# MODUL VII

# TEORI BAHASA DAN AUTOMATA

Tujuan :

Mahasiswa memahami ekspresi reguler dan dapat menerapkannya dalam berbagai penyelesaian persoalan.

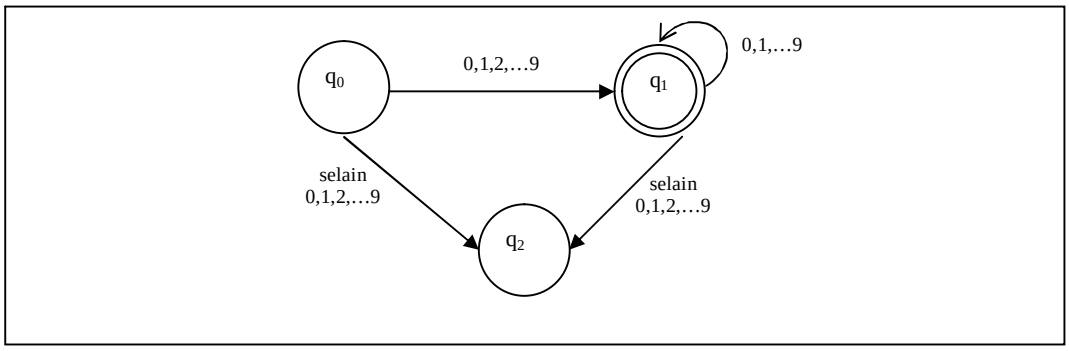
Materi :

- Penerapan Ekspresi Regular
- Notasi Ekspresi Regular
- Hubungan Ekspresi Regular dan Finite State Automata

# EKSPRESI REGULAR

## Penerapan Ekspresi Regular

Sebuah bahasa dinyatakan regular jika terdapat *finite state automata* yang dapat menerimanya. Bahasa-bahasa yang diterima oleh suatu *finite state automata* bisa dinyatakan secara sederhana dengan ekspresi regular (*regular expression*). Ekspresi regular selanjutnya kita sebut sebagai ER, memungkinkan menspesifikasikan atau mendefinisikan bahasa-bahasa. Ekspresi regular memberikan suatu pola (*pattern*) atau template untuk untai/*string* dari suatu bahasa. Untai yang menyusun suatu bahasa regular akan cocok (*match)* dengan pola bahasa itu. Banyak masalah pada perancangan perangkat lunak yang bisa disederhanakan dalam dengan melakukan pengubahan notasi ekspresi regular ke dalam implementasi computer dari finite state automata yang bersangkutan. Penerapan ekspresi regular yang tampak misalnya pencarian (*searching*) untai karakter (*string*) pada suatu file, biasanya fasilitas ini ada pada *text editor*. Dalam kasus itu dilakukan penerapan *finite state automata* pada untai-untai yang terdapat dalam file tersebut. Contoh penerapan yang lain adalah pembatasan data masukan yang diperkenankan, misalnya suatu *field* masukan hanya menerima *input* bilangan (0..9). Bisa kita lihat otomatanya pada gambar 1.




*Gambar 1. FSA menerima bilangan integer tak bertanda*

Bila dalam bahasa Indonesia bisa dikatakan bahwa otomata pada gambar menerima masukan symbol input antara 0 sampai 9 sedang ekspresi regularnya dinyatakan sebagai berikut:

*(digit)digit)**

dengan digit adalah 0..9

Dalam suatu kompilator, ekspresi regular bisa diaplikasikan untk melakukan analisis leksikal, yaitu mengidentifikasikan unit-unit leksikal yang dikenal dalam program. Unit leksikal ini biasanya disebut dengan *token*. *Token-token* pada suatu

bahsa pemrograman kebanyakan tanpa kecuali dinyatakan sebagai ekspresi regular . Misalkan suatu *identifier* baik huruf besar atau huruf kecil yang kemudian diikuti huruf atau digit, dengan tanpa pembatasan jumlah panjang bisa dinyatakan sebagai:

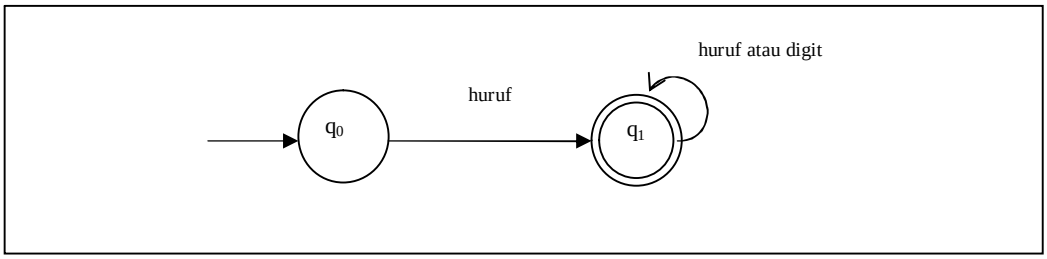
*(huruf)(huruf+digit)**

Contoh otomata pada gambar 2 berguna mengenali *identifier*, bila huruf A..Z, a..z, dan digit berupa 0..9.

Bila dalam bahasa FORTRAN dibatasi panjang *identifie*r maksimal 6 (enam), maka ekspresi regular untuk identifier pada FORTRAN bisa dinyatakan sebagai:

$(huruf)(huruf+digit)^5$

Dalam implementasinya suatu *finite state automata* akan diterjemahkan menjadi kode dalam sebuah bahasa pemrograman.




Gambar 2. FSA mengenali identifier

**Notasi Ekspresi Regular**

Sekarang kita akan memperkenalkan notasi baru disini yaitu ( * $^+$ + $\cup$ .) yaitu :

* (karakter asterisk), berarti bisa tidak muncul, bisa juga muncul berhingga kali *(0-n)*
$^+$ (pada posisi *superscript*/diatas) berarti minimal muncul satu kali *(1-n)*
+ atau $\cup$ berarti *union*
. (titik) berarti konkatensi, biasanya titik bisa dihilangkan,
misal *ab* bermakna seperti a.b

Contoh ekspresi regular (selanjutnya kita singkat ER):

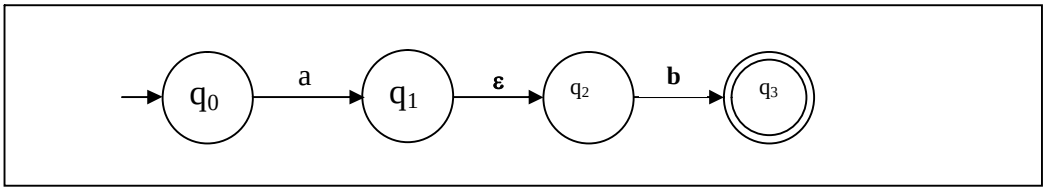
- ER: ab*cc
  contoh string yang dibangkitkan : abcc, abbcc, abbbcc, abbbbcc, acc, (b bisa tidak muncul atau muncul sejumlah berhingga kali)
- ER: 010*

contoh string yang dibangkitkan : 01, 010, 0100, 01000
(jumlah 0 diujung bisa tidak muncul, bisa muncul berhingga kali)

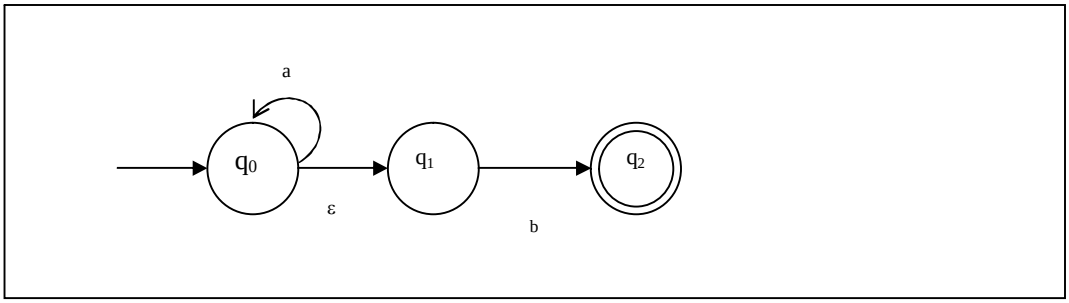
- ER: a*d
  contoh string yang dibangkitkan : d, ad, aad, aaad
- ER: $a^+d$
  contoh string yang dibangkitkan: ad, aad, aaad
  (a minimal muncul sekali)
- ER: a*∪b*(ingat '∪' berarti atau)
  contoh string yang dibangkitkan: a, b, aa, bb, aaa, bbb, aaaa, bbbb
- ER: (a∪b)
  contoh string yang dibangkitkan: a, b
- ER: (a∪b)*
  contoh string yang dibangkitkan: a, b, ab, ba, abb, bba, aaaa, bbbb
  (untai yang memuat a atau b)
  * perhatikan : notasi '∪' kadang dituliskan juga sebagai '+'
- ER: 01*+0
  contoh string yang dibangkitkan: 0, 01, 011, 0111, 01111,
  (string yang berawalan dengan 0, dan selanjutnya boleh diikuti deretan 1)

**Hubungan Ekspresi Regular dan Finite State Automata.**

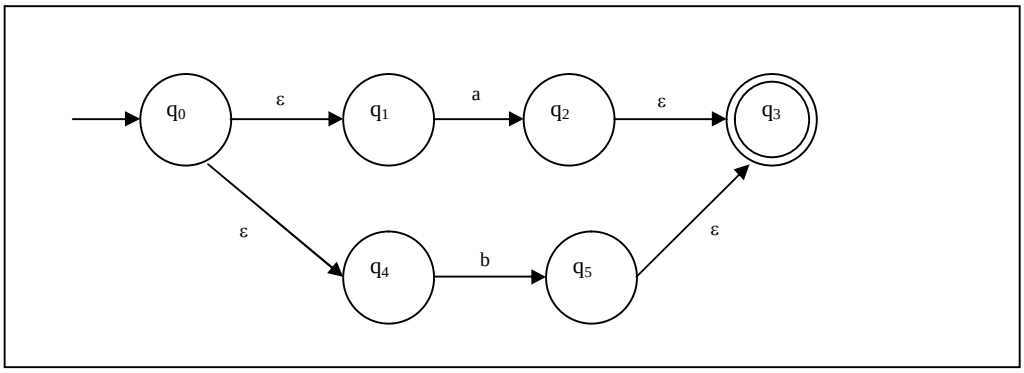
Untuk setiap ekspresi regular ada satu Non-deterministic Finite Automata dengan transisi ε (NFA ε-move) yang ekivalen. Sementara untuk *Deterministic Finite Automata* ada satu ekspresi regular dari bahasa yang diterima oleh *Deterministic Finite Automata.* Sederhananya kita bisa membuat suatu *Non-deterministic Finite Automata* ε-move dari suatu ekspresi regular. Bisa dilihat contohnya pada gambar 3,4,5. Yang perlu diperhatikan disitu, state akhir akan menandakan apakah suatu input diterima atau tidak.




Gambar 3. NFA ε-move untuk ER: ab




Gambar 4. NFA ε-move untuk ER: a*b

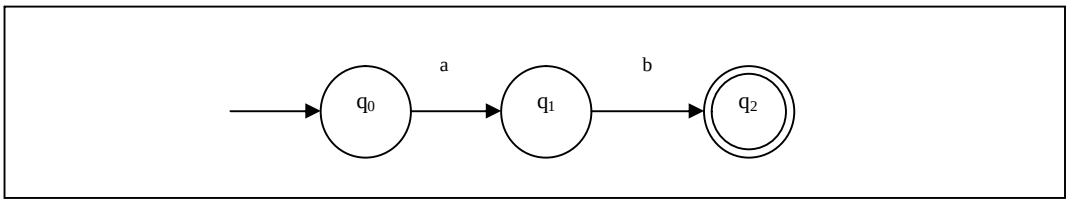



Gambar 5. NFA ε-move untuk ER: a ∪ b

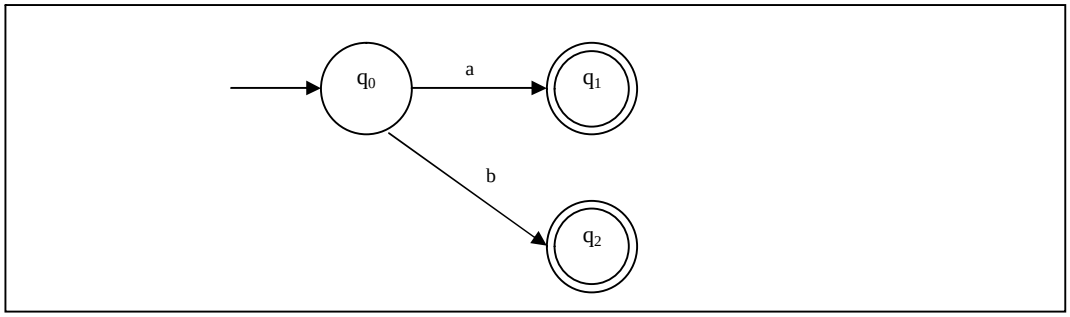
Kemudian dari *Non-deterministic Finite Automata* ε-move tersebut dapat kita ubah ke *Non-deterministic Finite Automata* dan selanjutnya ke *Deterministic Finite Automata,* atau prosesnya sebagai berikut:

NFA ε-move ⇒ NFA ⇒ DFA

Bila ekspresi regularnya cukup sederhana kita bisa saja langsung mengkonstruksi NFA-nya, tanpa melalui NFA ε-move. Misalkan saja NFA tanpa ε-move untuk ER: ab bisa dilihat pada gambar 6.

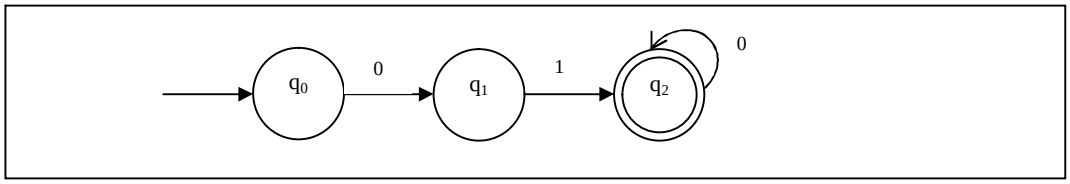



Gambar 6. NFA untuk ER: ab

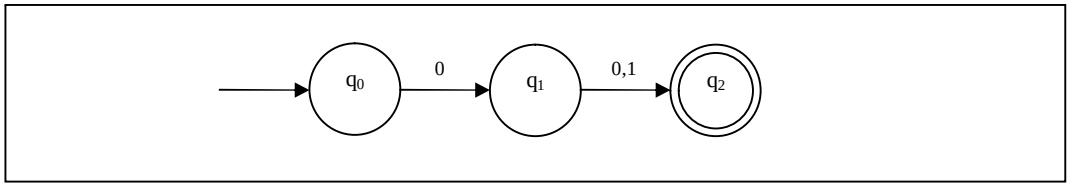



Gambar 7. NFA untuk ER: a ∪ b

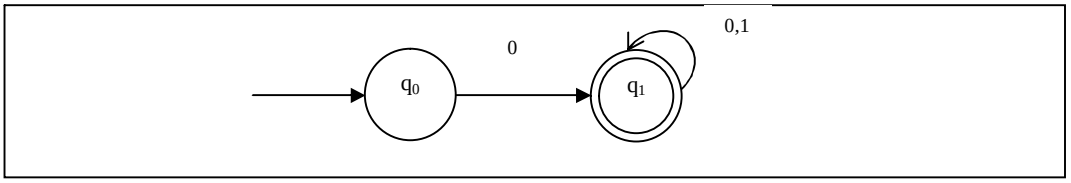
Contoh-contoh lain bisa dilihat pada gambar 8 sampai 15



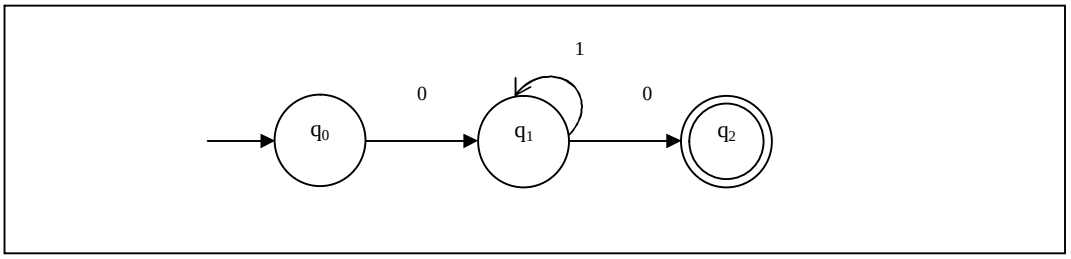

Gambar 8. NFA untuk ER: 010*



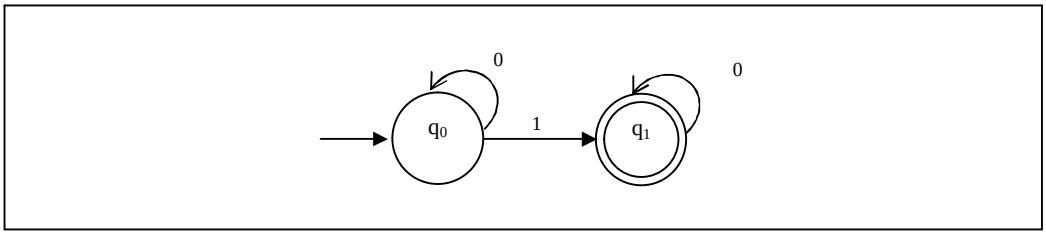

Gambar 9. NFA untuk ER: 0 (1∪0)



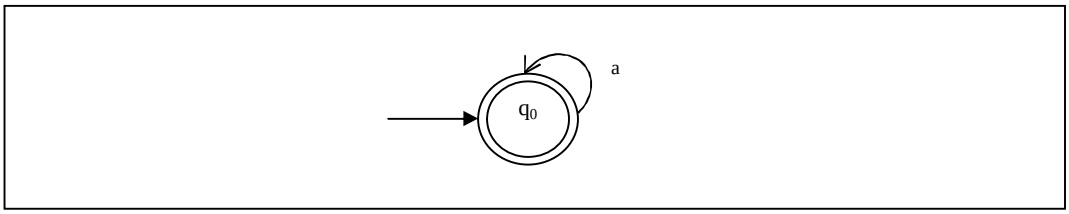

Gambar 10. NFA untuk ER: 0 (1∪0)*



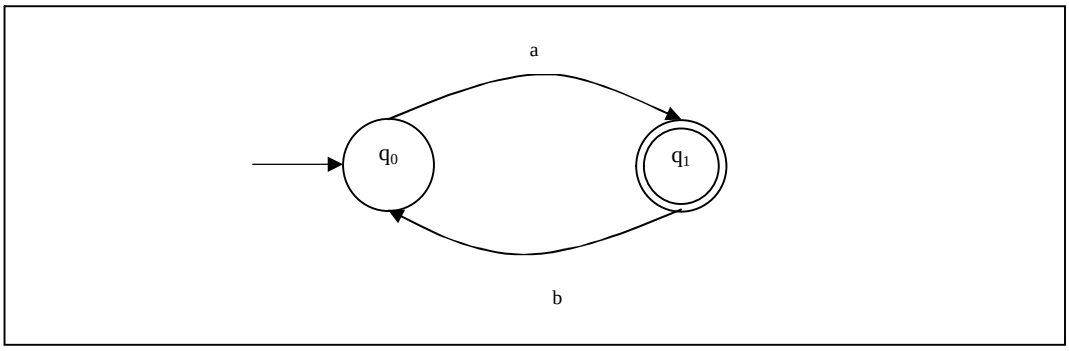

Gambar 11. NFA untuk ER: 01*0



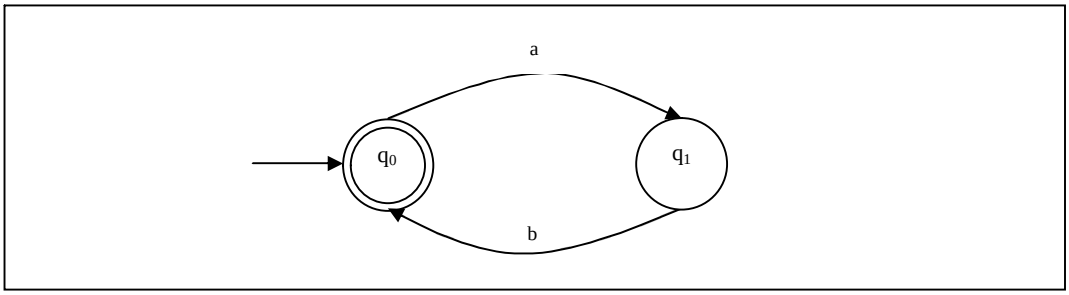

Gambar 12. NFA untuk ER: 0*10




Gambar 13. NFA untuk ER: a*




Gambar 14. NFA untuk ER: a (ba)*



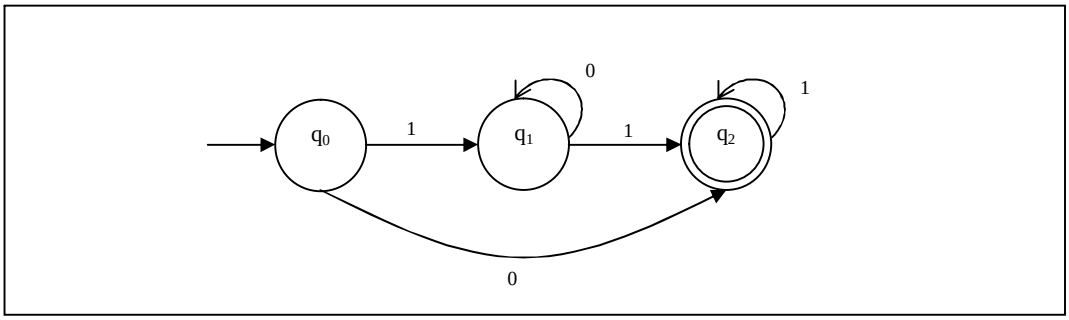

Gambar 15. NFA untuk ER:  (ab)*

Dari sebuah *finite state automata* (NFA atau DFA) kita bisa menentukan ekspresi regular yang diterima oleh otomata bersangkutan. Terdapat langkah-langkah secara formal untuk menentukan ekspresi regular dari suatu *finite state automata*,

tetapi kita bisa juga secara langsung menentukan ekspresi regular-nya dengan mengamati perilaku dari otomata tersebut.

Kita lihat dari gambar 16, input yang menuju pada final state (q2) adalah 0 atau 10*1, pada state q2 menerima input 1 dalam jumlah berapapun (1*) akan tetap di $q_2$. bisa dikatakan mesin itu menerima 01* atau 10*11*, yang dinyatakan dalam ekspresi regular

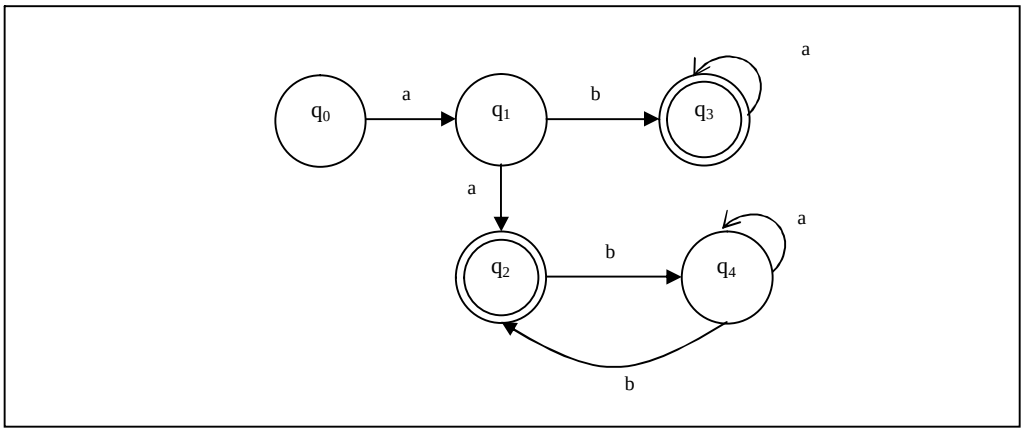
$$01^* \cup 10^*11^*$$




Gambar 16. Mesin FSA

Pada gambar 17 kita lihat final state adalah $q_2$ dan $q_3$. Input menuju $q_3$ adalah ab dengan di $q_3$ bisa menerima a*, sehingga bila digabung menjadi aba*. Input yang menuju $q_2$ adalah aa, selanjutnya dari $q_2$ menerima ba* b kembali ke $q_2$, sehingga bila digabung menjadi aa(ba*b)*. Akhirnya kita peroleh ekspresi regular untuk gambar17: aba*∪aa(ba*b)*, bisa pula dipersingkat menjadi:

$$a(ba^* \cup a(ba^*b)^*)$$

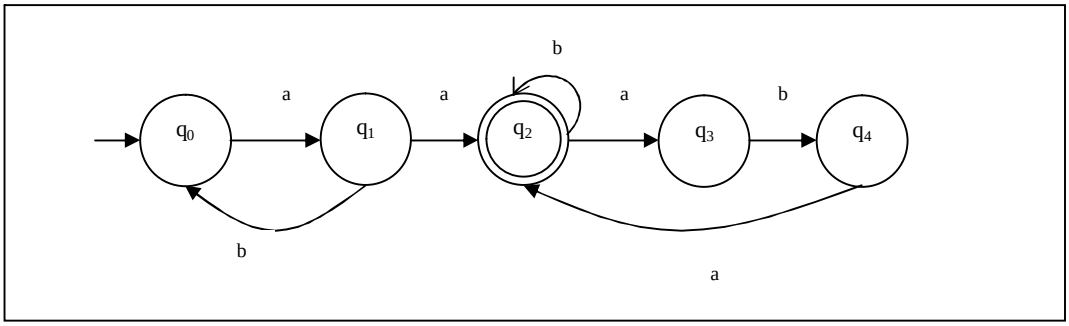



Gambar 17. Mesin FSA

Pada gambar 18 *input* yang menuju final *state* ($q_2$) adalah a(ba*)a, karena dari $q_1$ bila menerima (ba)* kembali ke $q_1$ lagi. Di $q_2$ bila menerima b* tetap di $q_2$. Selanjutnya

menerima (aba) kembali ke $q_2$, dan di $q_2$ bila menerima b* tetap di $q_2$, atau digabung (abab*). Maka ekspresi regularnya:

a(ba)*ab*(abab*)*




Gambar 18. Mesin FSA