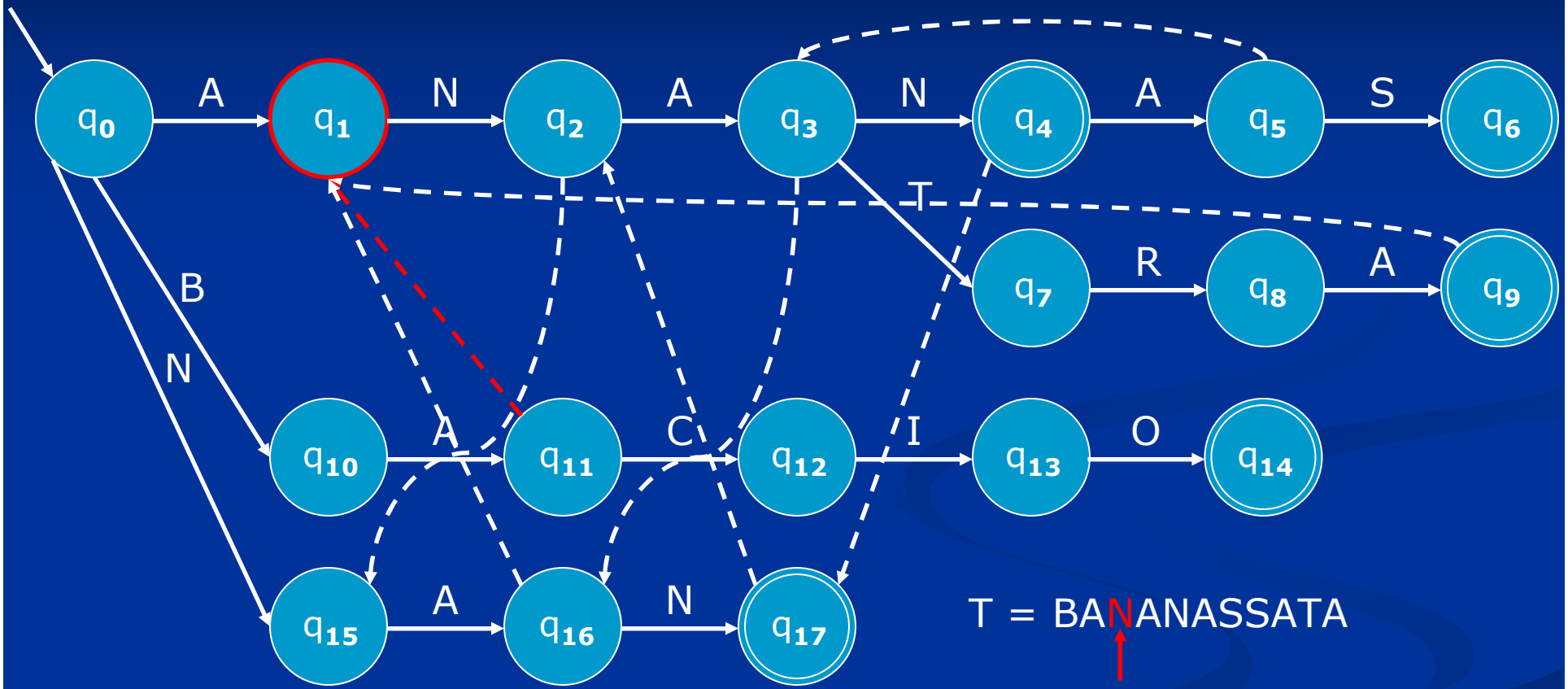
Ricerca: esempio



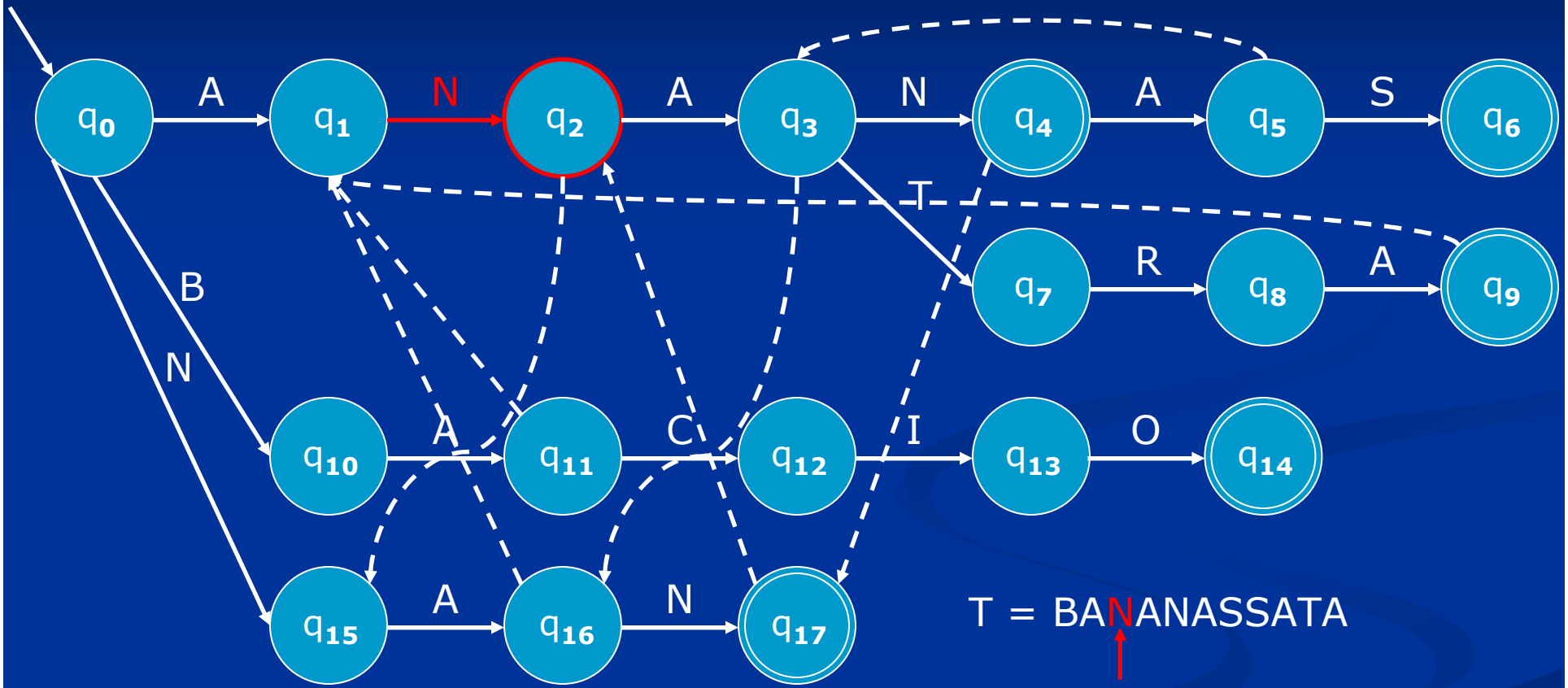
Ricerca: esempio



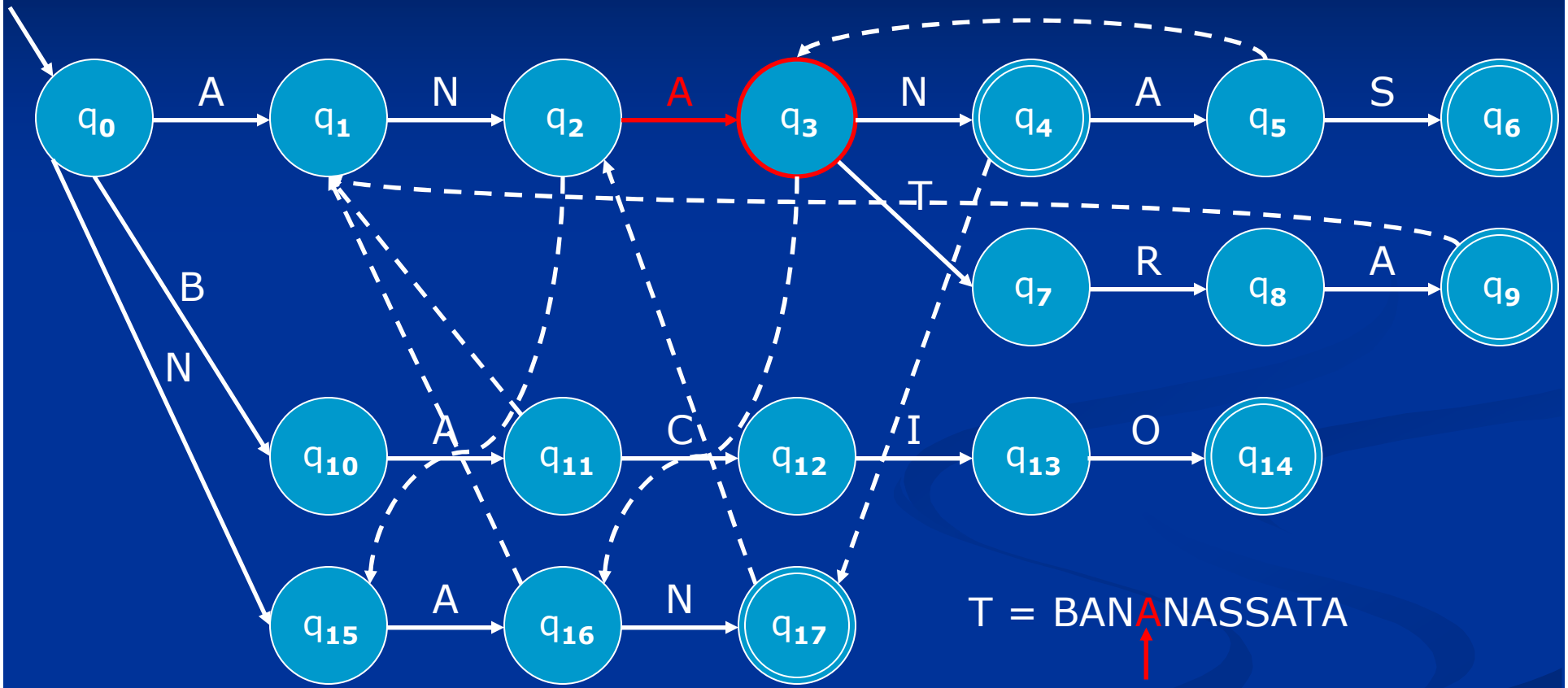
Ricerca: esempio



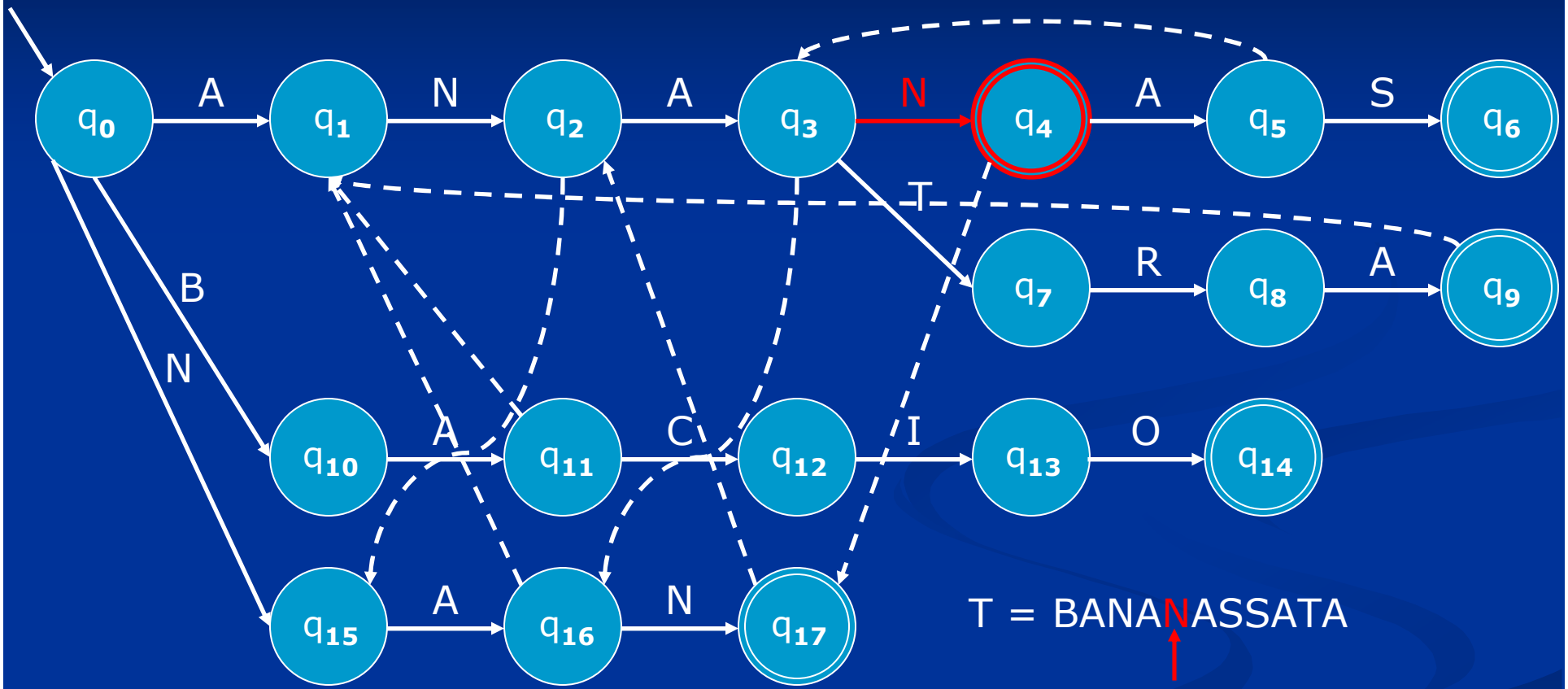
Ricerca: esempio



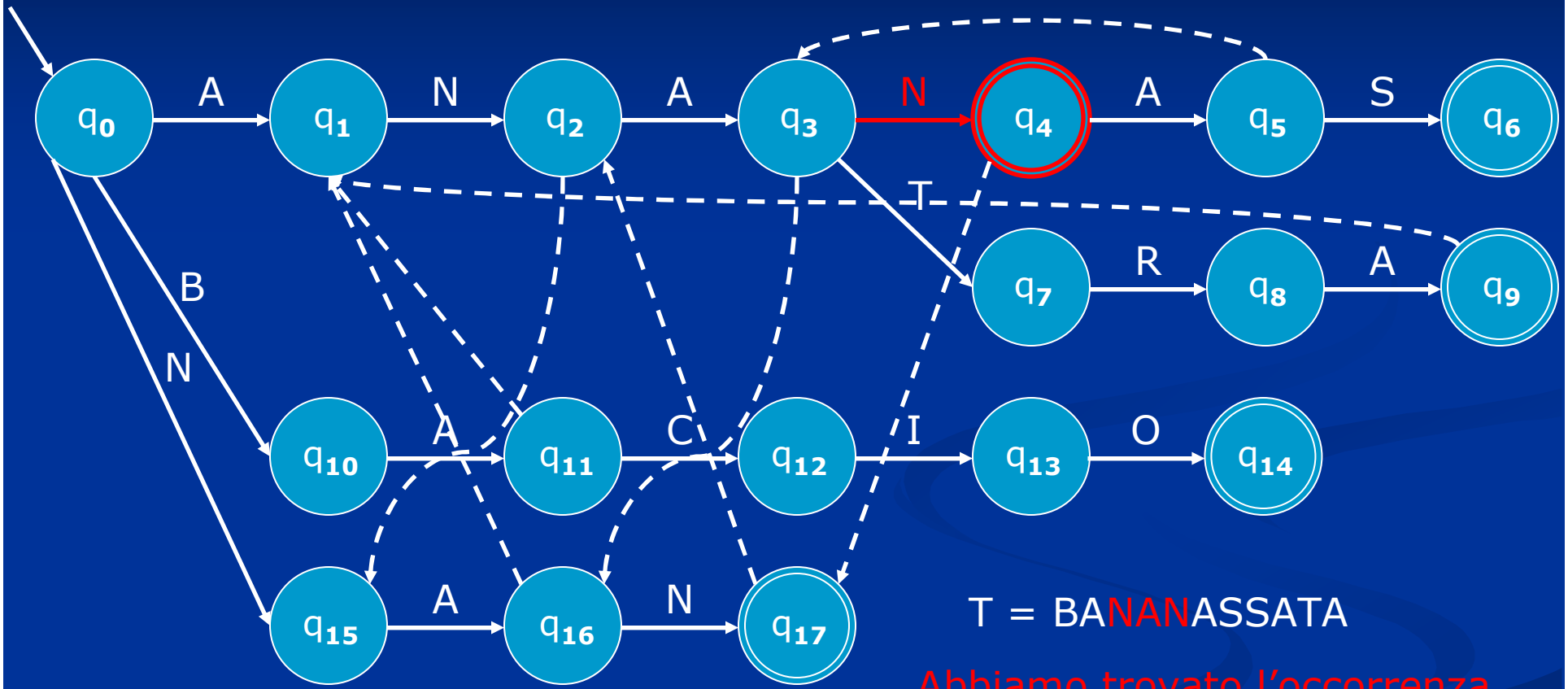
Ricerca: esempio



Ricerca: esempio



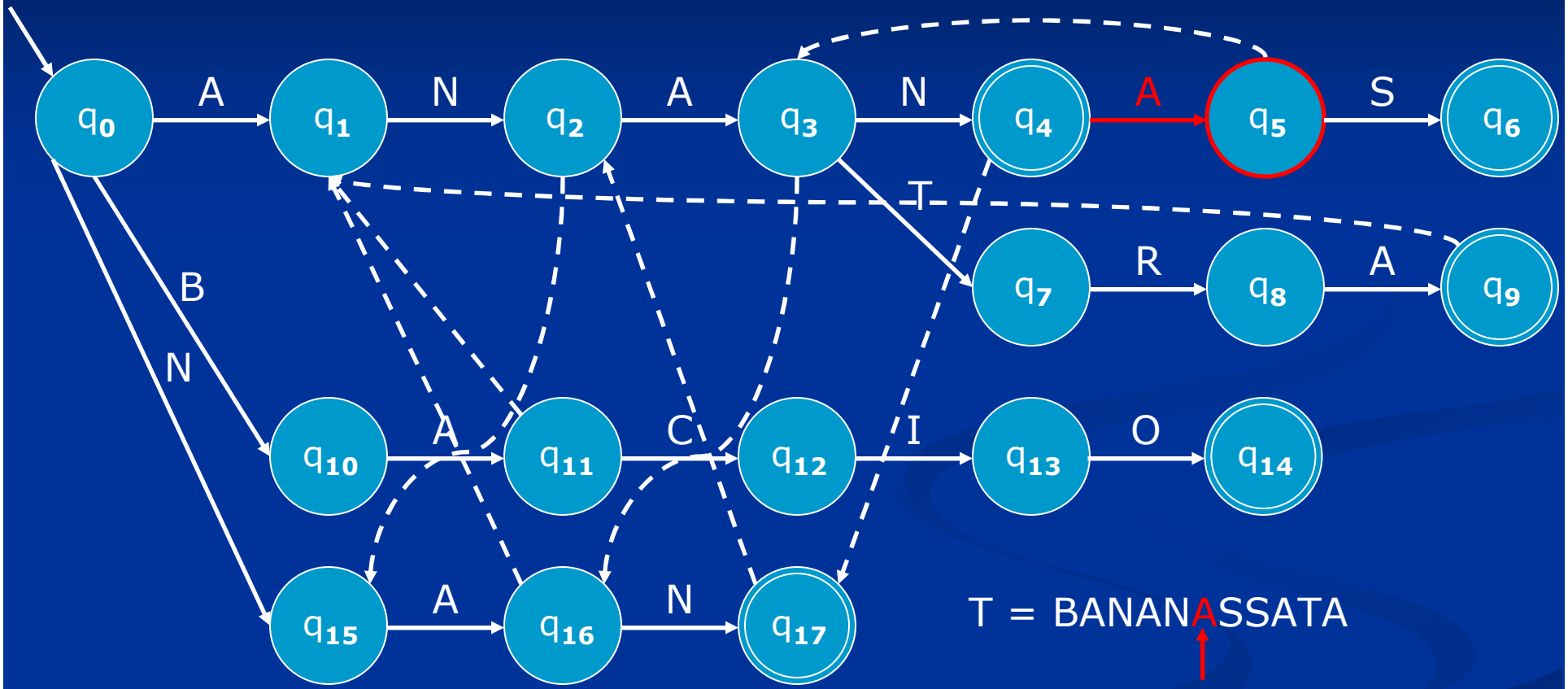
Ricerca: esempio



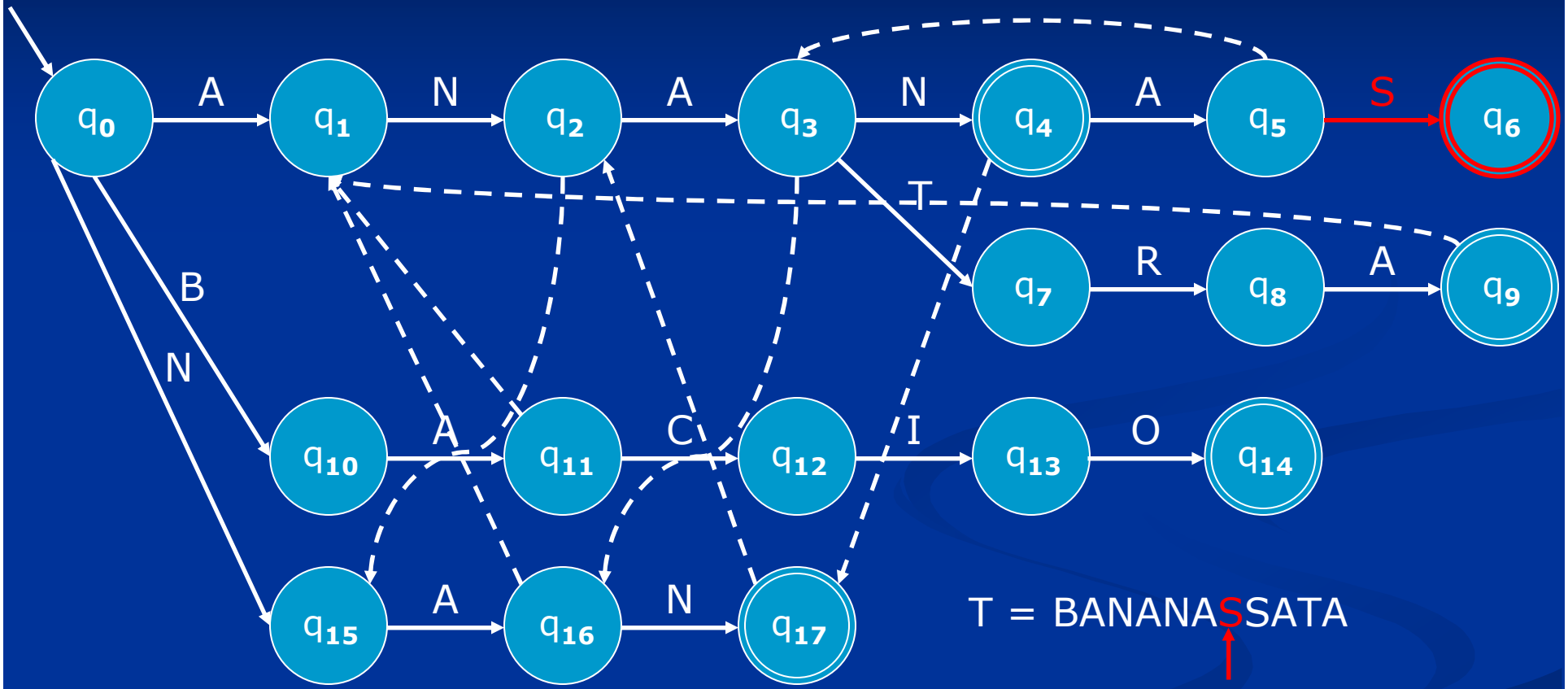
T = BANANASSATA

Abbiamo trovato l'occorrenza del pattern P_4 = NAN

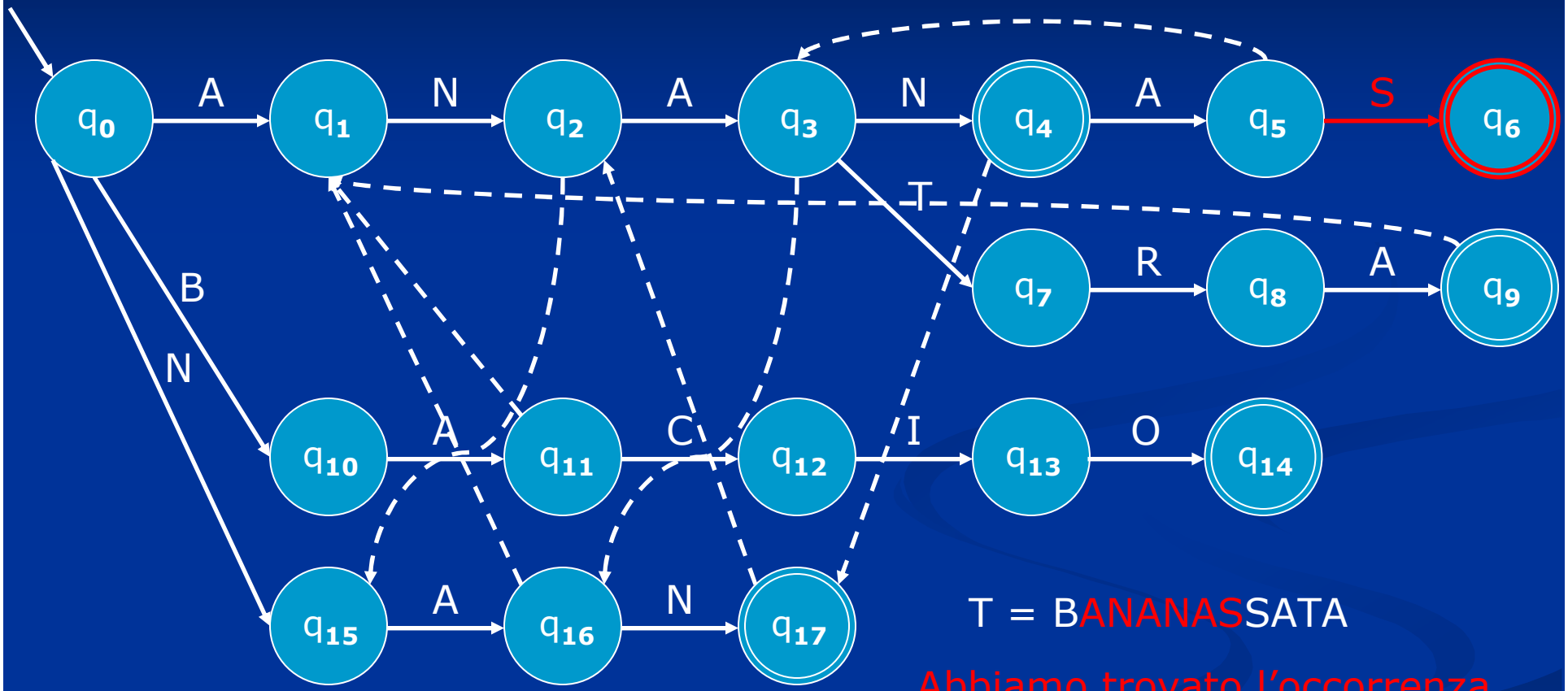
Ricerca: esempio



Ricerca: esempio



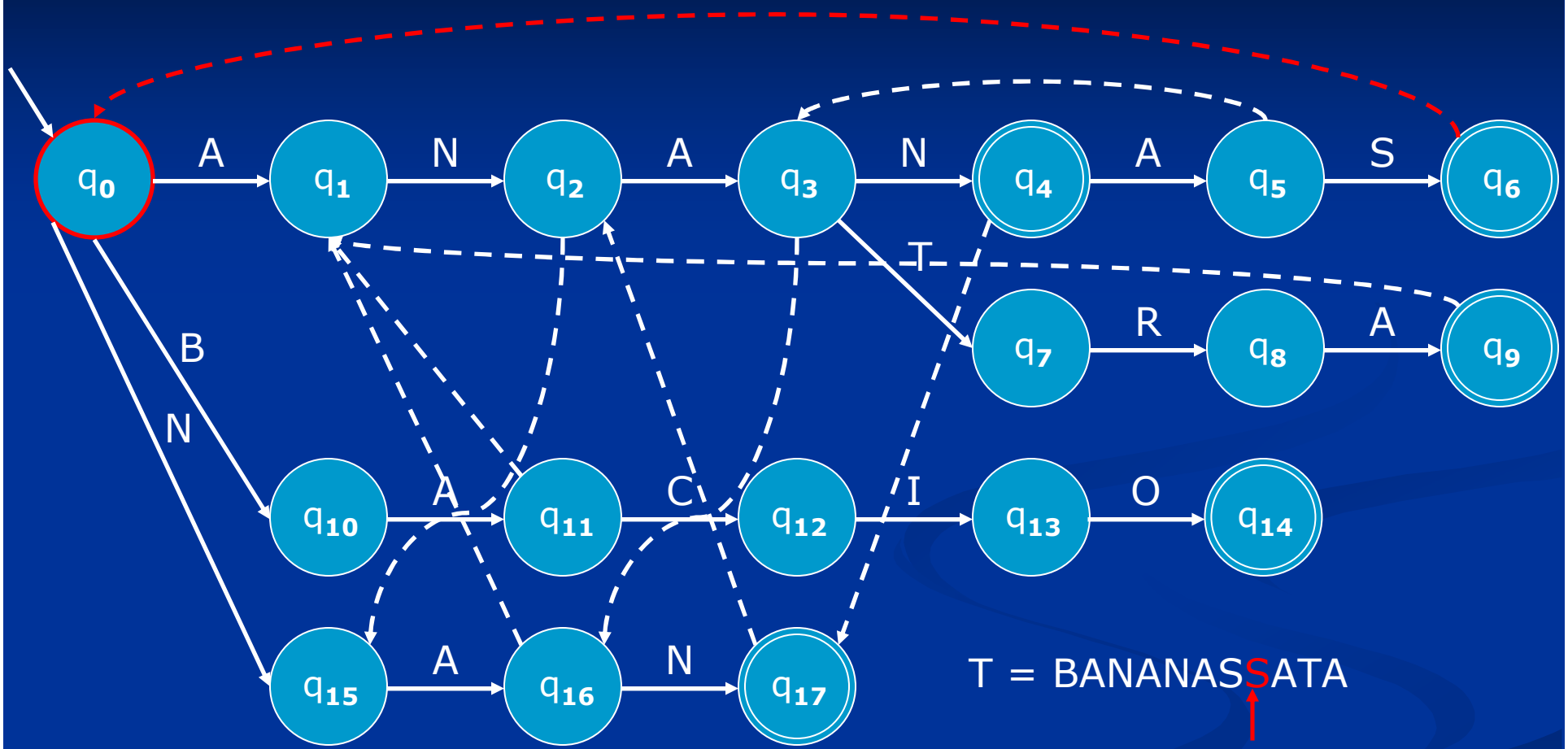
Ricerca: esempio



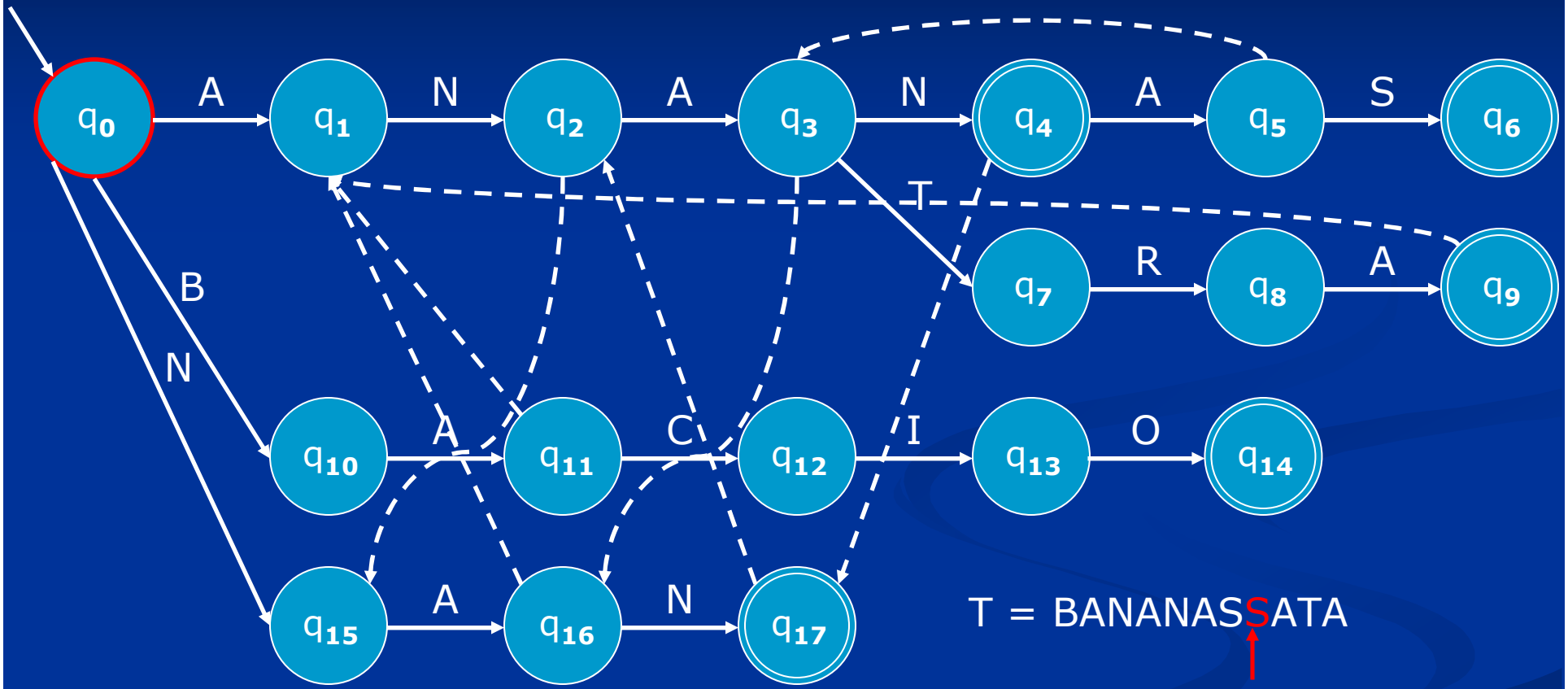
T = BANANASSATA

Abbiamo trovato l'occorrenza del pattern P_1 = ANANAS

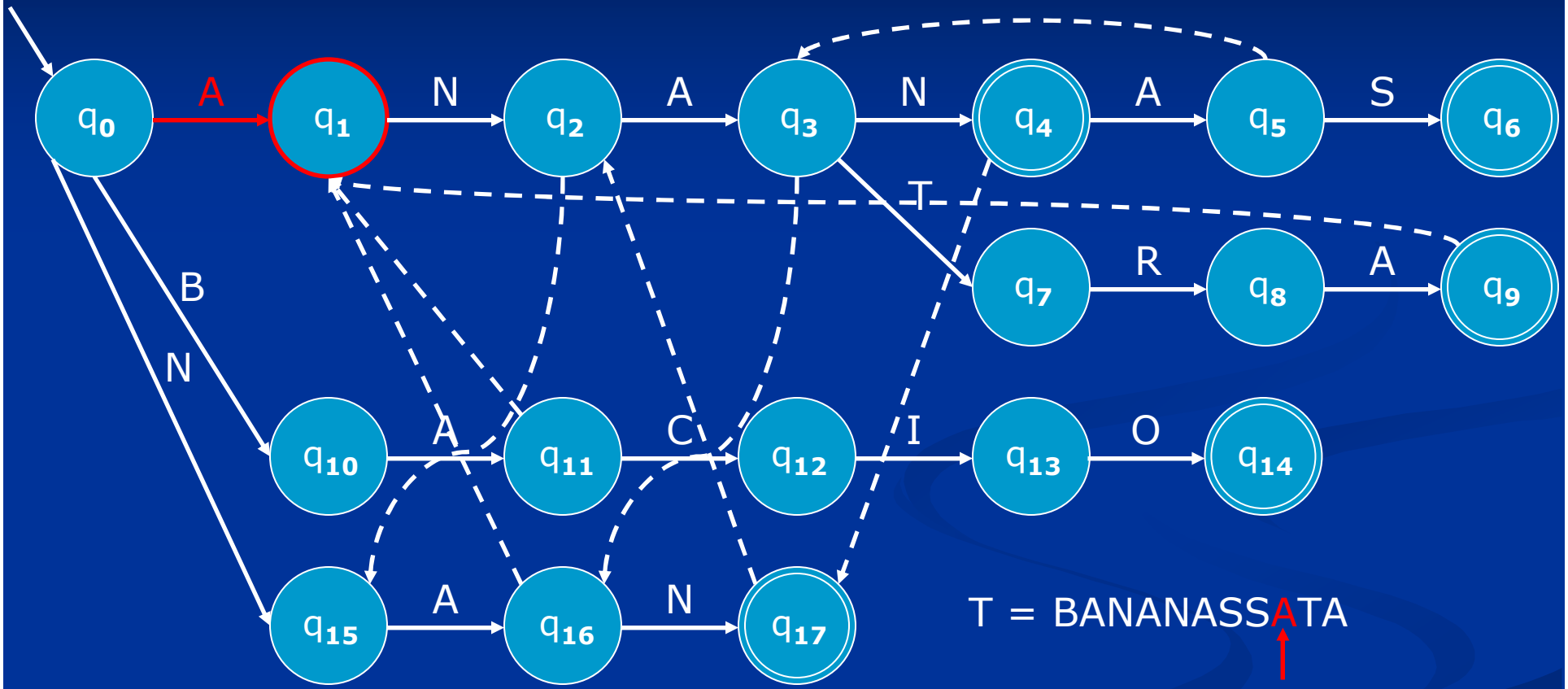
Ricerca: esempio



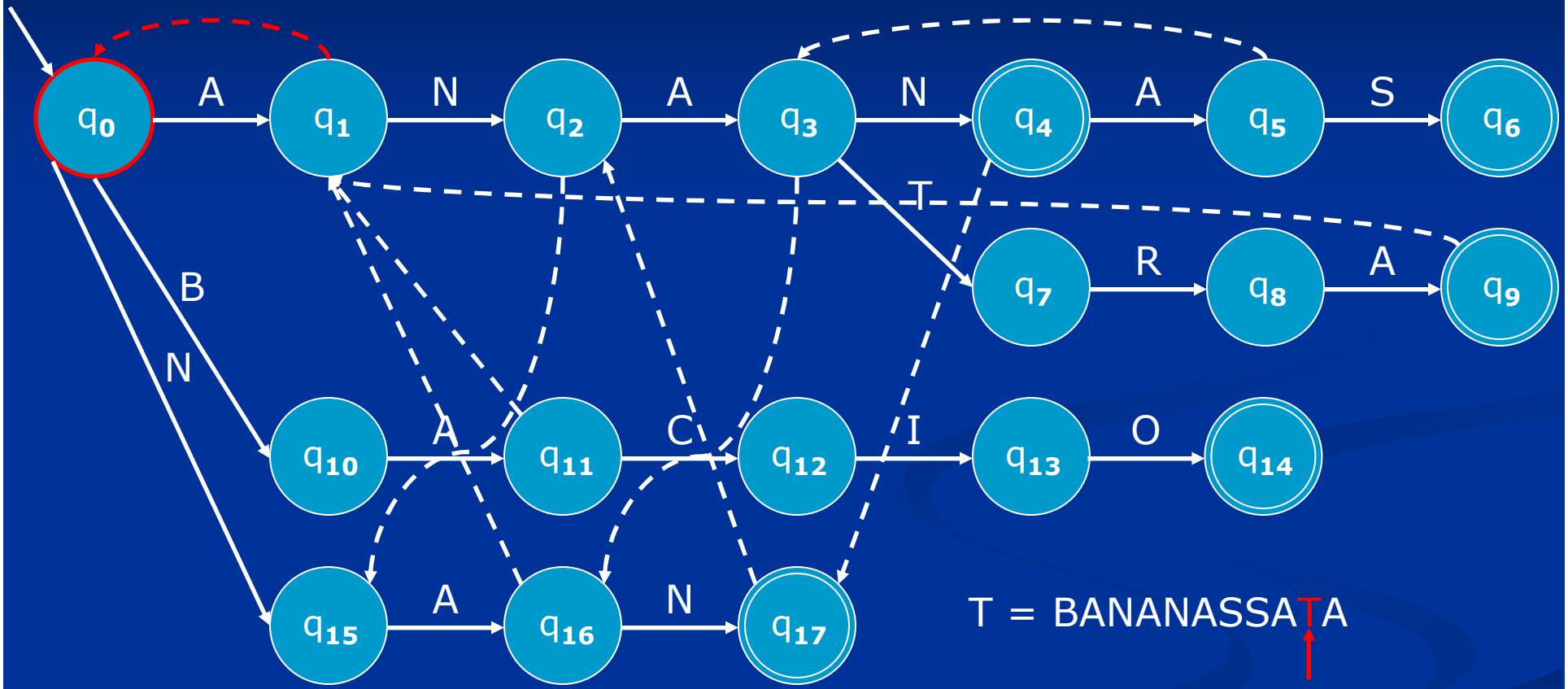
Ricerca: esempio



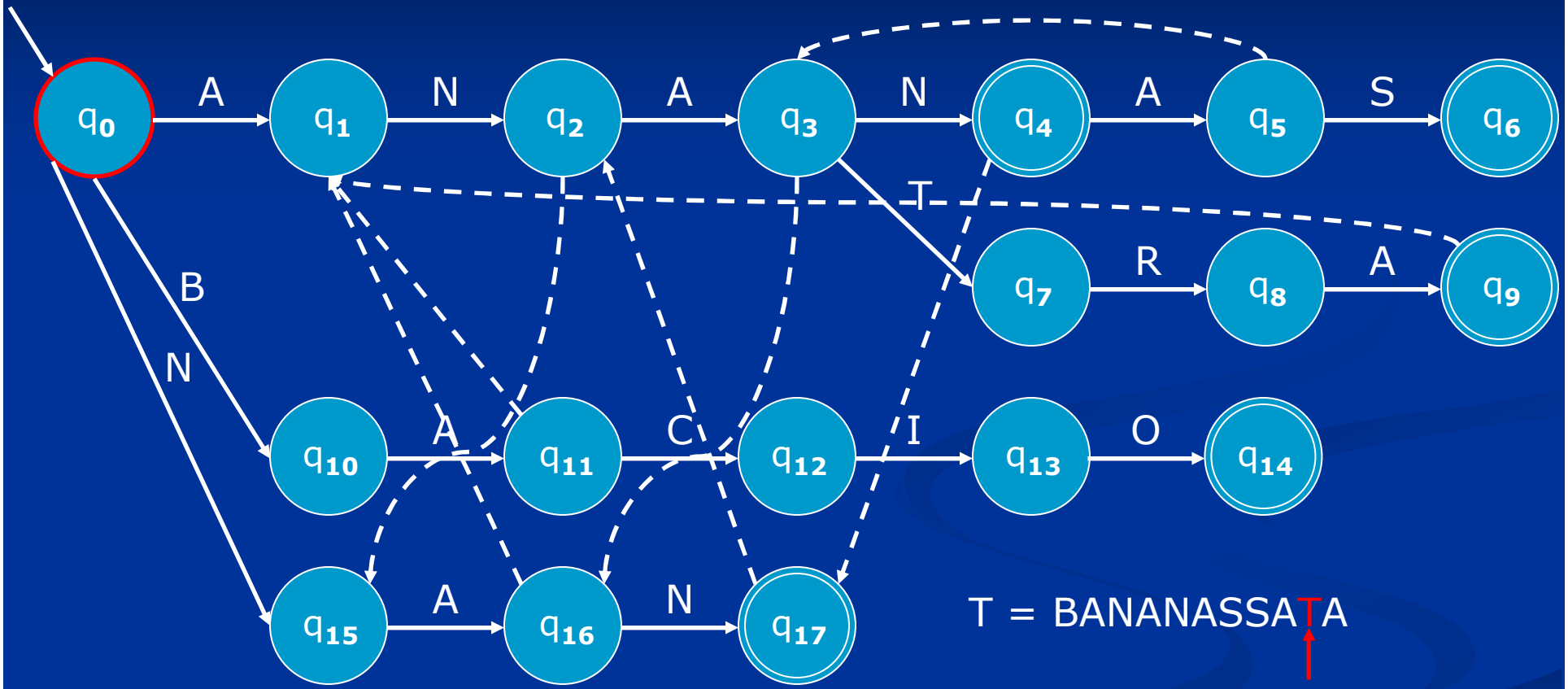
Ricerca: esempio



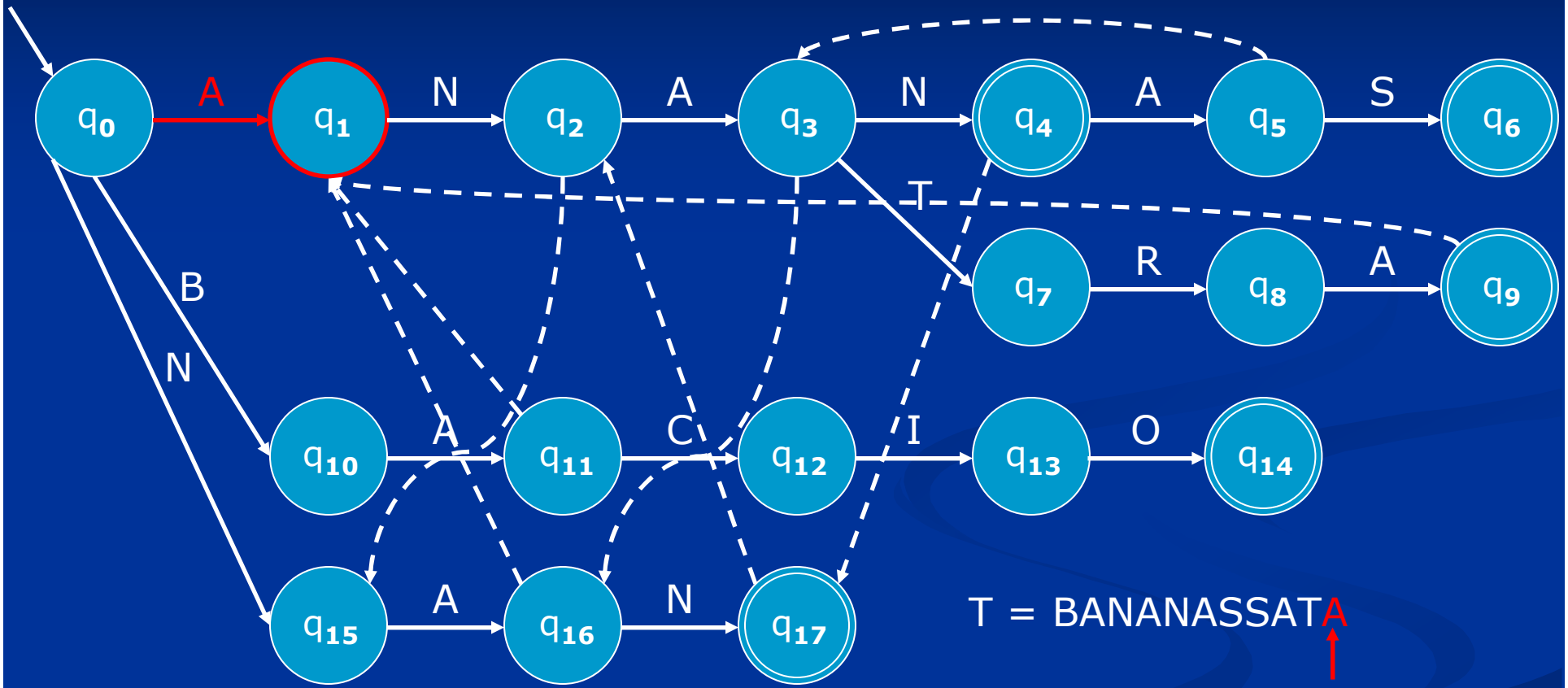
Ricerca: esempio



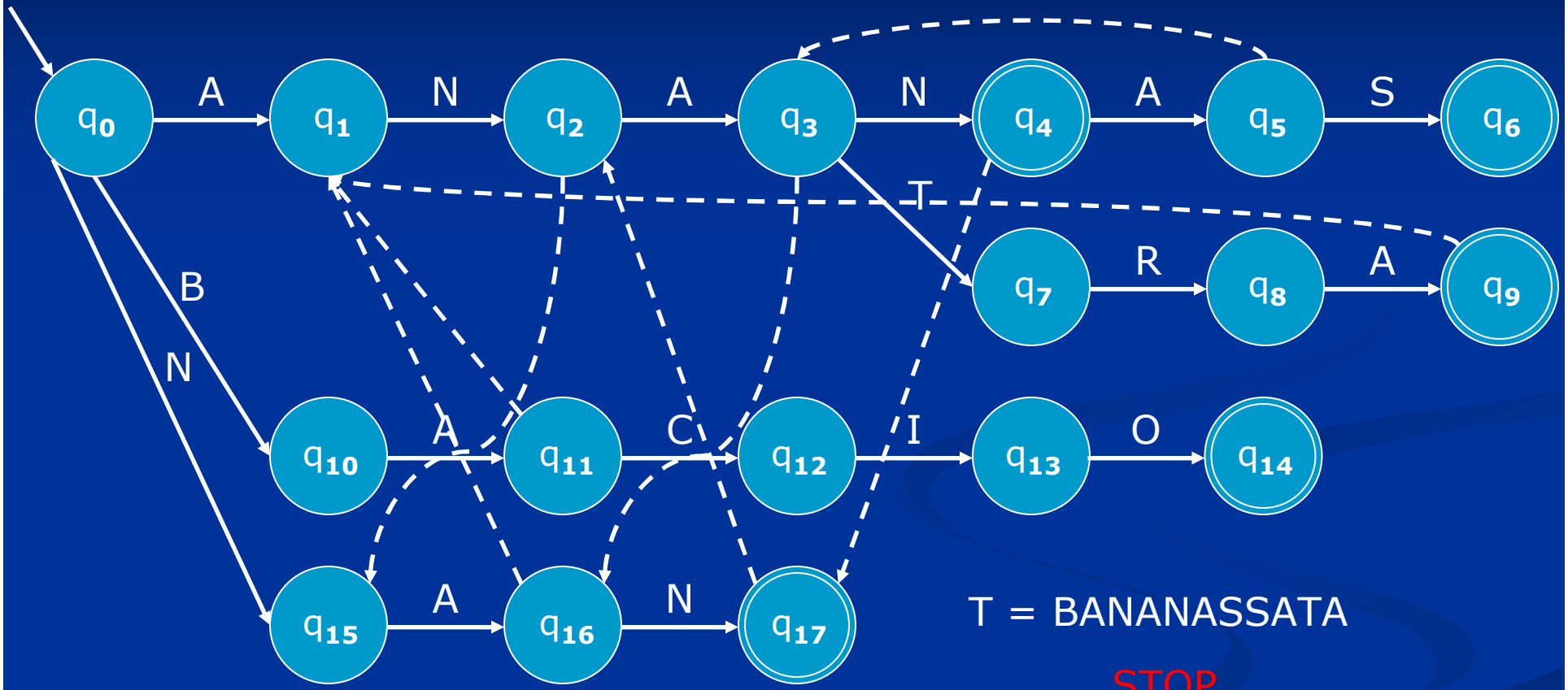
Ricerca: esempio



Ricerca: esempio



Ricerca: esempio



Estensione al caso bidimensionale

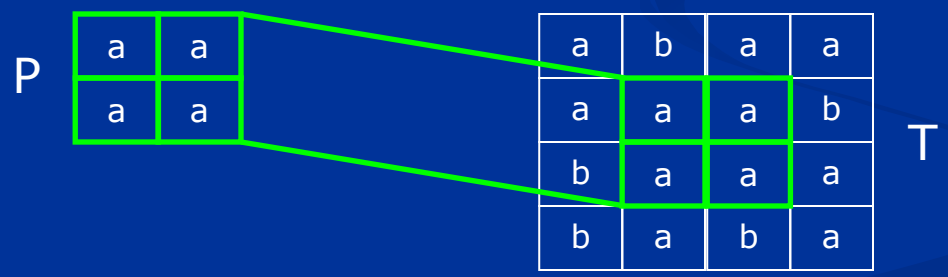
- Introduzione
- Nozioni Preliminari
  - Automi
- Pattern Matching Multiplo
  - Aho - Corasick
    - Bidimensionale
- Espressioni Regolari
- Conclusioni

Estensione al caso bidimensionale

- Gli automi di Aho-Corasick hanno una interessante applicazione al caso del pattern matching bidimensionale
  - Naturale generalizzazione del pattern matching singolo

Il Problema

- **Input:**
  - **Pattern P, matrice quadrata nxn**
  - **Testo T, matrice quadrata mxm**
- **Output: Determinare se P occorre come sottomatrice di T**

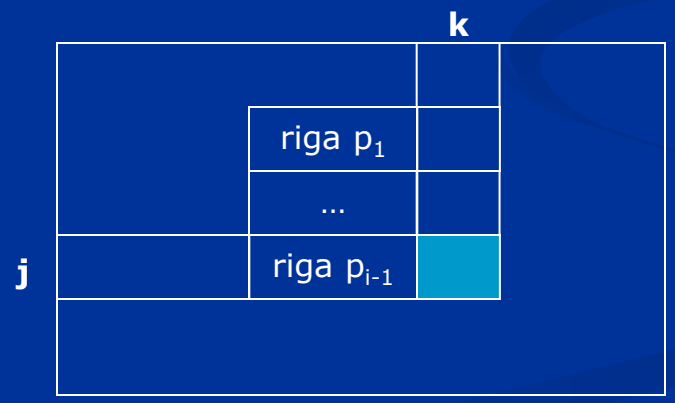


L'Idea

- Si utilizza Aho-Corasick per la ricerca simultanea di più pattern nel testo
  - Si considera ogni riga j di P come un singolo pattern p_j
  - Si costruisce l'automa di Aho-Corasick che riconosce i pattern $p_1,\ldots,p_n$

Algoritmo

- L'algoritmo mantiene un vettore a[], lungo m, tale che
  - a[k] = i se al passo corrente è stata trovata nel testo la sottomatrice di P formata dalle righe $p_1,\ldots,p_{i-1}$



Algoritmo

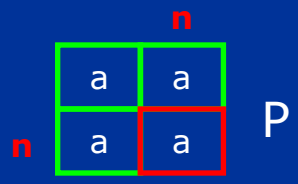
- **Inizializzazione:**
  - **a[k] = 1, 1 ≤ k ≤ m**
  - **posizione corrente = T[1,1]**
- **Ad ogni passo:**
  - **Si esamina la posizione corrente T[j,k]**
    - **Se nessuna riga del pattern termina in T[j,k], si passa ad esaminare la prossima posizione**
      - **T[j,k+1] se k < m**
      - **T[j+1,1] altrimenti (k = m)**
    - **altrimenti, se si trova un'occorrenza della riga i del pattern che termina in T[j,k]**

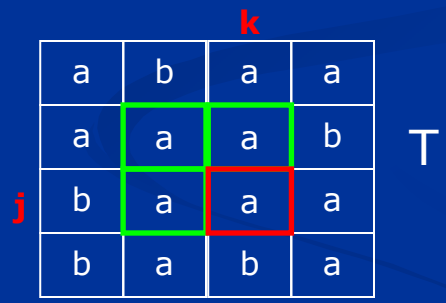
Algoritmo

- Se a[k] = i, la riga del pattern trovata è quella attesa si pone a[k] = a[k] +1
- altrimenti, se a[k] ≠ i, la riga del pattern trovata non è quella attesa, perciò eventuali sottomatrici di P trovate fino a questo momento non formano un'occorrenza valida
  - Se i =1 , l'occorrenza del pattern trovata è di nuovo la prima e si può ricominciare la ricerca di P a partire dalla seconda riga: si pone a[k] = 2
  - altrimenti (i ≠ 1), si deve ricominciare da capo la ricerca perciò a[k] = 1

Algoritmo

- Se a[k] = n + 1
  - P occorre come sottomatrice di T allineata con il suo angolo in basso a destra P[n,n] in T[j,k], perciò si restituisce la posizione (j,k)





Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

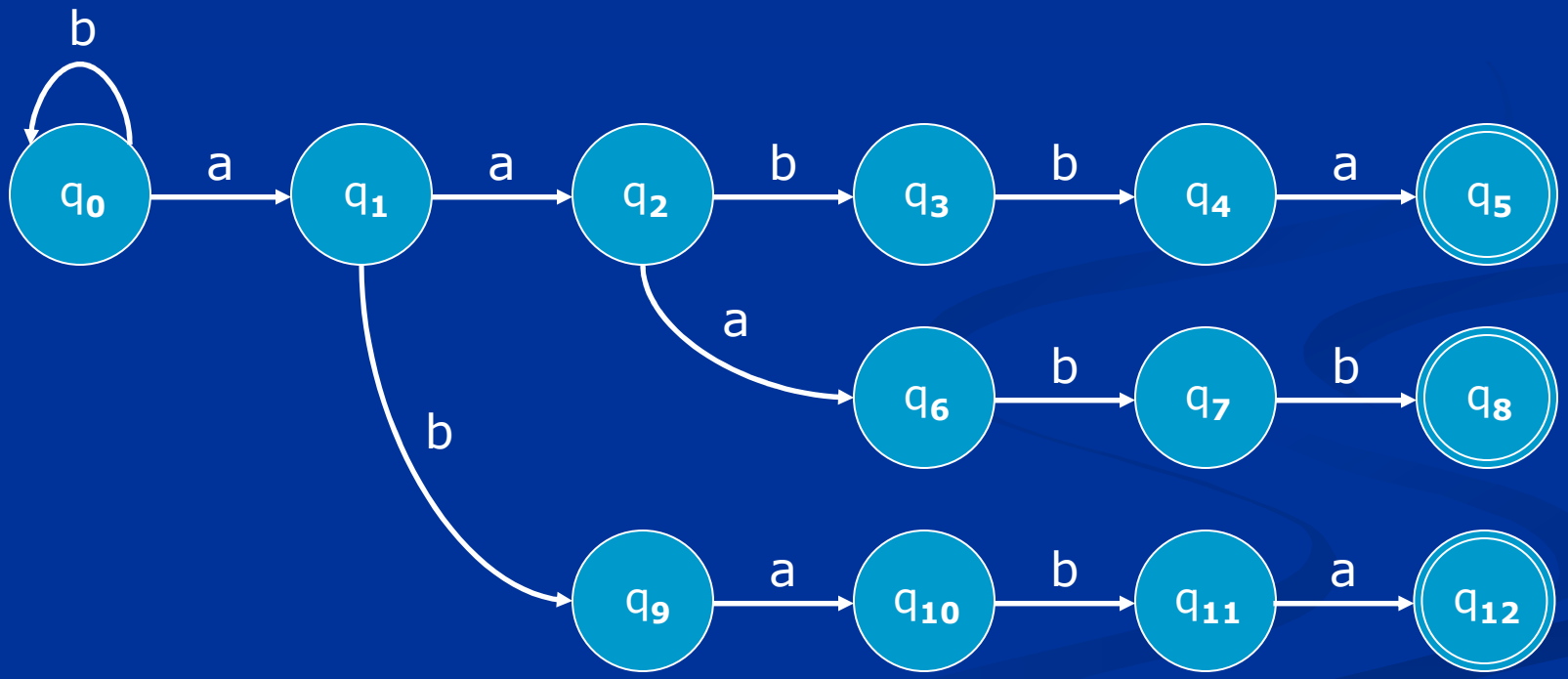
T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	1	1	1

Esempio 1

- **Automa di Aho-Corasick per P**



Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

no match

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	1	1	1

Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	1	1	1

Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

match

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	2	1	1

Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	2	1	1

Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

match

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	3	1	1

Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	3	1	1

Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

match

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	4	1	1

Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	4	1	1

Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

match

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	5	1	1

Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	5	1	1

Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

match

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	6	1	1

Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	6	1	1

Una occorrenza del pattern P trovata nel testo T

Esempio 1

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

						k		
	a	b	b	a	a	a	b	a
	b	a	a	b	b	a	a	a
	b	a	a	a	b	b	a	a
	b	a	b	a	b	a	a	a
	b	a	a	b	b	a	a	a
j	b	a	a	a	b	b	a	a
	a	b	b	b	b	b	b	a
	a	b	b	b	b	b	b	b

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	6	1	1

Restituisco la posizione (j,k)

Esempio 2

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

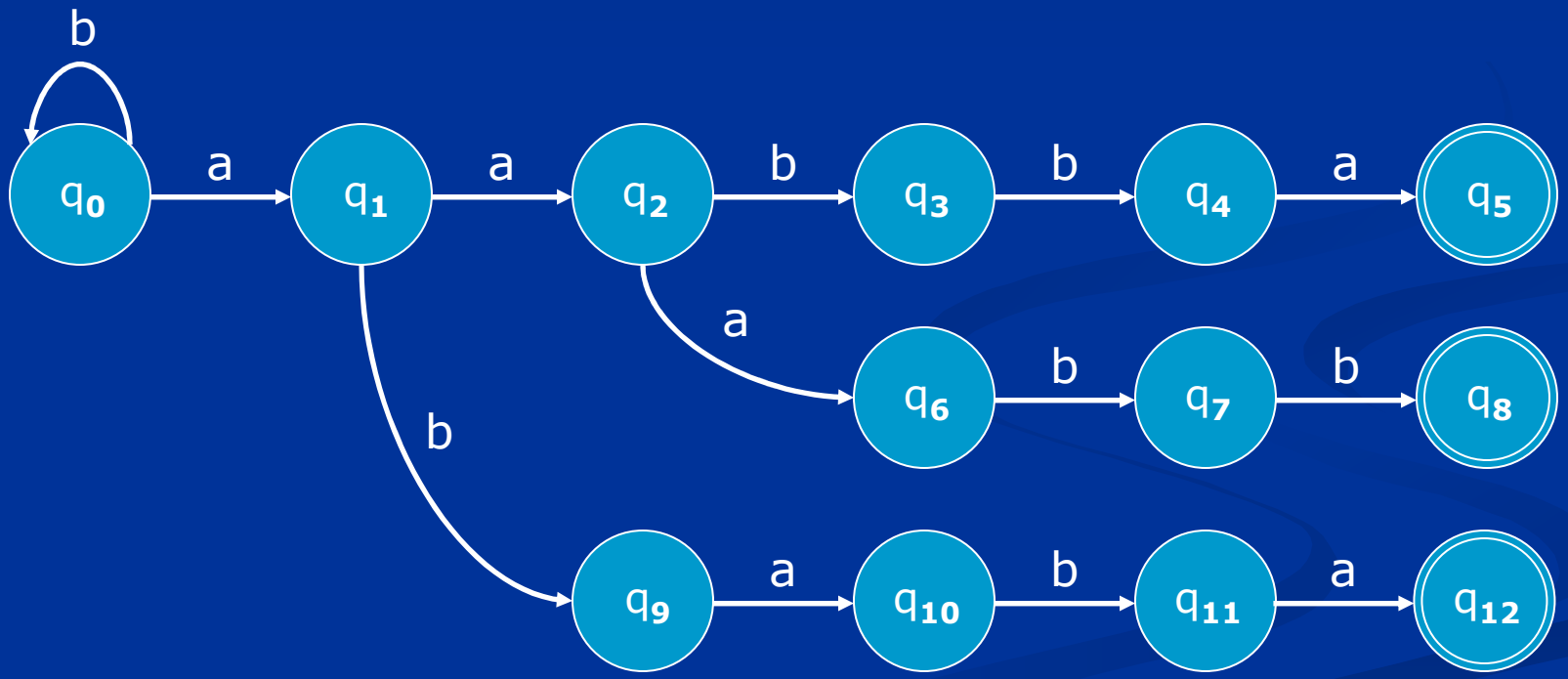
T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	a	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	1	1	1

Esempio 2

- **Automa di Aho-Corasick per P**



Esempio 2

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	a	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	5	1	1

Esempio 2

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
→ b	a	a	a	b	a	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

no match

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	1	1	1

Esempio 2

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	a	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

→ no match

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	1	1	1

Esempio 2

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	a	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

no match

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	1	1	1

Esempio 2

P

p1	a	a	b	b	a
p2	a	a	a	b	b
p3	a	b	a	b	a
p4	a	a	b	b	a
p5	a	a	a	b	b

p1 = aabba

p2 = aaabb

p3 = ababa

p4 = aabba

p5 = aaabb

T

a	b	b	a	a	a	b	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
b	a	b	a	b	a	a	a
b	a	a	b	b	a	a	a
b	a	a	a	b	b	a	a
a	b	b	b	b	b	b	a
a	b	b	b	b	b	b	b

	1	2	3	4	5	6	7	8
a[k]	1	1	1	1	1	1	1	1

Nessuna occorrenza del pattern P trovata nel testo T

Espressioni Regolari

- Introduzione
- Nozioni Preliminari
  - Automi
- Pattern Matching Multiplo
  - Aho - Corasick
    - Bidimensionale
- **Espressioni Regolari**
- Conclusioni

Espressioni Regolari

- **Insieme di semplici regole per rappresentare insiemi di stringhe**
- **Possono essere generate e riconosciute da automi a stati finiti**

Come si possono utilizzare

- Metodo conciso per definire
  - insiemi di stringhe anche infiniti
  - ricerche più o meno esatte di un pattern
  - ricerche di stringhe con alcune posizioni non specificate

Semplici Regole

- Espressioni Regolari, E_Σ , definite sull'alfabeto Σ
  1. $\varepsilon \in E_\Sigma$, la stringa vuota
  2. $a \in E_\Sigma$ per ogni $a \in \Sigma$, singolo carattere in Σ
  3. $e_1 e_2 \in E_\Sigma$ per ogni e_1, $e_2 \in E_\Sigma$, concatenazione
  4. $e_1 | e_2 \in E_\Sigma$ per ogni e_1, $e_2 \in E_\Sigma$, alternativa
  5. $e^* \in E_\Sigma$ per ogni $e \in E_\Sigma$, sequenza (stella di Kleene)

Costruzione Automa

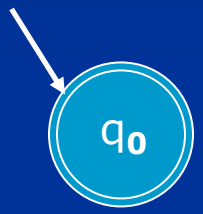
- **Metodo di Thompson: tecnica per costruire un automa a stati finiti partendo da un'espressione regolare**
  - **Complessità: O(|r|), con |r| la lunghezza dell'espressione regolare**
  - **Come funziona la tecnica ?**

Costruzione Automa

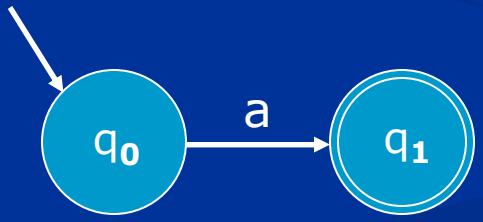
- Data un'espressione regolare
  1. Si decompone nelle sue sottoespressioni, le quali sono a loro volta decomposte e così via, fino ad arrivare ai casi base
     - Stringa Vuota
     - Singolo Carattere
  2. Si costruisce un automa per ciascun caso base

Stringa Vuota e Singolo Carattere

1. $\varepsilon \in E_\Sigma$



2. $a \in E_\Sigma$, per ogni $a \in \Sigma$



Esempio: Casi Base

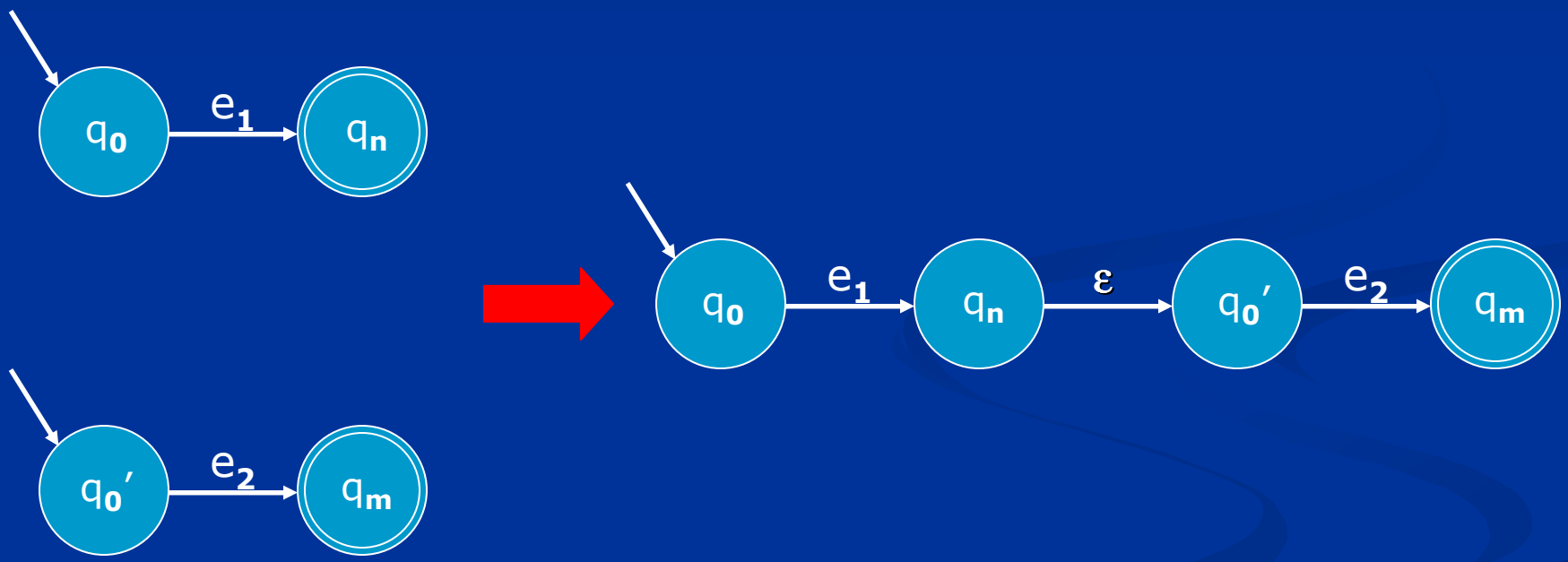
- r = (AT)*(C|G) con Σ = {A, T, C ,G}

Costruzione Automa

- **Data un'espressione regolare**
  3. **Si avvia una sequenza di passi in cui**
     - **ad ogni passo, si combina una coppia di automi ottenuti al passo precedente, applicando una delle regole di Thompson**
       - **Concatenazione**
       - **Alternativa**
       - **Sequenza**
     - **al passo finale, si ottiene l'automa completo che riconosce l'espressione regolare**

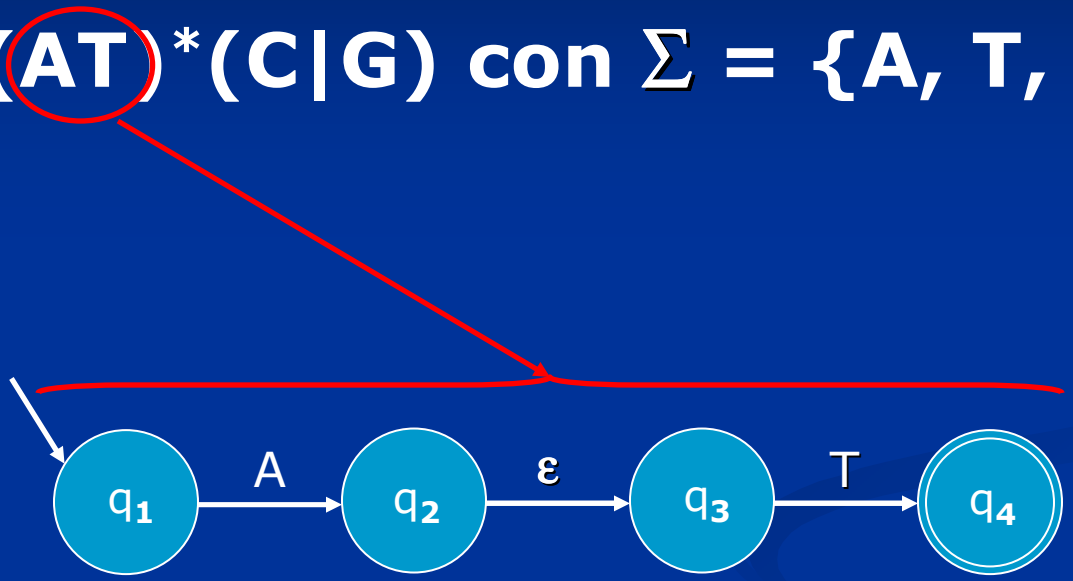
Concatenazione

3. $e_1 e_2 \in E_\Sigma$ per ogni $e_1, e_2 \in E_\Sigma$



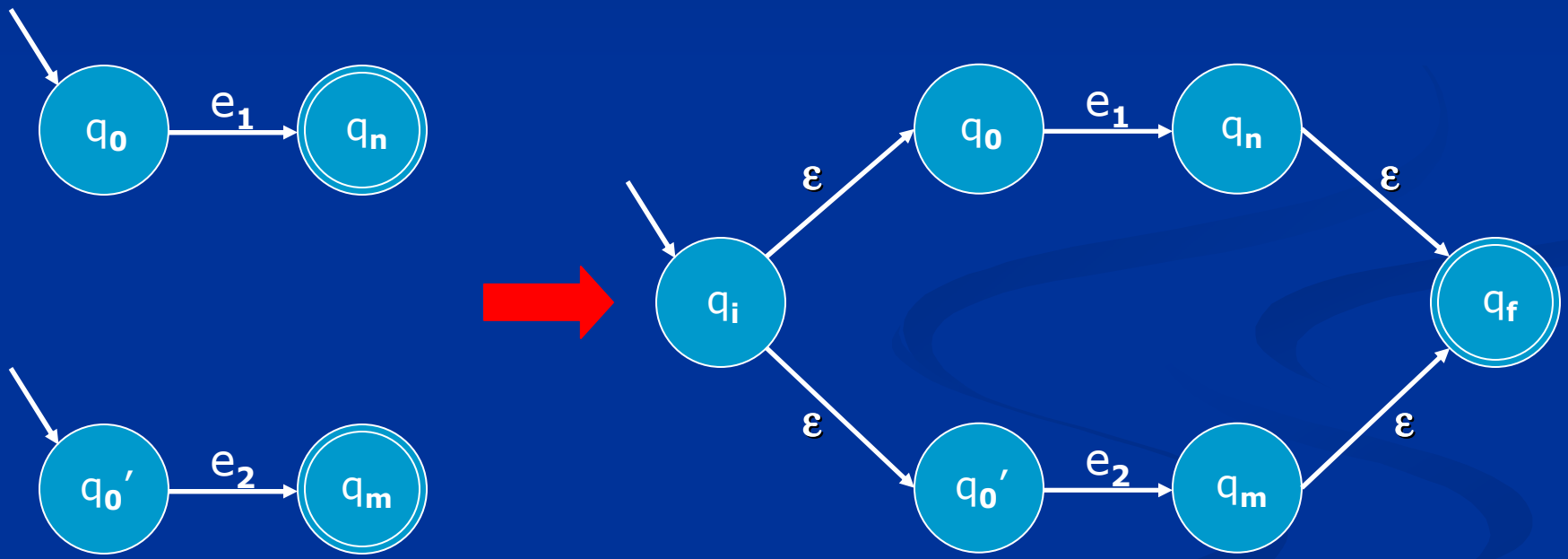
Esempio: Concatenazione

- r = (AT)*(C|G) con Σ = {A, T, C ,G}



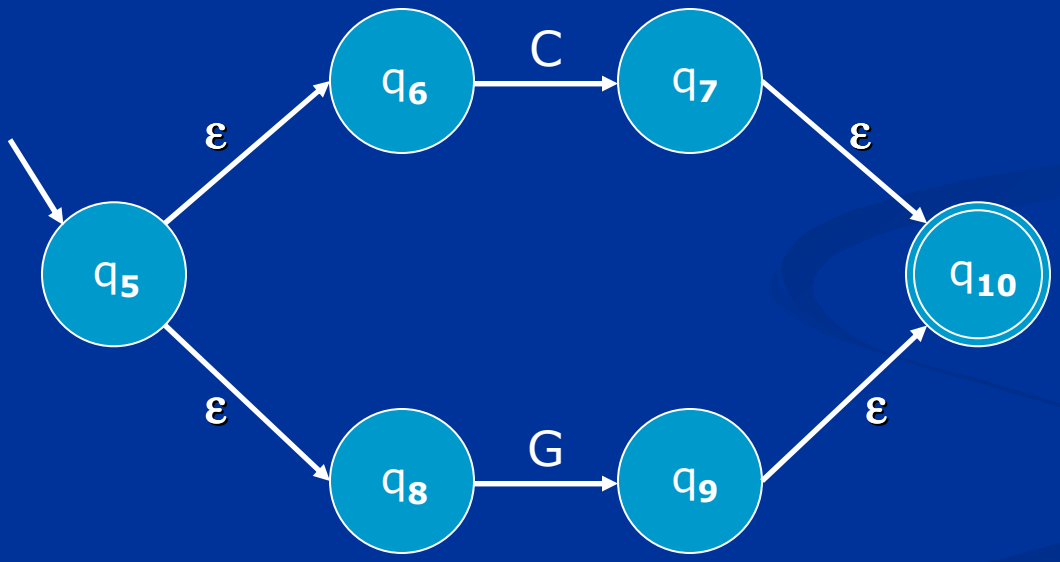
Alternativa

4. $e_1 | e_2 \in E_\Sigma$ per ogni $e_1, e_2 \in E_\Sigma$



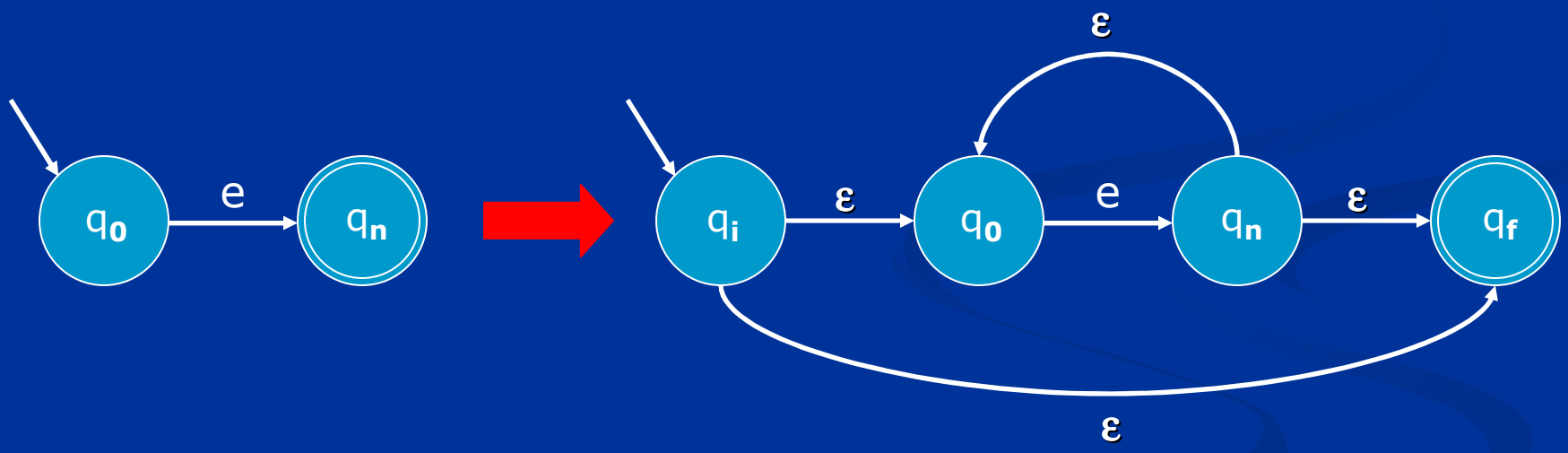
Esempio: Alternativa

- r = (AT)*(C|G) con Σ = {A, T, C ,G}



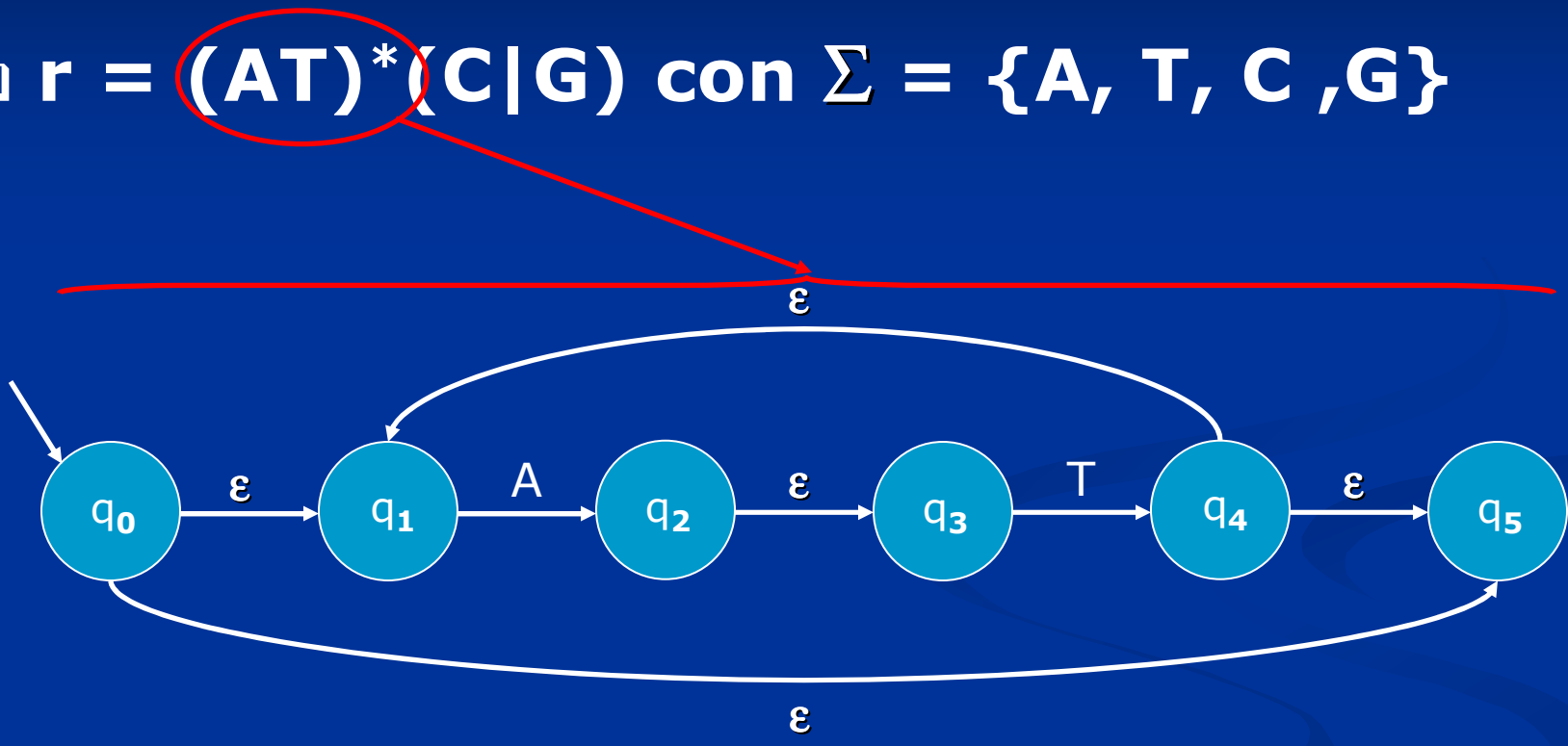
Sequenza (Stella di Kleene)

5. $e^* \in E_\Sigma$ per ogni $e \in E_\Sigma$



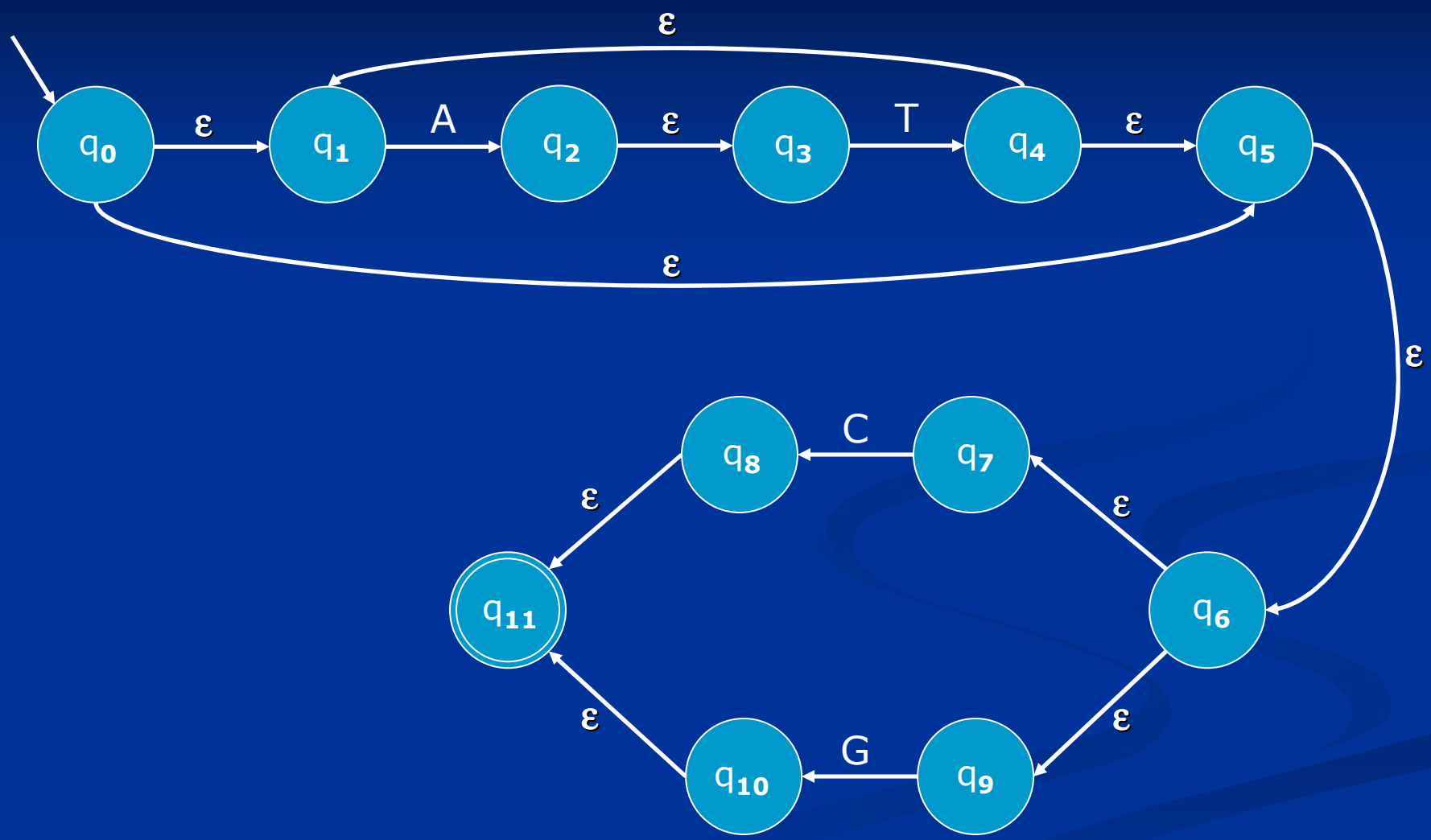
Esempio: Sequenza

- r = (AT)*(C|G) con Σ = {A, T, C ,G}



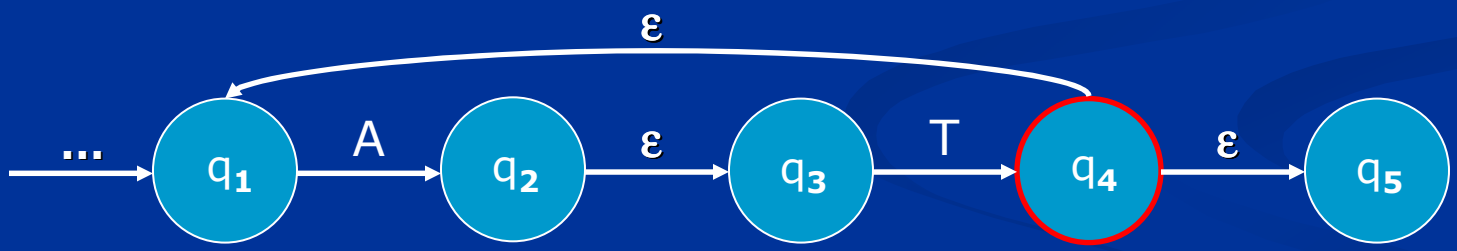
(AT)* = {ε, AT, ATAT, ATATAT, ATATATAT, ...}

Esempio: Automa Finale



ε-transizioni

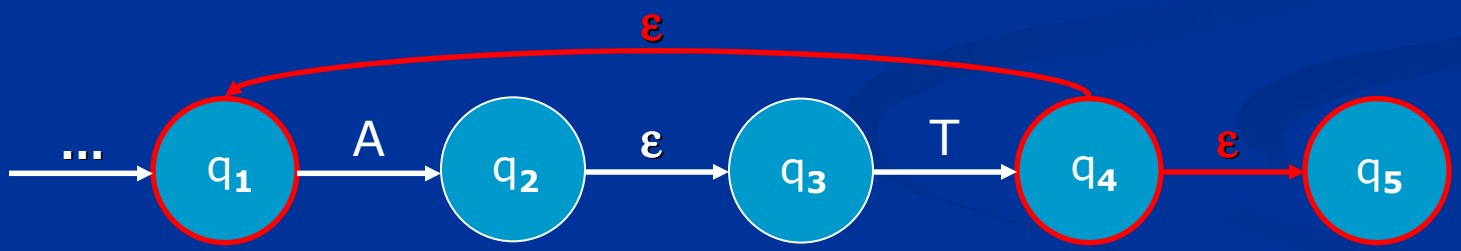
- **Automi non deterministici**
  - **ε-transizione, è una transizione di un arco dell'automa senza dover leggere alcun carattere del testo**



T = ...T...

ε-transizioni

- **Automi non deterministici**
  - **ε-transizione, è una transizione di un arco dell'automa senza dover leggere alcun carattere del testo**



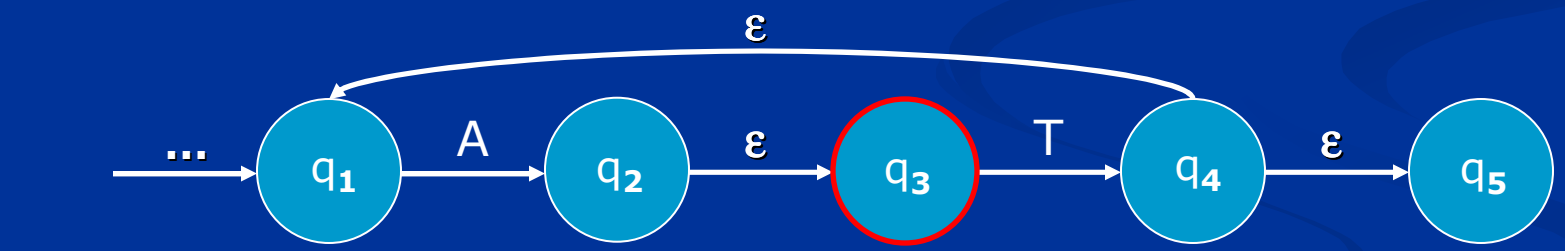
T = ...T...

Ricerca nell'Automa

- **Complessità: O(|r|*|T|)**
- **Transizione Σ sullo stato iniziale**
  - **permette la ricerca del pattern, in qualsiasi posizione, all'interno del testo T**
- **Visita dell'automa mediante la funzione δ', $Q' = Q \cup Q''$**
  - **Q'' è l'insieme degli stati raggiunti mediante transizioni ε dall'insieme di stati restituito dalla funzione δ, Q**

δ′: Esempio

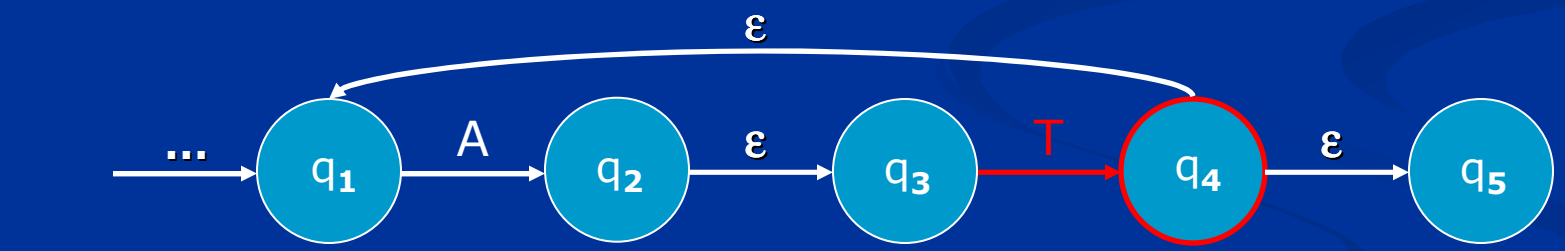
- Visita dell'automa mediante la funzione δ′, Q′= Q ∪ Q″
  - Q″ è l'insieme degli stati raggiunti mediante transizioni ε dall'insieme di stati restituito dalla funzione δ, Q



T = ...T...

δ′: Esempio

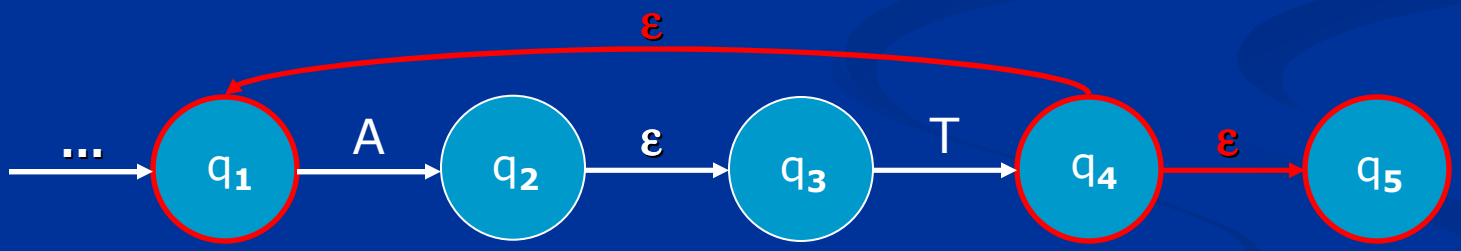
- Visita dell'automa mediante la funzione δ′, Q′= Q ∪ Q″
  - Q″ è l'insieme degli stati raggiunti mediante transizioni ε dall'insieme di stati restituito dalla funzione δ, Q



T = ...T... Q = {q_4}

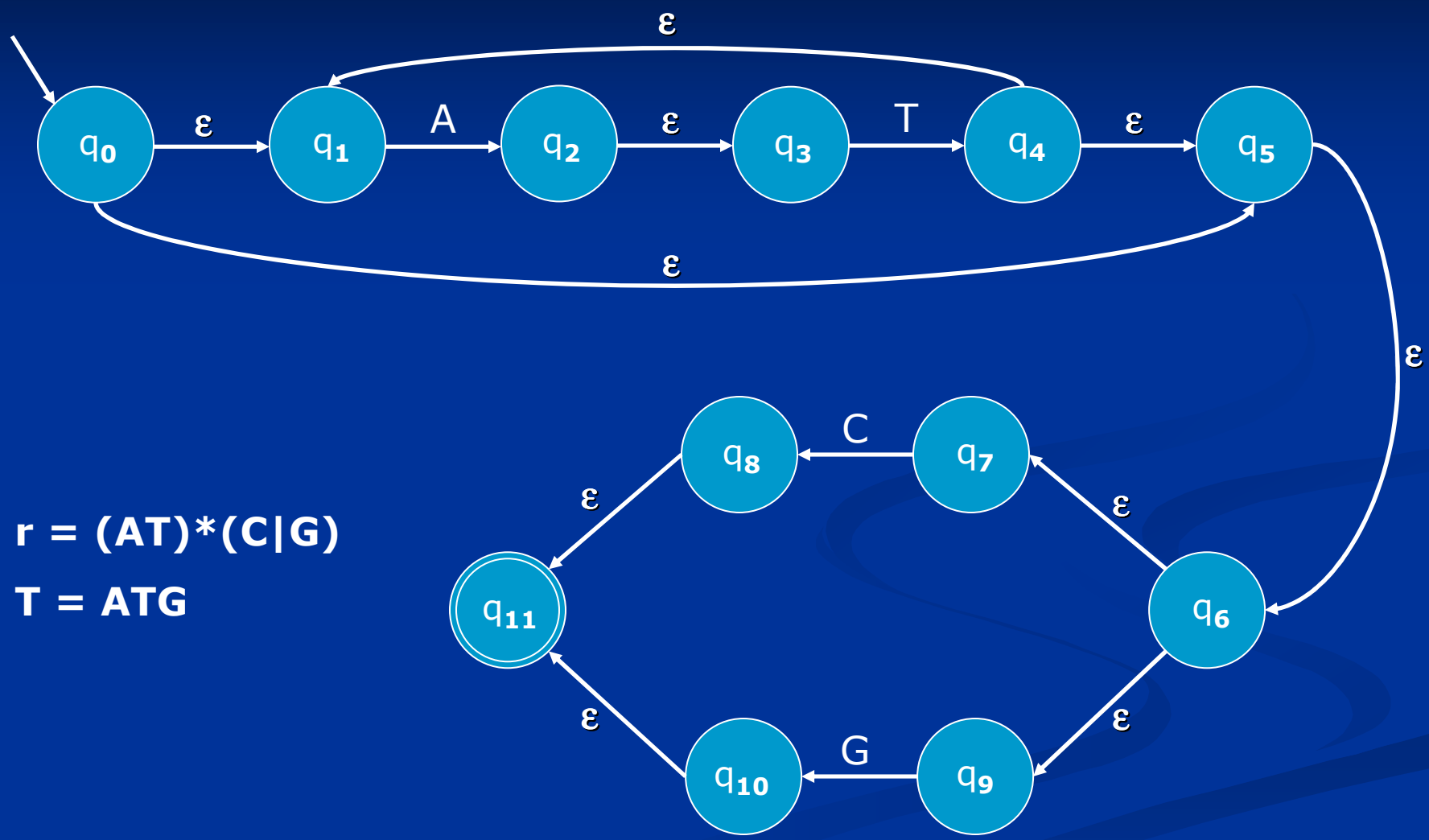
δ′: Esempio

- Visita dell'automa mediante la funzione δ′, Q′= Q ∪ Q″
  - Q″ è l'insieme degli stati raggiunti mediante transizioni ε dall'insieme di stati restituito dalla funzione δ, Q



T = ...T... Q = {q4} Q″ = {q5, q1} Q′ = {q4, q5, q1}

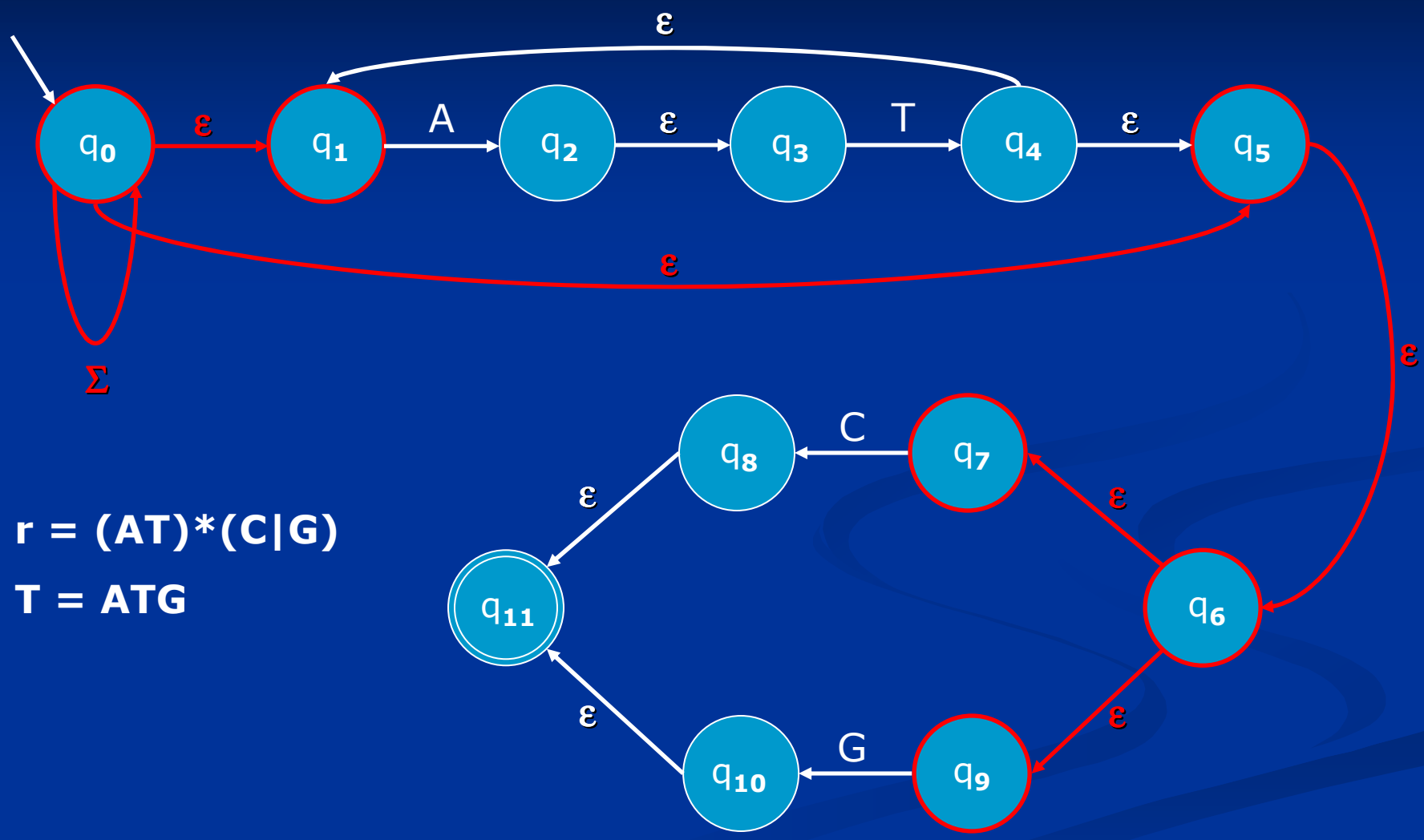
Esempio 1



r = (AT)*(C|G)

T = ATG

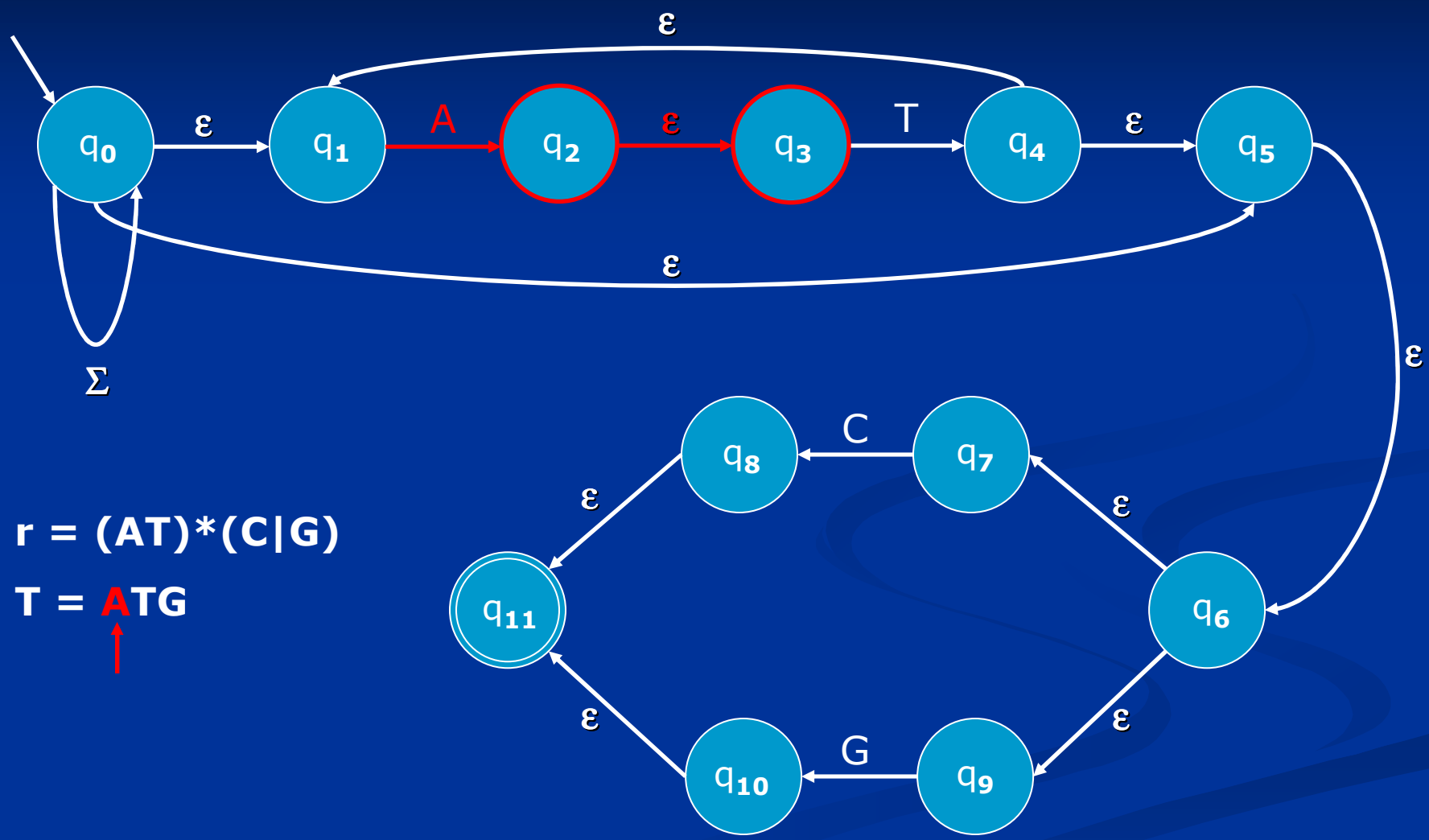
Esempio 1



r = (AT)*(C|G)

T = ATG

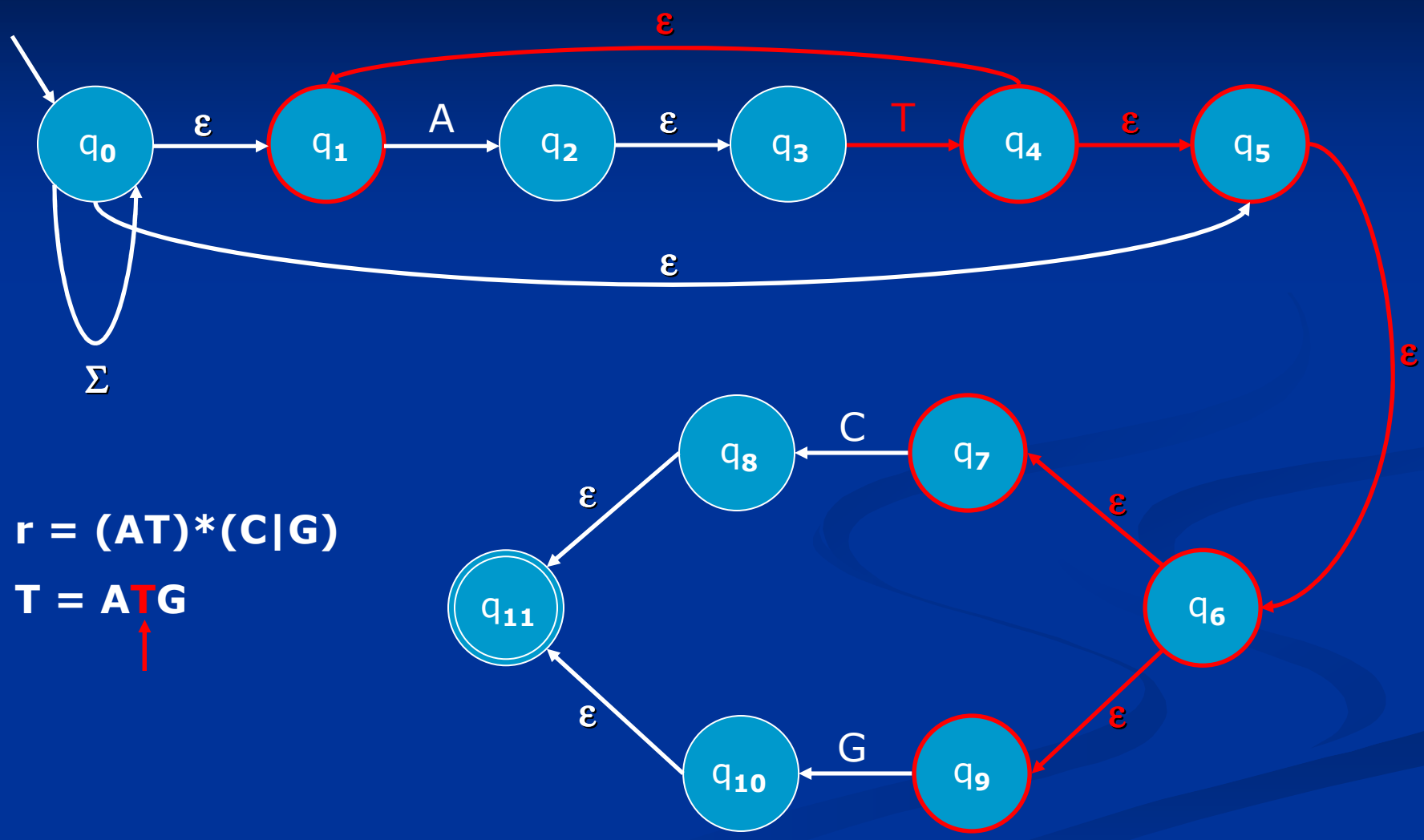
Esempio 1



r = (AT)*(C|G)

T = ATG

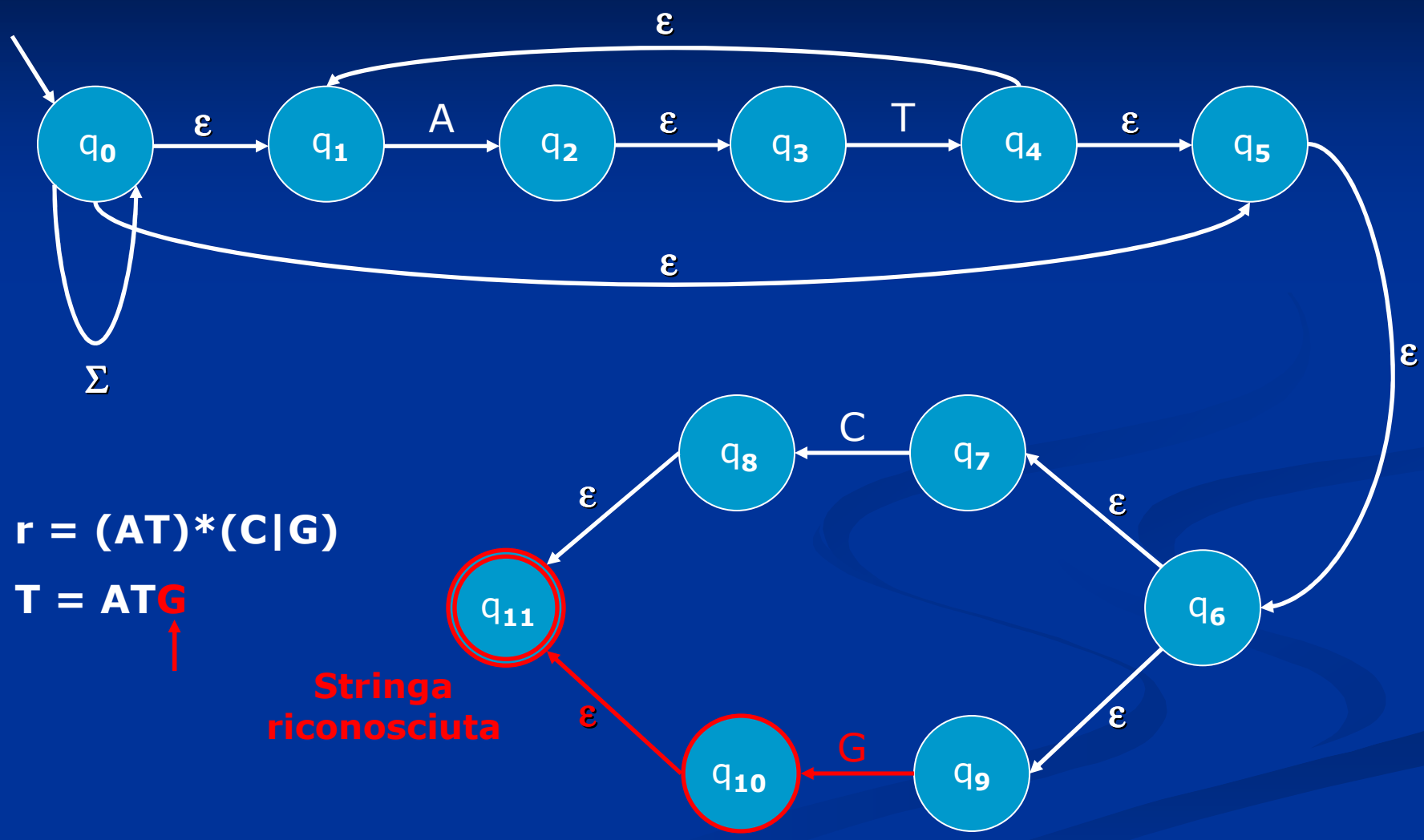
Esempio 1



r = (AT)*(C|G)

T = ATG

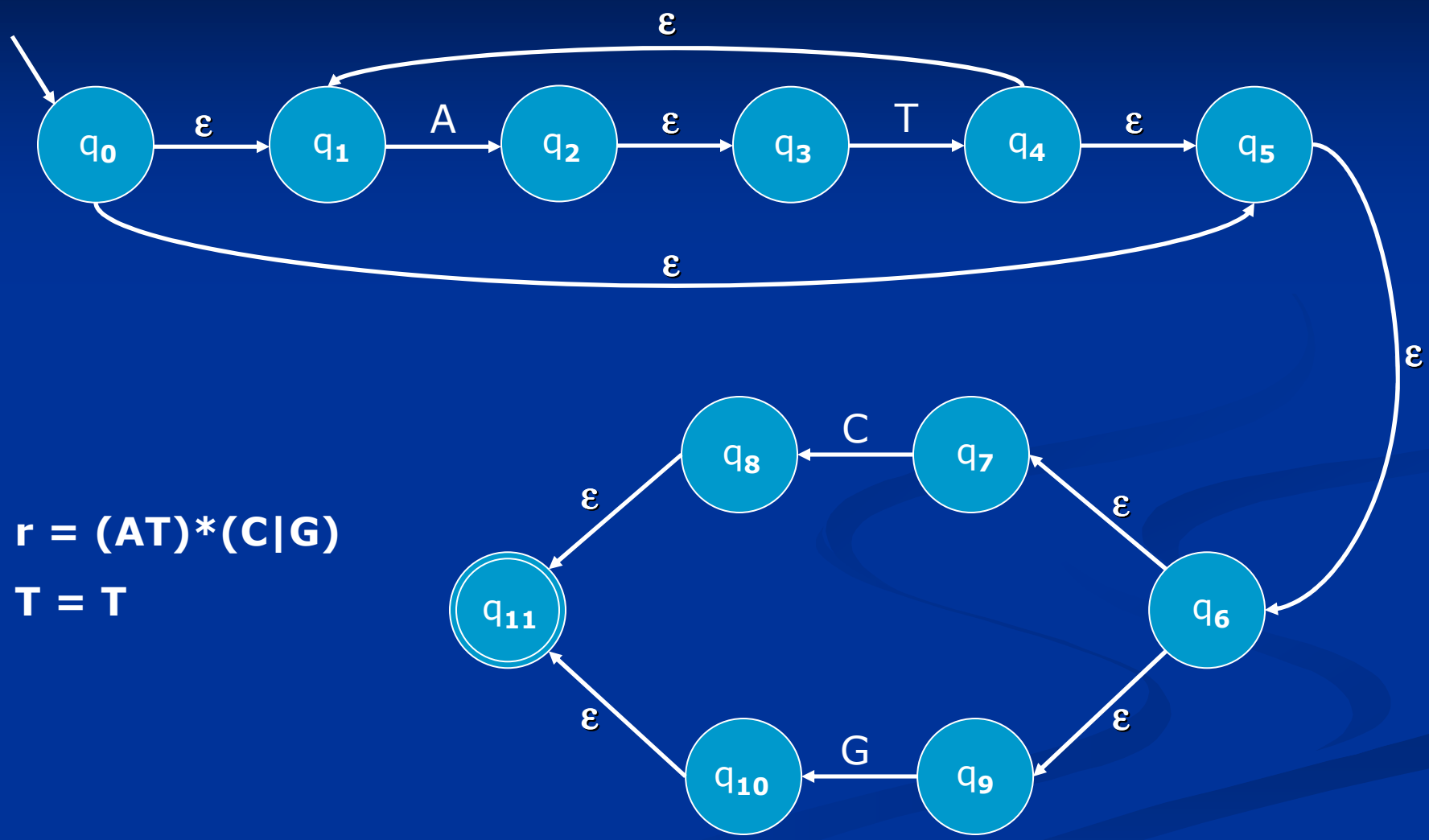
Esempio 1



r = (AT)*(C|G)

T = ATG

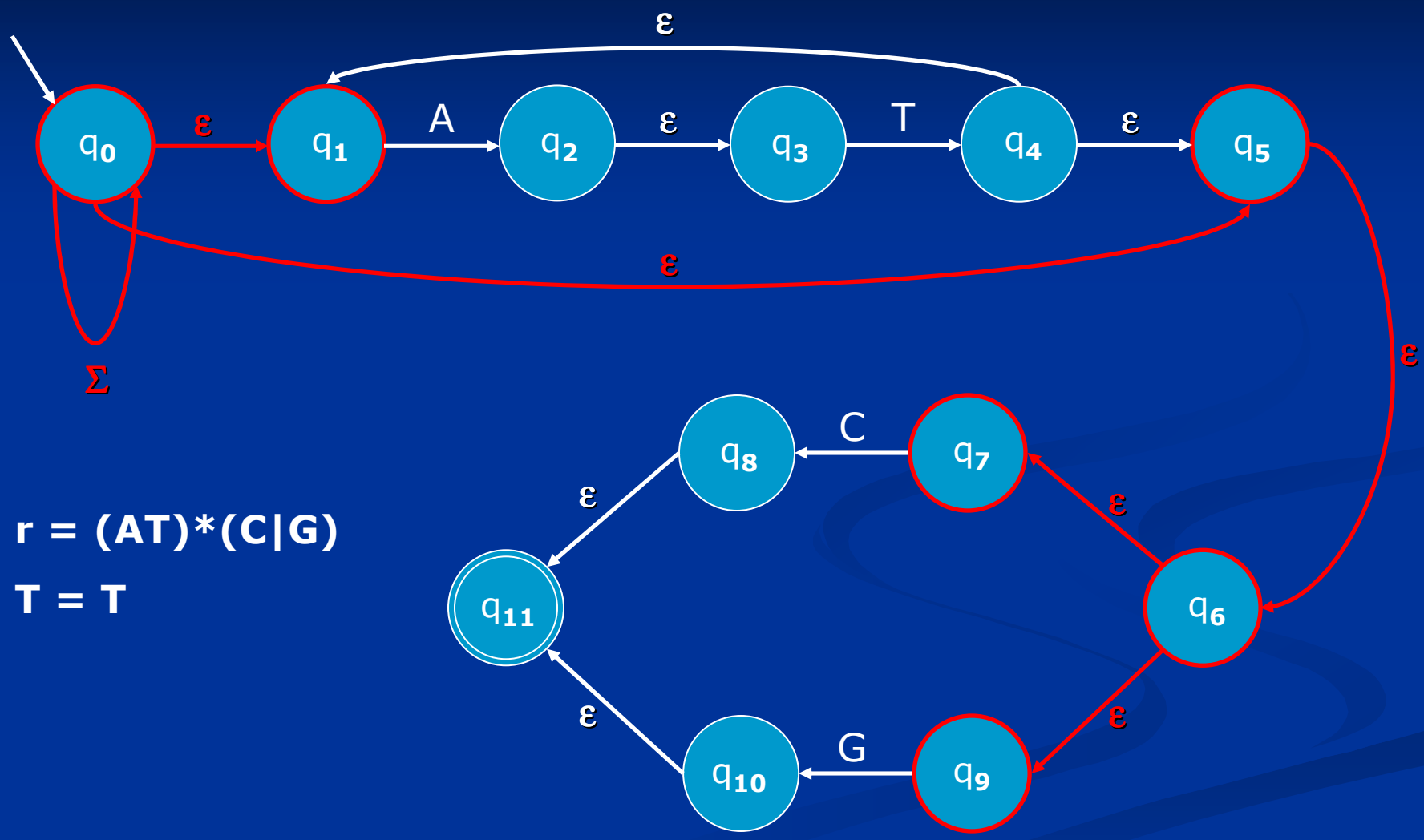
Esempio 2



r = (AT)*(C|G)

T = T

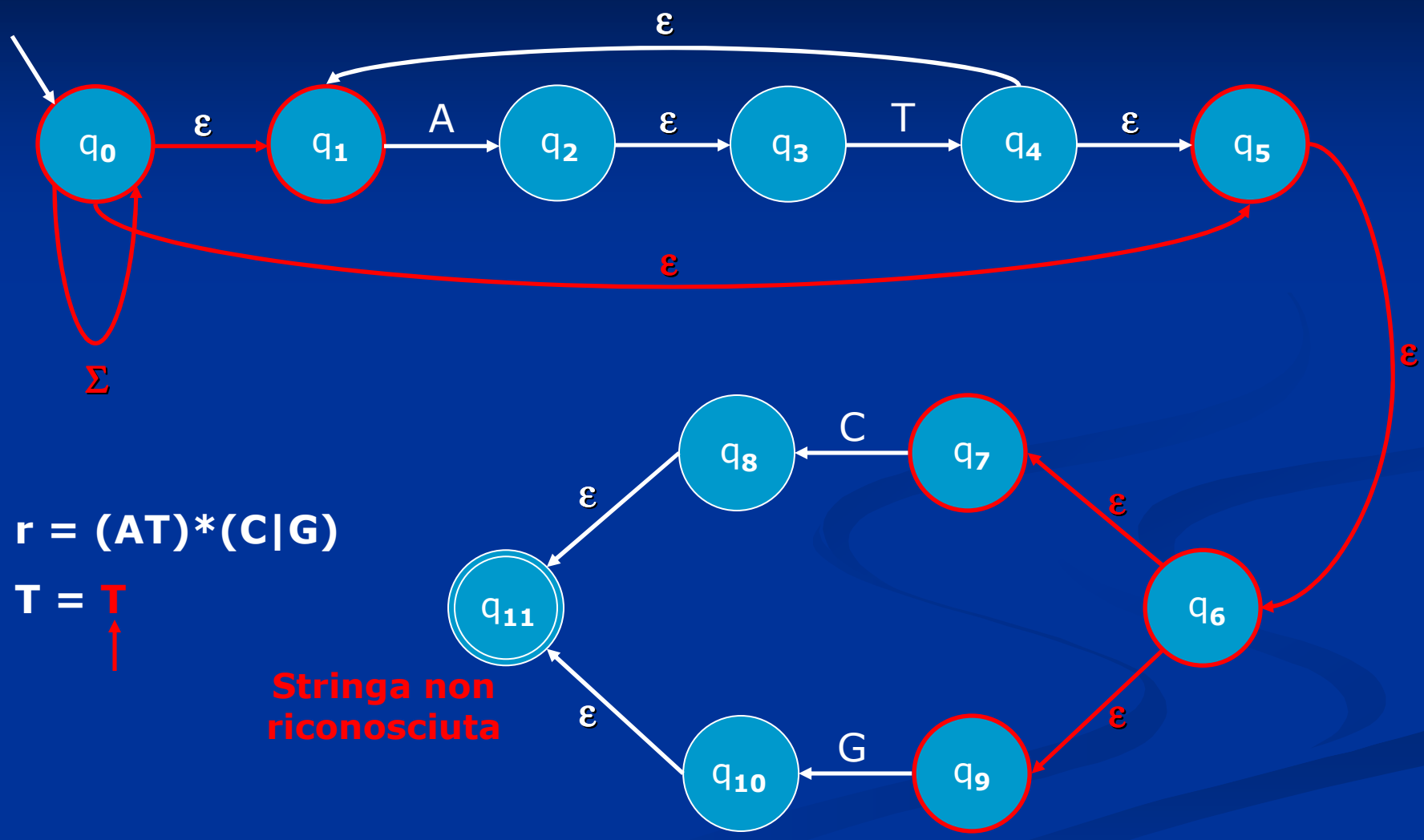
Esempio 2



r = (AT)*(C|G)

T = T

Esempio 2



r = (AT)*(C|G)

T = T

Conclusioni

- Introduzione
- Nozioni Preliminari
  - Automi
- Pattern Matching Multiplo
  - Aho - Corasick
    - Bidimensionale
- Espressioni Regolari
- Conclusioni

Conclusioni

- Le sequenze di proteine e di DNA possono essere viste come lunghi testi costruiti su specifici alfabeti
  - esempio: Σ_{DNA} = {A, C, G, T}
- La ricerca di sequenze specifiche di simboli in questi testi è un'operazione fondamentale per problemi come
  - ricostruire una sequenza di DNA dai singoli pezzi ottenuti da esperimenti

Conclusioni

- ricercare delle specifiche caratteristiche nelle sequenze di DNA
- determinare quanto sono differenti due sequenze di codice genetico

- Tutti questi problemi possono essere modellati come ricerche di una o più stringhe all'interno di un testo

Conclusioni

- Noi abbiamo visto come si possono utilizzare gli automi per affrontare problemi di
  - pattern matching multiplo
  - pattern matching mediante espressioni regolari
- In realtà, la ricerca esatta è poco idonea a risolvere i problemi reali di pattern matching nell'ambito della biologia, in quanto la struttura del DNA risulta essere molto complessa

Conclusioni

- **L'attenzione degli studiosi si è dunque rivolta alla ricerca approssimata, che ha prodotto soluzioni algoritmiche più efficienti ed affidabili**
- **Tuttavia il pattern matching approssimato risulta ancora oggi un campo in pieno sviluppo**

Bibliografia

- R. S. Bird, Two Dimensional Pattern Matching, Department of Computer Science, University of Reading, Whiteknights Park, Reading, Berkshire, U. K.
- R. Grossi, Algoritmi per Internet e Web: Ricerca e Indicizzazione dei Testi, Versione 1.3, Anno Accademico 2002-2003
- http://www.dia.unisa.it/~ads/BIOINFORMATICA/PatternMatching2006/PatternMartching.htm