



# **TEORI BAHASA AUTOMATA**

## DAFTAR ISI

BAB I .....	5
KONSEP DASAR AUTOMATA .....	5
1. Konsep Dasar .....	5
1.2 Teori Himpunan .....	5
1.3 String .....	8
Definisi .....	8
1.3 Operasi Dasar String.....	8
1.4 Sifat-sifat operasi String .....	10
1.5 Kompilasi.....	11
LATIHAN SOAL.....	16
BAB II .....	17
KONSEP BAHASA AUTOMATA.....	17
2.1 Konsep Bahasa dan Automata .....	17
2.1.1 Automata .....	17
2.1.2 Teori Bahasa.....	17
2.1.3 Hubungan Otomata dan Bahasa Formal.....	18
2.1.4 Bahasa Natural dan Bahasa Formal : .....	18
2.1.5 Elemen Bahasa Formal:.....	19
2.2 Grammar dan Bahasa.....	20
2.2.2 Simbol .....	20
2.2.3 Aturan Produksi .....	21
2.2.5 Hirarky Chomsky .....	22
LATIHAN SOAL.....	25
BAB III .....	27
FINITE STATE AUTOMATA .....	27
3.1 FSA.....	27
3.1.1 Konversi dari Tabel Transisi ke Diagram Transisi .....	29
3.2 Non Deterministic Finite Automata .....	30
LATIHAN SOAL.....	32
BAB IV .....	34
REDUKSI FINITE STATE AUTOMATA .....	34
4.1 Reduksi Jumlah State pada Finite State Automata .....	34
4.2 Ekuivalensi Non-Deterministic Finite Automata ke Deterministic Finite Automata .....	38
SOAL LATIHAN .....	42
BAB V.....	46

NON DETERMINISTIC FINITE AUTOMATA DENGAN $\epsilon$ -MOVE .....	46
5.1 NFA.....	46
5.2 $\epsilon$ – Closure .....	46
5.3 Ekuivalensi Non Deterministic Finite Automata dengan $\epsilon$ – Move ke Non Deterministic Finite Automata tanpa $\epsilon$ – Move .....	47
Penggabungan dan Konkatenasi Finite State Automata .....	53
LATIHAN SOAL.....	55
BAB VI.....	57
EKSPRESI REGULAR .....	57
6.1 ER (Ekspresi Regular).....	57
6.1.1 Notasi Ekspresi Regular.....	57
6.2 Hubungan Ekspresi Regular dan Finite State Automata .....	58
6.3 Aturan Produksi untuk suatu Tata Bahasa Regular.....	60
6.4 Finite State Automata untuk Suatu Tata Bahasa Regular .....	64
LATIHAN SOAL.....	66
BAB VII.....	68
TATA BAHASA BEBAS KONTEKS .....	68
7.3 Ambiguitas .....	70
7.4 Penyederhanaan Tata Bahasa Bebas Konteks .....	71
LATIHAN SOAL.....	83
BAB VIII.....	86
TATA BAHASA BEBAS KONTEKS .....	86
8.1 Bentuk Normal Chomsky .....	86
8.1.1 Pembentuk Bentuk Normal Chomsky .....	86
LATIHAN SOAL.....	91
BAB IX.....	92
MESIN LAIN .....	92
9.1 Mesin Moore.....	92
9.2 Mesin Mealy.....	93
9.3 Ekuivalensi Mesin Moore dan Mesin Mealy .....	94
9.3.1 Penghilangan Rekursif kiri .....	96
9.4 Bentuk Normal Greibach.....	100
SOAL LATIHAN.....	105



# BAB I

## KONSEP DASAR AUTOMATA

### 1. Konsep Dasar

Teori bahasa dan otomata merupakan salah satu mata kuliah yang wajib di jurusan teknik informatika maupun ilmu komputer. Teori bahasa dan otomata merupakan mata kuliah yang cenderung bersifat teoritis tidak memuat hal-hal yang ‘praktis’ untuk diterapkan langsung dalam praktik. Manfaat langsung dari mata kuliah teori bahasa dan otomata akan kita dapatkan ketika mempelajari mata kuliah Teknik Kompilasi.

Bahasa di dalam kamus adalah suatu sistem yang meliputi pengekspresian gagasan, fakta, konsep, termasuk sekumpulan simbol-simbol dan aturan untuk melakukan manipulasinya. Bahasa bisa juga disebut sebagai rangkaian simbol-simbol yang mempunyai makna.

### 1.2 Teori Himpunan

Definisi sebuah himpunan adalah kumpulan obyek atau simbol yang memiliki sifat yang sama. Anggota himpunan disebut elemen

Contoh:

Terdapat sebuah himpunan  $X = \{1, 2, 4\}$ , maka:

$1 \in X$  merupakan elemen dari himpunan  $X$

$3 \notin X$  bukan elemen dari himpunan  $X$

#### A. Himpunan Sama

Himpunan dikatakan sama bila memuat elemen - elemen yang sama.

Contoh :

Terdapat beberapa himpunan,

$$X = \{1, 2, 4\}$$

$$Y = \{4, 1, 2\}$$

$$Z = \{1, 2, 3\}$$

Maka,  $X = Y$ ,  $X \neq Z$

#### B. Himpunan Bagian (Subset)

Himpunan bagian adalah jika semua elemen dari himpunan  $A$  adalah elemen dari himpunan  $B$ .

Contoh :

Terdapat beberapa himpunan,

$$A = \{1, 2, 3\}$$

$$B = \{1, 2, 3, 4\}$$

$$C = \{1, 2, 3\}$$

Maka,

$$A \subseteq B \text{ dan } C \subseteq B, \text{ maka } A = C$$

### C. Himpunan Kosong

Himpunan kosong atau null ( $\emptyset$ ) merupakan bagian dari semua himpunan.

Contoh :

Terdapat himpunan,

$$A = \{1, 2, 3\}$$

$$B = \{1, 2, 3, 4\}$$

$$C = \{1, 2, 3\}$$

Maka,

$$\emptyset \subseteq A$$

$$\emptyset \subseteq B$$

$$\emptyset \subseteq C$$

### D. Operasi – operasi himpunan

#### 1. Gabungan (Union), dinyatakan dengan $\cup$

Misal,  $A = \{1, 2, 3\}$

$$B = \{2, 4\}$$

Jika  $P = A \cup B$ , maka

$$P = \{x \mid x \in A \text{ atau } x \in B\}$$

$$P = \{1, 2, 3, 4\}$$

Bisa dianalogikan dengan penjumlahan

$$A \cup B = A \vee B = A + B$$

#### 2. Irisan (*Intersection*), dinyatakan dengan $\cap$

Misal,  $A = \{1, 2, 3\}$

$$B = \{2, 4\}$$

$$C = \{10, 11\}$$

Jika  $P = A \cap B$ , maka

$$P = \{x \mid x \in A \text{ dan } x \in B\}$$

$$P = \{2\}$$

$$A \cap C = \emptyset, \text{ disebut saling lepas (disjoint)}$$

Bisa dianalogikan dengan perkalian

$$A \cap B = A \wedge B = A \cdot B$$

3. Komplemen, adalah semua elemen yang tidak menjadi elemen himpunan tersebut. Semua himpunan yang dibahas diasumsikan merupakan bagian dari suatu himpunan semesta (Universal).

Jadi komplemen  $A = \neg A = U - A$

Misal  $A = \{1, 2, 3\}$

$B = \{2, 4\}$

$C = \{10, 11\}$

Maka,  $A - B = \{1, 3\}$

$B - C = \emptyset$

E. Beberapa ketentuan pada operasi himpunan

- $\emptyset \cup A = A$
- $\emptyset \cap A = \emptyset$
- $A \subseteq B \rightarrow A \cap B = A$
- $A \subseteq B \rightarrow A \cup B = B$
- $A \cap A = A$
- $A \cup A = A$
- Komunikatif,  $A \cup B = B \cup A$   
 $A \cap B = B \cap A$
- Asosiatif,  $A \cup (B \cup C) = (A \cup B) \cup C$   
 $A \cap (B \cap C) = (A \cap B) \cap C$
- Distributif,  $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$   
 $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
- Hukum DeMorgan,  $A - B = A \cap \neg B$   
 $\neg(A \cap B) = \neg A \cup \neg B$
- $\neg(A \cup B) = \neg A \cap \neg B$
- $\neg(\neg A) = A$
- $\neg \emptyset = U$
- $\neg U = \emptyset$

### 1.3 String

#### Definisi

Simbol adalah sebuah entitas abstrak. Sebuah huruf atau sebuah angka adalah contoh simbol.

String adalah deretan terbatas (finite) simbol-simbol. Sebagai contoh, jika a, b, dan c adalah tiga buah simbol maka abcb adalah sebuah string yang dibangun dari ketiga simbol tersebut.

Alfabet adalah himpunan hingga (finite set) simbol-simbol

#### 1. Panjang String

Definisi 1: sebuah string dengan panjang n yang dibentuk dari himpunan A adalah barisan dari n simbol. Misalnya  $a_1a_2...a_n$  ;  $a_i \in A$  Jika w adalah sebuah string maka panjang string dinyatakan sebagai  $|w|$  dan didefinisikan sebagai cacahan (banyaknya) simbol yang menyusun string tersebut. Sebagai contoh, jika  $w = abcb$  maka  $|w| = 4$

#### 2. String hampa

Definisi 2 : String kosong/ String hampa (null string), dilambangkan dengan  $\epsilon$  (atau  $\wedge$ ) adalah untaian dengan panjang 0 dan tidak berisi apapun.

String hampa adalah sebuah string dengan nol buah simbol. Maka panjang string hampa  $|\epsilon| = 0$ .

String hampa dapat dipandang sebagai simbol hampa karena keduanya tersusun dari nol buah simbol.

### 1.3 Operasi Dasar String

Diberikan dua string :  $x = abc$ , dan  $y = 123$

- Prefik string w adalah string yang dihasilkan dari string w dengan menghilangkan nol atau lebih simbol-simbol paling belakang dari string w tersebut.

Contoh : abc, ab, a, dan  $\epsilon$  adalah semua Prefix(x)

- ProperPrefix string w adalah string yang dihasilkan dari string w dengan menghilangkan satu atau lebih simbol-simbol paling belakang dari string w tersebut.

Contoh : ab, a, dan  $\epsilon$  adalah semua ProperPrefix(x)

- Postfix (atau Sufix) string w adalah string yang dihasilkan dari string w dengan menghilangkan nol atau lebih simbol-simbol paling depan dari string w tersebut.

Contoh : abc, bc, c, dan  $\epsilon$  adalah semua Postfix(x)



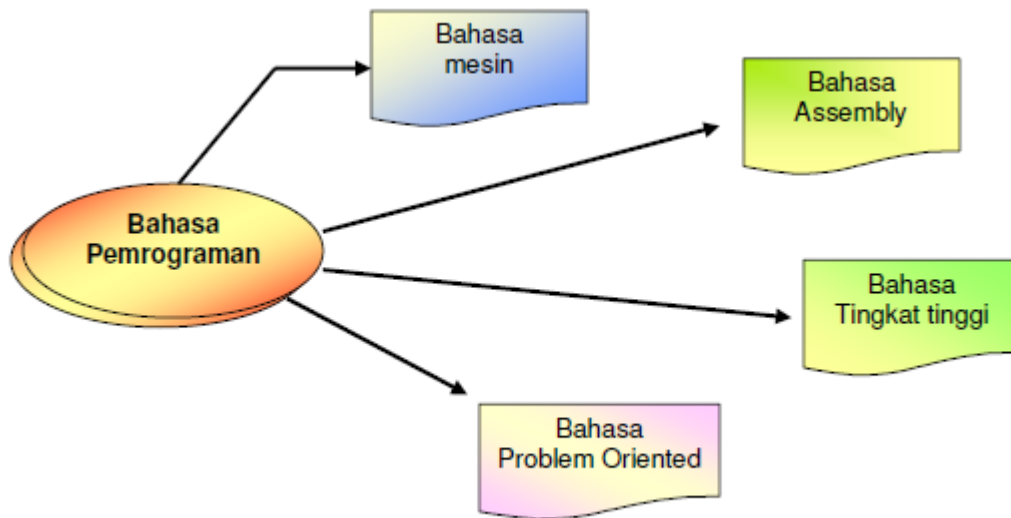
- ProperPostfix (atau PoperSufix) string w adalah string yang dihasilkan dari string w dengan menghilangkan satu atau lebih simbol-simbol paling depan dari string w tersebut. Contoh : bc, c, dan \_ adalah semua ProperPostfix(x)
- Head string w adalah simbol paling depan dari string w.  
Contoh : a adalah Head(x)
- Tail string w adalah string yang dihasilkan dari string w dengan menghilangkan simbol paling depan dari string w tersebut.  
Contoh : bc adalah Tail(x)
- Substring string w adalah string yang dihasilkan dari string w dengan menghilangkan nol atau lebih simbol-simbol paling depan dan/atau simboisimbol paling belakang dari string w tersebut.  
Contoh : abc, ab, bc, a, b, c, dan \_ adalah semua Substring(x)
- ProperSubstring string w adalah string yang dihasilkan dari string w dengan menghilangkan satu atau lebih simbol-simbol paling depan dan/atau simbol - simbol paling belakang dari string w tersebut.  
Contoh : ab, bc, a, b, c, dan \_ adalah semua Substring(x)
- Subsequence string w adalah string yang dihasilkan dari string w dengan menghilangkan nol atau lebih simbol-simbol dari string w tersebut.  
Contoh : abc, ab, bc, ac, a, b, c, dan \_ adalah semua Subsequence(x)
- ProperSubsequence string w adalah string yang dihasilkan dari string w dengan menghilangkan satu atau lebih simbol-simbol dari string w tersebut.  
Contoh : ab, bc, ac, a, b, c, dan \_ adalah semua Subsequence(x)
- Concatenation adalah penyambungan dua buah string. Operator concatenation adalah concate atau tanpa lambang apapun.  
Contoh :  $\text{concate}(xy) = xy = \text{abc123}$   
Alternation adalah pilihan satu di antara dua buah string. Operator alternation adalah alternate atau |. Contoh :  $\text{alternate}(xy) = x|y = \text{abc atau 123}$
- Kleene Closure :  $x^* = \_ |x|xx|xxx|\dots = \_ |x|x_2|x_3|\dots$   
o Positive Closure :  $x_+ = x|xx|xxx|\dots = x|x_2|x_3|\dots$

#### 1.4 Sifat-sifat operasi String

- Tidak selalu berlaku :  $x = \text{Prefix}(x)\text{Postfix}(x)$
- Selalu berlaku :  $x = \text{Head}(x)\text{Tail}(x)$
- Tidak selalu berlaku :  $\text{Prefix}(x) = \text{Postfix}(x)$  atau  $\text{Prefix}(x) \_ \text{Postfix}(x)$
- Selalu berlaku :  $\text{ProperPrefix}(x) \_ \text{ProperPostfix}(x)$
- Selalu berlaku :  $\text{Head}(x) \_ \text{Tail}(x)$
- Setiap  $\text{Prefix}(x)$ ,  $\text{ProperPrefix}(x)$ ,  $\text{Postfix}(x)$ ,  $\text{ProperPostfix}(x)$ ,  $\text{Head}(x)$ , dan
- $\text{Tail}(x)$  adalah  $\text{Substring}(x)$ , tetapi tidak sebaliknya
- Setiap  $\text{Substring}(x)$  adalah  $\text{Subsequence}(x)$ , tetapi tidak sebaliknya
- Dua sifat aljabar concatenation :
  - Operasi concatenation bersifat asosiatif :  $x(yz) = (xy)z$
  - Elemen identitas operasi concatenation adalah  $\_$  :  $\_x = x\_ = x$
- Tiga sifat aljabar alternation :
  - Operasi alternation bersifat komutatif :  $x|y = y|x$
  - Operasi alternation bersifat asosiatif :  $x|(y|z) = (x|y)|z$
  - Elemen identitas operasi alternation adalah dirinya sendiri :  $x|x = x$
- Sifat distributif concatenation terhadap alternation :  $x (y|z) = xy|xz$
- Beberapa kesamaan :
  - a. Kesamaan ke-1 :  $(x^*)^* = (x^*)$
  - b. Kesamaan ke-2 :  $\_|x+ = x+ \_ = x^*$
  - c. Kesamaan ke-3 :  $(x|y)^* = \_|x|y|xx|yy|xy|yx|\dots =$  semua string yang merupakan concatenation dari nol atau lebih  $x$ ,  $y$ , atau keduanya.

## 1.5 Kompilasi

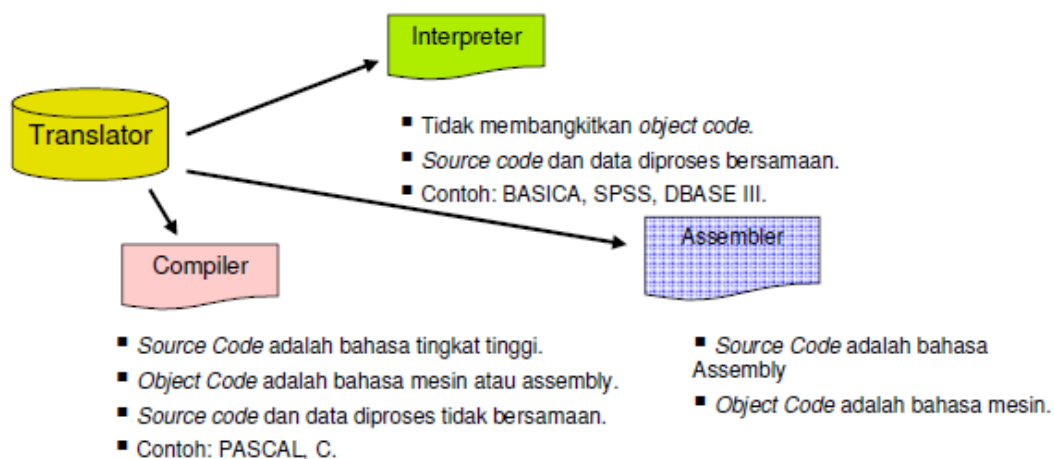
### A. Bahasa Pemrograman



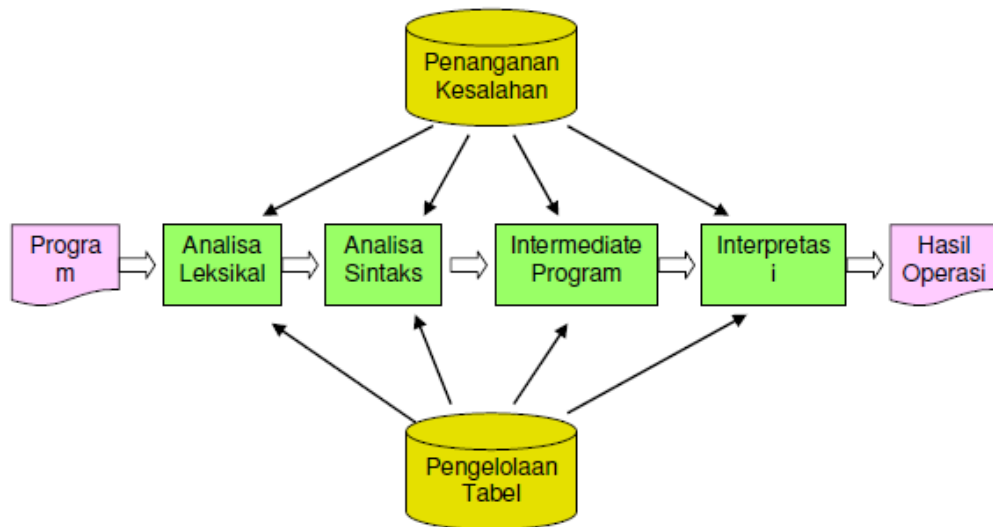
1. Bahasa mesin merupakan bentuk terendah dari bahasa komputer. Instruksi direpresentasikan dalam kode numerik.
2. Bahasa Assembly merupakan bentuk simbolik dari bahasa mesin. Kode misalnya ADD, MUL, dsb
3. Bahasa tingkat tinggi (user oriented) lebih banyak memberikan fungsi kontrol program, kalang, block, dan prosedur.
4. Bahasa problem oriented sering juga dimasukkan sebagai bahasa tingkat tinggi, misalnya SQL, Myob, dsb.

### B. Translator

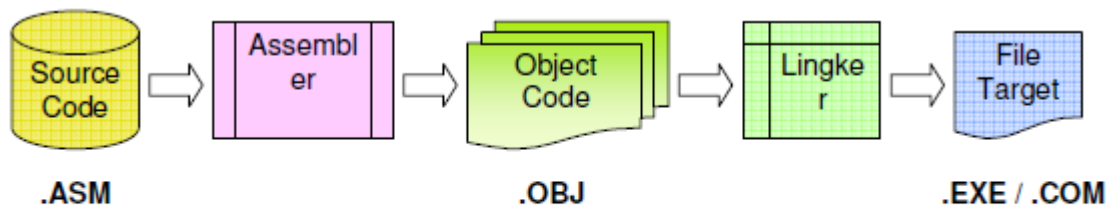
Translator melakukan pengubahan source code/ kode program kedalam target code/ object code. Interpreter dan Compiler termasuk dalam kategori translator



### C. Interpreter



### D. Assembler



Proses Sebuah Kompilasi pada Bahasa Assembler

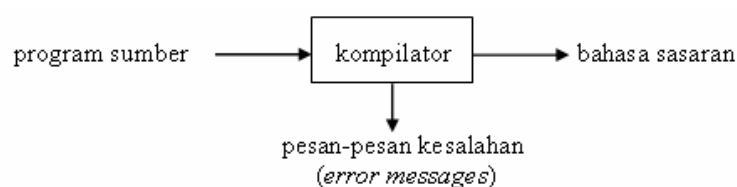
Keterangan :

- Source Code adalah bahasa Assembler, Object Code adalah bahasa Mesin.
- Object Code dapat berupa file object (.OBJ), file .EXE, atau file .COM
- Contoh : Turbo Assembler (dari IBM) dan Macro Assembler (dari Microsoft).

### E. Compiler

Kompilator (compiler) adalah sebuah program yang membaca suatu program yang ditulis dalam suatu bahasa sumber (source language) dan menterjemahkannya ke dalam suatu bahasa sasaran (target language).

Proses kompilasi dapat digambarkan melalui sebuah kotak hitam (black box) berikut :



## Fase Compiler

Proses kompilasi dikelompokkan ke dalam dua kelompok besar :

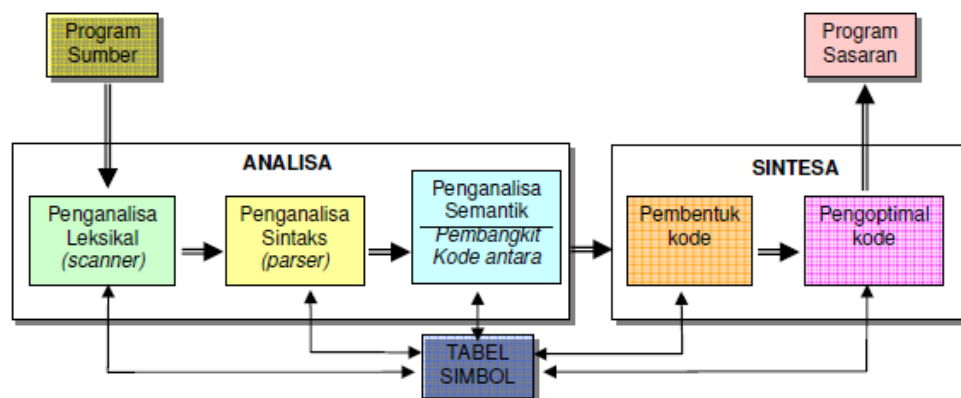
### 1. Analisis (front-end) :

- a. program sumber dipecah-pecah dan dibentuk menjadi bentuk antara (inter-mediate representation)
- b. Language Independent

### 2. Sintesis (back-end) :

- a. membangun program sasaran yang diinginkan dari bentuk antara
- b. Language dependent

## Blok Diagram Compiler



Bagan pokok proses kompilasi

## Keterangan

1. Program Sumber : ditulis dalam bahasa sumber (Pascal, Assembler, dsb)
2. Program Sasaran : dapat berupa bahasa pemrograman lain atau bahasa mesin pada suatu komputer.
3. Penganalisa Leksikal : membaca program sumber, karakter demi karakter. Sederetan (satu atau lebih) karakter dikelompokkan menjadi satu kesatuan mengacu kepada pola kesatuan kelompok karakter (token) yang ditentukan dalam bahasa sumber. Kelompok karakter yang membentuk sebuah token dinamakan lexeme untuk token tersebut. Setiap token yang dihasilkan disimpan di dalam tabel simbol. Sederetan karakter yang tidak mengikuti pola token akan dilaporkan sebagai token tak dikenal (unidentified token).

4. Penganalisa sintaks : memeriksa kesesuaian pola deretan token dengan aturan sintaks yang ditentukan dalam bahasa sumber. Sederetan token yang tidak mengikuti aturan sintaks akan dilaporkan sebagai kesalahan sintaks (syntax error). Secara logika deretan token yang bersesuaian dengan sintaks tertentu akan dinyatakan sebagai pohon parsing (parse tree).
5. Penganalisa semantik : memeriksa token dan ekspresi dari batasan - batasan yang ditetapkan. Batasan-batasan tersebut misalnya :
  - a. panjang maksimum token identifier adalah 8 karakter,
  - b. panjang maksimum ekspresi tunggal adalah 80 karakter,
  - c. nilai bilangan bulat adalah -32768 s/d 32767,
  - d. operasi aritmatika harus melibatkan operan-operan yang bertipe sama. Melakukan analisa semantik, biasanya dalam realisasi akan digabungkan Dengan intermediate code generator (bagian yang berfungsi membangkitkan kode antara).
6. Pembangkit kode antara : membangkitkan kode antara (intermediate code) berdasarkan pohon parsing. Pohon parse selanjutnya diterjemahkan oleh suatu penerjemah yang dinamakan penerjemah berdasarkan sintak (syntax-directed translator). Hasil penerjemahan ini biasanya merupakan perintah tiga alamat (three-address code) yang merupakan representasi program untuk suatu mesin abstrak. Perintah tiga alamat bisa berbentuk quadruples (op, arg1, arg2, result), tripels (op, arg1, arg2). Ekspresi dengan satu argumen dinyatakan dengan menetapkan arg2 dengan - (strip, dash)
7. Pembentuk Kode : membangkitkan kode objek dalam bahasa target tertentu (misalnya bahasa mesin).
8. Pengoptimal Kode : Melakukan optimasi (penghematan space dan waktu komputasi), jika mungkin, terhadap kode antara sehingga dapat memperkecil hasil dan mempercepat proses.
9. Tabel : Menyimpan semua informasi yang berhubungan dengan proses kompilasi.

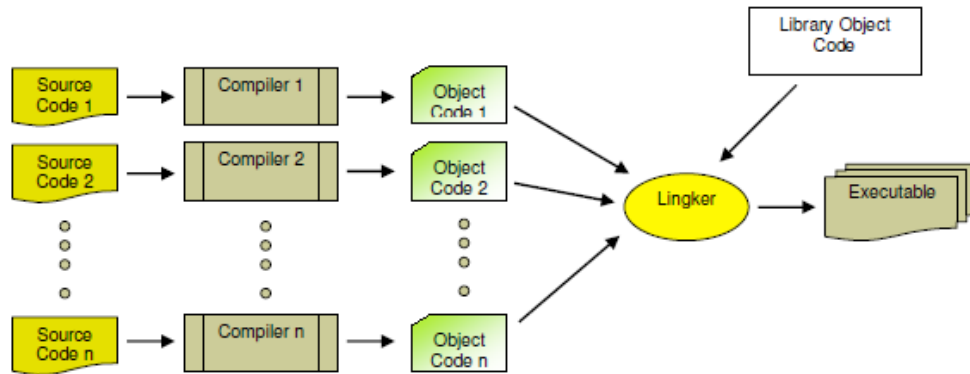
## **Pembuatan Compiler**

Pembuatan kompilator dapat dilakukan dengan:

1. Bahasa Mesin  
Tingkat kesulitannya tinggi, bahkan hampir mustahil dilakukan.
2. Bahasa Assembly  
Bahasa Assembly bisa dan biasa digunakan sebagai tahap awal pada proses pembuatan sebuah kompilator.
3. Bahasa Tingkat Tinggi lain pada mesin yang sama  
Proses pembuatan kopilator akan lebih mudah.

4. Bahasa tingkat tinggi yang sama pada mesin yang berbeda  
Misal, pembuatan kompilator C untuk DOS, berdasar C pada UNIX.
5. Bootstrap  
Pembuatan kompilator secara bertingkat.

Keterangan :



Keterangan :

- o Pembentukan file Executable berdasar dari beberapa Source Code.
- o Source Code dapat terdiri dari satu atau lebih bahasa pemrograman

### **Penggunaan Compiler selain untuk programming**

- o Mentranslasikan javadoc ke html.
- o Mendapatkan hasil dari SQL query (query optimization).
- o Printer parsing PostScript file.
- o Screen Scrapping.
- o Konversi LaTeX ke PDF.

## **LATIHAN SOAL**

1. Jelaskan apa pengertian otomata dan berikanlah contoh-contoh otomata dalam kehidupan sehari-hari?
2. Jelaskan perbedaan prinsip bahasa natural dengan bahasa formal !
3. Apa yang dimaksud dengan penurunan string, dalam pengertian bahasa natural dan pengertian bahasa formal?
4. Dalam tata bahasa formal dikenal istilah Non terminal, apa analoginya istilah non terminal ini dalam bahasa natural?
5. Apa bedanya string dalam bahasa formal dengan kalimat dalam bahasa natural?
6. Sebutkan Fase-fase kompilasi



## BAB II

### KONSEP BAHASA AUTOMATA

#### 2.1 Konsep Bahasa dan Automata

##### 2.1.1 Automata

Kata **otomata** merupakan bentuk jamak dari *automaton*. Kata ini berasal dari bahasa Yunani *automatos* yang berarti *self-acting*. Dalam Kamus *American- Heritage* kata ini diartikan sebagai :

( 1) *a robot*

(2) *one that behaves in automatic or mechanical fashion*

Istilah ini sudah dikenal sejak abad 17 yang terkait dengan misalnya : jam mekanik, *mechanical-duck* karya de Vaucanson (1738), mesin tenun otomatis (1745).

Automata adalah mesin abstrak yang dapat mengenali (recognize), menerima (accept), atau membangkitkan (generate) sebuah kalimat dalam bahasa tertentu. Automata berkaitan dengan teori mesin abstrak, yaitu mesin sekuensial yang menerima input, dan mengeluarkan output, dalam bentuk diskrit.

Contoh :

- Mesin Jaja / vending machine
- Kunci kombinasi
- Parser/compiler

##### Sifat-sifat Otomata :

1. Kelakuan mesin otomata bergantung pada rangkaian input yang diterima mesin tersebut.
2. Setiap saat berada pada status tertentu, dan dapat pindah ke status baru karena perubahan input

##### . 2.1.2 Teori Bahasa

Teori bahasa membicarakan bahasa formal (formal language), terutama untuk kepentingan perancangan kompilator (compiler) dan pemroses naskah (text processor). Bahasa formal adalah kumpulan kalimat. Semua kalimat dalam sebuah bahasa dibangkitkan oleh sebuah tata bahasa (grammar) yang sama. Sebuah bahasa formal bisa

dibangkitkan oleh dua atau lebih tata bahasa berbeda. Dikatakan bahasa formal karena grammar diciptakan mendahului pembangkitan setiap kalimatnya. Bahasa manusia bersifat sebaliknya; grammar diciptakan untuk meresmikan kata-kata yang hidup di masyarakat. Dalam pembicaraan selanjutnya ‘bahasa formal’ akan disebut ‘bahasa’ saja

### . 2.1.3 Hubungan Otomata dan Bahasa Formal

Hubungan otomata dengan bahasa formal dapat dilukiskan sebagai berikut :

- Rangkaian input diskreet pada mesin otomata dapat dianggap sebagai bahasa yang harus dikenali oleh otomata.
- Mesin otomata dapat pula digunakan untuk membangkitkan bahasa tertentu yang aturannya ditentukan oleh tata bahasa tertentu.

Dengan demikian dapat dilihat keterkaitan antara : mesin otomata, bahasa yang dibangkitkan atau dikenali oleh mesin dan tata bahasa yang membangkitkan sebuah bahasa.

Teori Otomata dan bahasa formal berkaitan dalam hal, yaitu:

- Pembangkitan kalimat/ generation : menghasilkan semua kalimat dalam bahasa L berdasarkan aturan yang dimilikinya.
- Pengenalan kalimat/ recognition : menentukan suatu string (kalimat) termasuk sebagai salah satu anggota himpunan L.

### 2.1.4 Bahasa Natural dan Bahasa Formal :

Perlu disini dibatasi pengertian bahasa formal dengan bahasa sehari-hari. Bahasa manusia sehari-hari (misalnya bahasa inggris) umumnya dinamakan sebagai bahasa alami (*natural language*). Bahasa alami memiliki tata bahasa dan aturan yang lebih luas dan luwes. Bahasa yang lebih kaku dengan aturan-aturan yang lebih ketat (misalnya bahasa pemrograman komputer) dinamakan dengan bahasa *formal* (*formal language*). Sehingga dengan demikian bahasa formal dapat lebih mudah dipelajari dan dianalisis dari pada bahasa alami. Sebaliknya analisis dan pengembangan riset tentang bahasa alami dapat dimulai dengan mempergunakan bahasa formal sebagai langkah awalnya. Ada dua hal penting yang terkait dengan bahasa formal, yaitu :

- **Pembangkitan kalimat** (*generation*) : Berkaitan dengan algoritma yang dapat menghasilkan semua kalimat dalam bahasa tertentu yang dikaji berdasarkan aturan yang dimiliki oleh bahasa tersebut. Aturan ini disebut tata bahasa (*grammar*). Penerapan konsep ini terjadi pada bahasa-bahasa pemrograman

visual seperti *Visual Basic*, *Delphi*, *Visual C*, *Java Net Bean* dan lain-lain yang mana programmer tidak menuliskan kode program tetapi kode tersebut dibangkitkan ketika sebuah aktivitas dilakukan, misalnya ketika programmer memasang *Button* pada sebuah *Form* maka nama variabel *Button* dan prosedur aktifitas *Button* akan dibangkitkan sehingga programmer tinggal mengisi kode intinya saja.

- **Pengenalan kalimat** (*recognition*) : Pembuatan algoritma yang dapat mengetahui apakah suatu string  $s$  (kalimat) termasuk anggota himpunan bahasa  $L$ . Algoritma ini memeriksa keanggotaan  $s$  dalam bahasa  $L$  berdasarkan aturan yang banyaknya terhingga. Penerapan ini terjadi pada saat sebuah kode program sudah *diparsing* menjadi token-token dan proses kompilasi akan dilakukan maka langkah pertama adalah pemeriksaan apakah token-token sudah berada dalam sintak yang benar sesuai dengan aturan bahasa yang ada. Jika belum memenuhi aturan bahasa maka proses kompilasi akan dihentikan, biasanya dengan memberikan pesan “*syntax error*”.

### 2.1.5 Elemen Bahasa Formal:

Beberapa istilah yang perlu dicatat berkaitan dengan bahasa formal adalah sebagai berikut :

**Abjad** (*alphabet*): Himpunan berhingga dari simbol-simbol yang dapat disusun untuk membentuk suatu kalimat. Dalam konteks teori bahasa: kalimat, string atau kata ketiganya digunakan merujuk kepada hal yang sama, yaitu rangkaian simbol-simbol yang dapat disusun dengan menggunakan simbol yang diambil dari himpunan abjad. Himpunan abjad biasa dinotasikan dengan simbol  $\Sigma$

**Bahasa** (*Language*): Himpunan seluruh string yang dapat dibangkitkan dari sebuah tata bahasa (*grammar*)  $G$ . Bahasa yang dibangkitkan oleh tata bahasa  $G$  biasa dinotasikan dengan  $L(G)$  atau  $L$  saja. Himpunan ini dapat berhingga atau tak berhingga.

**Aturan produksi** (*production rule*) : Adalah himpunan berhingga dari aturan-aturan penataan simbol dalam pembentukan sebuah string. Dengan aturan ini kita memproduksi sebuah string , anggota suatu bahasa. Himpunan aturan produksi biasa disimbolkan sebagai  $P$ .

## 2.2 Grammar dan Bahasa

### 2.2.1 Definisi

**Tata Bahasa (grammar)** bisa didefinisikan secara formal sebagai kumpulan dari himpunan-himpunan variabel, simbol-simbol terminal, simbol awal, yang dibatasi oleh aturan-aturan produksi.

**String / Kalimat** adalah deretan hingga (string) yang tersusun atas simbol-simbol terminal.

**Sentensial** adalah string yang tersusun atas simbol-simbol terminal atau simbol-simbol non terminal atau campuran keduanya. Kalimat adalah merupakan sentensial, sebaliknya belum tentu.

**Derivasi** adalah proses pembentukan sebuah kalimat atau sentensial. Sebuah derivasi

dilambangkan sebagai :  $\alpha \Rightarrow \beta$ .

### 2.2.2 Simbol

#### A. Simbol Terminal

Pengertian terminal berasal dari kata terminate (berakhir), maksudnya derivasi berakhir jika sentensial yang dihasilkan adalah sebuah kalimat (yang tersusun atas simbol-simbol terminal itu). Simbol terminal adalah simbol yang tidak dapat diturunkan lagi.

Simbol-simbol berikut adalah simbol terminal :

- huruf kecil awal alfabet, misalnya : a, b, c
- simbol operator, misalnya : +, −, dan ×
- simbol tanda baca, misalnya : ( , ), dan ;
- string yang tercetak tebal, misalnya : **if, then, dan else.**

#### B. Simbol Non Terminal

Pengertian non terminal berasal dari kata not terminate (belum/tidak berakhir), maksudnya derivasi belum/ tidak berakhir jika sentensial yang dihasilkan mengandung simbol non terminal. Simbol variabel /non terminal adalah simbol yang masih bisa diturunkan.

Simbol-simbol berikut adalah simbol non terminal :

- huruf besar awal alfabet, misalnya : A, B, C
- huruf S sebagai simbol awal

- string yang tercetak miring, misalnya :
- `expr` dan `stmt`

Huruf besar akhir alfabet melambangkan simbol terminal atau non terminal, misalnya : X, Y, Z.

Huruf kecil akhir alfabet melambangkan string yang tersusun atas simbol-simbol terminal, misalnya : x, y, z.

Huruf yunani melambangkan string yang tersusun atas simbol-simbol terminal atau simbol-simbol non terminal atau campuran keduanya, misalnya :  $\alpha$ ,  $\beta$ , dan  $\gamma$ .

### 2.2.3 Aturan Produksi

Aturan produksi men-spesifikasikan bagaimana suatu tatabahasa melakukan transformasi suatu string ke bentuk lainnya. Melalui aturan produksi didefinisikan suatu bahasa yang berhubungan dengan tata bahasa tersebut.

Sebuah produksi dilambangkan sebagai  $\alpha \rightarrow \beta$  (bisa dibaca  $\alpha$  menghasilkan  $\beta$ ), artinya : dalam sebuah derivasi dapat dilakukan penggantian simbol  $\alpha$  dengan simbol  $\beta$ .

Simbol  $\alpha$  dalam produksi berbentuk  $\alpha \rightarrow \beta$  disebut ruas kiri produksi sedangkan simbol  $\beta$  disebut ruas kanan produksi.

Contoh aturan produksi :

$T \rightarrow a$ , dibaca: T menghasilkan a

$E \rightarrow T \mid T+E$ , dibaca: E menghasilkan T atau E menghasilkan T+E

merupakan pemendekan dari aturan produksi :

$E \rightarrow T$

$E \rightarrow T+E$

### 2.2.4 Grammar

Tata bahasa (*grammar*) **G** didefinisikan sebagai *tuple-4*  $G(\Sigma, N, S, P)$

Dimana :

$\Sigma$  : Himpunan berhingga dari simbol-simbol abjad / alphabet / *vocabulary*. Simbol simbol elemen  $\Sigma$  dan rangkaian simbol-simbol yang terdiri dari elemen  $\Sigma$  dinamakan juga dengan simbol terminal. Simbol terminal dilambangkan dengan huruf kecil : a,b,c atau abjad 0,1,2.

**N** : Himpunan berhingga dari simbol-simbol yang disebut sebagai simbol Non Terminal, yaitu simbol-simbol yang dapat digantikan oleh simbol lain. Dalam bahasa *natural* simbol non terminal misalnya : S, NP , VP, ADJ dan lain-lain. Sedangkan dalam bahasa *formal* simbol non terminal dilambangkan dengan abjad huruf besar, dan cukup SATU HURUF BESAR saja, misalnya : A, B, atau C.

**S** : Sebuah simbol yang dinamakan simbol awal (*start symbol*). Simbol S merupakan awal penurunan seluruh string anggota bahasa yang dibangkitkan oleh tata bahasa **G** tersebut.

**P** : Himpunan berhingga aturan-aturan produksi. Aturan produksi merupakan ekspresi yang dapat dituliskan sebagai  $\alpha \rightarrow \beta$  , dengan  $\alpha$  dan  $\beta$  masing-masing adalah *string* (rangkaihan simbol-simbol) yang dapat terdiri dari simbol terminal dan atau simbol non terminal, misalnya :  $A \rightarrow Ba$

Catatan : Penulisan grammar dibeberapa buku ditulis sebagai **G ( V, T,S,P)**, dengan **V** adalah himpunan Vocabulary, atau himpunan seluruh simbol yang ada (baik simbol terminal maupun non terminal). Sehingga dalam hal ini **V** terdiri dari **S** (*start symbol*), **N** (Non terminal simbol) dan **T** (Terminal simbol), atau dapat ditulis :  $V = T \cup N$

### 2.2.5 Hirarky Chomsky

Tata bahasa (*grammar*) bisa didefinisikan secara formal sebagai kumpulan dari himpunan-himpunan variabel, simbol-simbol terminal, simbol awal, yang dibatasi oleh aturan-aturan produksi. Pada tahun 1959, seorang ahli bernama Noam Chomsky melakukan penggolongan tingkatan bahasa menjadi empat, yang disebut dengan hirarki Chomsky. Penggolongan tersebut bisa dilihat pada tabel berikut.

Bahasa	Mesin Otomata	Batasan Aturan Produksi
Regular	<i>Finite State Automata</i> (FSA) meliputi <i>Deterministic Finite Automata</i> (DFA) & <i>Non Deterministic Finite Automata</i> (NFA)	$\alpha$ adalah sebuah simbol variabel. $\beta$ maksimal memiliki sebuah simbol variabel yang bila ada terletak di posisi paling kanan
Bebas Konteks / <i>Context Free</i>	<i>Push Down Automata</i> (PDA)	$\alpha$ berupa sebuah simbol variabel
<i>Context Sensitive</i>	<i>Linier Bounded Automata</i>	$ \alpha  \leq  \beta $
<i>Unrestricted / Phase Structure / Natural Language</i>	Mesin Turing	Tidak ada batasan

Secara umum tata bahasa dirumuskan sebagai :

$\alpha \rightarrow \beta$ , yang berarti  $\alpha$  menghasilkan  $\beta$  atau  $\alpha$  menurunkan  $\beta$ .

Di mana  $\alpha$  menyatakan simbol-simbol pada ruas kiri aturan produksi (sebelah kiri tanda ' $\rightarrow$ ') dan  $\beta$  menyatakan simbol-simbol pada ruas kanan aturan produksi (sebelah kanan tanda ' $\rightarrow$ ')

Simbol variabel / non terminal adalah simbol yang masih bisa diturunkan dan ditandai dengan huruf besar seperti A, B, C, dst. Simbol terminal adalah simbol yang sudah tidak bisa diturunkan dan ditandai dengan huruf kecil seperti a, b, c, dst.

### **A. Tata Bahasa Regular**

**Aturan :**

- **Simbol pada Sebelah kiri harus berupa sebuah simbol variabel**
- **Simbol pada sebelah kanan maksimal hanya memiliki sebuah simbol variabel dan bila ada terletak di posisi paling kanan.**

Contoh :

$A \rightarrow b$  (Diterima)

$a \rightarrow B$  (Ditolak, karena simbol pada sebelah kiri harus berupa sebuah simbol variabel)

$A \rightarrow B$  (Diterima)

$A \rightarrow bC$  (Diterima)

$A \rightarrow Bc$  (Ditolak, karena simbol variabel pada sebelah kanan harus berada pada posisi paling kanan)

$A \rightarrow bcD$  (Diterima)

$A \rightarrow bCD$  (Ditolak, karena simbol pada sebelah kanan maksimal hanya memiliki sebuah simbol variabel)

$Ab \rightarrow c$  (Ditolak, karena simbol pada sebelah kiri harus berupa sebuah simbol variabel)

### **B. Tata Bahasa Bebas Konteks**

**Aturan:**

- **Simbol pada Sebelah kiri harus berupa sebuah simbol variabel**

Contoh :

$A \rightarrow b$  (Diterima)

$A \rightarrow B$  (Diterima)

$A \rightarrow bC$  (Diterima)

$A \rightarrow Bc$  (Diterima)

$A \rightarrow BcD$  (Diterima)

$A \rightarrow AAA$  (Diterima)

$a \rightarrow b$  (Ditolak, karena simbol pada sebelah kiri harus berupa sebuah simbol variabel)

$Ab \rightarrow c$  (Ditolak, karena simbol pada sebelah kiri harus berupa sebuah simbol variabel)

$AB \rightarrow c$  (Ditolak, karena simbol pada sebelah kiri harus berupa sebuah simbol variabel)

### C. Tata Bahasa *Context Sensitive*

**Aturan :**

- Simbol pada Sebelah kiri harus minimal ada sebuah simbol variabel
- Jumlah simbol pada ruas sebelah kiri harus lebih kecil atau sama dengan jumlah simbol pada ruas kanan

Contoh :

$A \rightarrow bc$  (Diterima)

$Ab \rightarrow cd$  (Diterima)

$AB \rightarrow CD$  (Diterima)

$ABC \rightarrow DE$  (Ditolak, karena jumlah simbol pada ruas sebelah kiri lebih banyak dari jumlah simbol pada ruas kanan)

$Ab \rightarrow cDe$  (Diterima)

$bA \rightarrow cd$  (Diterima)

$a \rightarrow b$  (Ditolak, karena simbol pada sebelah kiri harus minimal ada sebuah simbol variabel)

### D. Tata Bahasa *Unrestricted*

**Aturan :** - Simbol pada Sebelah kiri harus minimal ada sebuah simbol variabel

Contoh :

$Abcdef \rightarrow g$  (Diterima)

$aBCdE \rightarrow GHIJKL$  (Diterima)

$abcdef \rightarrow GHIJKL$  (Ditolak, karena simbol pada sebelah kiri tidak ada sebuah simbol variabel)



## LATIHAN SOAL

1. Sebutkan dan Jelaskan perbedaan antara setiap tata bahasa pada klasifikasi Chomsky
2. Tentukan apakah produksi-produksi berikut memenuhi aturan tata bahasa Regular
  - $A \rightarrow b$
  - $B \rightarrow bdB$
  - $B \rightarrow C$
  - $B \rightarrow bC$
  - $B \rightarrow Ad$
  - $B \rightarrow bcdef$
  - $B \rightarrow bcdefg$
  - $A \rightarrow aSa$
  - $A \rightarrow aSS$
  - $A \rightarrow \epsilon$
3. Tentukan apakah aturan produksi-produksi berikut memenuhi aturan tata bahasa bebas konteks.
  - $A \rightarrow aSa$
  - $A \rightarrow Ace$
  - $A \rightarrow ab$
  - $A \rightarrow \epsilon$
  - $B \rightarrow bcdef$
  - $B \rightarrow bcdefG$
  - $A \rightarrow aSa$
  - $A \rightarrow aSS$
  - $A \rightarrow BCDEF$
  - $Ad \rightarrow dB$
  - $A \rightarrow AAAAA$
  - $d \rightarrow A$
4. Tentukan apakah produksi-produksi berikut memenuhi aturan tata bahasa *context sensitive*.
  - $B \rightarrow bcdefG$
  - $A \rightarrow aSa$

- $A \rightarrow aSS$
  - $A \rightarrow BCDEF$
  - $Ad \rightarrow dB$
  - $A \rightarrow \epsilon$
  - $AB \rightarrow \epsilon$
  - $ad \rightarrow b$
  - $ad \rightarrow \epsilon$
  - $abC \rightarrow DE$
  - $abcDef \rightarrow ghijkl$
  - $AB \rightarrow cde$
  - $AAA \rightarrow BBB$
5. Tentukan apakah produksi-produksi berikut memenuhi aturan tata bahasa *unrestricted*.
- $A \rightarrow \epsilon$
  - $AB \rightarrow \epsilon$
  - $ad \rightarrow b$
  - $ad \rightarrow \epsilon$
  - $abC \rightarrow DE$
  - $AB \rightarrow cde$
  - $e \rightarrow a$
  - $ABCDEFGH \rightarrow h$
  - $bA \rightarrow CDEFGH$
6. . Jika dimiliki suatu tata-bahasa  $G(\Sigma, N, S, P)$  dengan  $\Sigma = \{ a, b \}$   $N = \{ A, B \}$  dan  $P = \{ S \rightarrow Aa ; S \rightarrow AB ; A \rightarrow aa ; B \rightarrow bB ; B \rightarrow \epsilon \}$ , tentukan bahasa yang dibangkitkan oleh grammar  $G$  di atas.
7. Dimiliki grammar  $G(\Sigma, N, S, P)$  dengan  $\Sigma = \{ a, b \}$ ,  $N = \{ A, B, S \}$  dan  $P = \{ S \rightarrow ABa, A \rightarrow BB, B \rightarrow ab, AB \rightarrow ab, BBB \rightarrow aa \}$ . Perhatikan bahwa string :  $aba$  dan  $abababa$  adalah string-string yang diproduksi oleh grammar tersebut.
8. Dimiliki grammar  $G(\Sigma, N, S, P)$  dengan  $\Sigma = \{ a, b \}$ ,  $N = \{ A, B, S \}$  dan  $P = \{ S \rightarrow ABC, A \rightarrow BB, B \rightarrow Bab, B \rightarrow \epsilon, C \rightarrow aa, A \rightarrow \epsilon \}$ . Termasuk tipe apakah tata bahasa tersebut? Apakah alasannya?
9. Dimiliki grammar  $G(\Sigma, N, S, P)$  dengan  $\Sigma = \{ a, , c \}$ ,  $N = \{ A, B, C, S \}$  dan  $P = \{ S \rightarrow ABC, A \rightarrow aA, A \rightarrow B, B \rightarrow bB, B \rightarrow C, C \rightarrow cC, C \rightarrow \epsilon \}$ . Tentukan bahasa yang dibangkitkan oleh tata bahasa tersebut.

10. Dimiliki grammar  $G(\Sigma, N, S, P)$  dengan  $\Sigma = \{ a, b, c \}$ ,  $N = \{ A, B, C, S \}$  dan  $P = \{ S \rightarrow ABC, A \rightarrow aA, A \rightarrow \epsilon, B \rightarrow bB, B \rightarrow \epsilon, C \rightarrow cC, C \rightarrow \epsilon \}$ . Tentukan bahasa yang dibangkitkan oleh tata bahasa tersebut.

## BAB III

### FINITE STATE AUTOMATA

#### 3.1 FSA

*Finite State Automata / State Otomata* berhingga, selanjutnya kita sebut sebagai FSA, bukanlah mesin fisik tetapi suatu model matematika dari suatu sistem yang menerima *input* dan *output* diskrit.

*Finite State Automata* merupakan mesin otomata dari bahasa regular. Suatu *Finite State Automata* memiliki *state* yang banyaknya berhingga, dan dapat berpindah-pindah dari suatu *state* ke *state* lain.

Kegunaan:

1. Software untuk mendesain dan mengecek perilaku sirkuit digital. Contoh: mesin ATM.
2. Bagian “Lexical Analyzer” dari berbagai kompiler yang berfungsi membagi teks input menjadi logical unit seperti Keyword, Identifier, dan punctuation.
3. Search engine \_ menscan web, dan menemukan kata, frase atau pola yang tepat.

Secara formal *finite state automata* dinyatakan oleh 5 tuple atau  $M = (Q, \Sigma, \delta, S, F)$ , dimana :

$Q$  = himpunan *state* / kedudukan

$\Sigma$  = himpunan simbol *input* / masukan / abjad

$\delta$  = fungsi transisi

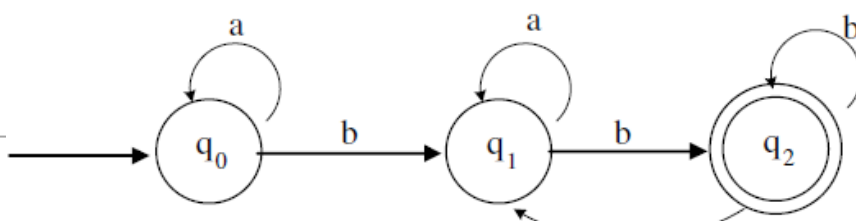
$S$  = *state* awal / kedudukan awal (*initial state*)

$F$  = himpunan *state* akhir

*Finite State Automata* yang memiliki tepat satu *state* berikutnya untuk setiap simbol masukan yang diterima disebut *Deterministic Finite Automata*.

Sebagai contoh;

A. kita memiliki sebuah otomata seperti pada gambar di bawah ini.



Konfigurasi *Deterministic Finite Automata* di atas secara formal dinyatakan sebagai berikut.

$$Q = \{ q_0, q_1, q_2 \}$$

$$\Sigma = \{ a, b \}$$

$$S = q_0$$

$$F = \{ q_2 \}$$

Fungsi transisi yang ada sebagai berikut.

$$\delta(q_0, a) = q_0$$

$$\delta(q_0, b) = q_1$$

$$\delta(q_1, a) = q_1$$

$$\delta(q_1, b) = q_2$$

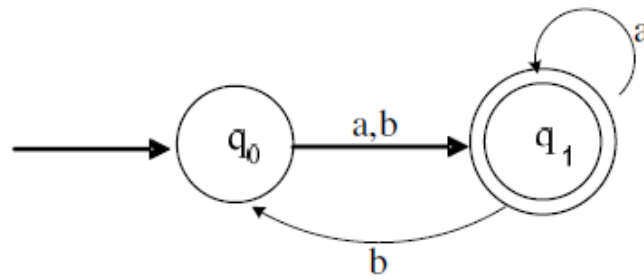
$$\delta(q_2, a) = q_1$$

$$\delta(q_2, b) = q_2$$

Biasanya fungsi-fungsi transisi ini kita sajikan dalam sebuah tabel transisi. Tabel transisi tersebut menunjukkan *state-state* berikutnya untuk kombinasi *state-state* dan *input*. Tabel transisi dari fungsi transisi di atas sebagai berikut

$\delta$	a	b
$q_0$	$q_0$	$q_1$
$q_1$	$q_1$	$q_2$
$q_2$	$q_1$	$q_2$

**B.** Contoh lain bisa dilihat pada gambar di bawah ini.



Tabel transisi dari gambar di atas adalah sebagai berikut:

$\delta$	a	b
$q_0$	$q_1$	$q_1$
$q_1$	$q_1$	$q_0$

### 3.1.1 Konversi dari Tabel Transisi ke Diagram Transisi

Kita juga dapat menggambar diagram transisi dari suatu tabel transisi.

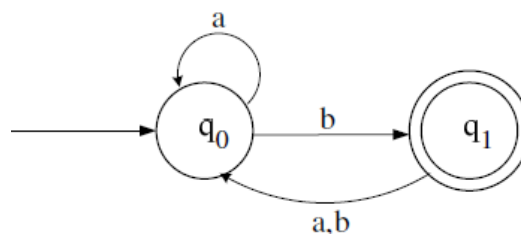
$\delta$	a	b
$q_0$	$q_0$	$q_1$
$q_1$	$q_0$	$q_0$

A. Contoh dengan;

$S = q_0$

$F = \{q_1\}$

Maka diagram transisinya adalah sebagai berikut.



B. Contoh Lain; terdapat tabel transisi sebagai berikut.

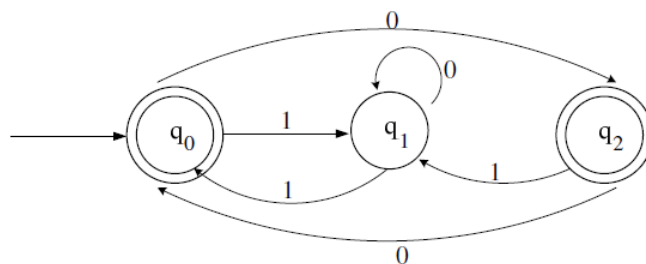
$\delta$	0	1
$q_0$	$q_2$	$q_1$
$q_1$	$q_1$	$q_0$
$q_2$	$q_0$	$q_1$

Dengan

$S = q_0$

$F = \{q_0, q_2\}$

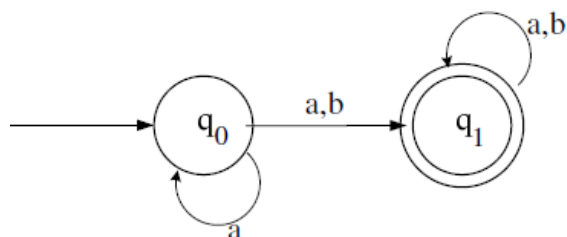
Diagram transisinya dapat kita lihat pada gambar di bawah ini.



Perhatikan pada contoh-contoh *Deterministic Finite Automata* pada contoh-contoh sebelumnya, terlihat bahwa dari setiap *state* selalu tepat ada satu *state* berikutnya untuk setiap simbol *input* yang ada. Berbeda halnya dengan *Non Deterministic Finite Automata* (NFA). Pada NFA, dari suatu input mungkin saja bisa dihasilkan lebih dari satu *state* berikutnya.

### 3.2 Non Deterministic Finite Automata

*Non Deterministic Finite Automata* didefinisikan pula dengan lima (5) tuple, sama seperti halnya pada *Deterministic Finite Automata*. Perhatikan contoh di bawah ini.

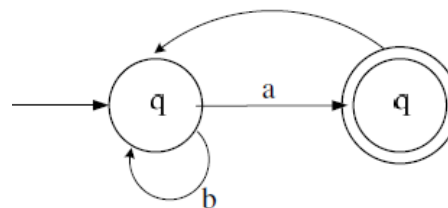


Perhatikan gambar di atas, bila *state*  $q_0$  mendapat input 'a' bisa berpindah ke *state*  $q_0$  atau  $q_1$ , yang secara formal dinyatakan :  $\delta(q_0, a) = \{q_0, q_1\}$  Maka otomata ini disebut non-deterministik (tidak pasti arahnya). Bisa kita lihat tabel transisinya seperti di bawah ini.

$\delta$	a	B
$q_0$	$\{q_0, q_1\}$	$\{q_1\}$
$q_1$	$\{q_1\}$	$\{q_1\}$

Perhatikan cara penulisan *state* hasil transisi pada tabel transisi untuk *Non Deterministic Finite Automata* digunakan kurung kurawal '{' dan '}' karena hasil transisinya merupakan suatu himpunan *state*

Contoh lainnya dapat ditunjukkan pada gambar di bawah ini :



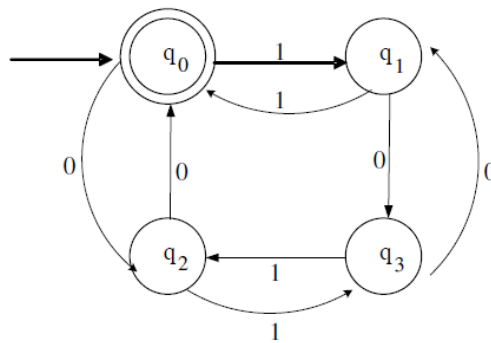
Kita bisa melihat tabel transisinya di bawah ini :

$\delta$	a	B
$q_0$	$\{q_1\}$	$\{q_0\}$
$q_1$	$\{q_0\}$	$\emptyset$

Seperti halnya pada *Deterministic Finite Automata*, pada *Non Deterministic Finite Automata* kita juga bisa membuat diagram transisinya dari tabel transisinya.

## LATIHAN SOAL

1. Buatlah tabel transisi dari *Deterministic Finite Automata* berikut.



2. Gambarkan diagram transisi dari *Deterministic Finite Automata* berikut

$$Q = \{q_0, q_1, q_2\}$$

$$\Sigma = \{a, b\}$$

$$S = q_0$$

$$F = \{q_0\}$$

Tabel transisi dari DFA tersebut :

$\delta$	a	B
$q_0$	$q_1$	$q_2$
$q_1$	$q_2$	$q_0$
$q_2$	$q_2$	$q_2$

3. Gambarkan diagram transisi dari *Deterministic Finite Automata* berikut.

$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{a, b\}$$

$$S = q_0$$

$$F = \{q_0, q_1, q_2\}$$



Fungsi transisi dari DFA tersebut :

$\delta$	a	B
$q_0$	$q_0$	$q_1$
$q_1$	$q_0$	$q_2$
$q_2$	$q_0$	$q_3$
$q_3$	$q_3$	$q_2$

4. Gambarlah diagram transisi untuk NFA berikut :

$$Q = \{q_0, q_1, q_2, q_3, q_4\}$$

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

$$S = q_0$$

$$F = \{q_2, q_4\}$$

Fungsi transisi dari NFA tersebut :

$\delta$	0	1
$q_0$	$\{q_0, q_3\}$	$\{q_0, q_1\}$
$q_1$	$\emptyset$	$\{q_2\}$
$q_2$	$\{q_2\}$	$\{q_2\}$
$q_3$	$\{q_4\}$	$\emptyset$
$q_4$	$\{q_4\}$	$\{q_4\}$

5. Gambarlah diagram transisi untuk NFA berikut :

$$Q = \{q_0, q_1\}$$

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

$$S = q_0$$

$$F = \{q_1\}$$

Fungsi transisi dari NFA tersebut :

$\delta$	0	1
$q_0$	$\{q_0, q_1\}$	$\{q_1\}$
$q_1$	$\emptyset$	$\{q_0, q_1\}$

## BAB IV

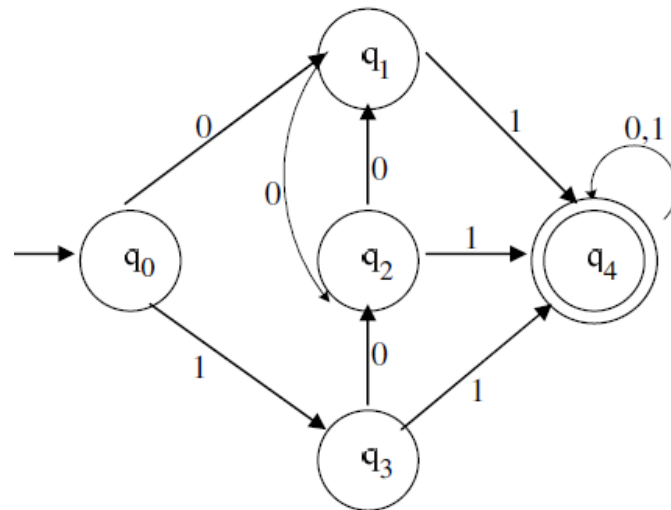
### REDUKSI FINITE STATE AUTOMATA

#### 4.1 Reduksi Jumlah State pada Finite State Automata

Untuk suatu bahasa regular, kemungkinan ada sejumlah *Deterministic Finite Automata* yang dapat menerimanya. Perbedaanannya hanyalah jumlah *state* yang dimiliki otomataotomata yang saling ekuivalen tersebut. Tentu saja, dengan alasan kepraktisan, kita memilih otomata dengan jumlah *state* yang lebih sedikit. Sasaran kita di sini adalah mengurangi jumlah *state* dari suatu *Finite State Automata*, dengan tidak mengurangi kemampuannya semula untuk menerima suatu bahasa. Ada dua buah istilah baru yang perlu kita ketahui yaitu :

1. *Distinguishable* yang berarti dapat dibedakan.
2. *Indistinguishable* yang berarti tidak dapat dibedakan

Sebagai contoh kita ingin menyederhanakan DFA berikut.



1. Identifikasilah setiap kombinasi *state* yang mungkin :

Kombinasi *state* yang mungkin adalah :

(q 0 , q1 )  
 (q 0 , q 2 )  
 (q 0 , q 3 )  
 (q 0 , q 4 )  
 (q1 , q 2 )  
 (q1 , q 3 )  
 (q1 , q 4 )  
 (q 2 , q 3 )  
 (q 2 , q 4 )  
 (q 3 , q 4 )

2. *State* yang berpasangan dengan *state* akhir (q 4 ) merupakan *state* yang distinguishable

(q 0 , q1 )  
 (q 0 , q 2 )  
 (q 0 , q 3 )  
 (q 0 , q 4 ) : Distinguishable  
 (q1 , q 2 )  
 (q1 , q 3 )  
 (q1 , q 4 ) : Distinguishable  
 (q 2 , q 3 )  
 (q 2 , q 4 ) : Distinguishable

$(q_3, q_4)$  : Distinguishable

3. Untuk pasangan *state* yang lain jika masing-masing *state* mendapat input yang sama, maka bila satu *state* mencapai *state* akhir dan yang lain tidak mencapai *state* akhir maka dikatakan distinguishable.

Untuk  $(q_0, q_1)$  :

$$\delta(q_0, 1) = q_3$$

$$\delta(q_1, 1) = q_4$$

$$\delta(q_0, 0) = q_1$$

$$\delta(q_1, 0) = q_2$$

Maka  $(q_0, q_1)$  : Distinguishable

Untuk  $(q_0, q_2)$  :

$$\delta(q_0, 1) = q_3$$

$$\delta(q_2, 1) = q_4$$

$$\delta(q_0, 0) = q_1$$

$$\delta(q_2, 0) = q_1$$

Maka  $(q_0, q_2)$  : Distinguishable

Untuk  $(q_0, q_3)$  :

$$\delta(q_0, 1) = q_3$$

$$\delta(q_3, 1) = q_4$$

$$\delta(q_0, 0) = q_1$$

$$\delta(q_3, 0) = q_2$$

Maka  $(q_0, q_3)$  : Distinguishable

Untuk  $(q_1, q_2)$

$$\delta(q_1, 1) = q_4$$

$$\delta(q_2, 1) = q_4$$

$$\delta(q_1, 0) = q_2$$

$$\delta(q_2, 0) = q_1$$

Maka  $(q_1, q_2)$  : Indistinguishable

Untuk  $(q_1, q_3)$

$$\delta(q_1, 1) = q_4$$

$$\delta(q_3, 1) = q_4$$

$$\delta(q_1, 0) = q_2$$

$$\delta(q_3, 0) = q_2$$

Maka  $(q_1, q_3)$  : Indistinguishable

Untuk  $(q_2, q_3)$

$$\delta(q_2, 1) = q_4$$

$$\delta(q_3, 1) = q_4$$

$$\delta(q_2, 0) = q_1$$

$$\delta(q_3, 0) = q_2$$

Maka  $(q_2, q_3)$  : Indistinguishable

4. Maka Didapatkan pasangan state sebagai berikut :

$(q_0, q_1)$  : Distinguishable

$(q_0, q_2)$  : Distinguishable

$(q_0, q_3)$  : Distinguishable

$(q_0, q_4)$  : Distinguishable

$(q_1, q_2)$  : Indistinguishable

$(q_1, q_3)$  : Indistinguishable

$(q_1, q_4)$  : Distinguishable

$(q_2, q_3)$  : Indistinguishable

$(q_2, q_4)$  : Distinguishable

$(q_3, q_4)$  : Distinguishable

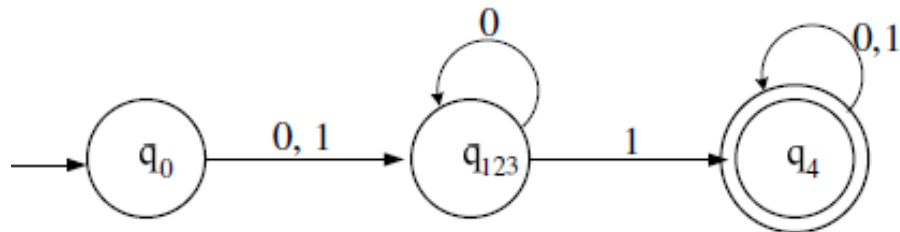
5. Kelompokkan pasangan state yang indistinguishable :

$(q_1, q_2)$  : Indistinguishable

$(q_1, q_3)$  : Indistinguishable

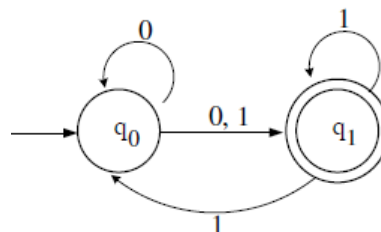
$(q_2, q_3)$  : Indistinguishable

6. Karena  $q_1$  indistinguishable dengan  $q_2$  dan  $q_2$  indistinguishable dengan  $q_3$ , maka bisa dikatakan bahwa  $q_1$ ,  $q_2$ , dan  $q_3$  saling indistinguishable dan dapat dijadikan satu state.
7. Sehingga hasil penyederhanaannya adalah sebagai berikut :



#### 4.2 Ekuivalensi Non-Deterministic Finite Automata ke Deterministic Finite Automata

Dari sebuah mesin *Non-Deterministic Finite Automata* dapat dibuat mesin *Deterministic Finite Automata*-nya yang ekuivalen (bersesuaian). Ekuivalen di sini artinya mampu menerima bahasa yang sama. Sebagai contoh, akan dibuat *Deterministic Finite Automata* dari *Non-Deterministic Finite Automata* berikut.



- Diketahui  $\Sigma = \{0,1\}$

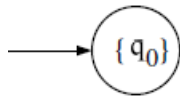
Adapun langkah-langkahnya adalah sebagai berikut.

1. Buatlah tabel transisi dari diagram transisi di atas.

$\delta$	0	1
$q_0$	$\{q_0, q_1\}$	$\{q_1\}$
$q_1$	$\emptyset$	$\{q_0, q_1\}$

2. Buatlah diagram transisi untuk *finite state automata* dari tabel transisi di atas.

a. Kita mulai dari state awal yaitu  $q_0$



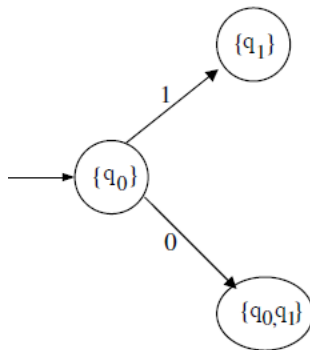
Catatan :

Perhatikan bahwa di sini pada gambar setiap *state* kita tuliskan sebagai himpunan *state*

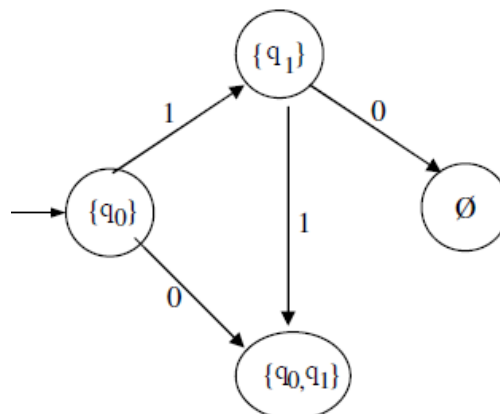
b. Selanjutnya, kita telusuri lebih lanjut tentang  $q_0$ , yaitu :

Bila *state*  $q_0$  mendapat input 0 menjadi *state*  $\{q_0, q_1\}$

Bila *state*  $q_0$  mendapat input 1 menjadi *state*  $\{q_1\}$ , seperti yang tampak pada gbr



c. Selanjutnya kita telusuri untuk *state*  $q_1$ , yaitu : Bila *state*  $q_1$  mendapat *input* 0 maka menjadi *state*  $\emptyset$  Bila *state*  $q_1$  mendapat *input* 1 maka menjadi *state*  $\{q_0, q_1\}$ , sehingga diperoleh gbr.



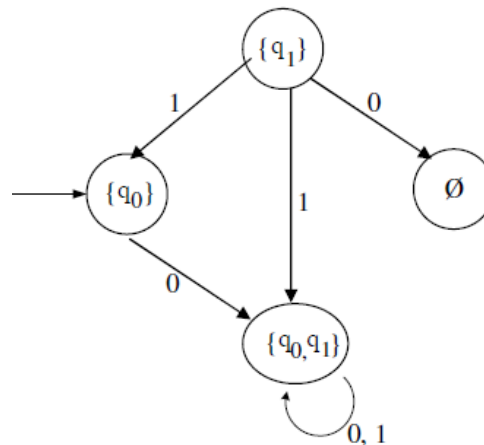
- d. Selanjutnya kita telusuri untuk *state*  $\{q_0, q_1\}$ , yang merupakan penggabungan dari *state*  $q_0$  dan *state*  $q_1$ , sehingga hasil *state*  $\{q_0, q_1\}$  merupakan penggabungan dari hasil *state*  $q_0$  dan *state*  $q_1$ .

Bila *state*  $q_0$  mendapat input 0 menjadi *state*  $\{q_0, q_1\}$

Bila *state*  $q_1$  mendapat input 0 maka menjadi *state*  $\emptyset$  Sehingga diperoleh jika *state*  $\{q_0, q_1\}$  mendapat input 0 menjadi *state*  $\{q_0, q_1\}$

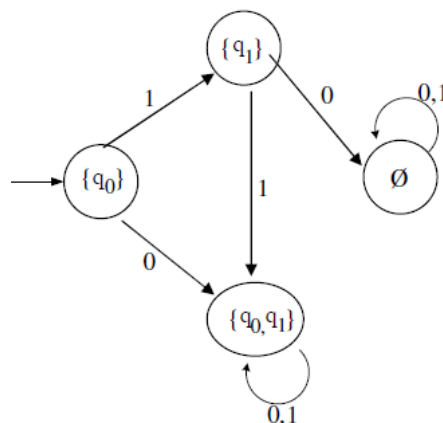
Bila *state*  $q_0$  mendapat input 1 menjadi *state*  $\{q_1\}$

Bila *state*  $q_1$  mendapat input 1 maka menjadi *state*  $\{q_0, q_1\}$  Sehingga diperoleh jika *state*  $\{q_0, q_1\}$  mendapat input 0 menjadi *state*  $\{q_0, q_1\}$  Maka diagram transisi menjadi :



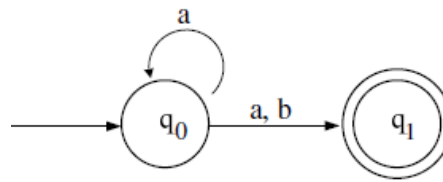
- e. Selanjutnya kita telusuri *state*  $\emptyset$ , yaitu :

Bila *state*  $\emptyset$  mendapat input 0 dan 1 maka tetap menghasilkan  $\emptyset$  Sehingga diperoleh diagram transisi berikut.



- Contoh lain, buatlah DFA dari NFA berikut :



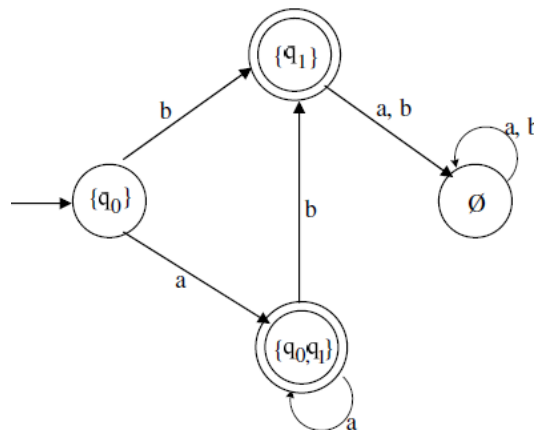


Diketahui  $\Sigma = \{a,b\}$

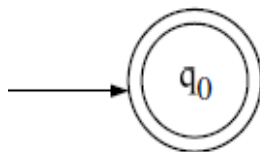
Tabel Transisi untuk NFA pada gambar di atas adalah sebagai berikut.

$\delta$	a	b
$q_0$	$\{q_0, q_1\}$	$\{q_1\}$
$q_1$	$\emptyset$	$\emptyset$

Mesin *Deterministic Finite Automata* yang ekuivalen adalah sebagai berikut.



- Buatlah DFA dari NFA berikut

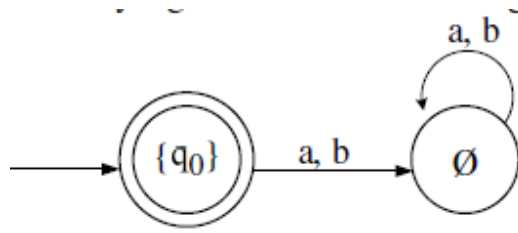


Diketahui  $\Sigma = \{a,b\}$

Tabel transisi untuk NFA pada gambar di atas adalah sebagai berikut.

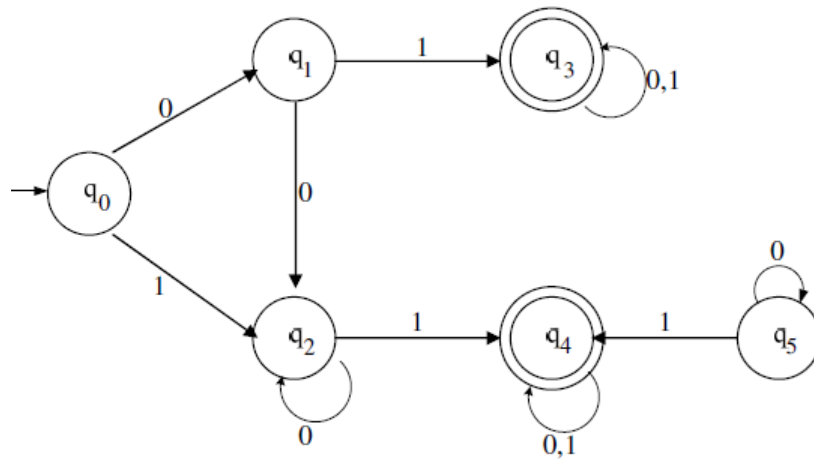
$\delta$	a	b
$q_0$	$\emptyset$	$\emptyset$

Mesin DFA yang ekuivalen adalah sebagai berikut.

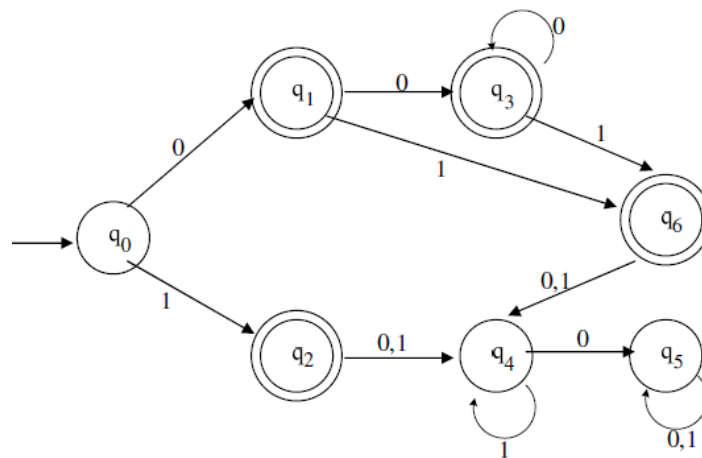


### SOAL LATIHAN

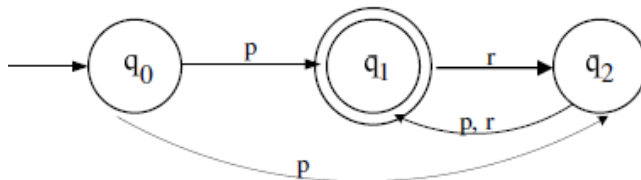
1. Lakukan reduksi jumlah *state* pada *Deterministic Finite Automata* pada gambar berikut.



2. Lakukan reduksi jumlah state pada *Deterministic Finite Automata* berikut.



3. Buatlah DFA dari NFA berikut.



Diketahui  $\Sigma = \{p, r\}$

Tabel transisinya adalah sebagai berikut

$\delta$	p	R
$q_0$	$\{q_1, q_2\}$	$\emptyset$
$q_1$	$\emptyset$	$\{q_2\}$
$q_2$	$\{q_1\}$	$\{q_1\}$

4. Buatlah *Deterministic Finite Automata* yang ekuivalen dengan *Non Deterministic Finite Automata* berikut.

$Q = \{p, q, r, s\}$

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

$$S = p$$

$$F = \{s\}$$

Fungsi transisinya dinyatakan dalam tabel transisi berikut.

$\delta$	0	1
p	{p, q}	{p}
q	{r}	{r}
r	{s}	-
s	s	s

5. Buatlah *Deterministic Finite Automata* yang ekuivalen dengan *Non-Deterministic Finite Automata* berikut.

$$Q = \{p, q, r, s\}$$

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

$$S = p$$

$$F = \{q, s\}$$

Fungsi transisinya dinyatakan dalam tabel transisi berikut.

$\delta$	0	1
p	{q, s}	{q}
q	{r}	{q, r}
r	{s}	{p}
s	-	{p}

6. Buatlah *Deterministic Finite Automata* yang ekuivalen dengan *Non Deterministic Finite Automata* berikut.

$$Q = \{q_0, q_1, q_2\}$$

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

$$S = q_0$$

$$F = \{q_1\}$$

Fungsi transisinya dinyatakan dalam tabel transisi berikut.

$\delta$	0	1
$q_0$	$\{q_0\}$	$\{q_2\}$
$q_1$	$\{q_1\}$	$\emptyset$
$q_2$	$\{q_0, q_1\}$	$\{q_1\}$

7. Buatlah *Deterministic Finite Automata* yang ekuivalen dengan *Non-Deterministic Finite Automata* berikut.

$$Q = \{q_0, q_1, q_2\}$$

$$\Sigma = \{a, b\}$$

$$S = q_0$$

$$F = \{q_1\}$$

Fungsi transisinya dinyatakan dalam tabel transisi berikut

$\delta$	a	b
$q_0$	$\{q_1, q_2\}$	$\{q_2\}$
$q_1$	$\{q_1\}$	$\{q_2\}$
$q_2$	$\emptyset$	$\{q_0, q_2\}$

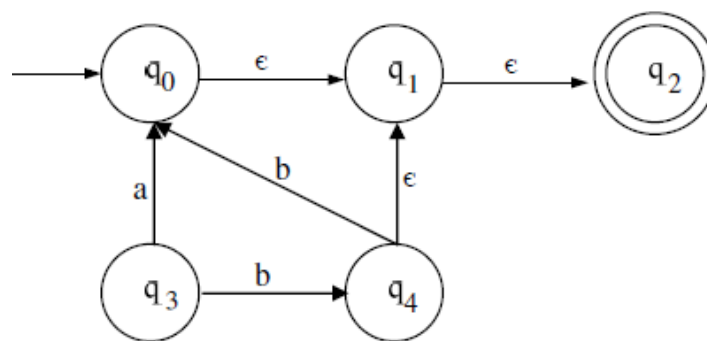
## BAB V

### NON DETERMINISTIC FINITE AUTOMATA DENGAN $\epsilon$ -MOVE

#### 5.1 NFA

Di sini kita mempunyai jenis otomata baru yang disebut *Non Deterministic Finite Automata* dengan  $\epsilon$  – Move (  $\epsilon$  di sini bisa dianggap sebagai 'empty'). Pada *Non – deterministic Finite Automata* dengan  $\epsilon$  – move (transisi  $\epsilon$  ), diperbolehkan mengubah *state* tanpa membaca *input*. Disebut dengan transisi  $\epsilon$  karena tidak bergantung pada suatu *input* ketika melakukan transisi.

- Contoh:



Penjelasan gambar :

- Dari  $q_0$  tanpa membaca *input* dapat berpindah ke  $q_1$
- Dari  $q_1$  tanpa membaca *input* dapat berpindah ke  $q_2$
- Dari  $q_4$  tanpa membaca *input* dapat berpindah ke  $q_1$

#### 5.2 $\epsilon$ – Closure

**$\epsilon$  – Closure untuk Suatu *Non-Deterministic Finite Automata* dengan  $\epsilon$  – Move**

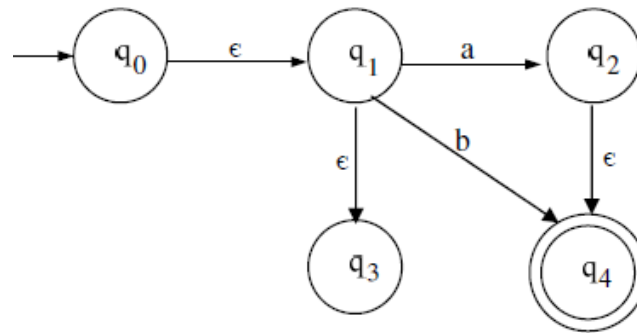
$\epsilon$  – Closure adalah himpunan *state-state* yang dapat dicapai dari suatu *state* tanpa membaca *input*. Perhatikan gambar sebelumnya, maka diperoleh :

$$\epsilon - \text{Closure} ( q_1 ) = \{ q_1, q_2 \}$$

$$\epsilon - \text{Closure} ( q_2 ) = \{ q_2 \}$$

$$\epsilon - \text{Closure} ( q_3 ) = \{ q_3 \}$$

- Contoh lain, dapat dilihat pada gambar di bawah ini.



Dari gambar di atas, kita ketahui  $\epsilon$  – Closure untuk setiap *state* adalah sebagai berikut.

$\epsilon$  – Closure (  $q_0$  ) = {  $q_0, q_1, q_3$  }

$\epsilon$  – Closure (  $q_1$  ) = {  $q_1, q_3$  }

$\epsilon$  – Closure (  $q_2$  ) = {  $q_2, q_4$  }

$\epsilon$  – Closure (  $q_3$  ) = {  $q_3$  }

$\epsilon$  – Closure (  $q_4$  ) = {  $q_4$  }

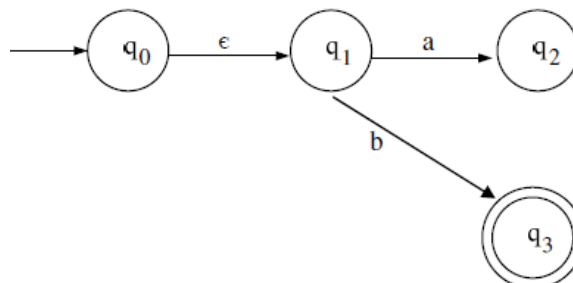
Catatan :

Perhatikan bahwa pada suatu *state* yang tidak memiliki transisi  $\epsilon$  , maka  $\epsilon$  closure – nya adalah *state* itu sendiri

### 5.3 Ekuivalensi Non Deterministic Finite Automata dengan $\epsilon$ – Move ke Non Deterministic Finite Automata tanpa $\epsilon$ – Move

Dari sebuah *Non-Deterministic Finite Automata* dengan  $\epsilon$  – move dapat kita peroleh *Non– Deterministic Finite Automata* tanpa  $\epsilon$  – move yang ekuivalen.

Contohnya, bila kita punya NFA  $\epsilon$  – move, seperti pada gambar di bawah ini



- Dari NFA  $\epsilon$  – move di atas, akan dibuat NFA yang ekuivalen

1. Buatlah tabel transisi dari NFA  $\epsilon$  – move di atas.

$\delta$	a	b
$q_0$	$\emptyset$	$\emptyset$
$q_1$	$\{q_2\}$	$\{q_3\}$
$q_2$	$\emptyset$	$\emptyset$
$q_3$	$\emptyset$	$\emptyset$

2. Tentukan  $\epsilon$ -closure untuk setiap *state*

$$\epsilon - \text{Closure} (q_0) = \{q_0, q_1\}$$

$$\epsilon - \text{Closure} (q_1) = \{q_1\}$$

$$\epsilon - \text{Closure} (q_2) = \{q_2\}$$

$$\epsilon - \text{Closure} (q_3) = \{q_3\}$$

3. Carilah setiap fungsi transisi hasil dari pengubahan NFA  $\epsilon$  - move ke NFA tanpa

$\epsilon$  -move. Fungsi transisi itu ditandai dengan simbol  $\delta'$ ,

$$\delta'(q_0, a) = \epsilon\_cl(\delta(\epsilon\_cl(q_0), a))$$

$$= \epsilon\_cl(q_2)$$

$$= \{q_2\}$$

$$\delta'(q_0, b) = \epsilon\_cl(\delta(\epsilon\_cl(q_0), b))$$

$$= \epsilon\_cl(q_3)$$

$$= \{q_3\}$$

$$\delta'(q_1, a) = \epsilon\_cl(\delta(\epsilon\_cl(q_1), a))$$

$$= \epsilon\_cl(q_2)$$

$$= \{q_2\}$$

$$\delta'(q_1, b) = \epsilon\_cl(\delta(\epsilon\_cl(q_1), b))$$

$$= \epsilon\_cl(q_2)$$

$$= \{q_3\}$$

$$\delta'(q_2, a) = \epsilon\_cl(\delta(\epsilon\_cl(q_2), a))$$

$$= \epsilon\_cl(\emptyset)$$

$$= \emptyset$$

$$\delta'(q_2, b) = \epsilon\_cl(\delta(\epsilon\_cl(q_2), b))$$

$$= \epsilon\_cl(\emptyset)$$

$$= \emptyset$$



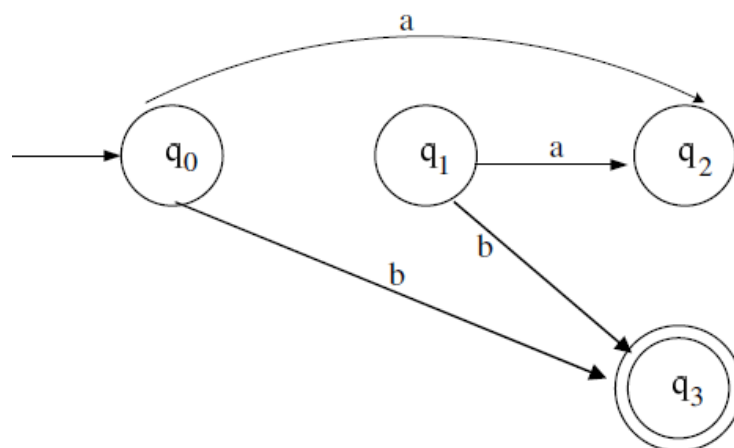
$$\begin{aligned}
 \delta^*(q_3, a) &= \epsilon\_cl(\delta(\epsilon\_cl(q_3), a)) \\
 &= \epsilon\_cl(\emptyset) \\
 &= \emptyset \\
 \delta^*(q_3, b) &= \epsilon\_cl(\delta(\epsilon\_cl(q_3), b)) \\
 &= \epsilon\_cl(\emptyset) \\
 &= \emptyset
 \end{aligned}$$

4. Buatlah tabel transisi dari fungsi transisi yang telah dibuat pada langkah sebelumnya.

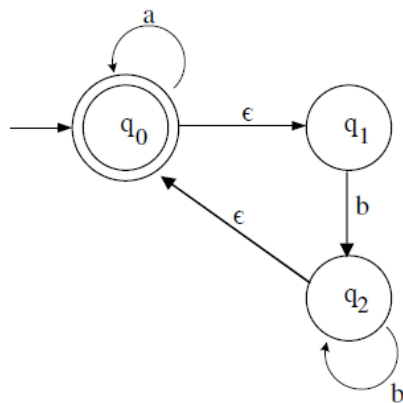
$\delta$	a	b
$q_0$	$\{q_2\}$	$\{q_3\}$
$q_1$	$\{q_2\}$	$\{q_3\}$
$q_2$	$\emptyset$	$\emptyset$
$q_3$	$\emptyset$	$\emptyset$

5. Kemudian, tentukanlah himpunan *state* akhir untuk NFA tanpa  $\epsilon$  – move ini.

Himpunan *state* akhir semula adalah  $\{q_3\}$ . Karena tidak ada *state* lain yang  $\epsilon$  – closure – nya memuat  $q_3$ , maka himpunan *state* akhir sekarang tetap  $\{q_3\}$ . Sehingga diperoleh diagram transisi sebagai berikut.



- Contoh lain:



- Buatlah tabel transisi dari NFA  $\epsilon$  - move di atas.

$\delta$	a	b
$q_0$	$\{ q_0 \}$	$\emptyset$
$q_1$	$\emptyset$	$\{ q_2 \}$
$q_2$	$\emptyset$	$\{ q_2 \}$

- Tentukan  $\epsilon$  -closure untuk setiap *state*

$$\epsilon - \text{Closure} (q_0) = \{ q_0, q_1 \}$$

$$\epsilon - \text{Closure} (q_1) = \{ q_1 \}$$

$$\epsilon - \text{Closure} (q_2) = \{ q_0, q_1, q_2 \}$$

- Carilah setiap fungsi transisi hasil dari pengubahan NFA  $\epsilon$  - move ke NFA tanpa  $\epsilon$  -move. Fungsi transisi itu ditandai dengan simbol  $\delta'$

$$\delta' (q_0, a) = \epsilon\_cl ( \delta ( \epsilon\_cl(q_0), a) )$$

$$= \epsilon\_cl (q_0)$$

$$= \{ q_0, q_1 \}$$

$$\delta' (q_0, b) = \epsilon\_cl ( \delta ( \epsilon\_cl(q_0), b) )$$

$$= \epsilon\_cl (q_2)$$

$$= \{ q_0, q_1, q_2 \}$$

$$\delta' (q_1, a) = \epsilon\_cl ( \delta ( \epsilon\_cl(q_1), a) )$$

$$= \epsilon\_cl (\emptyset)$$

$$= \emptyset$$

$$\delta' (q_1, b) = \epsilon\_cl ( \delta ( \epsilon\_cl(q_1), b) )$$

$$= \epsilon\_cl (q_2)$$

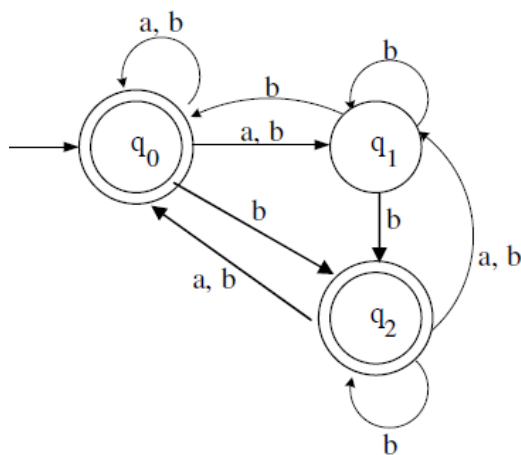
$$= \{ q_0, q_1, q_2 \}$$

$$\begin{aligned}
 \delta'(q_2, a) &= \epsilon\_cl(\delta(\epsilon\_cl(q_2), a)) \\
 &= \epsilon\_cl(q_0) \\
 &= \{q_0, q_1\} \\
 \delta'(q_2, b) &= \epsilon\_cl(\delta(\epsilon\_cl(q_2), b)) \\
 &= \epsilon\_cl(q_2) \\
 &= \{q_0, q_1, q_2\}
 \end{aligned}$$

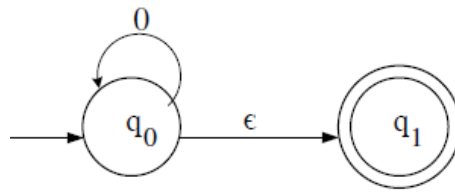
4. Buatlah tabel transisi dari fungsi transisi yang telah dibuat pada langkah sebelumnya.

$\delta$	a	b
$q_0$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$
$q_1$	$\emptyset$	$\{q_0, q_1, q_2\}$
$q_2$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$

5. Kemudian, tentukanlah himpunan *state* akhir untuk NFA tanpa  $\epsilon$  – move ini. Himpunan *state* akhir semula adalah  $\{q_0\}$ . Kita lihat  $\epsilon\_cl(q_2) = \{q_0, q_1, q_2\}$ , maka himpunan *state* akhir sekarang adalah  $\{q_0, q_2\}$ . Sehingga diperoleh diagram transisi sebagai berikut.



▪ Contoh Lain



$$\Sigma = \{0\}$$

1. Buatlah tabel transisi dari NFA  $\epsilon$  - move di atas.

$\delta$	0
$q_0$	$\{ q_0 \}$
$q_1$	$\emptyset$

2. Tentukan  $\epsilon$  -closure untuk setiap *state*

$$\epsilon - \text{Closure} ( q_0 ) = \{ q_0, q_1 \}$$

$$\epsilon - \text{Closure} ( q_1 ) = \{ q_1 \}$$

3. Carilah setiap fungsi transisi hasil dari pengubahan NFA  $\epsilon$  - move ke NFA

tanpa  $\epsilon$  - move. Fungsi transisi itu ditandai dengan simbol  $\delta'$ .

$$\delta' ( q_0, 0 ) = \epsilon\_cl ( \delta ( \epsilon\_cl(q_0), 0 ) )$$

$$= \epsilon\_cl ( q_0 )$$

$$= \{ q_0, q_1 \}$$

$$\delta' ( q_1, 0 ) = \epsilon\_cl ( \delta ( \epsilon\_cl(q_1), 0 ) )$$

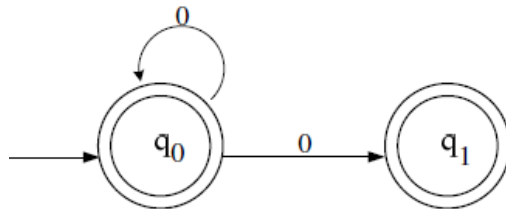
$$= \epsilon\_cl ( \emptyset )$$

$$= \emptyset$$

4. Buatlah tabel transisi dari fungsi transisi yang telah dibuat pada langkah sebelumnya.

$\delta$	0
$q_0$	$\{ q_0, q_1 \}$
$q_1$	$\emptyset$

5. Kemudian, tentukanlah himpunan *state* akhir untuk NFA tanpa  $\epsilon$  – move ini. Himpunan *state* akhir semula adalah  $\{q_1\}$ . Kita lihat  $\epsilon\_cl(q_0) = \{q_0, q_1\}$ , maka himpunan *state* akhir sekarang adalah  $\{q_0, q_1\}$ . Sehingga diperoleh diagram transisi sebagai berikut.



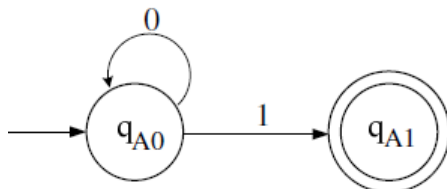
### Penggabungan dan Konkatenasi Finite State Automata

#### A. Penggabungan Finite State Automata

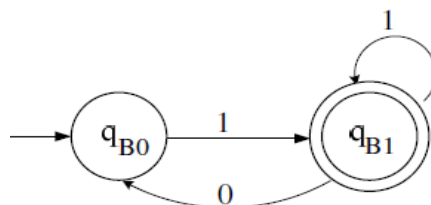
Pada dua mesin *Finite State Automata*, misalkan M1 dan M2 dapat dilakukan penggabungan yang menghasilkan mesin M3 dengan cara :

1. Tambahkan *state* awal untuk M3, hubungkan dengan *state* awal M1 dan *state* awal M2 menggunakan transisi  $\epsilon$ .
2. Tambahkan *state* akhir untuk M3, hubungkan dengan *state-state* akhir M1 dan *state-state* akhir M2 menggunakan transisi  $\epsilon$ .

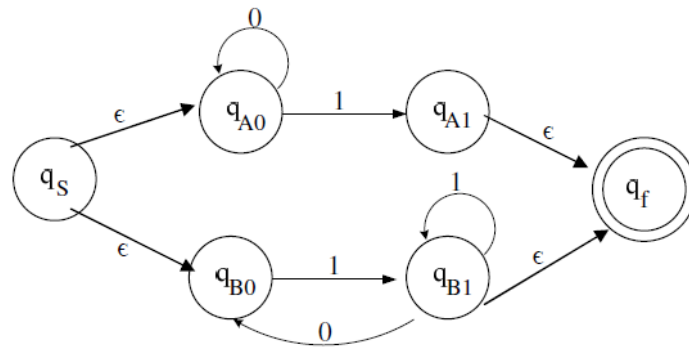
Gambar Mesin M1



Gambar Mesin M2



Adapun hasil penggabungan dari Mesin M1 dan M2 dapat dilihat pada gambar di bawah ini.

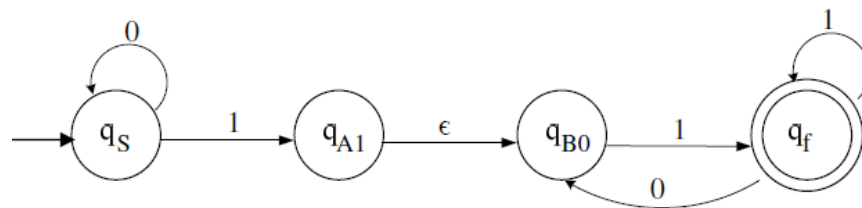


## B. Konkatenasi *Finite State Automata*

Pada dua mesin *Finite State Automata*, misalkan M1 dan M2 dapat dilakukan konkatenasi yang menghasilkan mesin M4 dengan cara :

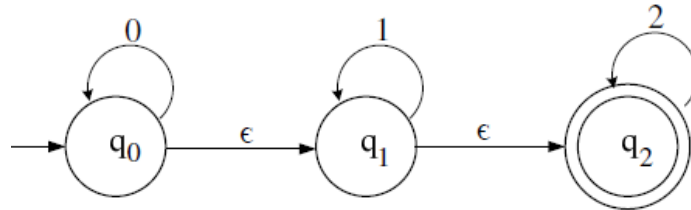
1. *State* awal M1 menjadi *state* awal M4
2. *State-state* akhir M2 menjadi *state* akhir M4
3. Hubungkan *state-state* akhir M1 dengan *state* awal M2 menggunakan transisi  $\epsilon$ .

Kita dapat melihat hasil operasi konkatenasi ini pada gambar di bawah ini.

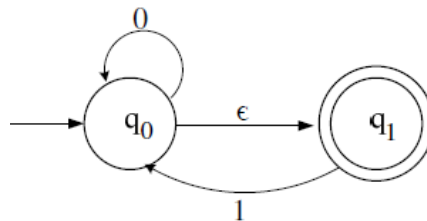


## LATIHAN SOAL

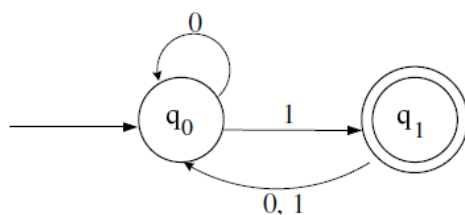
1. Buatlah NFA tanpa  $\epsilon$  – move yang ekuivalen dengan NFA  $\epsilon$  – Move pada gambar berikut ini.  $\Sigma = \{0, 1, 2\}$



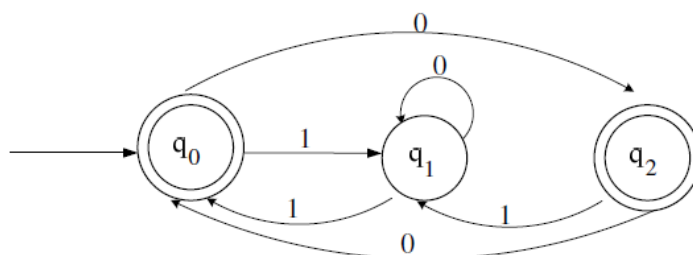
2. Buatlah NFA tanpa  $\epsilon$  – Move yang ekuivalen dengan NFA  $\epsilon$  – Move pada gambar di bawah ini.  $\Sigma = \{0, 1\}$



3. Bila diketahui  $L(M1)$  adalah bahasa yang diterima oleh  $M1$  pada gambar 1, dan  $L(M2)$  adalah bahasa yang diterima oleh  $M2$  pada gambar 2. Diketahui  $L(M3) = L(M1) + L(M2)$ , serta  $L(M4) = L(M1) L(M2)$ . Gambarkan :
  - a. Mesin  $M3$  yang menerima bahasa  $L(M3)$ .
  - b. Mesin  $M4$  yang menerima bahasa  $L(M4)$ .

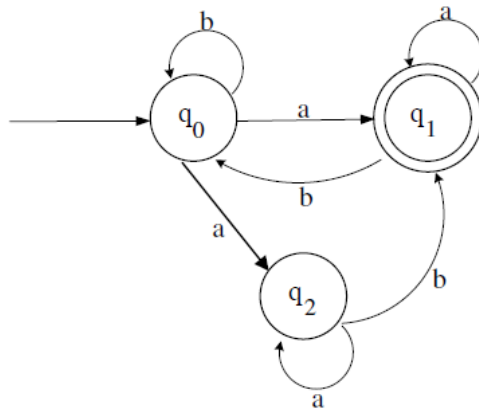


Mesin  $M_1$

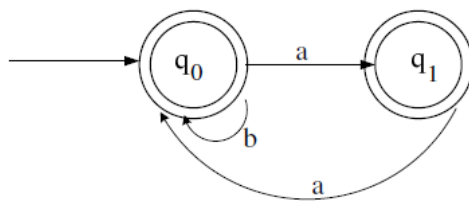


Mesin  $M_2$

4. Bila diketahui  $L(M_1)$  adalah bahasa yang diterima oleh  $M_1$  pada gambar 1, dan  $L(M_2)$  adalah bahasa yang diterima oleh  $M_2$  pada gambar 2. Diketahui  $L(M_3) = L(M_1) + L(M_2)$ , serta  $L(M_4) = L(M_1) L(M_2)$ . Gambarkan :
- Mesin  $M_3$  yang menerima bahasa  $L(M_3)$ .
  - Mesin  $M_4$  yang menerima bahasa  $L(M_4)$ .



Mesin  $M_1$



Mesin  $M_2$



## BAB VI

### EKSPRESI REGULAR

#### 6.1 ER (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. Contoh pemakaian ekspresi regular adalah pada perancangan suatu *text editor*.

##### 6.1.1 Notasi Ekspresi Regular

Notasi Ekspresi Regular yang sering dipakai adalah sebagai berikut.

1. \* yaitu karakter asterisk, yang berarti bisa tidak muncul, bisa juga muncul lebih dari satu kali.
2. + yaitu minimal muncul satu kali
3. + atau U berarti *union*
4. . (Titik) berarti konkatenasi, biasanya titik bisa dihilangkan. Misalnya : ab bermakna sama seperti a.b.

Contoh ekspresi regular (selanjutnya kita singkat sebagai ER) adalah sebagai berikut.

- 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

- ER :  $a^* \cup b^*$  (ingat  $\cup$  berarti atau)

Contoh string yang dibangkitkan : a, b, aa, bb, aaa, bbb, aaaa, bbbb

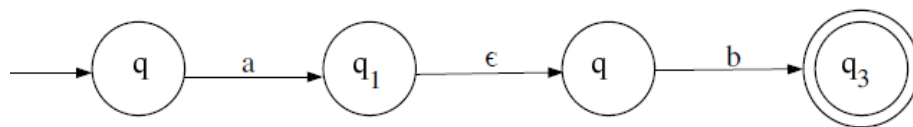
- ER :  $a \cup b$

Contoh string yang dibangkitkan : a, b

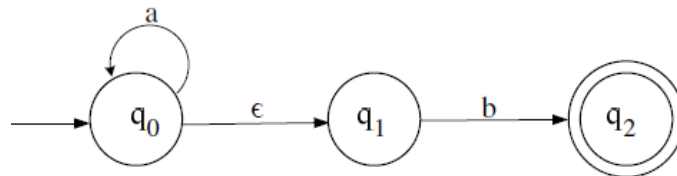
- ER :  $01^* + 0$

Contoh string yang dibangkitkan : 0, 01, 011, 0111, 01111

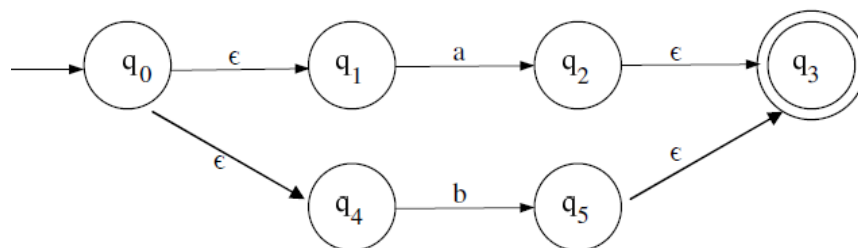
## 6.2 Hubungan Ekspresi Regular dan Finite State Automata



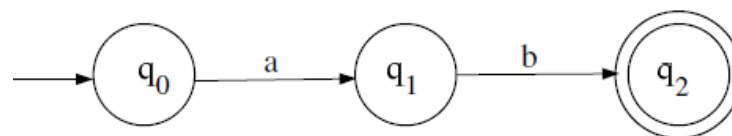
NFA  $\epsilon$  – move untuk ER :  $ab$



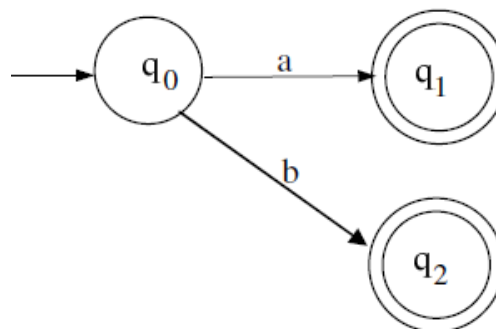
NFA  $\epsilon$  – move untuk ER :  $a^*b$



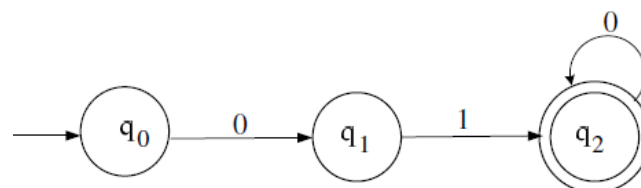
NFA  $\epsilon$  – move untuk ER :  $a \cup b$



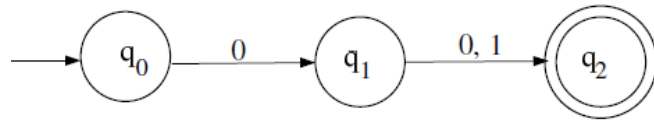
NFA untuk ER :  $ab$



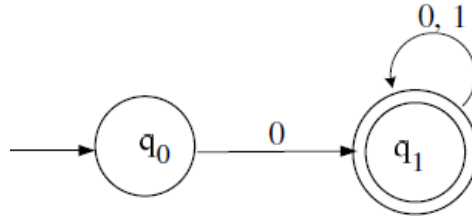
NFA untuk ER :  $a \cup b$



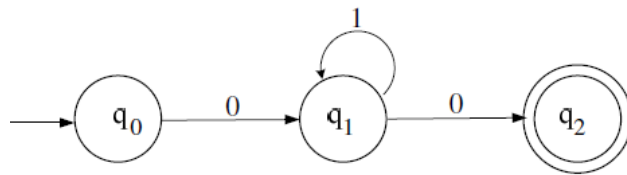
NFA untuk ER :  $010^*$



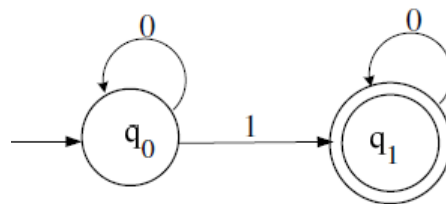
NFA untuk ER :  $0(1 \cup 0)$



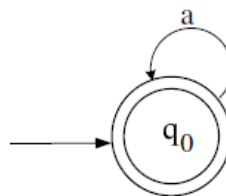
NFA untuk ER :  $0(1 \cup 0)^*$



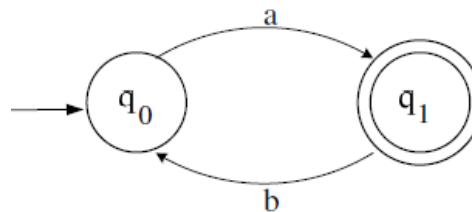
NFA untuk ER :  $01^*0$



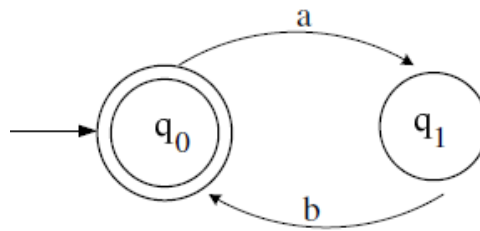
NFA untuk ER :  $0^*10^*$



NFA untuk ER :  $a^*$



NFA untuk ER :  $a(ba)^*$



NFA untuk ER :  $(ab)^*$

### 6.3 Aturan Produksi untuk suatu Tata Bahasa Regular

Batasan aturan produksi untuk bahasa regular :

$$\alpha \rightarrow \beta$$

Suatu tata bahasa (*grammar*) didefinisikan dengan 4 Tupel yaitu : V, T, P, dan S

Di mana,

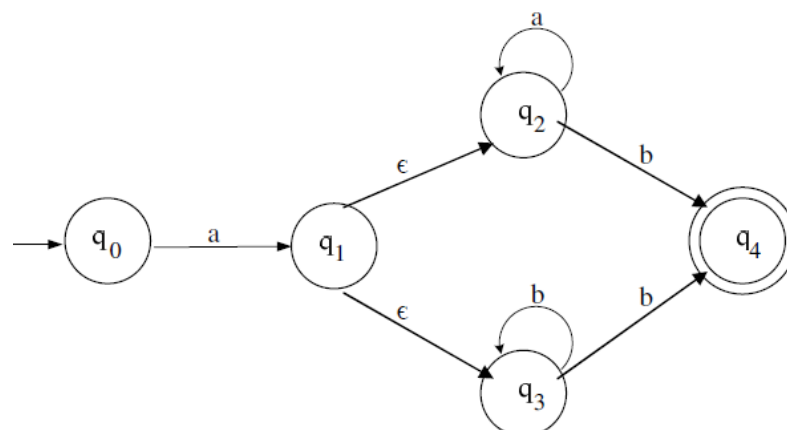
V = Himpunan simbol variabel / non terminal

T = Himpunan simbol terminal

P = Kumpulan aturan produksi

S = Simbol awal

- Sebagai contoh terdapat Mesin FSA berikut



Mesin *finite state automata* pada gambar di atas memiliki simbol input 'a' dan 'b'. Simbol 'a' dan 'b' akan menjadi simbol terminal pada aturan produksi yang akan kita bentuk.

Misalnya kita tentukan simbol awal adalah S. Kita identikkan S dengan *state* awal  $q_0$ .

Dari  $q_0$  mendapat *input* a menjadi  $q_1$ . Bisa kita tuliskan sebagai aturan produksi :

$$S \rightarrow aE$$

Di sini kita gunakan sebagai E dan bukan A karena menyatakan bagian yang belum

terbangkitkan mulai dari state  $q_1$ .

Dari  $q_1$  mendapat transisi  $\epsilon$  (tanpa menerima *input*) ke  $q_2$  dan  $q_3$ . Bisa kita tuliskan :

$$E \rightarrow A$$
$$E \rightarrow B$$

(Di sini kita identikkan  $q_2$  sebagai A dan  $q_3$  sebagai B)

Dari  $q_2$  jika mendapat input a menuju ke *state*  $q_2$  itu sendiri dan jika mendapat *input* b menuju ke *state*  $q_4$  yang merupakan *state* akhir dan tidak menuju ke *state* yang lainnya sehingga dapat dituliskan menjadi :

$$A \rightarrow aA$$

$$A \rightarrow b$$

Dari  $q_3$  jika mendapat input b menuju ke *state*  $q_3$  itu sendiri dan jika mendapat *input* b juga menuju ke *state*  $q_4$  yang merupakan *state* akhir dan tidak menuju ke *state* yang lainnya sehingga dapat dituliskan menjadi :

$$B \rightarrow bB$$

$$B \rightarrow b$$

Kumpulan aturan produksi yang kita peroleh bisa kita tuliskan sebagai berikut.

$$S \rightarrow aE$$

$$E \rightarrow A \mid B$$

$$A \rightarrow aA \mid b$$

$$B \rightarrow bB \mid b$$

Secara formal tata bahasa yang diperoleh dari otomata adalah sebagai berikut.

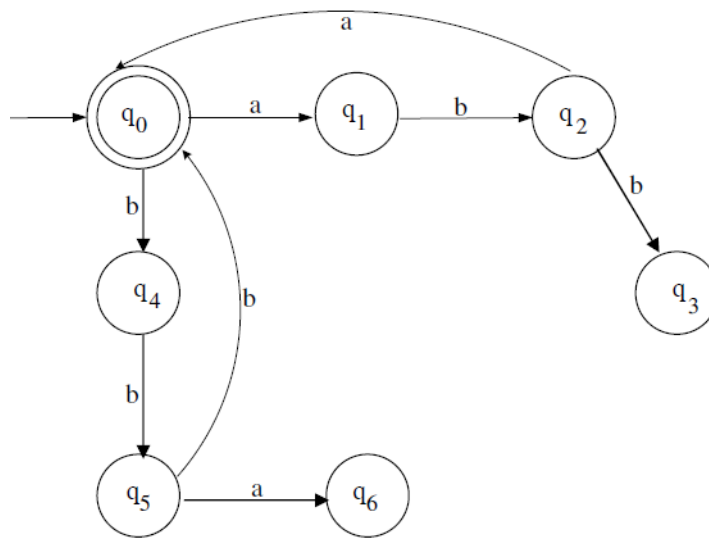
$$V = \{S, E, A, B\}$$

$$T = \{a, b\}$$

$$P = \{ S \rightarrow aE, E \rightarrow A \mid B, A \rightarrow aA \mid b, B \rightarrow bB \mid b \}$$

$$S = S$$

- Contoh lain dapat dilihat pada gambar di bawah ini.



Kita bisa mengkonstruksi aturan produksi untuk otomata tersebut.

$T = \{a, b\}$

$S = S$

Kita mulai dari *state* awal yaitu  $q_0$  yang dalam hal ini dilambangkan dengan  $S$ .

- ✓ Bila  $S$  mendapat input  $a$  maka menuju ke  $q_1$  yang dalam hal ini dilambangkan dengan  $A$ .

$S \rightarrow aA$

- ✓ Bila  $S$  mendapat input  $b$  maka menuju ke  $q_4$  yang dalam hal ini dilambangkan dengan  $B$ .

$S \rightarrow bB$

- ✓ Karena  $q_0$  dalam hal ini sebagai *state* akhir dan masih memiliki transisi keluar, maka untuk menandainya sebagai *state* akhir kita buat :

$S \rightarrow \epsilon$

Kemudian setelah itu kita lihat  $q_1$  yang tadi telah kita lambangkan sebagai  $A$ .

- ✓ Jika  $A$  mendapat input  $b$  maka menuju  $q_2$  yang dalam hal ini dilambangkan sebagai  $C$ .

$A \rightarrow bC$

Kemudian kita lihat  $q_4$  yang telah kita identikkan sebagai  $B$ .

- ✓ Jika  $B$  mendapat input  $b$  maka menuju ke  $q_5$  yang kita lambangkan sebagai  $D$ .

$B \rightarrow bD$

Kemudian kita lihat  $q_2$  yang telah kita lambangkan sebagai  $C$ .

- ✓ Jika C mendapat input b maka menuju ke q3 (Tetapi karena q3 tidak mempunyai transisi keluar dan bukan merupakan *state* akhir maka dapat kita abaikan).
- ✓ Jika C mendapat input a maka menuju ke S.

$$C \rightarrow aS$$

Kemudian kita lihat q5 yang telah kita lambangkan sebagai D.

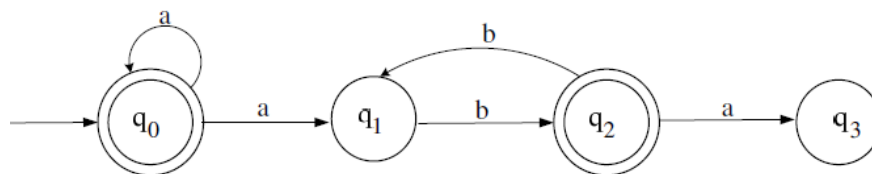
- ✓ Jika D mendapat input b maka menuju ke S.
- ✓ Jika D mendapat input a maka menuju ke q6 (tetapi karena q6 bukan merupakan *state* akhir dan tidak ada transisi keluar dari q6 maka dapat diabaikan)

Maka Diperoleh :

$$V = \{S, A, B, C, D\}$$

$$P = \{S \rightarrow aA \mid bB \mid \epsilon, A \rightarrow bC, B \rightarrow bD, C \rightarrow aS, D \rightarrow bS\}$$

- Contoh lain dapat dilihat pada gambar di bawah ini.



$$T = \{a, b\}$$

$$S = S$$

Kita mulai dari *state* awal yaitu q<sub>0</sub> yang dalam hal ini dilambangkan dengan S.

- ✓ Bila S mendapat input a maka menuju ke q<sub>1</sub> yang dalam hal ini dilambangkan dengan A.

$$S \rightarrow aA$$

- ✓ Bila S mendapat input a maka menuju ke S sendiri

$$S \rightarrow aS$$

- ✓ Karena S adalah *state* awal dan bukan satu-satunya *state* akhir maka tidak perlu ditambahkan  $S \rightarrow \epsilon$

Kemudian kita lihat q<sub>1</sub> yang dalam hal ini dilambangkan dengan A.

- ✓ Jika A mendapat input b maka menuju ke *state* q<sub>2</sub> yang dalam hal ini dilambangkan dengan B.

$$A \rightarrow bB$$

Kemudian kita lihat  $q_2$  yang dalam hal ini dilambangkan dengan B

- ✓ Jika B mendapat input b maka menuju ke state  $q_1$  yang dalam hal ini dilambangkan dengan A.

$B \rightarrow bA$

- ✓ Jika B mendapat input a maka menuju ke state  $q_3$ , tetapi karena  $q_3$  bukan merupakan state akhir dan tidak mempunyai transisi keluar maka dapat diabaikan.

- ✓ Karena B merupakan state akhir dan mempunyai transisi keluar maka untuk menandainya ditulis.

$B \rightarrow \epsilon$

Maka diperoleh :

$V = \{S, A, B\}$

$P = \{S \rightarrow aA \mid aS, A \rightarrow bB, B \rightarrow bA \mid \epsilon\}$

#### 6.4 Finite State Automata untuk Suatu Tata Bahasa Regular

Bila sebelumnya dari suatu diagram transisi *Finite State Automata* kita bisa membuat aturan – aturan produksinya, sebaliknya kita juga bisa mengkonstruksi diagram transisi *finite state automata* untuk suatu tata bahasa regular yang diketahui aturan – aturan produksinya.

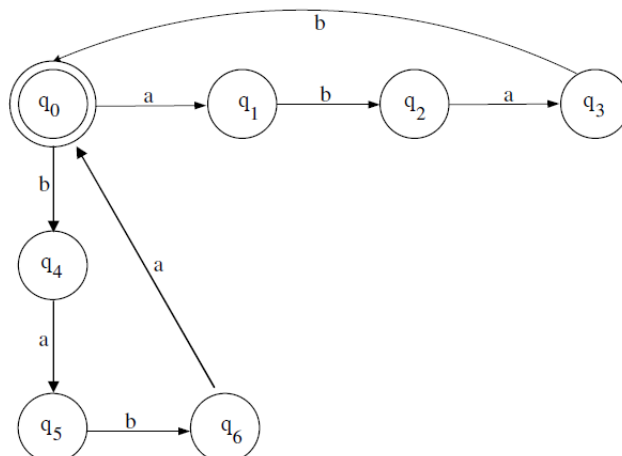
- Misalkan terdapat tata bahasa regular dengan aturan produksi.

$S \rightarrow aB \mid bA \mid \epsilon$

$A \rightarrow abaS$

$B \rightarrow babS$

Maka diagram transisinya dapat digambar sebagai berikut



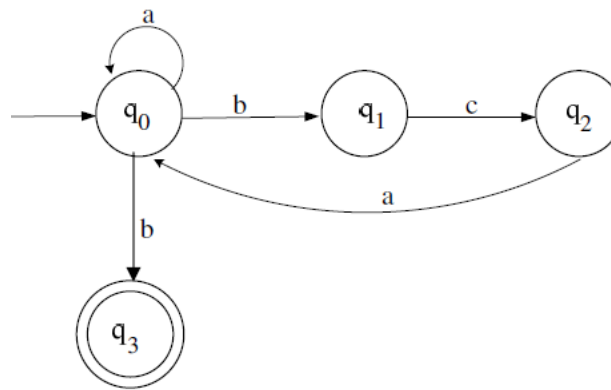


- Contoh lain, akan dibuat diagram transisi untuk tata bahasa regular :

$S \rightarrow aS \mid bB \mid b$

$B \rightarrow cC$

$C \rightarrow aS$



## LATIHAN SOAL

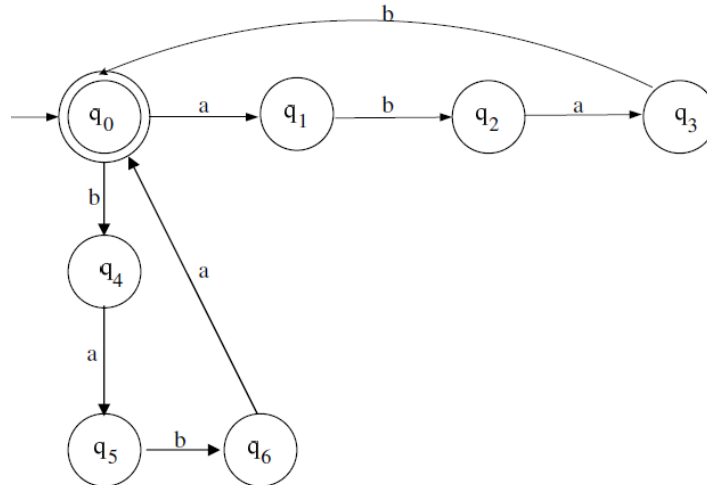
1. akan dibuat diagram transisi untuk tata bahasa regular :

$$S \rightarrow aB \mid bA \mid \epsilon$$

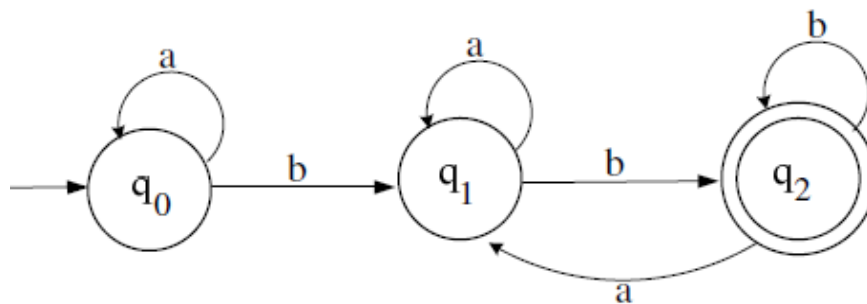
$$A \rightarrow abaS$$

$$B \rightarrow babS$$

2. Konstruksikanlah tata bahasa regular untuk automata berikut.

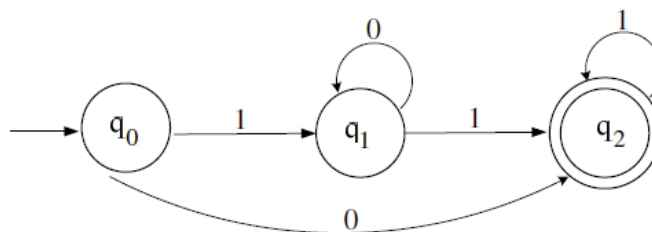


3. Konstruksikanlah tata bahasa regular untuk automata berikut

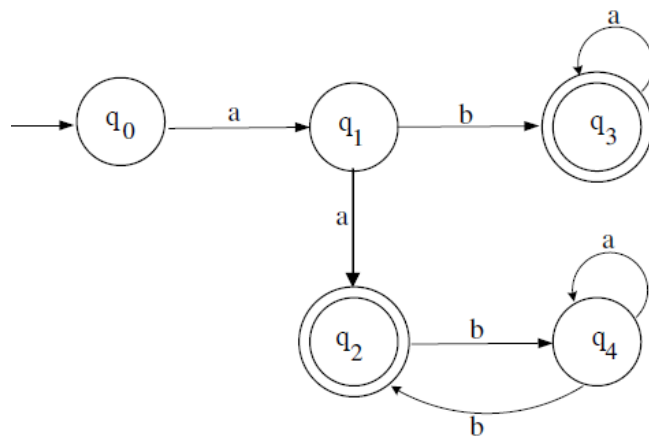


4. Deskripsikan himpunan string yang akan diterima oleh Finite State Automata seperti di bawah:

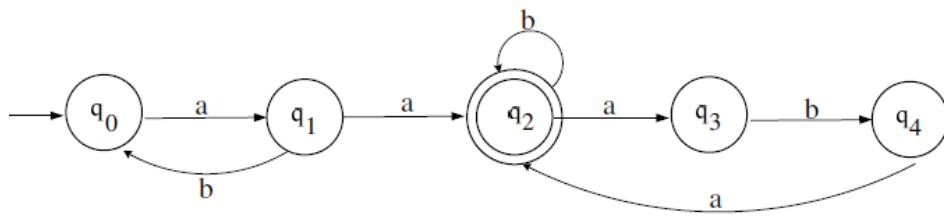
a.



b.



c.



## BAB VII

### TATA BAHASA BEBAS KONTEKS

#### 7.1 Free Contexts

Bila pada tata bahasa regular terdapat pembatasan pada ruas kanan atau hasil produksinya, maka pada tata bahasa bebas konteks / *Context Free Grammar*, selanjutnya

kita sebut sebagai CFG, tidak terdapat pembatasan hasil produksinya.

Sebagai contoh :

$$B \rightarrow CDeFg$$

$$D \rightarrow BcDe$$

#### 7.2 Pohon Penurunan (Derivation Tree)

Pohon penurunan (*derivation tree* / *parse tree*) berguna untuk menggambarkan bagaimana memperoleh suatu *string* (untai) dengan cara menurunkan simbol-simbol variabel menjadi simbol-simbol terminal. Setiap simbol variabel akan diturunkan menjadi terminal sampai tidak ada yang belum tergantikan.

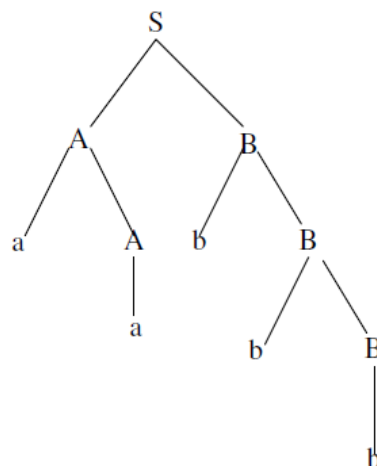
❖ Misalkan terdapat tata bahasa bebas konteks dengan aturan produksi :

$$S \rightarrow AB$$

$$A \rightarrow aA \mid a$$

$$B \rightarrow bB \mid b$$

Akan kita gambarkan pohon penurunan untuk memperoleh untai : 'aabbb'

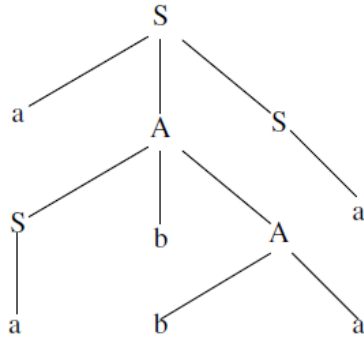


❖ Contoh lain, terdapat tata bahasa bebas konteks :

$$S \rightarrow aAS \mid a$$

$$A \rightarrow SbA \mid ba$$

Gambarkan pohon penurunan untuk memperoleh untai 'aabbaa'



❖ Contoh lain, terdapat tata bahasa bebas konteks memiliki aturan produksi sebagai berikut

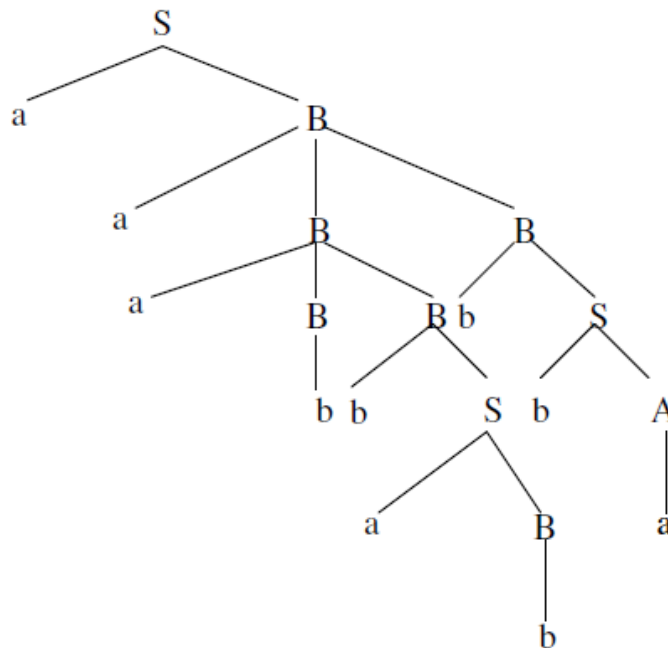
$$S \rightarrow aB \mid bA$$

$$A \rightarrow a \mid aS \mid bAA$$

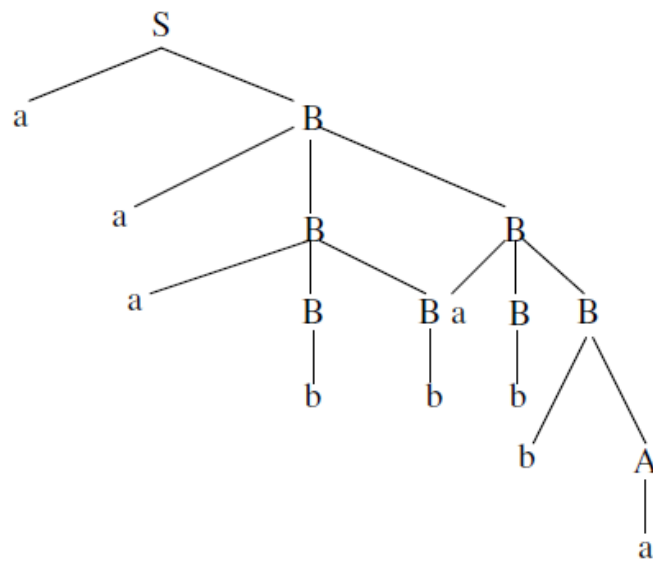
$$B \rightarrow b \mid bS \mid aBB$$

Gambarkan pohon penurunan untuk memperoleh untai 'aaabbabbba'

Jawab: Versi 1



Versi 2



### 7.3 Ambiguitas

Ambiguitas / kedwitarian terjadi bila terdapat lebih dari satu pohon penurunan yang berbeda untuk memperoleh suatu untai.

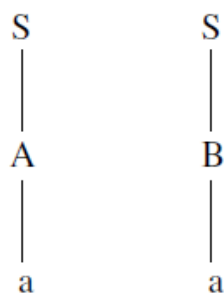
Misalkan terdapat bebas konteks :

$$S \rightarrow A \mid B$$

$$A \rightarrow a$$

$$B \rightarrow a$$

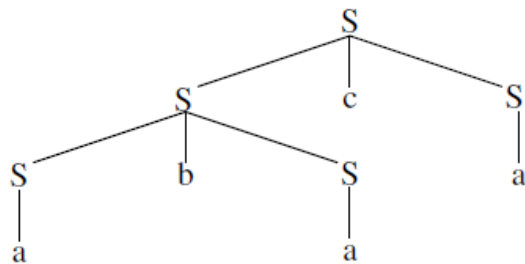
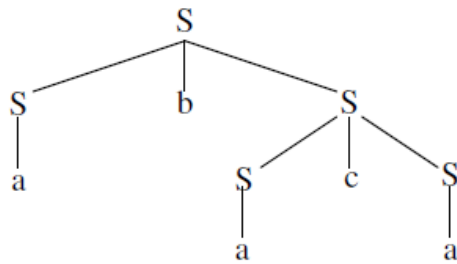
Untuk memperoleh suatu untai 'a' bisa terdapat dua cara penurunan seperti yang ditunjukkan pada pohon penurunan berikut ini.



Contoh lain, terdapat tata bahasa bebas konteks :

$$S \rightarrow SbS \mid ScS \mid a$$

Kita dapat memperoleh untai 'abaca' dalam dua cara berikut ini.



#### 7.4 Penyederhanaan Tata Bahasa Bebas Konteks

Penyederhanaan tata bahasa bebas konteks bertujuan untuk melakukan pembatasan sehingga tidak menghasilkan pohon penurunan yang memiliki kerumitan yang tak perlu atau aturan produksi yang tidak berarti. Misalkan terdapat tata bahasa bebas konteks :

$$S \rightarrow AB \mid a$$

$$A \rightarrow a$$

Kelemahan tata bahasa bebas konteks di atas, aturan produksi  $S \rightarrow AB$  tidak berarti karena B tidak memiliki penurunan.

Untuk tata bahasa bebas konteks berikut.

$$S \rightarrow A$$

$$A \rightarrow B$$

$$B \rightarrow C$$

$$C \rightarrow D$$

$$D \rightarrow a \mid A$$

Memiliki kelemahan terlalu panjang jalannya padahal berujung pada  $S \rightarrow a$ , produksi

$D \rightarrow A$  juga memiliki kerumitan.

Suatu tata bahasa bebas konteks dapat disederhanakan dengan melakukan cara berikut ini.

1. Penghilangan produksi *useless* (tidak berguna)
2. Penghilangan produksi unit.
3. Penghilangan produksi  $\epsilon$

#### A. Penghilangan Produksi Useless

Penghilangan produksi *useless* dilakukan dengan cara sebagai berikut.

1. Menghilangkan produksi yang memuat simbol variabel yang tidak memiliki penurunan yang akan menghasilkan simbol terminal.
2. Produksi yang tidak akan pernah dicapai dengan penurunan apapun dari simbol awal sehingga produksi itu redundan (berlebih).

❖ Contoh, terdapat tata bahasa bebas konteks :

$$S \rightarrow aSa \mid Abd \mid Bde$$
$$A \rightarrow Ada$$
$$B \rightarrow BBB \mid a$$

Kita bisa melihat bahwa :

1. Simbol variabel A tidak memiliki penurunan yang menuju terminal, sehingga bisa dihilangkan.
2. Karena simbol A telah dihilangkan, maka dengan sendirinya  $S \rightarrow Abd$  juga dihilangkan.

Maka dari tata bahasa bebas konteks di atas, produksi yang *useless* :

$$A \rightarrow Ada$$
$$S \rightarrow Abd$$

Maka tata bahasa bebas konteks setelah disederhanakan menjadi :

$$S \rightarrow aSa \mid Bde$$
$$B \rightarrow BBB \mid a$$

❖ Contoh lain, terdapat tata bahasa bebas konteks sebagai berikut.

$$S \rightarrow Aa \mid B$$
$$A \rightarrow ab \mid D$$



$$B \rightarrow b \mid E$$

$$C \rightarrow bb$$

$$E \rightarrow aEa$$

Kita bisa melihat bahwa :

1. Aturan produksi  $A \rightarrow D$ , simbol variabel  $D$  tidak memiliki penurunan, sehingga bisa dihilangkan
2. Aturan produksi  $C \rightarrow bb$ , bila kita coba melakukan penurunan dari simbol awal  $S$ , dengan jalan manapun tidak akan pernah mencapai  $C$ , sehingga bisa dihilangkan.
3. Simbol variabel  $E$  tidak memiliki aturan produksi yang menuju terminal, sehingga bisa dihilangkan.
4. Konsekuensi dari aturan no. 3 maka aturan produksi  $B \rightarrow E$ , juga mesti dihilangkan.

Maka dari tata bahasa bebas konteks tersebut produksi yang *useless* adalah sebagai berikut.

$$A \rightarrow D$$

$$C \rightarrow bb$$

$$E \rightarrow aEa$$

$$B \rightarrow E$$

Maka tata bahasa bebas konteks setelah disederhanakan menjadi :

$$S \rightarrow Aa \mid B$$

$$A \rightarrow ab$$

$$B \rightarrow b$$

❖ Contoh lain, terdapat tata bahasa bebas konteks berikut.

$$S \rightarrow aAb \mid cEB$$

$$A \rightarrow dBE \mid eeC$$

$$B \rightarrow ff$$

$$C \rightarrow ae$$

$$D \rightarrow h$$

Kita bisa melihat bahwa :

1.  $S \rightarrow cEB$ ,  $E$  tidak memiliki penurunan, sehingga bisa dihilangkan.

2.  $A \rightarrow dBE$ , E tidak memiliki penurunan sehingga bisa dihilangkan.
3.  $D \rightarrow h$ , tidak dapat dicapai melalui penurunan manapun sehingga redundan.
4.  $B \rightarrow ff$ , karena  $S \rightarrow cEB$  dan  $A \rightarrow dBE$  sudah dihilangkan, maka tidak akan dapat dicapai melalui penurunan manapun sehingga bisa dihilangkan.

Maka aturan produksi yang *useless* adalah sebagai berikut.

$$S \rightarrow cEB$$

$$A \rightarrow dBE$$

$$D \rightarrow h$$

$$B \rightarrow ff$$

Sehingga tata bahasa bebas konteks hasil penyederhanaan adalah sebagai berikut.

$$S \rightarrow aAb$$

$$A \rightarrow eeC$$

$$C \rightarrow ae$$

## B. Penghilangan Produksi Unit

Produksi unit adalah produksi di mana ruas kiri dan kanan aturan produksi hanya berupa satu simbol variabel, misalkan :  $A \rightarrow B$ ,  $C \rightarrow D$ . Keberadaan produksi unit membuat tata bahasa memiliki kerumitan yang tak perlu atau menambah panjang penurunan. Penyederhanaan ini dilakukan dengan melakukan penggantian aturan produksi unit.

❖ Contoh tata bahasa bebas konteks :

$$S \rightarrow Sb$$

$$S \rightarrow C$$

$$C \rightarrow D$$

$$C \rightarrow ef$$

$$D \rightarrow dd$$

Langkah-langkahnya adalah sebagai berikut :

1. Kita lihat untuk  $S \rightarrow C$

$C \rightarrow D$  lalu  $D \rightarrow dd$  sehingga  $C \rightarrow dd$

$C \rightarrow ef$

Maka  $S \rightarrow C$  dapat diubah menjadi  $S \rightarrow dd \mid ef$

2. Kita lihat untuk  $C \rightarrow D$

$C \rightarrow D$  lalu  $D \rightarrow dd$  sehingga  $C \rightarrow dd$

Maka  $C \rightarrow D$  dapat diubah menjadi  $C \rightarrow dd$

Sehingga tata bahasa bebas konteks setelah penyederhanaan adalah sebagai berikut.

$S \rightarrow Sb$

$S \rightarrow dd \mid ef$

$C \rightarrow dd$

$C \rightarrow ef$

$D \rightarrow dd$

❖ Contoh lain, terdapat tata bahasa bebas konteks :

$S \rightarrow A$

$S \rightarrow Aa$

$A \rightarrow B$

$B \rightarrow C$

$B \rightarrow b$

$C \rightarrow D$

$C \rightarrow ab$

$D \rightarrow b$

Penggantian yang dilakukan :

1.  $S \rightarrow A$

Untuk  $S \rightarrow A$  maka setelah ditelusuri menghasilkan  $S \rightarrow b \mid ab$

2.  $A \rightarrow B$

Untuk  $A \rightarrow B$  maka setelah ditelusuri menghasilkan  $A \rightarrow b \mid ab$

3.  $B \rightarrow C$

Untuk  $B \rightarrow C$  maka setelah ditelusuri menghasilkan  $B \rightarrow b \mid ab$

Untuk  $B \rightarrow b$  sudah ada, maka cukup ditulis  $B \rightarrow ab$

4.  $C \rightarrow D$

Untuk  $C \rightarrow D$  maka setelah ditelusuri menghasilkan  $C \rightarrow b \mid ab$  Untuk  $C \rightarrow$

$ab$  sudah ada, maka cukup ditulis  $C \rightarrow b$

Sehingga tata bahasa bebas konteks setelah disederhanakan menjadi :

$S \rightarrow ab \mid b$

$S \rightarrow Aa$

$A \rightarrow ab \mid b$

$B \rightarrow ab$

$B \rightarrow b$

$C \rightarrow b$

$C \rightarrow ab$

$D \rightarrow b$

❖ Contoh lain, terdapat tata bahasa bebas konteks :

$S \rightarrow Cba \mid D$

$A \rightarrow bbC$

$B \rightarrow Sc \mid ddd$

$C \rightarrow eA \mid f \mid C$

$D \rightarrow E \mid SABC$

$E \rightarrow gh$

Penggantian yang dilakukan :

1.  $D \rightarrow E$  menjadi  $D \rightarrow gh$

2.  $C \rightarrow C$ , kita hapus

3.  $S \rightarrow D$  menjadi  $S \rightarrow gh \mid SABC$

Sehingga tata bahasa bebas konteks setelah penyederhanaan menjadi :

$S \rightarrow Cba \mid gh \mid SABC$

$$A \rightarrow bbC$$

$$B \rightarrow Sc \mid ddd$$

$$C \rightarrow eA \mid f$$

$$D \rightarrow gh \mid SABC$$

$$E \rightarrow gh$$

### C. Penghilangan Produksi $\epsilon$

Produksi  $\epsilon$  adalah produksi dalam bentuk :

$$\alpha \rightarrow \epsilon$$

atau bisa dianggap sebagai produksi kosong (*empty*). Penghilangan produksi dilakukan dengan melakukan penggantian produksi yang memuat variabel yang bisa menuju ke produksi  $\epsilon$ , atau biasa disebut *nullable*. Prinsip penggantinya bisa dilihat pada kasus berikut ini.

$$S \rightarrow bcAd$$

$$A \rightarrow \epsilon$$

Pada kasus diatas  $A$  *nullable*, serta  $A \rightarrow \epsilon$  satu-satunya produksi dari  $A$ , maka variabel  $A$  bisa ditiadakan, hasil penyederhanaan tata bahasa bebas konteks menjadi:

$$S \rightarrow bcd$$

Tetapi bila kasusnya :

$$S \rightarrow bcAd$$

$$A \rightarrow bd \mid \epsilon$$

Pada kasus di atas  $A$  *nullable*, tetapi  $A \rightarrow \epsilon$  bukan satu-satunya produksi dari  $A$ , maka hasil penyederhanaan :

$$S \rightarrow bcAd \mid bcd$$

$$A \rightarrow bd$$

❖ Contoh, terdapat tata bahasa bebas konteks :

$$S \rightarrow Ab \mid Cd$$

$$A \rightarrow d$$

$$C \rightarrow \epsilon$$

Variabel yang *nullable* adalah variabel C. Karena penurunan  $C \rightarrow \epsilon$  merupakan penurunan satu-satunya dari C, maka kita ganti  $S \rightarrow Cd$  menjadi  $S \rightarrow d$ .

Kemudian produksi  $C \rightarrow \epsilon$  kita hapus.

Tata bahasa bebas konteks setelah penyederhanaan :

$$S \rightarrow Ab \mid d$$

$$A \rightarrow d$$

❖ Contoh lain, terdapat tata bahasa bebas konteks :

$$S \rightarrow dA \mid Bd$$

$$A \rightarrow bc$$

$$A \rightarrow \epsilon$$

$$B \rightarrow c$$

Variabel yang *nullable* adalah variabel A.  $A \rightarrow \epsilon$  bukan penurunan satu satunya dari A (terdapat  $A \rightarrow bc$ ), maka kita ganti  $S \rightarrow dA$  menjadi  $S \rightarrow dA \mid d$ .  $A \rightarrow \epsilon$  kita hapus.

Setelah penyederhanaan :

$$S \rightarrow dA \mid d \mid Bd$$

$$A \rightarrow bc$$

$$B \rightarrow c$$

❖ Contoh lain, terdapat tata bahasa bebas konteks :

$$S \rightarrow AaCD$$

$$A \rightarrow CD \mid AB$$

$$B \rightarrow b \mid \epsilon$$

$$C \rightarrow d \mid \epsilon$$

$$D \rightarrow \epsilon$$

Variabel yang *nullable* adalah variabel B, C, D. Kemudian kita lihat  $A \rightarrow CD$ ,

maka variabel A juga *nullable*, karena D hanya memiliki penurunan  $D \rightarrow \epsilon$ ,

maka kita

sederhanakan dulu :

$$S \rightarrow AaCD \text{ menjadi } S \rightarrow AaC$$

$$A \rightarrow CD \text{ menjadi } A \rightarrow C$$

$$D \rightarrow \epsilon \text{ kita hapus}$$

Selanjutnya kita lihat variabel B dan C memiliki penurunan  $\epsilon$ , meskipun bukan

satu-satunya penurunan, maka kita lakukan penggantian :

$$A \rightarrow AB \text{ menjadi } A \rightarrow AB \mid A \mid B$$

$$S \rightarrow AaC \text{ menjadi } S \rightarrow AaC \mid aC \mid Aa \mid a$$

$$B \rightarrow \epsilon \text{ dan } C \rightarrow \epsilon \text{ kita hapus}$$

Setelah penyederhanaan :

$$S \rightarrow AaC \mid aC \mid Aa \mid a$$

$$A \rightarrow C \mid AB \mid A \mid B$$

$$B \rightarrow b$$

$$C \rightarrow d$$

❖ Contoh lain, terdapat tata bahasa bebas konteks :

$$S \rightarrow AB$$

$$A \rightarrow abB \mid aCa \mid \epsilon$$

$$B \rightarrow bA \mid BB \mid \epsilon$$

$$C \rightarrow \epsilon$$

Variabel yang *nullable* adalah A, B, dan C. Dari  $S \rightarrow AB$ , maka S juga *nullable*.

Kita lakukan penggantian :

$S \rightarrow AB$  menjadi  $S \rightarrow AB \mid A \mid B$

$A \rightarrow abB$  menjadi  $A \rightarrow abB \mid ab$

$A \rightarrow aCa$  menjadi  $A \rightarrow aa$

$B \rightarrow bA$  menjadi  $B \rightarrow bA \mid b$

$B \rightarrow BB$  menjadi  $B \rightarrow BB \mid B$

$C \rightarrow \epsilon$ ,  $B \rightarrow \epsilon$ , dan  $A \rightarrow \epsilon$  kita hapus.

Hasil akhir penyederhanaan menjadi :

$S \rightarrow AB \mid A \mid B$

$A \rightarrow abB \mid ab \mid aa$

$B \rightarrow bA \mid b \mid BB \mid B$

❖ Contoh lain, terdapat tata bahasa bebas konteks :

$S \rightarrow aAb$

$A \rightarrow aAb \mid \epsilon$

Hasil akhir penyederhanaan menjadi :

$S \rightarrow aAb \mid ab$

$A \rightarrow aAb \mid ab$

❖ Contoh lain, terdapat tata bahasa bebas konteks :

$S \rightarrow AbaC$

$A \rightarrow BC$

$B \rightarrow b \mid \epsilon$

$C \rightarrow D \mid \epsilon$

$D \rightarrow d$

Hasil penyederhanaan :



$$S \rightarrow ABaC \mid AaC \mid Aba \mid BaC \mid aC \mid Aa \mid Ba \mid a$$

$$A \rightarrow BC \mid B \mid C$$

$$B \rightarrow b$$

$$C \rightarrow D$$

$$D \rightarrow d$$

Pada prakteknya, ketiga penyederhanaan tersebut (penghilangan *useless*, unit, dan  $\epsilon$ ) dilakukan bersama pada suatu tata bahasa bebas konteks, yang nantinya menyiapkan tata bahasa bebas konteks tersebut untuk diubah ke dalam suatu *bentuk normal chomsky* yang akan dibahas pada bagian selanjutnya.

Adapun urutan pengerjaannya adalah sebagai berikut.

1. Hilangkan produksi  $\epsilon$
2. Hilangkan produksi unit
3. Hilangkan produksi *useless*.

❖ Kita coba untuk melakukan ketiga penyederhanaan tersebut pada aturan produksi berikut.

$$S \rightarrow AA \mid C \mid bd$$

$$A \rightarrow Bb \mid \epsilon$$

$$B \rightarrow AB \mid d$$

$$C \rightarrow de$$

Pertama – tama kita lakukan penghilangan produksi  $\epsilon$ , sehingga aturan produksi menjadi:

$$S \rightarrow AA \mid A \mid C \mid bd$$

$$A \rightarrow Bb$$

$$B \rightarrow AB \mid B \mid d$$

$$C \rightarrow de$$

Anda bisa melihat bahwa penghilangan produksi  $\epsilon$  berpotensi untuk menghasilkan produksi unit baru yang sebelumnya tidak ada. Selanjutnya kita lakukan penghilangan produksi unit menjadi :

$$S \rightarrow AA \mid Bb \mid de \mid bd$$

$$A \rightarrow Bb$$

$$B \rightarrow AB \mid d$$

$$C \rightarrow de$$

Anda bisa melihat bahwa penghilangan produksi unit bisa menghasilkan produksi *useless*. Terakhir kita lakukan penghilangan produksi *useless* :

$$S \rightarrow AA \mid Bb \mid de \mid bd$$

$$A \rightarrow Bb$$

$$B \rightarrow AB \mid d$$

## LATIHAN SOAL

1. Untuk tata bahasa bebas konteks berikut.

$$S \rightarrow AA$$

$$A \rightarrow AAA \mid a \mid bA \mid Ab$$

Gambarkan pohon penurunan untuk memperoleh untai 'bbabaaba'

2. Untuk tata bahasa bebas konteks berikut.

$$S \rightarrow aAd \mid aB$$

$$A \rightarrow b \mid c$$

$$B \rightarrow ccd \mid ddc$$

Gambarkan pohon penurunan untuk memperoleh untai 'accd'

3. Untuk tata bahasa bebas konteks berikut.

$$S \rightarrow AB$$

$$A \rightarrow Aa \mid bB$$

$$B \rightarrow a \mid Sb$$

Berikanlah pohon penurunan untuk memperoleh untai 'baabaab'

4. Untuk tata bahasa bebas konteks berikut.

Gambarkan pohon penurunan untuk memperoleh untai 'bbaaaabb'

5. Buktikan bahwa tata bahasa bebas konteks berikut ambigu :

$$S \rightarrow aB \mid bA$$

$$A \rightarrow a \mid aS \mid bAA$$

$$B \rightarrow b \mid bS \mid aBB$$

6. Sederhanakan aturan produksi yang *useless* tata bahasa bebas konteks berikut.

$$S \rightarrow aB$$

$$A \rightarrow bcD \mid dAC$$

$$B \rightarrow e \mid Ab$$

$$C \rightarrow bCb \mid adF \mid ab$$

$$F \rightarrow cFB$$

7. Sederhanakan aturan produksi yang *useless* tata bahasa bebas konteks berikut.

$$S \rightarrow aBD$$

$$B \rightarrow cD \mid Ab$$

$$D \rightarrow ef$$

$$A \rightarrow Ed$$

$$F \rightarrow dc$$

8. Hilangkanlah semua aturan produksi yang *useless* dari tata bahasa bebas konteks berikut

$$S \rightarrow AB \mid CA$$

$$B \rightarrow BC \mid AB$$

$$A \rightarrow a$$

$$C \rightarrow aB \mid b$$

9. Hilangkan semua aturan produksi yang *useless* dari tata bahasa bebas konteks berikut.

$$S \rightarrow aS \mid A \mid C$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

$$C \rightarrow aCb$$

10. Hilangkan semua aturan produksi yang *useless* dari tata bahasa bebas konteks berikut.

$$S \rightarrow A$$

$$A \rightarrow aA \mid \epsilon$$

$$B \rightarrow bA$$

11. Hilangkanlah semua aturan produksi unit dari tata bahasa bebas konteks berikut.

$$S \rightarrow Aa \mid B$$

$$B \rightarrow A \mid bb$$

$$A \rightarrow a \mid bc \mid B$$

12. Hilangkan semua aturan produksi unit dari tata bahasa bebas konteks berikut.

$$S \rightarrow AbaC \mid BaC \mid AaC \mid Aba \mid aC \mid Aa \mid Ba \mid a$$

$$A \rightarrow B \mid C \mid BC$$

$$B \rightarrow b$$

$$C \rightarrow D$$

$$D \rightarrow d$$

13. Hilangkanlah semua produksi  $\epsilon$  dari tata bahasa bebas konteks berikut.

$$S \rightarrow AbaC$$

$$A \rightarrow Bd$$

$$B \rightarrow b \mid \epsilon$$

$$C \rightarrow D \mid \epsilon$$

$$D \rightarrow Bca$$

14. Hilangkanlah semua produksi  $\epsilon$  dari tata bahasa bebas konteks berikut.

$$S \rightarrow AaB \mid aaB$$

$$A \rightarrow \epsilon$$

$$B \rightarrow bbA \mid \epsilon$$

15. Hilangkanlah semua produksi  $\epsilon$  dari tata bahasa bebas konteks berikut.

$$S \rightarrow aSb \mid SS \mid \epsilon$$

16. Hilangkanlah semua produksi dari tata bahasa bebas konteks berikut.

$$S \rightarrow AB$$

$$A \rightarrow aA \mid abB \mid aCa$$

$$B \rightarrow bA \mid BB \mid \epsilon$$

$$C \rightarrow \epsilon$$

$$D \rightarrow dB \mid BCB$$

17. Lakukan penghilangan produksi unit, *useless*, dan  $\epsilon$  dari tata bahasa bebas konteks berikut.

$$S \rightarrow a \mid aA \mid B \mid C$$

$$A \rightarrow aB \mid \epsilon$$

$$B \rightarrow Aa$$

$$C \rightarrow cCD$$

$$D \rightarrow ddd$$

18. . Lakukan penghilangan produksi unit, *useless*, dan  $\epsilon$  dari tata bahasa bebas konteks berikut.

$$S \rightarrow aB \mid aaB$$

$$A \rightarrow \epsilon$$

$$B \rightarrow bA$$

$$B \rightarrow \epsilon$$

19. . Hilangkan semua aturan produksi *useless* dan unit dari tata bahasa bebas konteks berikut.

$$S \rightarrow AaC \mid aC \mid Aa \mid a$$

$$A \rightarrow C \mid AB \mid A \mid B$$

$$B \rightarrow b$$

$$C \rightarrow d$$

## BAB VIII

### TATA BAHASA BEBAS KONTEKS

#### 8.1 Bentuk Normal Chomsky

*Bentuk Normal Chomsky* (Chomsky Normal Form / CNF) merupakan salah satu bentuk normal yang sangat berguna untuk tata bahasa bebas konteks (CFG). *Bentuk Normal Chomsky* dapat dibuat dari sebuah tata bahasa bebas konteks yang telah mengalami penyederhanaan, yaitu penghilangan produksi *useless*, unit, dan  $\epsilon$ .

Aturan produksi dalam *bentuk normal Chomsky* ruas kanannya tepat berupa sebuah terminal atau dua variabel. Misalkan :

$$A \rightarrow BC$$

$$A \rightarrow b$$

$$B \rightarrow a$$

$$C \rightarrow BA \mid d$$

##### 8.1.1 Pembentuk Bentuk Normal Chomsky

Langkah – langkah pembentukan *bentuk normal Chomsky* secara umum sebagai berikut.

- Biarkan aturan produksi yang sudah dalam bentuk *normal Chomsky*.
- Lakukan penggantian aturan produksi yang ruas kanannya memuat simbol terminal dan panjang ruas kanan  $> 1$ .
- Lakukan penggantian aturan produksi yang ruas kanannya memuat  $> 2$  simbol variabel.
- Penggantian – penggantian tersebut bisa dilakukan berkali – kali sampai akhirnya semua aturan produksi dalam *bentuk normal Chomsky*.
- Selama dilakukan penggantian, kemungkinan kita akan memperoleh aturan – aturan produksi baru, dan juga memunculkan simbol – simbol variabel baru.

1) Contoh tata bahasa konteks sebagai berikut.

$$S \rightarrow bA \mid aB$$

$$A \rightarrow bAA \mid aS \mid a$$

$$B \rightarrow aBB \mid bS \mid b$$

Aturan produksi yang sudah dalam bentuk normal Chomsky adalah sebagai berikut.

$$A \rightarrow a$$

$$B \rightarrow b$$

Dilakukan penggantian aturan produksi yang belum *bentuk normal Chomsky*.

$$S \rightarrow bA \text{ menjadi } S \rightarrow P_1 A$$

$$S \rightarrow aB \text{ menjadi } S \rightarrow P_2 B$$

$$A \rightarrow bAA \text{ menjadi } A \rightarrow P_1 AA \text{ menjadi } A \rightarrow P_1 P_3$$

$$A \rightarrow aS \text{ menjadi } A \rightarrow P_2 S$$

$$B \rightarrow aBB \text{ menjadi } B \rightarrow P_2 BB \text{ menjadi } B \rightarrow P_2 P_4$$

$$B \rightarrow bS \text{ menjadi } B \rightarrow P_1 S$$

Terbentuk aturan produksi dan simbol variabel baru :

$$P_1 \rightarrow b$$

$$P_2 \rightarrow a$$

$$P_3 \rightarrow AA$$

$$P_1 \rightarrow BB$$

Hasil akhir aturan produksi dalam bentuk normal Chomsky adalah sebagai berikut.

$$A \rightarrow a$$

$$B \rightarrow b$$

$$S \rightarrow P_1 A$$

$$S \rightarrow P_2 B$$

$$A \rightarrow P_1 P_3$$

$$A \rightarrow P_2 S$$

$$B \rightarrow P_2 P_4$$

$$B \rightarrow P_1 S$$

$$P_1 \rightarrow b$$

$$P_2 \rightarrow a$$

$$P_3 \rightarrow AA$$

$$P_4 \rightarrow BB$$

2) Contoh lain, tata bahasa bebas konteks :

$$S \rightarrow aB \mid CA$$

$$A \rightarrow a \mid bc$$

$$B \rightarrow BC \mid Ab$$

$$C \rightarrow aB \mid b$$

Aturan produksi yang sudah dalam bentuk normal Chomsky :

$$S \rightarrow CA$$

$$A \rightarrow a$$

$$B \rightarrow BC$$

$$C \rightarrow b$$

Penggantian aturan produksi yang belum dalam bentuk normal Chomsky :

$$S \rightarrow aB \text{ menjadi } S \rightarrow P_1 B$$

$$A \rightarrow bc \text{ menjadi } S \rightarrow P_2 P_3$$

$$B \rightarrow Ab \text{ menjadi } B \rightarrow A P_2$$

$$C \rightarrow aB \text{ menjadi } C \rightarrow P_1 B$$

Terbentuk aturan produksi dan simbol variabel baru :

$$P_1 \rightarrow a$$

$$P_2 \rightarrow b$$

$$P_3 \rightarrow c$$

Hasil akhir aturan produksi dalam bentuk normal Chomsky adalah sebagai berikut.

$$S \rightarrow CA$$

$$A \rightarrow a$$

$$B \rightarrow BC$$

$$C \rightarrow b$$

$$S \rightarrow P_1 B$$

$$S \rightarrow P_2 P_3$$

$$B \rightarrow A P_2$$

$$C \rightarrow P_1 B$$



$$P_1 \rightarrow a$$

$$P_2 \rightarrow b$$

$$P_3 \rightarrow c$$

3) Contoh, tata bahasa bebas konteks :

$$S \rightarrow aAB \mid ch \mid CD$$

$$A \rightarrow dbE \mid eEC$$

$$B \rightarrow ff \mid DD$$

$$C \rightarrow ADB \mid aS$$

$$D \rightarrow i$$

$$E \rightarrow jD$$

Aturan produksi yang sudah dalam bentuk normal Chomsky :

$$S \rightarrow CD$$

$$B \rightarrow DD$$

$$D \rightarrow i$$

Penggantian aturan produksi :

$$S \rightarrow aAB \text{ menjadi } S \rightarrow P_1 P_2$$

$$S \rightarrow ch \text{ menjadi } S \rightarrow P_3 P_4$$

$$A \rightarrow dbE \text{ menjadi } A \rightarrow P_5 P_6$$

$$A \rightarrow eEC \text{ menjadi } A \rightarrow P_8 P_9$$

$$B \rightarrow ff \text{ menjadi } B \rightarrow P_{10} P_{10}$$

$$C \rightarrow ADB \text{ menjadi } C \rightarrow A P_{11}$$

$$C \rightarrow aS \text{ menjadi } C \rightarrow P_1 S$$

$$E \rightarrow jD \text{ menjadi } E \rightarrow P_{12} D$$

Terbentuk aturan produksi baru :

$$P_1 \rightarrow A$$

$$P_2 \rightarrow AB$$

$$P_3 \rightarrow c$$

$$P_4 \rightarrow h$$

$$\begin{aligned}
P_5 &\rightarrow d \\
P_6 &\rightarrow P_7 E \\
P_7 &\rightarrow b \\
P_8 &\rightarrow e \\
P_9 &\rightarrow EC \\
P_{10} &\rightarrow f \\
P_{11} &\rightarrow DB \\
P_{12} &\rightarrow j
\end{aligned}$$

Hasil akhir dalam bentuk normal Chomsky adalah sebagai berikut.

$$\begin{aligned}
S &\rightarrow CD \\
B &\rightarrow DD \\
D &\rightarrow i \\
S &\rightarrow P_1 P_2 \\
S &\rightarrow P_3 P_4 \\
A &\rightarrow P_5 P_6 \\
A &\rightarrow P_8 P_9 \\
B &\rightarrow P_{10} P_{10} \\
C &\rightarrow A P_{11} \\
C &\rightarrow P_i S \\
E &\rightarrow P_{12} D \\
P_1 &\rightarrow A \\
P_2 &\rightarrow AB \\
\\ 
P_3 &\rightarrow c \\
P_4 &\rightarrow h \\
P_5 &\rightarrow d \\
P_6 &\rightarrow P_7 E \\
P_7 &\rightarrow b \\
P_8 &\rightarrow e \\
P_9 &\rightarrow EC \\
P_{10} &\rightarrow f \\
P_{11} &\rightarrow DB \\
P_{12} &\rightarrow j
\end{aligned}$$

## LATIHAN SOAL

1. Transformasikan tata bahasa bebas konteks berikut ke dalam bentuk normal Chomsky:

$$S \rightarrow aSb \mid ab$$

2. Transformasikan tata bahasa bebas konteks berikut ke dalam bentuk normal Chomsky:

$$S \rightarrow aSaA \mid A$$

$$A \rightarrow abA \mid b$$

3. Transformasikan tata bahasa bebas konteks berikut ke dalam bentuk normal Chomsky:

$$S \rightarrow abAB$$

$$A \rightarrow bAB \mid \epsilon$$

$$B \rightarrow Baa \mid A \mid \epsilon$$

## BAB IX

### MESIN LAIN

#### 9.1 Mesin Moore

Mesin *Moore* adalah suatu *Finite State Automata* yang memiliki keputusan beberapa keluaran / *output*.

Mesin Moore didefinisikan dalam 6 (enam) tuple,  $M = (Q, \Sigma, \delta, S, \Delta, \lambda)$ , di mana:

$Q$  = himpunan *state*

$\Sigma$  = himpunan simbol *input*

$\delta$  = fungsi transisi

$S$  = *state* awal

$\Delta$  = himpunan *output*

$\lambda$  = fungsi *output* untuk setiap *state*

Kita lihat contoh penerapan dari Mesin Moore. Misal kita ingin memperoleh pembagian (modulus) suatu bilangan dengan 3. Di mana *input* dinyatakan dalam biner. Mesin Moore yang bersesuaian bisa dilihat pada gambar di bawah ini. Konfigurasi mesinnya adalah sebagai berikut.

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0,1\}$  (*input* dalam biner)

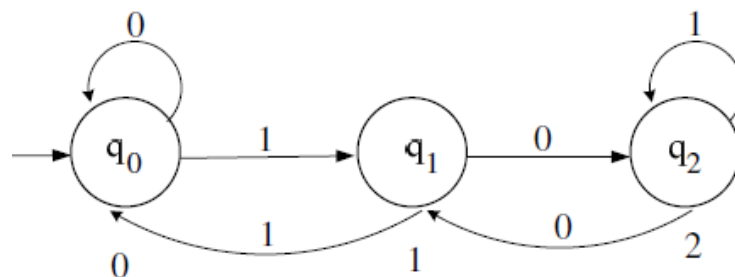
$\Delta = \{0, 1, 2\}$  (untuk *output*-nya pada kasus *mod* dengan 3, maka sisanya kemungkinan adalah 0, 1, 2)

$S = q_0$

$\lambda(q_0) = 0$

$\lambda(q_1) = 1$

$\lambda(q_2) = 2$



Gambar Mesin Moore untuk modulus 3

Misalkan saja :

- $5 \bmod 3 = ?$

Input 5 dalam biner 101

Bila kita masukkan 101 ke dalam mesin, urutan *state* yang dicapai :  $q_0, q_1, q_2, q_2$

Perhatikan state terakhir yang dicapai adalah  $q_2$ ,  $\lambda(q_2) = 2$ , maka  $5 \bmod 3 = 2$

- $10 \bmod 3 = ?$

Input 10 dalam biner 1010

Bila kita masukkan ke dalam mesin, urutan *state* yang dicapai :  $q_0, q_1, q_2, q_2, q_1$

$\lambda(q_1) = 1$ , maka  $10 \bmod 3 = 1$

## 9.2 Mesin Mealy

Bila *output* pada mesin Moore berasosiasi dengan *state*, maka *output* pada mesin Mealy akan berasosiasi dengan transisi. Mesin Mealy sendiri didefinisikan dalam 6 tupel,

$M = (Q, \Sigma, \delta, S, \Delta, \lambda)$ , di mana :

$Q$  = himpunan *state*

$\Sigma$  = himpunan simbol *input*

$\delta$  = fungsi transisi

$S$  = state awal

$\Delta$  = himpunan *output*

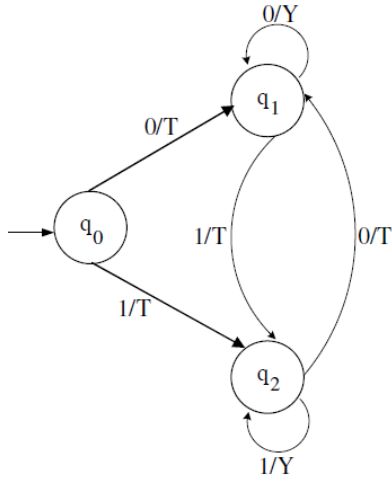
$\lambda$  = fungsi *output* untuk setiap transisi

Contoh penerapan Mesin Mealy dapat dilihat pada gambar di bawah ini.

Mesin itu akan mengeluarkan *output* apakah menerima (Y) atau menolak (T), suatu masukan, di mana mesin akan mengeluarkan output 'Y' bila menerima untai yang memiliki akhiran 2 simbol berturutan yang sama.

Contoh input yang diterima :

01011, 01100, 1010100, 10110100, 00, 11, 100, 011, 000, 111, dll.



Konfigurasi dari mesin Mealy tersebut adalah sebagai berikut.

$$Q = \{ q_0, q_1, q_2 \}$$

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

$$\Delta = \{Y, T\}$$

$$S = q_0$$

$$\lambda(q_0, 0) = T$$

$$\lambda(q_0, 1) = T$$

$$\lambda(q_1, 0) = Y$$

$$\lambda(q_1, 1) = T$$

$$\lambda(q_2, 0) = T$$

$$\lambda(q_2, 1) = Y$$

### 9.3 Ekuivalensi Mesin Moore dan Mesin Mealy

Dari suatu mesin Moore dapat dibuat mesin Mealy yang ekuivalen, begitu juga sebaliknya. Untuk mesin Mealy pada contoh sebelumnya dapat kita buat mesin Moore yang ekuivalen. Bisa kita lihat bahwa *state* pada mesin Moore dibentuk dari kombinasi *state* pada mesin Mealy dan banyaknya *output*. Karena jumlah *state* pada mesin Mealy = 3 dan jumlah *output* = 2, maka jumlah *state* pada mesin Moore yang ekuivalen = 6. Konfigurasi mesin Moore yang dibentuk adalah sebagai berikut

$$Q = \{ q_0Y, q_0T, q_1Y, q_1T, q_2Y, q_2T \}$$

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

$$\Delta = \{Y,T\}$$

$$S = q_0T$$

$$\lambda(q_0Y) = Y$$

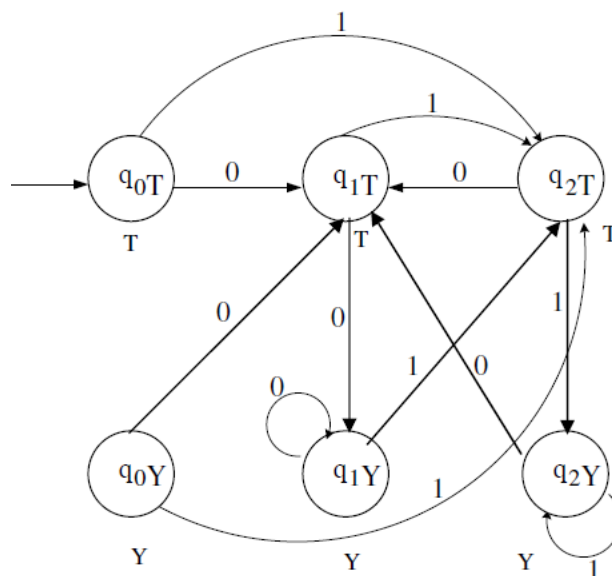
$$\lambda(q_0T) = T$$

$$\lambda(q_1Y) = Y$$

$$\lambda(q_1T) = T$$

$$\lambda(q_2Y) = Y$$

$$\lambda(q_2T) = T$$



Untuk memperoleh ekuivalensi mesin Mealy dari suatu mesin Moore caranya lebih mudah, cukup dengan menambahkan label *output* ke setiap transisi, dan menghapus label *output* pada setiap *state*. Gambar berikut merupakan mesin Mealy yang ekuivalen dengan mesin Moore.

Konfigurasi mesin Mealy tersebut adalah sebagai berikut.

$$Q = \{q_0, q_1, q_2\}$$

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

$$\Delta = \{0, 1, 2\}$$

$$S = q_0$$

$$\lambda(q_0, 0) = 0$$

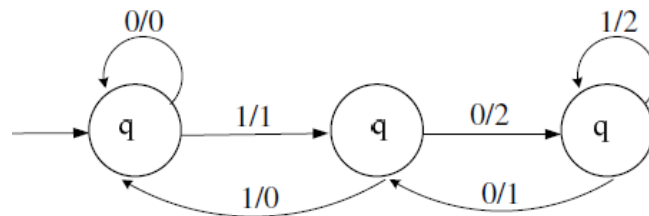
$$\lambda(q_0, 1) = 1$$

$$\lambda(q_1, 0) = 2$$

$$\lambda(q_1, 1) = 0$$

$$\lambda(q_2, 0) = 1$$

$$\lambda(q_2, 1) = 2$$



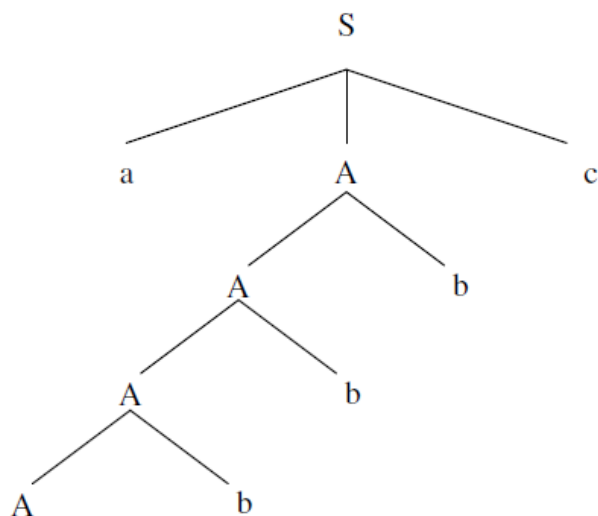
### 9.3.1 Penghilangan Rekursive kiri

Aturan produksi yang rekursif memiliki ruas kanan (hasil produksi) yang memuat simbol variabel pada ruas kiri. Produksi yang rekursif kiri menyebabkan pohon penurunan tumbuh ke kiri. Dalam banyak penerapan tata bahasa, rekursif kiri tak diinginkan. Untuk menghindari penurunan yang bisa mengakibatkan *loop* kita perlu menghilangkan sifat rekursif kiri dari aturan produksi. Penghilangan rekursif kiri di sini memungkinkan tata bahasa bebas konteks nantinya diubah ke dalam *bentuk normal greibach*.

❖ Contoh aturan produksi yang rekursif kiri :

$$S \rightarrow aAc$$

$$A \rightarrow Ab \mid \epsilon$$



Langkah – langkah penghilangan rekursif kiri adalah sebagai berikut



- Pisahkan aturan produksi yang rekursif kiri dan yang tidak.
- Lakukan penggantian aturan produksi yang rekursif kiri, menjadi sebagai berikut.

$$1) A \rightarrow \beta_1 Z \mid \beta_2 Z \mid \dots \mid \beta_m Z$$

$$2) Z \rightarrow \alpha_1 \mid \alpha_2 \mid \alpha_3 \mid \dots \mid \alpha_n$$

$$3) Z \rightarrow \alpha_1 Z \mid \alpha_2 Z \mid \alpha_3 Z \mid \dots \mid \alpha_n Z$$

Penggantian di atas dilakukan untuk setiap aturan produksi dengan simbol ruas kiri yang sama, bisa muncul simbol variabel baru  $Z_1$ ,  $Z_2$ , dan seterusnya, sesuai banyaknya variabel yang menghasilkan produksi yang rekursif kiri.

- Hasil akhir berupa aturan produksi pengganti ditambah dengan aturan produksi semula yang tidak rekursif kiri.

Contoh, tata bahasa bebas konteks :

$$S \rightarrow Sab \mid aSc \mid dd \mid ff \mid Sbd$$

Pertama – tama kita lakukan pemisahan aturan produksi.

Aturan produksi yang rekursif kiri :

$$S \rightarrow Sab \mid Sbd$$

Aturan produksi yang tidak rekursif kiri :

$$S \rightarrow aSc \mid dd \mid ff$$

Kita lakukan penggantian aturan produksi yang rekursif kiri.

$$Z_1 \rightarrow ab \mid bd$$

$$Z_1 \rightarrow abZ_1 \mid bdZ_1$$

Untuk aturan produksi yang tidak rekursif kiri diubah menjadi :

$$S \rightarrow aScZ_1 \mid ddZ_1 \mid ffZ_1$$

Hasil akhir setelah penghilangan rekursif kiri adalah sebagai berikut.

$$S \rightarrow aSc \mid dd \mid ff$$

$$S \rightarrow aScZ_1 \mid ddZ_1 \mid ffZ_1$$

$$Z_1 \rightarrow ab \mid bd$$

$$Z_1 \rightarrow abZ_1 \mid bdZ_1$$

❖ **Contoh lain, terdapat tata bahasa bebas konteks :**

$$S \rightarrow Sab \mid Sb \mid cA$$

$$A \rightarrow Aa \mid a \mid bd$$

Pertama – tama kita lakukan pemisahan aturan produksi.

Aturan produksi yang rekursif kiri :

$$S \rightarrow Sab \mid Sb$$

$$A \rightarrow Aa$$

Aturan produksi yang tidak rekursif kiri :

$$S \rightarrow cA$$

$$A \rightarrow a \mid bd$$

Kita lakukan penggantian untuk aturan produksi yang rekursif kiri untuk yang memiliki

simbol ruas kiri S.

$$Z_1 \rightarrow ab \mid b$$

$$Z_1 \rightarrow abZ_1 \mid bZ_1$$

Kita lakukan penggantian untuk aturan produksi yang rekursif kiri untuk yang memiliki

simbol ruas kiri A.

$$Z_2 \rightarrow a$$

$$Z_2 \rightarrow aZ_2$$

Untuk aturan produksi yang tidak rekursif diubah menjadi.

$$S \rightarrow cAZ_1$$

$$A \rightarrow a Z_2 \mid bdZ_2$$

Hasil akhir setelah penghilangan rekursif kiri adalah sebagai berikut.

$$S \rightarrow cA$$

$$A \rightarrow a \mid bd$$

$$S \rightarrow cAZ_1$$

$$Z_1 \rightarrow ab \mid b$$

$$Z_1 \rightarrow abZ_1 \mid bZ_1$$

$$A \rightarrow aZ_2 \mid bdZ_2$$

$$Z_2 \rightarrow a$$

$$Z_2 \rightarrow aZ_2$$

**Contoh lain, terdapat tata bahasa bebas konteks :**

$$S \rightarrow Sa \mid aAc \mid c \mid \epsilon$$

$$A \rightarrow Ab \mid ba$$

Pertama – tama kita lakukan pemisahan aturan produksi.

Aturan produksi yang rekursif kiri.

$$S \rightarrow Sa$$

$$A \rightarrow Ab$$

Aturan produksi yang tidak rekursif kiri.

$$S \rightarrow aAc \mid c \mid \epsilon$$

$$A \rightarrow ba$$

Kita lakukan penggantian aturan produksi yang rekursif kiri untuk yang memiliki simbol ruas kiri S.

$$Z_1 \rightarrow a$$

$$Z_1 \rightarrow aZ_1$$

Kita lakukan penggantian aturan produksi yang rekursif kiri untuk yang memiliki simbol ruas kiri A.

$$Z_2 \rightarrow b$$

$$Z_2 \rightarrow bZ_2$$

Untuk aturan produksi yang tidak rekursif diubah menjadi :

$$S \rightarrow aAcZ_1 \mid cZ_1 \mid Z_1$$

$$A \rightarrow baZ_2$$

Hasil akhir setelah penghilangan rekursif kiri adalah sebagai berikut.

$$S \rightarrow aAc \mid c \mid \epsilon$$

$$S \rightarrow aAcZ_1 \mid cZ_1 \mid Z_1$$

$$A \rightarrow ba$$

$$A \rightarrow baZ_2$$

$$Z_1 \rightarrow a$$

$$Z_1 \rightarrow aZ_1$$

$$Z_2 \rightarrow b$$

$$Z_2 \rightarrow bZ_2$$

#### 9.4 Bentuk Normal Greibach

Bentuk Normal *Greibach* / *Greibach Normal Form* adalah suatu tata bahasa bebas konteks di mana hasil produksinya (ruas kanan) diawali dengan satu simbol terminal, selanjutnya bisa diikuti dengan rangkaian simbol variabel. Contoh tata bahasa bebas konteks dalam bentuk normal *Greibach*.

$$S \rightarrow a \mid aAB$$

$$A \rightarrow aB$$

$$B \rightarrow cS$$

Untuk dapat diubah ke dalam bentuk normal *Greibach*, tata bahasa semula harus memenuhi syarat – syarat sebagai berikut.

- udah dalam bentuk normal *chomsky*
- Tidak bersifat rekursif kiri
- Tidak menghasilkan  $\epsilon$

Secara umum langkah – langkah untuk mendapatkan *bentuk normal Greibach* adalah sebagai berikut.

1. Tentukan urutan simbol – simbol variabel yang ada dalam tata bahasa. Misalkan, terdapat  $m$  variabel dengan urutan  $A_1, A_1, \dots, A_m$
2. Berdasarkan urutan simbol yang ditetapkan pada langkah (1) seluruh aturan produksi yang ruas kanannya diawali dengan simbol variabel dapat dituliskan dalam bentuk :

$$A_h \rightarrow A_i \gamma$$

Di mana  $h < i$  (rekursif kiri sudah dihilangkan),  $\gamma$  bisa berupa simbol – simbol variabel.

- a. Jika  $h < i$ , aturan produksi ini sudah benar (tidak perlu diubah).
- b. Jika  $h > i$ , aturan produksi belum benar. Lakukan substitusi berulang – ulang terhadap  $A_i$  (ganti  $A_i$  pada produksi ini dengan ruas kanan produksi dari variabel  $A_i$ ) sehingga suatu saat diperoleh produksi dalam bentuk :

$$A_h \rightarrow A_p \gamma \text{ ( di mana } h \leq p \text{ )}$$

- i) Jika  $h = p$ , lakukan penghilangan rekursif kiri.
  - ii) Jika  $h < p$ , aturan produksi sudah benar.
3. Jika terjadi penghilangan rekursif kiri pada tahap (2b), sejumlah simbol variabel baru yang muncul dari operasi ini dapat disisipkan pada urutan variabel semula di mana saja asalkan ditempatkan tidak sebelum  $A_h$  ( di kiri ).
  4. Setelah langkah (2) & (3) dikerjakan, maka aturan – aturan produksi yang ruas kanannya dimulai simbol variabel sudah berada dalam urutan yang benar.

$$A_x \rightarrow A_y \gamma \text{ (di mana } x < y \text{ )}.$$

Produksi – produksi yang lain ada dalam bentuk :

$$A_x \rightarrow a \gamma \text{ (a = simbol terminal)}$$

$$B_x \rightarrow \gamma$$

(  $B_x$  = simbol variabel baru yang muncul sebagai akibat dari operasi penghilangan rekursif kiri).

5. Bentuk normal *Greibach* diperoleh dengan cara melakukan substitusi mundur mulai dari variabel  $A_m$ , lalu  $A_{m-1}$ ,  $A_{m-1}$ , ... Dengan cara ini aturan produksi dalam bentuk  $A_x \rightarrow A_y \gamma$  dapat diubah sehingga ruas kanannya dimulai dengan simbol terminal.

6. Produksi yang dalam bentuk  $B_x \rightarrow \gamma$  juga dapat diubah dengan cara substitusi seperti pada langkah (5).

Contoh (tata bahasa bebas konteks sudah dalam bentuk normal *Chomsky* dan memenuhi syarat untuk diubah ke bentuk normal *Greibach*), simbol awal adalah S :

$$S \rightarrow CA$$

$$A \rightarrow a \mid d$$

$$B \rightarrow b$$

$$C \rightarrow DD$$

$$D \rightarrow AB$$

Kita tentukan urutan simbol variabel, misalnya S, A, B, C, D ( $S < A < B < C < D$ ).

Kita periksa aturan produksi yang simbol pertama pada ruas kanan adalah simbol variabel, apakah sudah memenuhi ketentuan urutan variabel :

- $S \rightarrow CA$  (sudah memenuhi karena  $S < C$ )
- $C \rightarrow DD$  (sudah memenuhi karena  $C < D$ )
- $D \rightarrow AB$  (tidak memenuhi karena  $D > A$ )

Yang belum memenuhi urutan yang telah kita tentukan adalah :

$$D \rightarrow AB$$

Karena ruas kiri  $>$  simbol pertama pada ruas kanan. Maka kita lakukan substitusi pada simbol variabel A, aturan produksi menjadi :

$$D \rightarrow aB \mid dB$$

Setelah semua aturan produksi sudah memenuhi ketentuan urutan variabel, kita lakukan substitusi mundur pada aturan produksi yang belum dalam bentuk normal *Greibach*.

- $C \rightarrow DD$  diubah menjadi  $C \rightarrow aBD \mid dBD$
- $S \rightarrow CA$  diubah menjadi  $S \rightarrow aBDA \mid dBDA$

Perhatikan bahwa substitusi mundur dimulai dari aturan produksi yang memiliki ruas kiri dengan urutan variabel paling akhir (kasus di atas :  $S < A < B < C < D$ , maka C lebih dulu disubstitusi daripada S).

Hasil akhir aturan produksi yang sudah dalam bentuk normal *Greibach* adalah sebagai berikut.

$$S \rightarrow aBDA \mid dBDA$$

$$A \rightarrow a \mid d$$

$$B \rightarrow b$$

$$C \rightarrow aBD \mid dBD$$

$$D \rightarrow aB \mid dB$$

**Contoh lain (simbol Awal A) :**

$$A \rightarrow BC$$

$$B \rightarrow CA \mid b$$

$$C \rightarrow AB \mid a$$

Kita tentukan urutan simbol : A, B, C ( $A < B < C$ ).

$A \rightarrow BC$  (sudah memenuhi karena  $A < B$ )

$B \rightarrow CA$  (sudah memenuhi karena  $B < C$ )

$C \rightarrow AB$  (Padahal  $C > A$  sehingga harus diubah)

Yang belum memenuhi urutan yang telah kita tentukan adalah :

$$C \rightarrow AB$$

Karena ruas kiri  $>$  simbol pertama pada ruas kanan. Maka kita lakukan substitusi pada simbol variabel A, aturan produksi menjadi :

$$C \rightarrow AB \text{ diubah menjadi } C \rightarrow BCB \text{ diubah lagi menjadi } C \rightarrow bCB$$

Setelah semua aturan produksi sudah memenuhi ketentuan urutan variabel, kita lakukan substitusi mundur pada aturan produksi yang belum dalam bentuk normal *Greibach*.

- $B \rightarrow CA$  diubah menjadi  $B \rightarrow bCBA$
- $A \rightarrow BC$  diubah menjadi  $A \rightarrow bCBAC$

Hasil akhir aturan produksi yang sudah dalam bentuk normal *Greibach* adalah sebagai berikut.

$$A \rightarrow \text{bCBAC}$$

$$B \rightarrow \text{bCBA} \mid \text{b}$$

$$C \rightarrow \text{bCB} \mid \text{a}$$



## SOAL LATIHAN

Rancanglah mesin *moore* untuk perhitungan :

1. Modulus 2
2. Modulus 4
3. Modulus 5
4. Modulus 6
5. Lakukan penghilangan rekursif kiri pada tata bahasa bebas konteks berikut.

$$A \rightarrow Aa \mid aBc$$

6. Lakukan penghilangan rekursif kiri pada tata bahasa bebas konteks berikut.

$$A \rightarrow AbAB \mid \epsilon$$

$$B \rightarrow Baa \mid A \mid \epsilon$$

7. Lakukan penghilangan rekursif kiri pada tata bahasa bebas konteks berikut.

$$S \rightarrow Sba \mid Ab$$

$$A \rightarrow Sa \mid Aab \mid a$$

$$B \rightarrow Sb \mid Bba \mid b$$

8. Lakukan penghilangan rekursif kiri pada tata bahasa bebas konteks berikut.

$$S \rightarrow SSC \mid SSB \mid abg$$

$$B \rightarrow abc \mid BSb \mid BCd$$

$$C \rightarrow ab$$

9. Buatlah bentuk normal *Greibach* dari tata bahasa bebas konteks berikut (tata bahasa bebas konteks sudah dalam bentuk normal *Chomsky* dan memenuhi syarat untuk diubah ke GNF ).

$$S \rightarrow AS \mid a$$

$$A \rightarrow a$$

10. Buatlah bentuk normal *Greibach* dari tata bahasa bebas konteks berikut.

$$S \rightarrow AA \mid d$$

$$A \rightarrow SS \mid b$$

11. Buatlah bentuk normal *Greibach* dari tata bahasa bebas konteks berikut.

$$S \rightarrow a \mid AS \mid AA$$

$$A \rightarrow SA \mid a$$