ESPRESSIONI REGOLARI

Le espressioni regolari sono una notazione per rappresentare insiemi di stringhe (parole).

Sono state introdotte da KLEENE negli anni ‘50,

poi utilizzate per la prima volta da Thompson, negli anni ‘70, in un editor poi incorporato in *Unix* in alcuni comandi di gestione di testi (per esempio *grep “Global* search for *Regular Expression and Print”*)

e nel *Perl* di Larry Wall alla fine degli anni ‘80.

Definizione formale: sintassi

Dato un alfabeto Σ, un'espressione regolare su Σ, in breve r.e.(Σ), è ricorsivamente definita come segue:

base:

∅ e ε sono r.e.(Σ),

ogni a ∈ Σ è una r.e.(Σ)

passo ricorsivo: se r e s sono r.e.(Σ), allora (r+s), (rs) e (r)* sono r.e.(Σ).

Definizione formale: semantica

Ogni r.e.(Σ) r denota un linguaggio L(r), ricorsivamente definito come segue:

base:

se r = ∅, allora L(r) = ∅, se r = ε, allora L(r) = {ε},

se r = a, allora L(r) = {a}

passo ricorsivo:

se r = (u+v) allora L(r) = L(u) ∪ L(v),

se r= (uv), allora L(r) = L(u)L(v) e

se r = (u)*, allora L(r) = L(u)*.

Due espressioni regolari che denotano lo stesso linguaggio si dicono equivalenti. Scriviamo *r*=*s* se *r* ed *s* sono equivalenti, cioè L(*r*)=L(*s*).

Esempi

1. a*ba*ba* denota il linguaggio delle parole su {a,b} che contengono esattamente due b.

2. (b+abb)* denota il linguaggio delle parole su {a,b} in cui ogni occorrenza di a è immediatamente seguita da due b.

3. b(ab)* denota il linguaggio delle parole in cui a e b si alternano, cominciando e finendo per b, lo stesso linguaggio è denotato anche da (ba)*b, quindi le due espressioni regolari sono equivalenti.

Risultato principale

Sia L(r.e.) la classe dei linguaggi denotati da una r.e. al variare dell'alfabeto.

Teorema (Kleene). L(r.e.) = L(NFA).

Poichè sappiamo che L(NFA) = L(DFA),

si ha

$$L(\text{r.e.}) = L(\text{DFA}) = L(\text{NFA}).$$

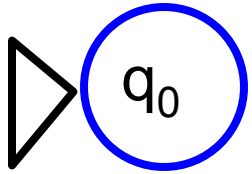
L(r.e.) ⊆ L(NFA)

Si dimostra facilmente per induzione strutturale, ossia sulla definizio di r.e.:

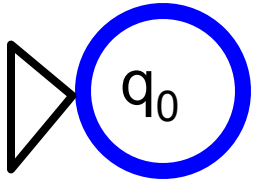
Sia Σ l'alfabeto.

Passo base:

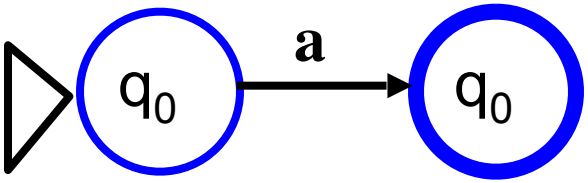
Un automa per il ∅ :



Un automa per ε :



Un automa per ogni a ∈ Σ :



L(r.e.) ⊆ L(NFA) (continua)

Passo induttivo:

se r e s sono r.e.(Σ), e A(r) e A(s) sono gli NFA che riconoscono L(r) e L(s), allora gli NFA che riconoscono i linguaggi denotati da

(r+s), (rs) e (r)*

si ottengono facilmente applicando le costruzioni viste per dimostrare le proprietà di chiusura della classe dei linguaggi accettati da un NFA rispetto a unione, prodotto e stella di Kleene.

L(NFA) $\subseteq$ L(r.e.)

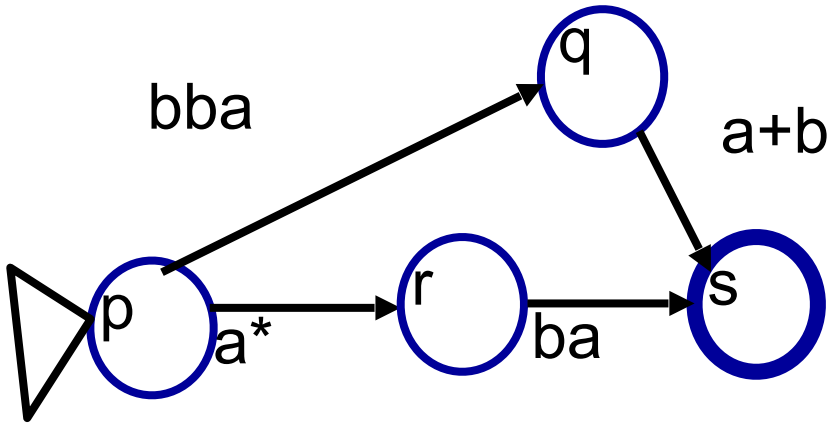
Per provare questa direzione, al posto di quella originale e algebrica di Kleene, forniremo un algoritmo che riceve in input un NFA e fornisce in output la r.e. equivalente.

L'algoritmo si deve a R. Mc Naughton e H. Yamada (1960)

GNFA: definizione

Un GNFA, (Generalized Non deterministic Finite Automaton) è un diagramma degli stati in cui ogni arco è etichettato da un'espressione regolare.

Esempo:



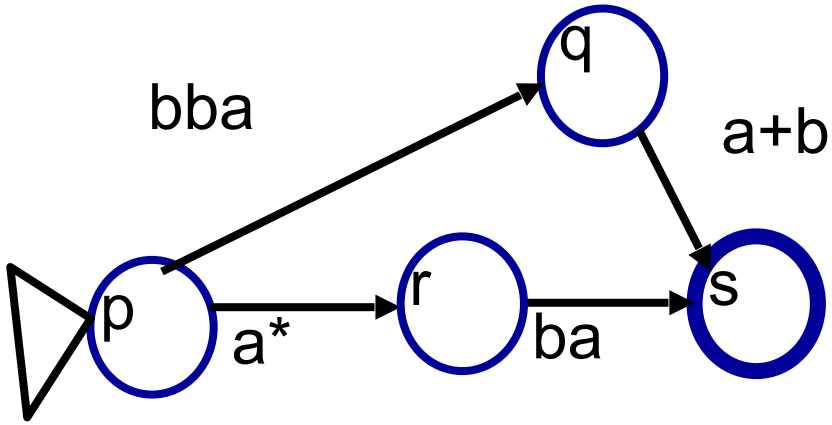
Il cammino pqs è etichettato dall'r.e bba(a+b), ottenuta concatenando le r.e. che etichettano gli archi (p,q) e (q,s)

GNFA: accettazione

Un GNFA accetta una parola w se è contenuta nel linguaggio denotato dall'espressione regolare che etichetta un cammino dallo stato iniziale a uno finale.

Il linguaggio accettato da un grafo di espressione è l'insieme delle parole accettate.

GNFA: esempio

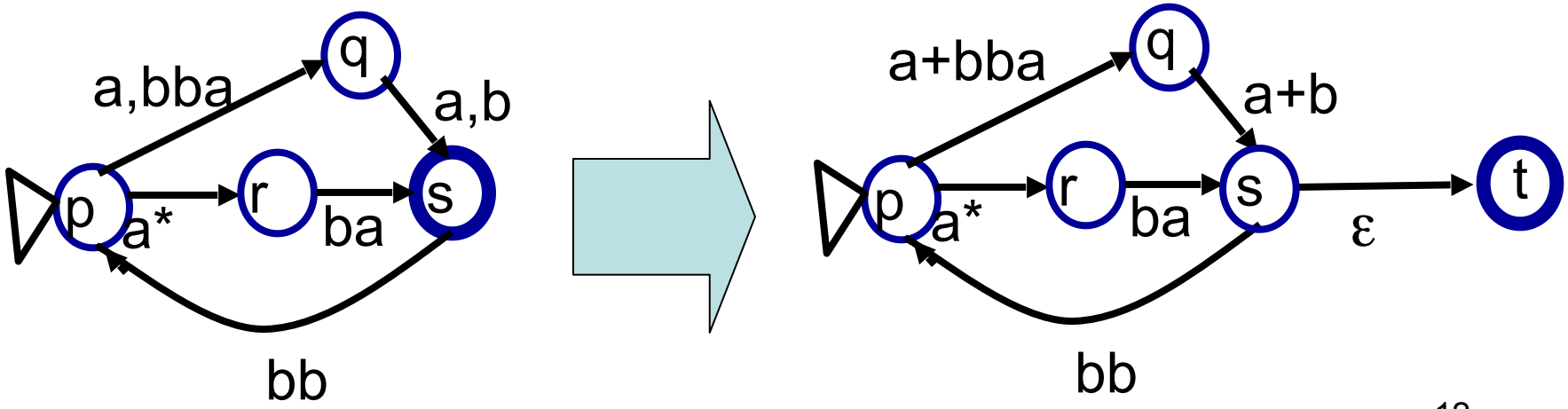


Per il GNFA dell'esempio il linguaggio accettato è l'insieme delle parole formata da zero o più *a* e che terminano per *ba*, oppure da quelle che iniziano con *bba*.

GNFA: Forma normale

Un GNFA è in una forma normale se

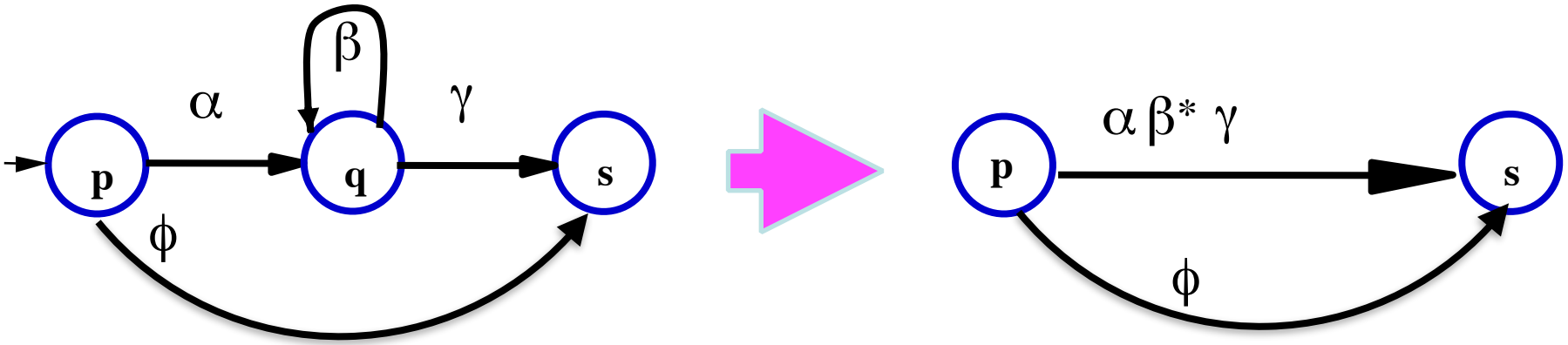
- **ha esattamente uno stato finale diverso da quello iniziale e**
- **lo stato finale non ha archi uscenti**
- **per ogni coppia di nodi (i,j) c'è al più un arco etichettato $w_{i,j}$ da i a j.**



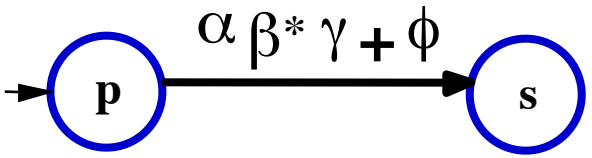
forma normale

L(NFA) ⊆ L(r.e.) (continua)

Regola di eliminazione di uno stato:



In forma normale:



L'espressione regolare che etichetta il nuovo arco da *p* a *s* denota il linguaggio delle parole che "consentono" il passaggio da *p* a *s* passando per *q* o *direttamente*.
Si noti che *p* non è necessariamente diverso da *s*.

L(NFA) ⊆ L(r.e.) (continua)

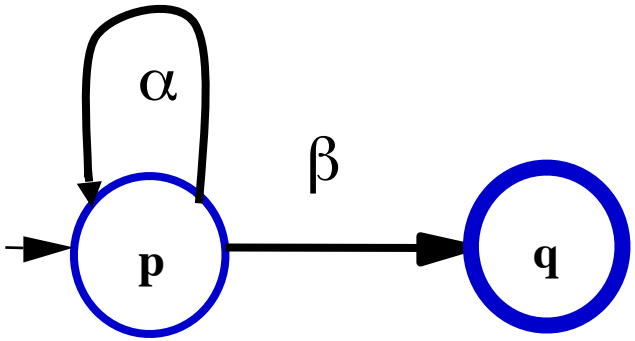
Algoritmo.

Passo iniziale: trasforma il diagramma di transizione dell'NFA in input M in un GNFA in forma normale

ripeti, finchè ci sono stati diversi da quello finale o iniziale, il passo di eliminazione di uno stato q per ogni coppia di stati p,s per i quali q è intermedio.

L(NFA) $\subseteq$ L(r.e.) (continua)

Al termine si potrà avere un diagramma del tipo:

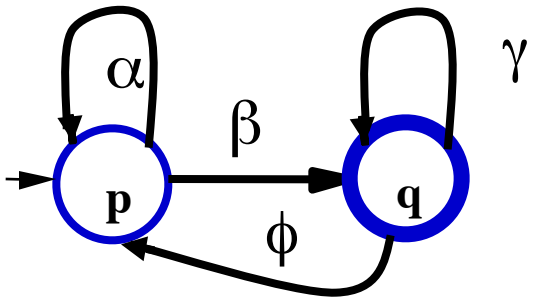


Quindi l'espressione regolare è $\alpha^*\beta$

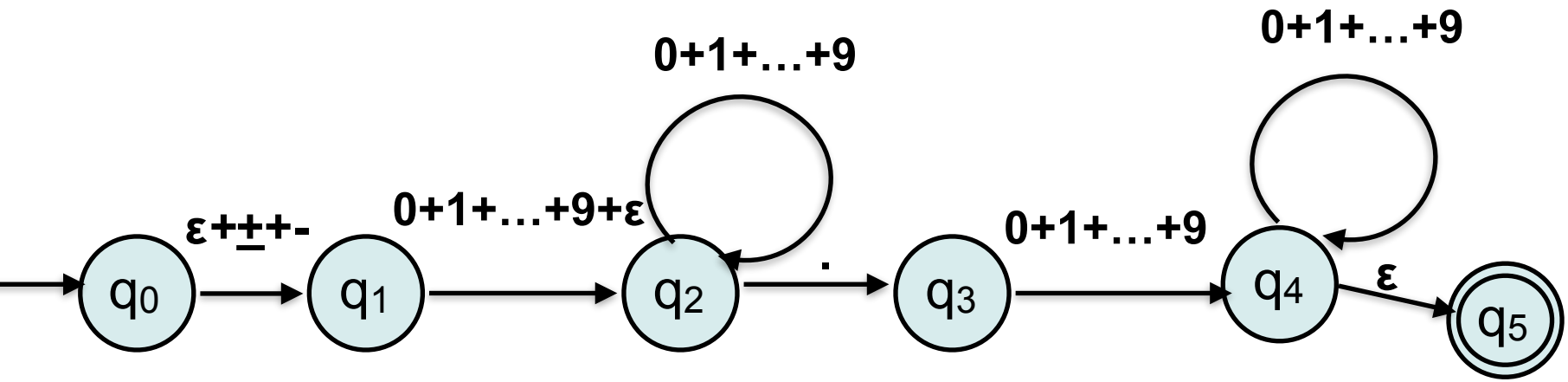
L(NFA) ⊆ L(r.e.) (fine)

La correttezza dell'algoritmo si basa sul fatto che ogni passo di eliminazione di uno stato conserva il linguaggio accettato dal GNFA e la forma normale del grafo.

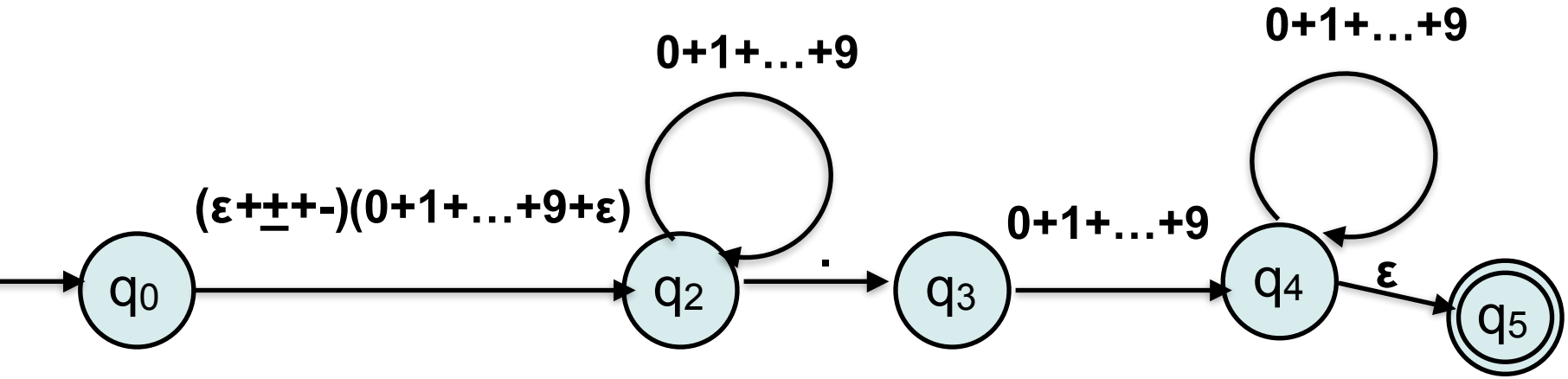
In alcune varianti (p.e. in JFLAP) di questo algoritmo si fa riferimento a una forma normale semplificata. Questo complica leggermente il passo finale e il GNFA ottenuto può essere del tipo:

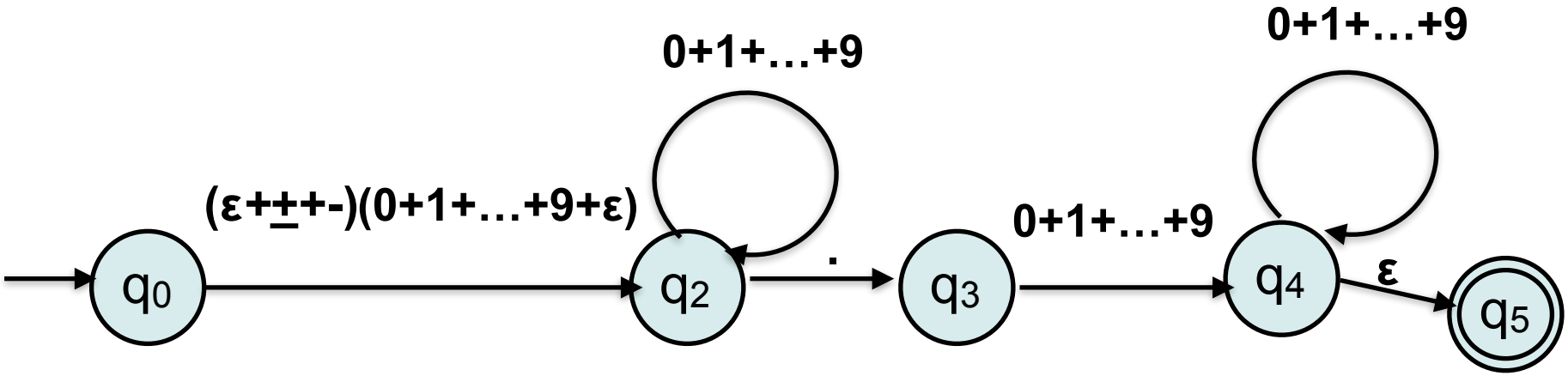


e l'espressione regolare è
(α*βγ*φ)*α*βγ *

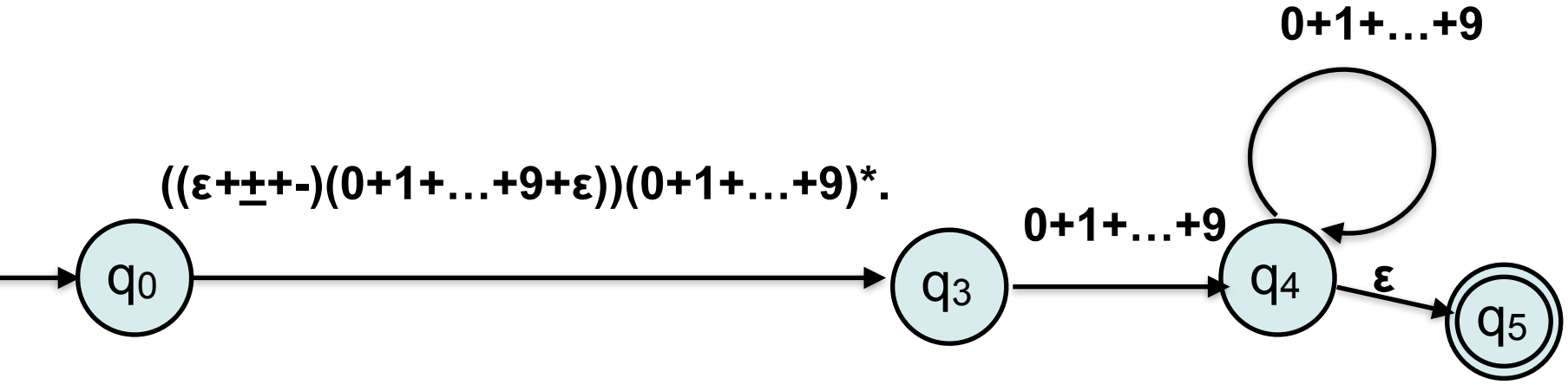


Eliminazione q_1

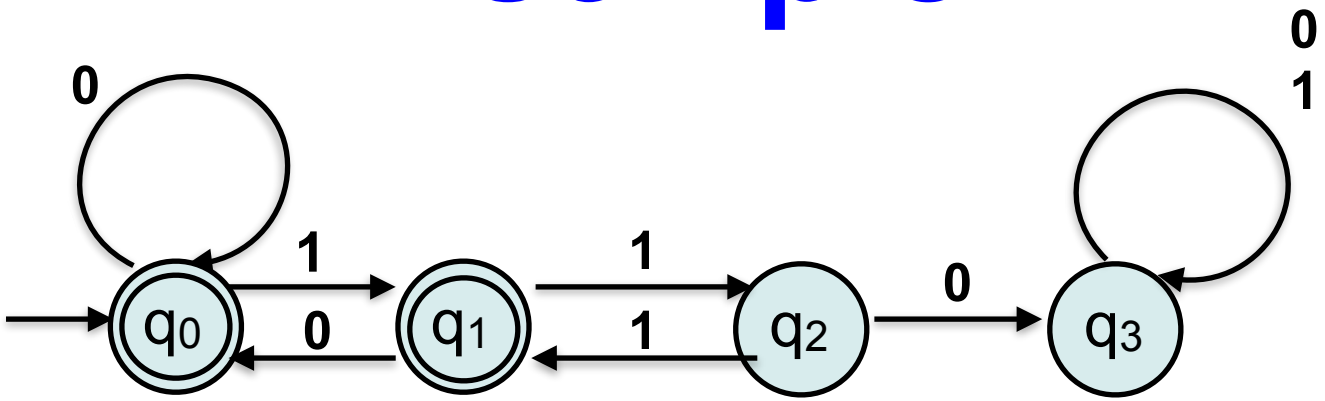




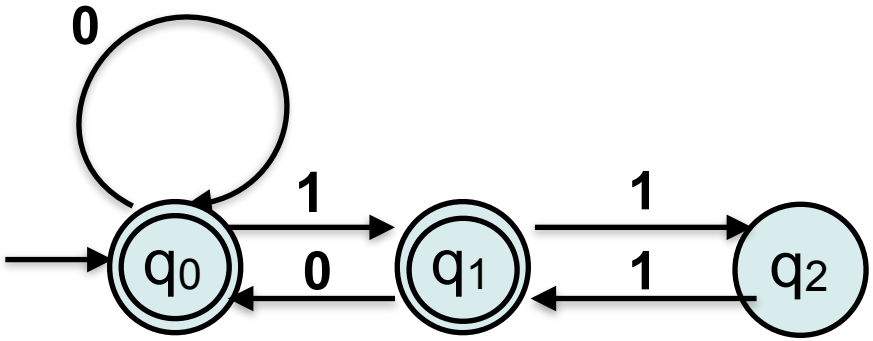
Eliminazione q_2

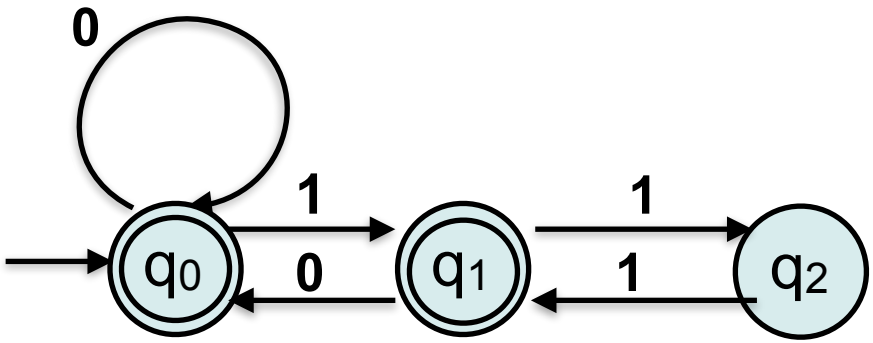


Esempio 2

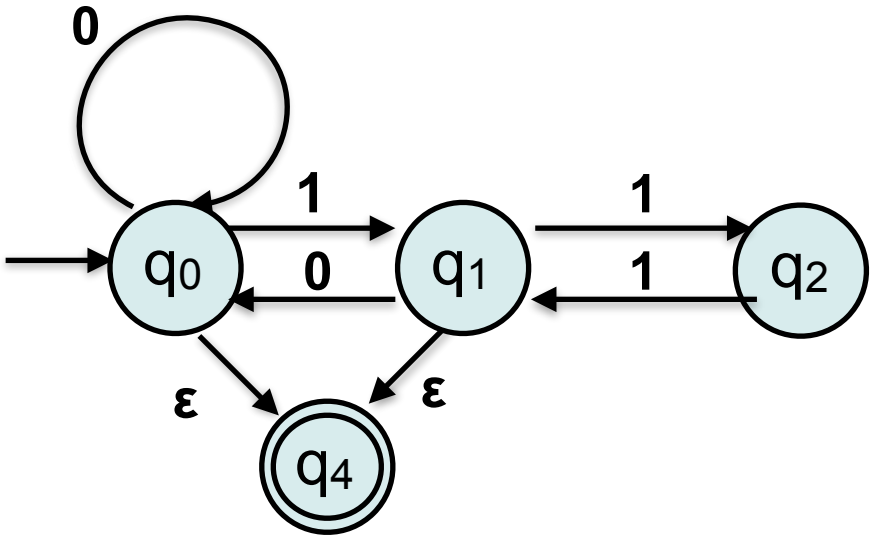


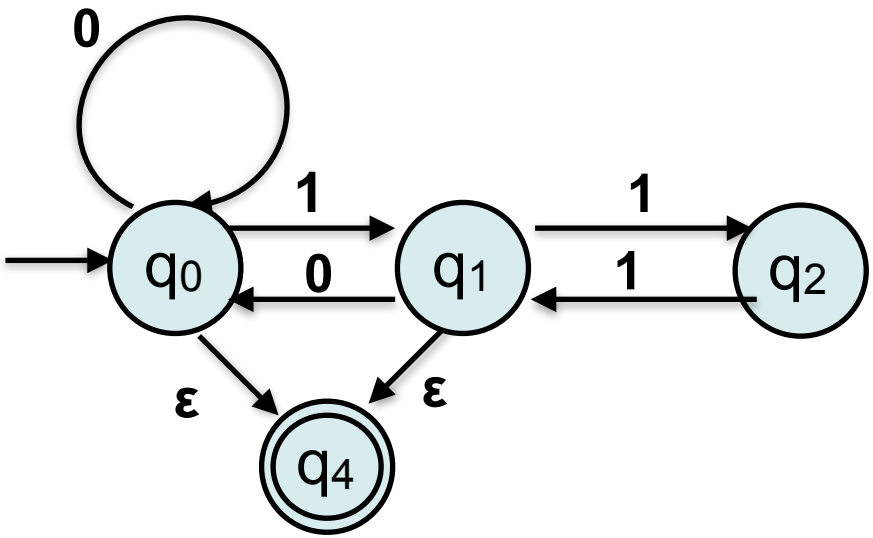
Eliminazione stati inutili



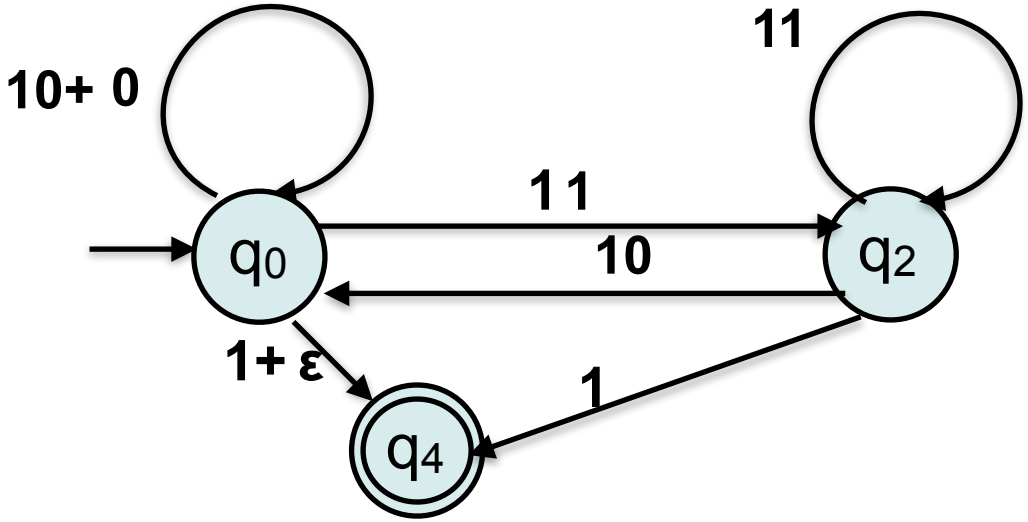


Forma normale





Eliminazione q_1



via q1:
cammino q_0-q_0

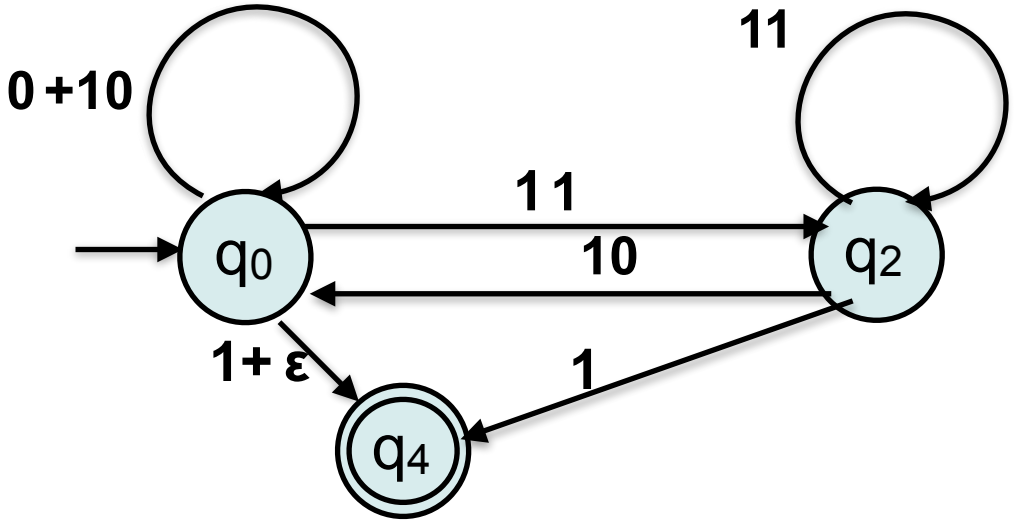
via q1:
cammino q_0-q_2

via q1:
cammino q_0-q_4

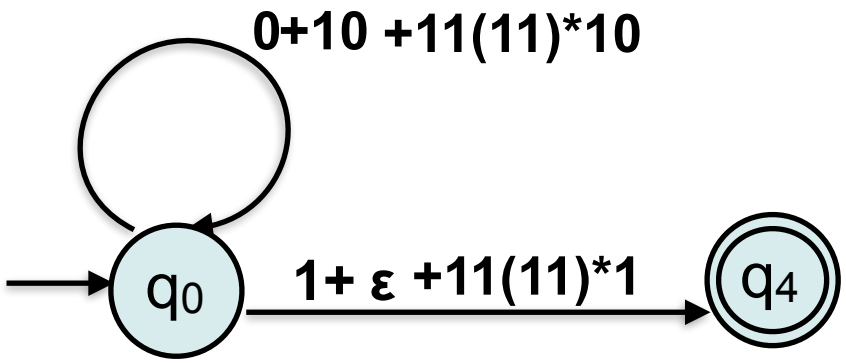
via q1:
cammino q_2-q_0

via q1:
cammino q_2-q_2

via q1:
cammino q_2-q_4



Eliminazione q_2



via q2:
cammino q_0-q_0

via q2:
cammino q_0-q_4

R.E. = (0+10+11(11)*10)*(1+ε+11(11)*1)

Problemi di decisione

Sia L un linguaggio regolare su Σ, si delinei un algoritmo che decide se L = Σ*

Dati L ed L' regolari su Σ, si delinei un algoritmo che decide se hanno una stringa in comune.