

EKSPRESI REGULAR DAN BAHASA REGULAR

OBJEKTIF :

1. Mahasiswa mampu memahami konsep Ekspresi Regular
2. Mahasiswa mampu memahami konsep Pumping Lemma
3. Mahasiswa mampu memahami konsep Ekuivalensi Ekspresi Regular dan Finite Automata

3.1 Ekspresi Regular

3.1.1 Mengenal Ekspresi Regular

Ekspresi Regular disusun dari bahasa regular atau dapat dikatakan bahwa bahasa regular dinyatakan secara sederhana menggunakan ekspresi regular. Dua atau lebih ekspresi regular disebut ekuivalen apabila merupakan bahasa yang sama. Suatu bahasa regular dapat dioperasikan dengan bahasa regular lainnya yang menghasilkan bahasa regular. Operator ekspresi regular tersebut, yaitu gabungan, konkatenasi, dan *kleen*(^{*}).

Suatu bahasa disebut regular apabila terdapat *finite state automata* yang dapat menerimanya. Contoh penerapan dari ekspresi regular adalah pencarian string pada suatu file dan pemberian kondisi pada data yang akan di input.

Contoh:

Jika Σ merupakan himpunan karakter alfabet, maka ekspresi regularnya dapat didefinisikan sebagai berikut:

- Jika \emptyset termasuk anggota R, maka \emptyset adalah ekspresi regular.
- Jika λ adalah anggota R, maka λ adalah ekspresi regular.
- Untuk setiap $x \in \Sigma$, $\{x\}$ adalah anggota dari R, maka x adalah ekspresi regular.
- Jika L dan M adalah anggota R, dan r_1 dan r_2 adalah ekspresi regular, maka:
 - $L \cup M$ anggota R, ekspresi regularnya adalah $(r_1 + r_2)$
 - LM anggota R, ekspresi regularnya adalah $(r_1 r_2)$
 - L^* anggota R, ekspresi regularnya adalah r_1^*

Berikut adalah korespondensi sederhana antara ekspresi regular dan bahasa yang didenotasikannya:

Ekspresi Regular	Bahasa Regular
x , untuk setiap $x \in \Sigma$	$\{x\}$
λ	$\{\lambda\}$
\emptyset	$\{ \}$
(r_1)	$L(r_1)$
r_1^*	$\{L(r_1)^*\}$
$r_1 r_2$	$L(r_1) L(r_2)$
$r_1 + r_2$	$L(r_1) \cup L(r_2)$

Penjelasan:

- Pada ekspresi regular x , untuk setiap x merupakan himpunan dari alfabet maka bahasa regularnya adalah $\{x\}$
- Pada ekspresi regular λ atau himpunan kosong, maka bahasa regularnya adalah $\{\lambda\}$
- Pada ekspresi regular \emptyset atau string kosong, maka bahasa regularnya adalah $\{\emptyset\}$
- Pada ekspresi regular (r_1) , maka bahasa regularnya adalah $L(r_1)$
- Pada ekspresi regular r_1^* , maka bahasa regularnya adalah $\{L(r_1)^*\}$
- Pada ekspresi regular $r_1 r_2$, maka bahasa regularnya adalah $L(r_1) L(r_2)$
- Pada ekspresi regular $r_1 + r_2$, maka bahasa regularnya adalah $L(r_1) \cup L(r_2)$

3.1.2 Operator Ekspresi Regular

Sebuah bahasa regular jika dioperasikan dengan bahasa regular lainnya maka akan menghasilkan bahasa regular. Terdapat beberapa operator yang ada pada ekspresi regular, yaitu:

1. Gabungan

Gabungan dari dua bahasa L dan M dinotasikan dengan $(L \cup M)$, yang memiliki arti gabungan dari L dan M merupakan himpunan dari string yang ada dalam bahasa L maupun M .

Contoh: Terdapat bahasa $L = \{\lambda, ab, abb, aba, abba\}$ dan $M = \{ab, abb, baa, abba\}$. Kemudian diberikan operator gabungan, maka $(L \cup M)$ akan menghasilkan string $\{\lambda, ab, abb, aba, baa, abba\}$

$$(L \cup M) = \{\lambda, ab, abb, aba, baa, abba\}$$

2. Konkatenasi

Konkatenasi dari bahasa L dan M dinotasikan dengan (LM) , yang memiliki arti himpunan string yang dapat dibentuk dengan mengambil string apa saja dari bahasa L dan disambungkan dengan string apa saja dari bahasa M .

Contoh: Terdapat bahasa $L = \{\lambda, ab, abb, aba, abba\}$ dan $M = \{ab, abb, baa, abba\}$. Kemudian dilakukan konkatenasi, maka operasi LM akan menghasilkan $\{ab, abb, baa, abab, abbab, abbaa, \dots\}$

$$(L \cup M) = \{ab, abb, baa, abab, abbab, abbaa, \dots\}$$

3. Kleene Star/Closure

Suatu bahasa apabila dinotasikan dengan kleene star atau $(*)$ memiliki arti himpunan string yang dapat dibentuk dengan mengambil sejumlah string dari anggota himpunan itu sendiri dan memungkinkan juga dengan dilakukannya perulangan.

Contoh: Terdapat bahasa $L = \{\lambda, ab, abb, aba, abba\}$, kemudian diberikan Kleene. Maka L^* dapat menghasilkan string $\{ab, abab, abbb, ababb, \dots\}$

$$L^* = \{ab, abab, abbb, ababb, \dots\}$$

3.1.3 Notasi Ekspresi Regular

Notasi	Arti	Ekspresi Regular	String
* (Kleene)	String dapat muncul 0..n kali	$L(M) = a^*$	$L(M) = \{\epsilon, a, aa, aaa, \dots\}$
+ (Posisi superscript)	String muncul 1..n kali	$L(M) = a^+$	$L(M) = \{a, aa, aaa, \dots\}$
+ atau \cup	Gabungan atau <i>Union</i>	$L(M) = a + b$	$L(M) = \{a, b\}$
.	Penyambungan atau Konkatenasi	$L(M) = a.b$	$L(M) = \{ab\}$

Penjelasan:

- Perbedaan dari notasi *kleen* dan *postscript* adalah pada notasi kleen string yang dihasilkan dapat berupa string kosong sedangkan pada notasi superscript tidak ada string kosong atau minimal muncul 1 kali.
- Pada ekspresi regular notasi gabungan, string a dan string b merupakan string yang berbeda dan tidak bisa digabungkan, misalnya seperti ab dan $abab$.
- Pada konkatenasi tidak harus menggunakan titik ($.$).

3.1.4 Menentukan Ekspresi Regular

Tentukan String dari Ekspresi Regular berikut!

1. $ER = ab^*$

Hasil string yang dapat dibuat adalah $ab, abb, abbb, abbbb, \dots$ dst . String b bisa tidak muncul atau juga dapat muncul berkali-kali.

2. $ER = ab^*cc$

Hasil string yang dapat dibuat adalah $abcc, acc, abbcc, abbbcc, \dots$ dst. String b bisa tidak muncul atau juga dapat muncul berkali-kali.

3. $ER = 10^+$

Hasil string yang dapat dibuat adalah $10, 100, 1000, \dots$ dst.

4. $ER = 110^+ 1^*$

Hasil string yang dapat dibuat adalah $110, 1101, 11001, 110011, \dots$ dst.

Tentukan Ekspresi Regularnya!

1. Tuliskan ekspresi regular dari bahasa $L = \{a^n b^m; n \geq 1, m \geq 1\}$

Jawab:

$L = \{a^n b^m; n \geq 1, m \geq 1\}$ dapat menghasilkan string $L = \{ab, aabb, aaabbb, abbb, aab, abb, \dots\}$

String ab dihasilkan dari a^1 dan b^1 . String $aabb$, dihasilkan dari a^2 dan b^2 dan seterusnya. Dapat dilihat minimal munculnya ab adalah 1x yaitu pada string pertama ab . Setelah membuat string yang dapat dibentuk oleh bahasa L , barulah kita dapat menentukan ekspresi regularnya. Ekspresi regular yang dapat dibentuk oleh bahasa L tersebut adalah $ER = aa^*bb^*$ atau a^+b^+

Penjelasan: Kedua ekspresi sama-sama menghasilkan string bahasa L tersebut.



2. Tuliskan ekspresi regular dari bahasa $L = \{a^{2n} b^{2m+2}; n \geq 0, m \geq 0\}$

Jawab:

$L = \{a^{2n} b^{2m+2}; n \geq 0, m \geq 0\}$ dapat menghasilkan string $L = \{bb, aabb, aabbbb, aaaabb, \dots\}$, Ekspresi regular yang dapat dibentuk oleh bahasa L tersebut adalah $ER = (aa)^* (bb)^* bb$

Penjelasan:

- bb , dihasilkan dari apabila nilai n dan m adalah 0, seperti berikut $(aa)^0 (bb)^0 bb$.
- $aabb$, dihasilkan apabila nilai $n = 1$ dan $m = 0$, seperti berikut $(aa)^1 (bb)^0 bb$.
- $aabbbb$, dihasilkan apabila nilai $n = 1$ dan $m = 1$, seperti berikut $(aa)^1 (bb)^1 bb$.
- $aaaabb$, dihasilkan apabila nilai $n = 2$ dan $m = 0$, seperti berikut $(aa)^2 (bb)^0 bb$.

3.2 Pumping Lemma

Misalnya, L adalah sebuah bahasa regular. Terdapat sebuah bilangan m yang disebut sebagai *pumping length* yang sedemikian rupa sehingga setiap *string* w pada L atau ($w \in L$) dengan $|w| \geq m$ (panjang string w lebih dari sama dengan m) harus dapat dibagi menjadi 3 bagian, $w = xyz$. Dengan ketentuan:


- Apabila ketiga ketentuan tersebut terpenuhi, maka L dapat dikatakan sebagai bahasa regular.

Terdapat bahasa $L = \{ab^n ; n > 0\}$. Buktikan apakah bahasa L termasuk dalam bahasa reguler atau tidak!




Tentukan terlebih dahulu string dari bahasa L, string yang didapat adalah $L = \{ab, abb, abbb, \dots\}$, kemudian

- Pengujian:

- Ketentuan 1: $y \neq \text{empty} \rightarrow y = b$ <<<<<<<<<< **Terpenuhi** ✓
Pada ketentuan 1, nilai y tidak boleh kosong, disini kita sudah menentukan nilai y adalah b , yang berarti y tidak kosong/empty. Sehingga ketentuan pertama terpenuhi.

- Ketentuan 3: Untuk semua $i \geq 0$, string xy^iz harus merupakan string dari bahasa L
- Terpenuhi 

Sebagai penjelasan:

- o Jika $i = 0$ maka terbentuk string $ab \in L$ 
(Apabila $i = 0$, maka akan menghasilkan string ab dan string ab tersebut termasuk ke dalam bahasa L)
- o Jika $i = 2$ maka terbentuk string $abbb \in L$ 
(Apabila $i = 2$, maka akan menghasilkan string $abbb$ dan string $abbb$ tersebut termasuk ke dalam bahasa L)
- o Jika $i = 5$ maka terbentuk string $abbbbb \in L$ 
(Apabila $i = 5$, maka akan menghasilkan string $abbbbb$ dan string $abbbbb$ tersebut termasuk ke dalam bahasa L)

Karena semua ketentuan terpenuhi, maka $L = \{ab^n \mid n \geq 0\}$ adalah bahasa reguler.

Contoh 2:


Terdapat bahasa $L = \{a^n b^n ; n \geq 1\}$. Buktikan apakah bahasa L termasuk dalam bahasa reguler atau tidak!

Jawab:

Tentukan terlebih dahulu string dari bahasa L, string yang didapat adalah $L = \{ab, aabb, aaabbb, \dots\}$, kemudian

- Asumsikan $m = 4$, (Pada asumsi ini dapat menggunakan nilai berapa saja, tetapi pada asumsi ini akan menggunakan nilai m atau *pumping length*nya adalah 4)
- Tentukan string $w \in L$ dimana $|w| \geq m \rightarrow$; abb , (Tentukan string w yang terdapat pada L , dimana panjang dari string w harus lebih besar sama dengan m , disini kita akan gunakan string $aabb$)
- $w = xyz \rightarrow x = a$; $y = ab$; $z = b$, (Kemudian bagi string w menjadi 3 bagian, yaitu x , y , dan z . x memiliki string a , y memiliki string ab , dan z memiliki string b)

Pengujian:

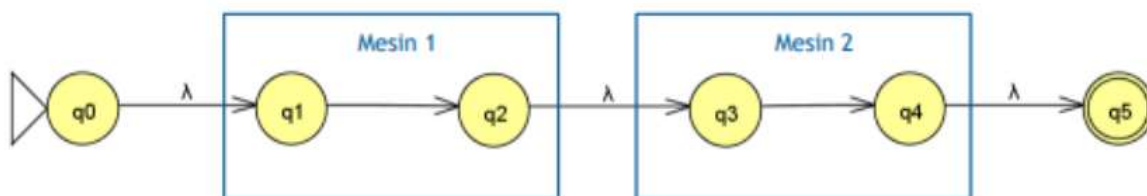
- Ketentuan 1: $y \neq \text{empty} \rightarrow ; y = ab \llllllllll$ **Terpenuhi** 
- Pada ketentuan 1, nilai y tidak boleh kosong, disini kita sudah menentukan nilai y adalah ab , yang berarti y tidak kosong/empty. Sehingga ketentuan pertama terpenuhi.

- Ekspresi regular \emptyset menandakan bahasa \emptyset . Tidak ada string pada bahasa ini, empty string pun tidak.



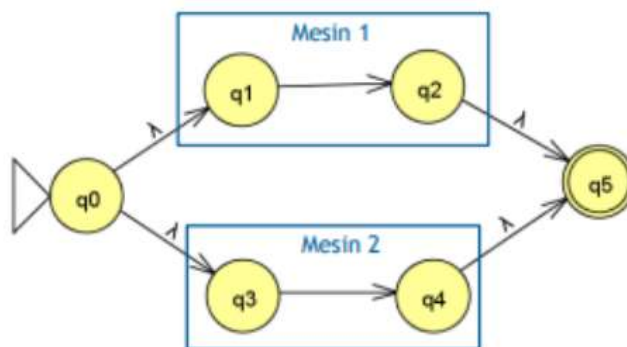
NFA \emptyset

- Untuk konkatenasi, string dalam L_1 diikuti oleh string dalam L_2 . Jadi untuk NFA nya disambungkan seperti finite automata berikut: (NFA mesin 1 disambungkan dengan input kosong ke NFA mesin 2)



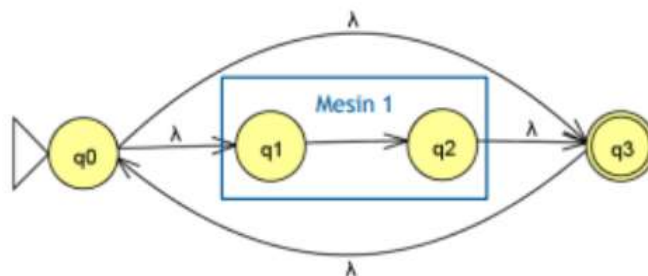
NFA L_1L_2

- Untuk operator "+" menandakan gabungan (operator or) pada ekspresi regular, sehingga NFA nya memiliki 2 atau lebih jalur.



NFA $L_1 \cup L_2$

- Untuk operator *Kleene Star* (*) menandakan nol atau lebih aplikasi ekspresi regular, sehingga terdapat perulangan (loop) pada NFA.

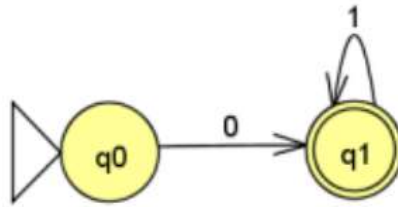


NFA *Kleene Star*

Contoh 1:

Diketahui ekspresi regular \emptyset^* . Rancanglah finite automata dari ER tersebut!

Jawab:

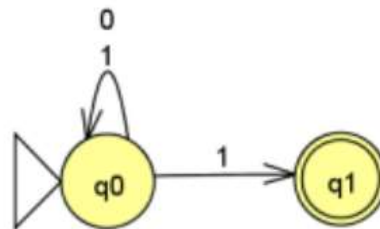


Penjelasan: Perhatikan operatornya! Terdapat operator bintang (*), sehingga 1 akan terus melakukan perulangan atau *looping*. Dari q0 akan melakukan input 0 ke q1, kemudian pada q1 terdapat perulangan dengan input 1.

Contoh 2:

Rancanglah finite automata dari ekspresi regular $(0+1)^*1$

Jawab:

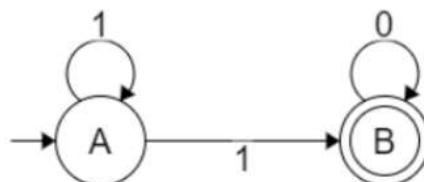


Penjelasan: Perhatikan tanda kurung dan operatornya! Pada contoh ini, yang dioperasikan dengan bintang (*) adalah 0 dan 1 atau $0+1$, yang berarti 0 dan 1 akan melakukan perulangan. 0 dan 1 akan melakukan perulangan pada q0 kemudian akan melakukan input 1 ke q1.

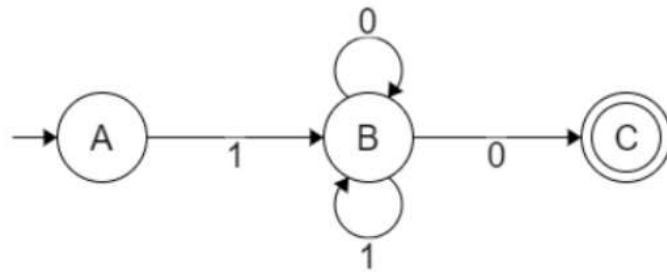
Contoh 3: 011 (Memiliki 4 State)



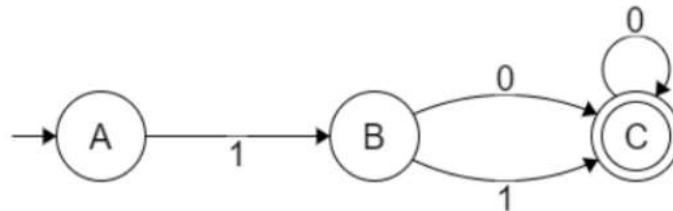
Contoh 4: 1^*10^*



Contoh 5: $1(0+1)^*0$, Untuk transisi $(0+1)^*$ dibuat menjadi 2 transisi pada state yang sama dan cara ini **khusus** digunakan pada soal di **Website Praktikum CodeRunner**. Apabila menggunakan **JFLAP**, ikuti **Contoh 2**.



Contoh 6: $1(01)0^*$, Transisi (01) memiliki tujuan yang sama yaitu ke State C.



3.3.2 Finite Automata ke Ekspresi Regular

Untuk mengubah finite automata ke ekspresi regular, terdapat algoritma yang harus terpenuhi, yaitu:

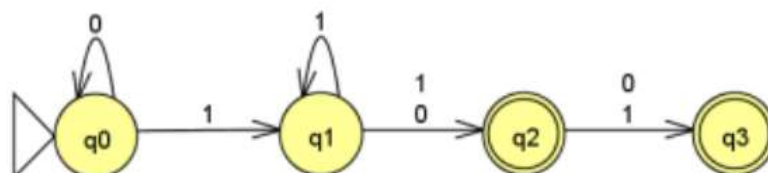
1. Untuk setiap final state q , hilangkan final state lainnya dengan menjadikan final state tersebut menjadi non final state kemudian ditransisikan dengan λ .
2. Hilangkan state dari NFA satu per satu, hingga tersisa satu start state dan satu final state.
3. Baca ekspresi regular akhir dari dua state automata tersebut.

Setiap bahasa yang didefinisikan oleh ekspresi regular, dapat juga didefinisikan oleh finite automata apabila syarat berikut terpenuhi, yaitu:

- Hanya terdapat satu final state
- Tidak ada transisi ke start state
- Tidak ada transisi ke final state

Contoh 1:

Terdapat NFA sebagai berikut, kemudian tentukan ekspresi regularnya!

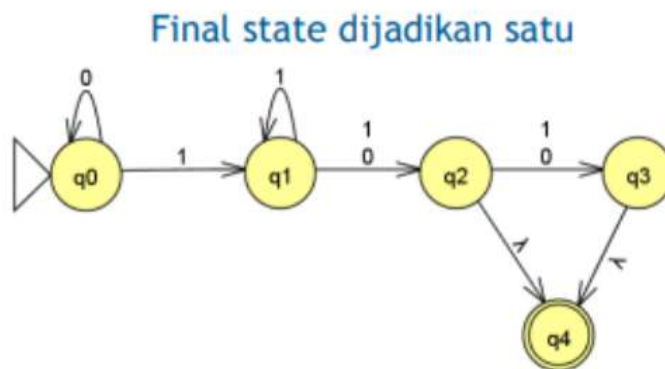


Jawab:

- Langkah 1:
Karena pada FA tersebut tersebut terdapat 2 final state, yaitu q_2 dan q_3 . Maka, diperlukan perubahan pada FA tersebut sehingga mempunyai 1 final state dengan cara

menambahkan state baru dan memberikan transisi input kosong ke state baru tersebut dari kedua final state awal.

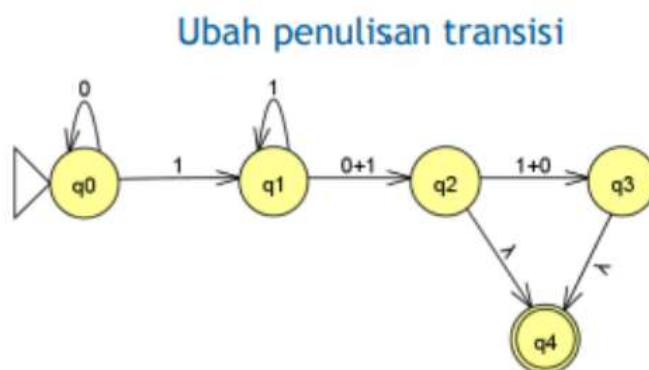
Sehingga membentuk FA seperti berikut:



Terdapat state baru yaitu q4 dan jadikan q4 tersebut menjadi final state baru. Sehingga state q2 dan q3 tidak menjadi final state. Lalu transisikan menggunakan input kosong ke q4.

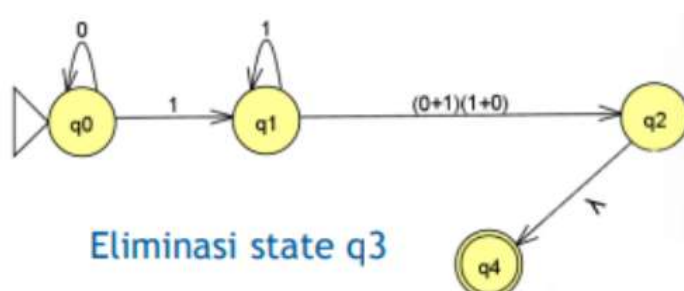
- Langkah 2:

Selanjutnya lakukan perubahan transisi antara q1 dengan q2 dan transisi antara q2 dan q3, karena terdapat 2 transisi di antara kedua state yaitu transisi dengan input 0 dan transisi dengan input 1. Untuk melakukan perubahan tersebut dapat menggunakan operator gabungan atau (+). Sehingga bentuk FAny akan menjadi seperti berikut, dimana input 1 dan input 0 digabung menjadi $0+1$ dan $1+0$ (keduanya sama saja).



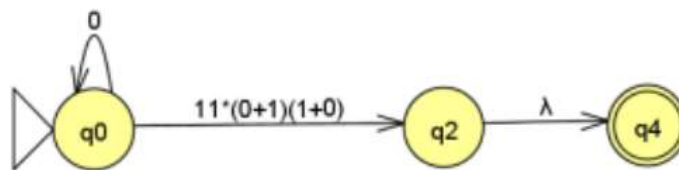
- Langkah 3:

Pada tahap selanjutnya, diperlukan eliminasi state. Eliminasi dimulai dari state q3. Setelah state q3 di eliminasi, maka bentuk FAny menjadi seperti berikut, sehingga transisi $0+1$ akan disambungkan dengan transisi $1+0$.



- Langkah 4:

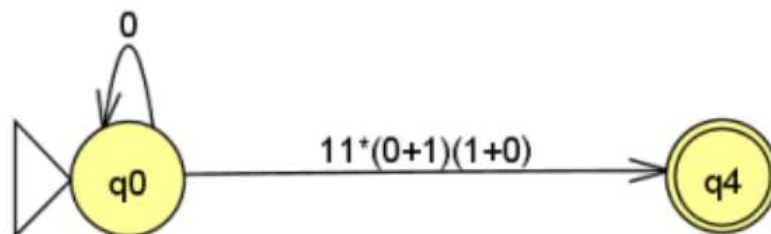
Selanjutnya akan melakukan eliminasi pada state q1. Berikut adalah bentuk FA setelah q1 dieliminasi. Pada state q1 sebelumnya, terdapat perulangan dengan input 1 oleh karena itu ekspresi regularnya menggunakan tanda bintang (*).



Eliminasi state q1

- Langkah 5:

Langkah terakhir adalah melakukan eliminasi pada state q2, sehingga bentuk FAny menjadi seperti berikut.

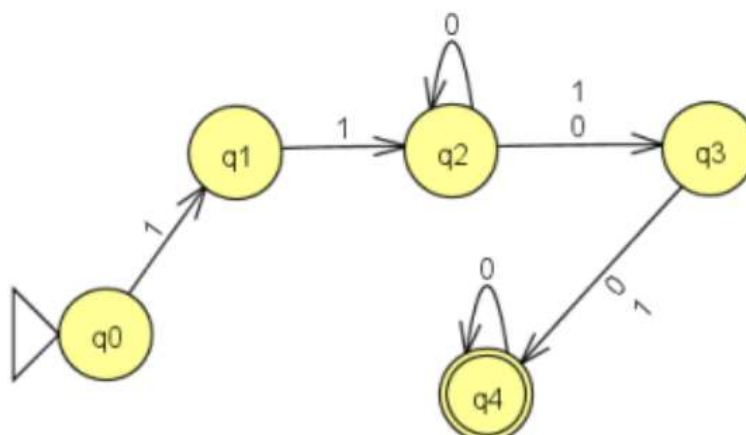


Eliminasi state q2

Setelah melakukan eliminasi state satu per satu dan menyisakan satu state awal dan state akhir, barulah ekspresi regularnya dapat ditentukan dari NFA tersebut. Karena pada state q0 terdapat perulangan 0, maka bentuk ekspresi regular dari NFA tersebut adalah $0^*11^*(0+1)(1+0)$

Contoh 2:

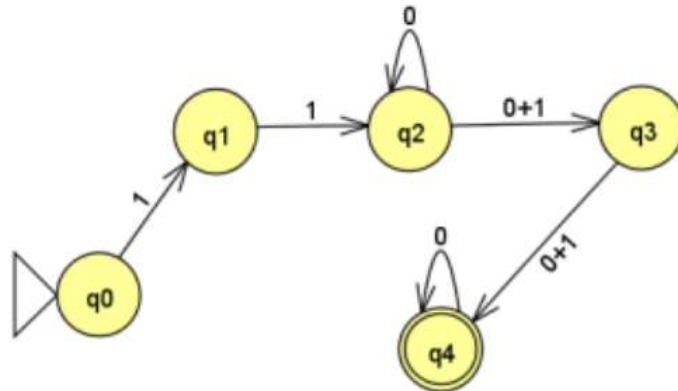
Konversikan NFA berikut menjadi ekspresi regular!



Jawab:

- Langkah 1:

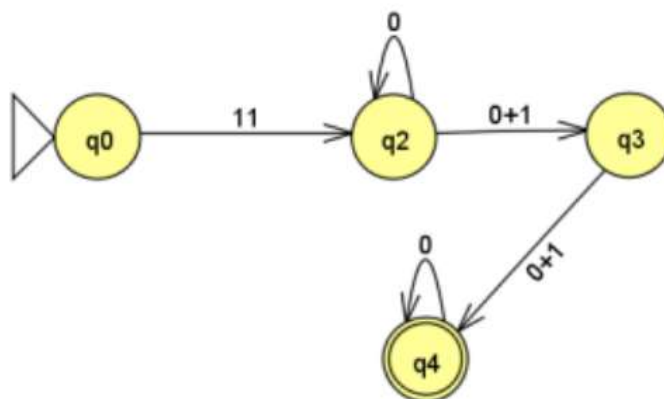
Karena hanya terdapat satu final state, maka dapat langsung dilakukan eliminasi state satu per satu. Sebelum melakukan eliminasi, diperlukan perubahan pada transisi yang memiliki 2 input seperti pada transisi q2 antara q3 dan transisi antara q3 dan q4 menggunakan operator gabungan atau (+) sehingga bentuk FAnyanya berubah menjadi seperti berikut:



Ubah penulisan transisi

- Langkah 2:

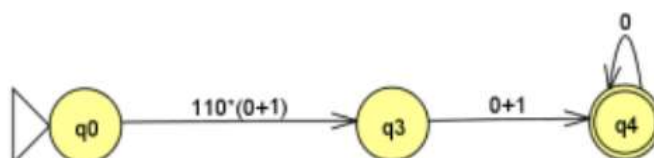
Kemudian lakukan eliminasi pada state q1, sehingga bentuknya berubah menjadi seperti berikut:



Eliminasi state q1

- Langkah 3:

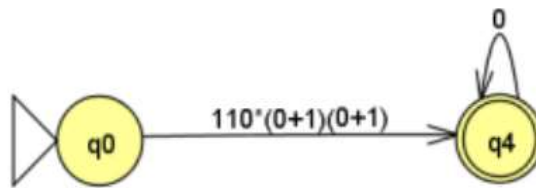
Lakukan eliminasi pada state q2.



Eliminasi state q2

- Langkah 4:

Selanjutnya eliminasi pada state q_3 . Setelah dilakukan eliminasi pada state q_3 , bentuk akhir dari finite automatanya adalah seperti berikut:



Eliminasi state q_3

Dari bentuk akhir finite automata tersebut dapat ditentukan ekspresi reguleranya, yaitu $110^*(0+1)(0+1)0^*$

3.4 RANGKUMAN

- Suatu bahasa yang didefinisikan oleh ekspresi regular, didefinisikan juga oleh finite automata.
- Ekspresi regular dapat dibentuk dari finite automata, begitu juga sebaliknya.
- Ekspresi regular dapat terbentuk dari finite automata apabila hanya ada satu final state, tidak ada transisi ke start state, tidak ada transisi dari final state.

REFERENSI

- [1] Harya Bima Dirgantara. 2016. Teori Bahasa dan Automata: Merancang Automaton Dengan JFLAP. Jakarta: Penerbit Teknosain.
- [2] John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. 2007. Teori Bahasa dan Otomata Edisi Kedua. Yogyakarta: Penerbit Andi.