MATERI KULIAH

TEORI BAHASA DAN OTOMATA

Oleh :

R. Joko Musridho, S.T., M.Phil.

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAHLAWAN TUANKU TAMBUSAI
2021

PERTEMUAN I
Teori Bahasa dan Otomata

Buku
Teori Bahasa dan Otomata, Firrar Utdirartatmo
An Introduction to Formal Language and Automata, Peter Linz

Otomata
Arti menurut American Heritage Dictionary:
1. a robot
2. one that behaves in an automatic or mechanical fashion

Arti dalam dunia matematika
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

Teori Otomata dan bahasa formal, berkaitan dalam hal :

- 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.

Bahasa Formal
Suatu kalimat dibentuk dengan menerapkan serangkaian aturan produksi pada sebuah simbol 'akar'. Proses penerapan aturan produksi dapat digambarkan sebagai suatu diagram pohon.

Teori dasar
Def. 1 sebuah string dengan panjang n yang dibentuk dari himpunan A adalah barisan dari n simbol

$a_1a_2...a_n \qquad a_i \in A$

Panjang string x dituliskan dengan |x|

Def 2. String kosong (null string), dilambangkan dengan ε adalah untaian dengan panjang 0 dan tidak berisi apapun. Panjang string x dituliskan dengan |x|

Def 3. dua buah string $a = a_1a_2...a_m$ dan $b=b_1b_2...b_n$ dapat disambungkan menjadi string c dengan panjang m+n sebagai berikut $c = a_1a_2...a_mb_1b_2...b_n$
Operasi penyambungan tersebut dapat pula diterapkan pada himpunan

$Z=XY = \{st \mid s \in X \wedge t \in Y\}$

Def 4. (Closure) . A^n adalah himpunan string dengan panjang n yang dibentuk dari simbol-simbol di himpunan simbol/alfabet A:
Transitif Closure/Kleen Closure adalah himpunan seluruh string yang dapat dibentuk dari A dengan berbagai panjang

$A^* = A^0 \cup A^1 \cup A^2 \cup A^3 \cup ...$

Jika string kosong dikeluarkan , akan diperoleh positive closure

$$A^+ = A^1 \cup A^2 \cup A^3 \cup ...$$

Tata Bahasa

Aturan yang disebutkan pada proses pengenalan dan pembangkitan kalimat.
Secara formal, tata bahasa terdiri dari 4 komponen yaitu :

1. Himpunan berhingga, tidak kosong dari simbol-simbol non terminal T^1
2. Himpunan berhingga, dari simbol-simbol non-terminal N
3. Simbol awal $S \in N$, yang merupakan salah satu anggota dari himpunan simbol non-terminal.
4. Himpunan berhingga aturan produksi P yang setiap elemennya dituliskan dalam bentuk :

$$\alpha \rightarrow \beta$$

dimana α dan β adalah string yang dibentuk dari himpunan $T \cup N$ dan α harus berisi paling sedikit satu simbol non-terminal.
Keempat komponen tersebut sering dituliskan sbb :

$$G = (T,N,S,P)$$

Bahasa yang dihasilkan oleh G ditulis sebagai L(G), yaitu himpunan string yang dapat diturunkan dari simbol awal S dengan menerapkan aturan-aturan produksi yang terdapat pada P.

Aturan Produksi

Aturan produksi $\alpha \rightarrow \beta$ yang diterapkan pada suatu string $w=a\alpha c$ mengganti kemunculan. α menjadi β, sehingga string tersebut berubah menjadi $w=a\beta c$, sehingga dapat dituliskan $a\alpha c \Rightarrow a\beta c$ ($a\alpha c$ memproduksi $a\beta c$).
Produksi tersebut dapat diterapkan berkali-kali

$$w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow ... \Rightarrow w_n$$

atau dapat dituliskan

$$w_1 \Rightarrow^* w_n$$

jika minimal harus ada 1 aturan produksi yang diterapkan :

$$w_1 \Rightarrow^+ w_n$$

Contoh 1

Tatabahasa G = {{S} , {a,b}, S , P } dengan aturan produksi P adalah

$$S \rightarrow aSb$$

$$S \rightarrow \varepsilon$$

maka dapat dihasilkan suatu string

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

sehingga dapat dituliskan

$$S \Rightarrow^* aabb$$

Bahasa yang dihasilkan dari tatabahasa tersebut adalah

$$L(G) = \{ \varepsilon , ab, aabb , aaabbb , aaaabbbb, ... \}$$

atau dapat pula dituliskan

$$L(G) = \{a^n b^n \mid n \geq 0 \}$$

Contoh 2

Tatabahasa G = {{S,A} , {a,b}, S , P } dengan aturan produksi P adalah

$S \rightarrow Ab$

$A \rightarrow aAb$

$A \rightarrow \varepsilon$

maka dapat dihasilkan suatu string

$S \Rightarrow Ab \Rightarrow b$

$S \Rightarrow Ab \Rightarrow aAbb \Rightarrow abb$

$S \Rightarrow Ab \Rightarrow aAbb \Rightarrow aaAbbb \Rightarrow aaAbbb$

Bahasa yang dihasilkan dari tatabahasa tersebut adalah

$L(G) = \{ b , abb, aabbb , aaabbbb , aaaabbbbb, ... \}$

atau dapat pula dituliskan

$L(G) = \{a^n b^{n+1} \mid n \geq 0 \}$

Hirarki Bahasa

Kelas	Mesin pengenal
Regular language	Finite State Automata
Context free language	Push Down Automata
Context sensitive language	Linear Bounded Automata
Unrestricted language	Turing Machine

Kelas	Ruas kiri	Ruas Kanan	Contoh
Regular	$\alpha \in N$	$\leq$ 1 non terminal (paling kiri/kanan)	$P \rightarrow abR$ $Q \rightarrow abc$ $R \rightarrow Scac$
Context free	$\alpha \in N$	-	$P \rightarrow aQb$ $Q \rightarrow abPRS$
Context sensitive	$\alpha \in (T \cup N)^+$	$\lvert\alpha\rvert \leq \lvert\beta\rvert$	$aD \rightarrow Da$ $AD \rightarrow aCD$
Unrestricted	$\alpha \in (T \cup N)^+$	-	$CB \rightarrow DB$ $ADc \rightarrow \varepsilon$

NB : Ruas kiri harus memuat simbol non-terminal

Pelajari sendiri

Teori Himpunan

Relasi dan fungsi

Teori Pembuktian

Graph dan Tree

PR /Latihan di buku Firrar, bab I

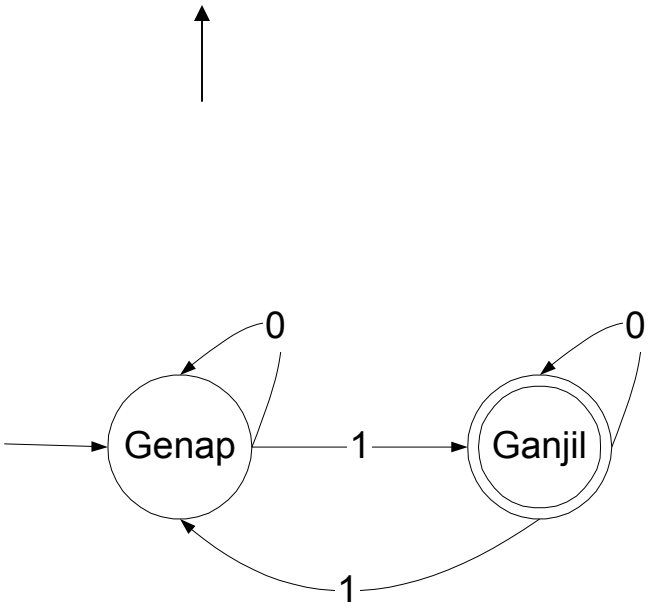
PERTEMUAN II
Finite State Automata (FSA)

- model matematika yang dapat menerima input dan mengeluarkan output
- Memiliki state yang berhingga banyaknya dan dapat berpindah dari satu state ke state lainnya berdasar input dan fungsi transisi
- Tidak memiliki tempat penyimpanan/memory, hanya bisa mengingat state terkini.
- Mekanisme kerja dapat diaplikasikan pada : elevator, text editor, analisa leksikal, pencek parity.

0	1	0	1	1	0	1

FA

Contoh pencek parity ganjil



Misal input : 1101

Genap 1 Ganjil 1 Genap 0 Genap 1 Ganjil
diterima mesin

Misal input : 1100

Genap 1 Ganjil 1 Genap 0 Genap 0 Genap
ditolak mesin

Def 1. Finite State Automata dinyatakan oleh 5 tuple

$M=(Q , \Sigma , \delta , S , F)$

Q = himpunan state

Σ = himpunan simbol input

δ = fungsi transisi $\delta : Q \times \Sigma$
S = state awal / initial state , $S \in Q$
F = state akhir, $F \subseteq Q$

Contoh diatas,
Q = {Genap, Ganjil}
Σ = {0,1}
S = Genap
F = {Ganjil }

δ	0	1
Genap	Genap	Ganjil
Ganjil	Ganjil	Genap

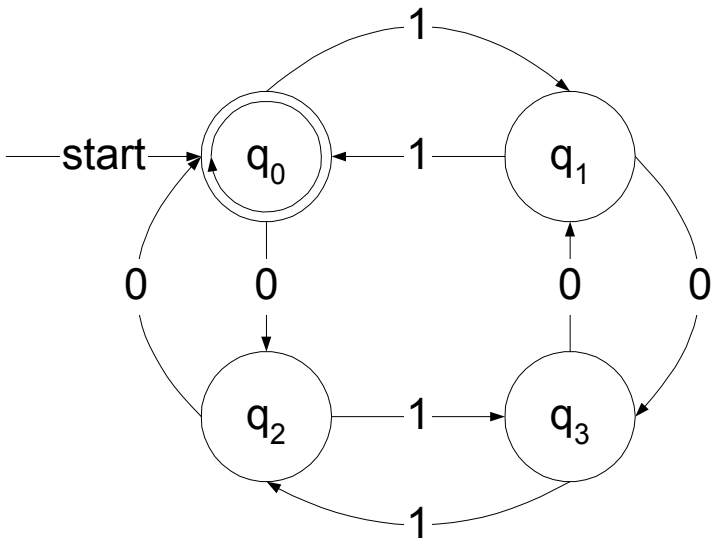
atau

δ(Genap,0) = Genap
δ(Genap,1) = Ganjil
δ(Ganjil,0) = Ganjil
δ(Ganjil,1) = Genap

Jenis FSA
Deterministic Finite Automata (DFA) : dari suatu state ada tepat **satu** state berikutnya untuk setiap simbol masukan yang diterima
Non-deterministic Finite Automata (NFA) : dari suatu state ada **0, 1 atau lebih** state berikutnya untuk setiap simbol masukan yang diterima

Deterministic Finite Automata

- Contoh : pengujian parity ganjil.
- Contoh lain : Pengujian untuk menerima bit string dengan banyaknya 0 genap, serta banyaknya 1 genap.
  - 0011 : diterima.
  - 10010 : ditolak, karena banyaknya 0 ganjil
- Diagram transisi-nya :

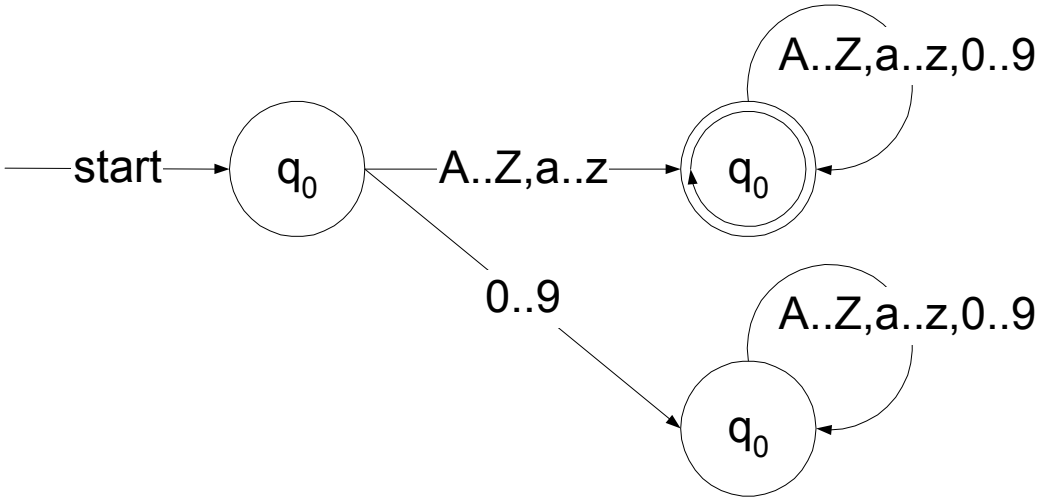


- DFA nya
  $Q = \{q_0 , q_1 , q_2 , q_3 \}$
  $\Sigma = \{0,1\}$
  $S = q_0$
  $F = \{ q_0\}$
  fungsi transisi

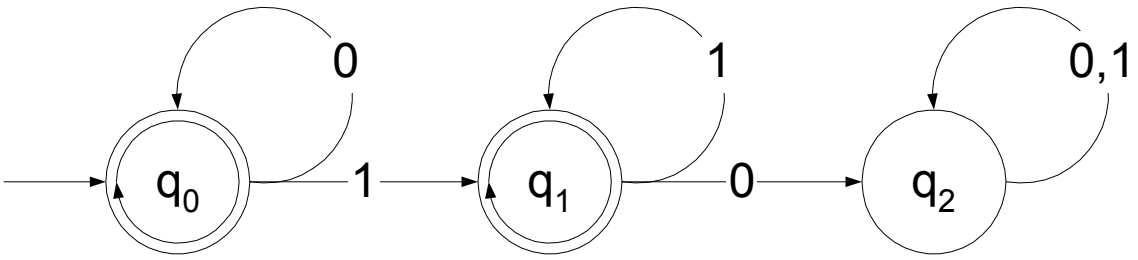
δ	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

$\delta(q_0,011)= \delta(q_2,11) =\delta(q_3,1)= q_2$ Ditolak
$\delta(q_0,1010)= \delta(q_1,010) =\delta(q_3,10)=\delta(q_2,0)= q_0$ Diterima

- Contoh lain DFA : Variabel dalam bahasa pascal diawali oleh huruf (besar/kecil), dan diikuti dengan huruf atau angka.



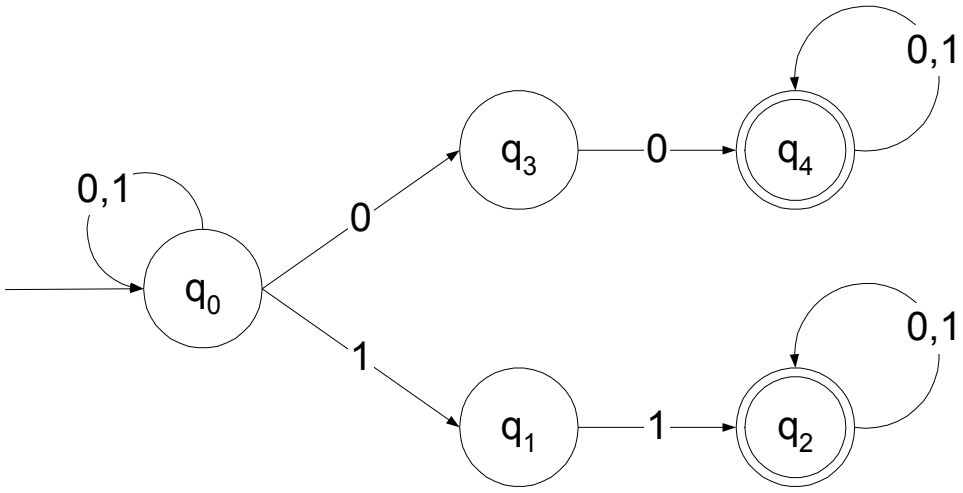
- Contoh DFA lainnya :



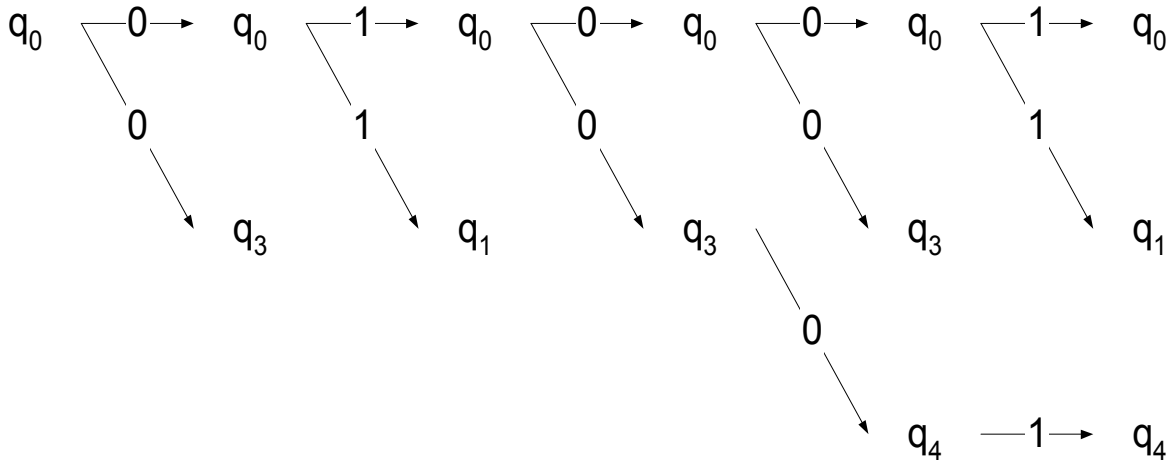
PERTEMUAN III
Nondeterministic Finite Automata

- Perbedaan dengan NFA: fungsi transisi dapat memiliki 0 atau lebih fungsi transisi
- G = ({q_0 , q_1 , q_2 , q_3, q_4 }, {0,1}, δ , q_0 , { q_2 , q_4}}

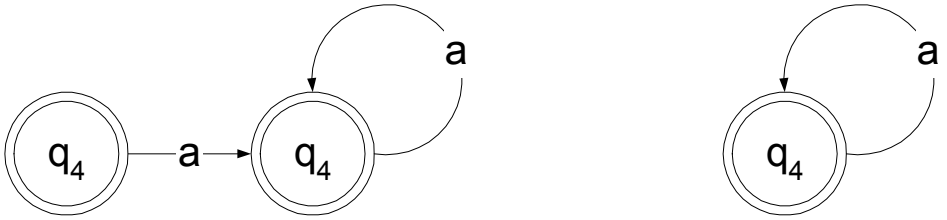
δ	0	1
q_0	{ q_0,q_3}	{q_0,q_1}
q_1	ε	{q_2}
q_2	{q_2}	{q_2}
q_3	{q_4}	ε
q_4	{q_4}	{q_4}



- String diterima NFA bila terdapat suatu urutan transisi berdasar input, dari state awal ke state akhir.
- harus mencoba semua kemungkinan.
- Contoh : string 01001



Def 2. Dua buah FSA disebut ekuivalen apabila kedua FSA tersebut menerima bahasa yang sama
Contoh : FSA yang menerima bahasa {a^n | n≥0 }



Def 3. Dua buah state dari FSA disebut indistinguishable (tidak dapat dibedakan) apabila :

$\delta(q,w) \in F$ sedangkan $\delta(p,w) \notin F$ dan

$\delta(q,w) \notin F$ sedangkan $\delta(p,w) \in F$ untuk semua $w \in \Sigma^*$

Def 4. Dua buah state dari FSA disebut distinguishable (dapat dibedakan) bila terdapat $w \in \Sigma^*$ sedemikian hingga:

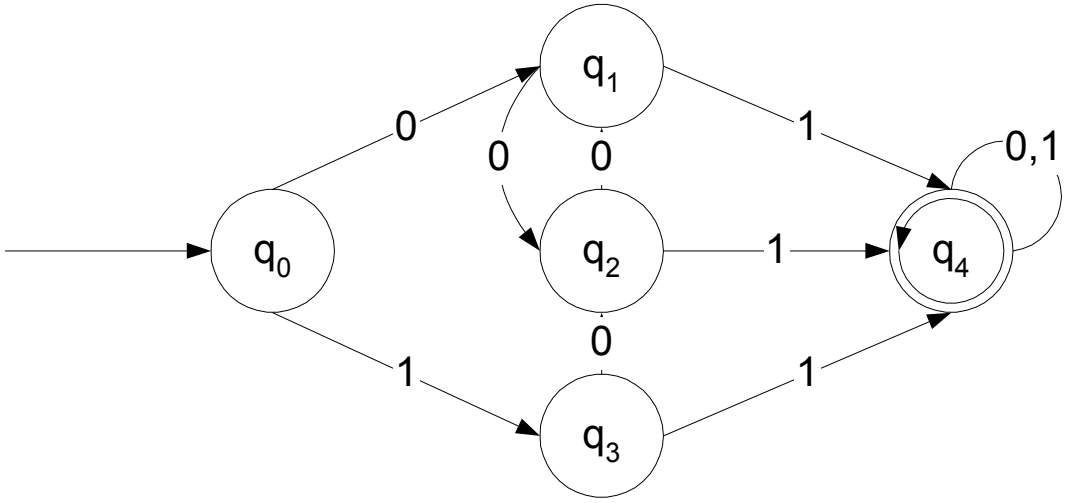
$\delta(q,w) \in F$ sedangkan $\delta(p,w) \notin F$ dan

$\delta(q,w) \notin F$ sedangkan $\delta(p,w) \in F$ untuk semua $w \in \Sigma^*$

Prosedur menentukan pasangan status indistinguishable

1. Hapus semua state yang tak dapat dicapai dari state awal.
2. Catat semua pasangan state (p,q) yang distinguishable, yaitu $\{(p,q) \mid p \in F \wedge q \notin F\}$
3. Untuk setiap pasangan (p,q) sisanya,
   untuk setiap $a \in \Sigma$, tentukan $\delta(p,a)$ dan $\delta(q,a)$

Contoh



1. Hapus state yang tidak tercapai -> tidak ada
2. Pasangan distinguishable (q_0,q_4), (q_1,q_4), (q_2,q_4), (q_3,q_4).
3. Pasangan sisanya (q_0,q_1), (q_0,q_2), (q_0,q_3), (q_1,q_2) (q_1,q_3) (q_2,q_3)

pasangan	state 1		state 2		hasil
	0	1	0	1	
(q_0,q_1)	q1	q3	q2	q4	distinguishable
(q_0,q_2)	q1	q3	q1	q4	distinguishable
(q_1,q_2)	q2	q4	q1	q4	indistinguishable
(q_0,q_3)	q1	q3	q2	q4	distinguishable
(q_1,q_3)	q2	q4	q2	q4	indistinguishable
(q_2,q_3)	q1	q4	q2	q4	indistinguishable

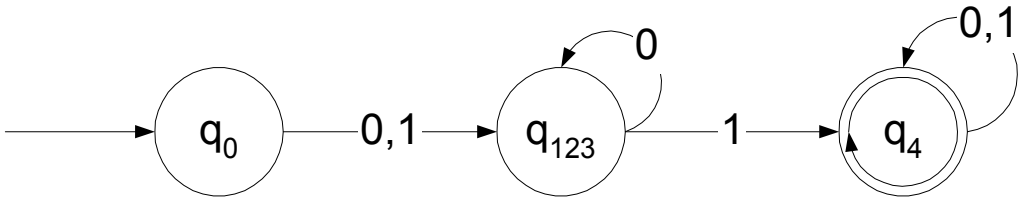
Catatan : jumlah pasangan seluruhnya : $C\binom{5}{2} = \frac{5!}{2!3!} = 10$

Prosedur Reduksi DFA

1. Tentukan pasangan status indistinguishable.
2. Gabungkan setiap group indistinguishable state ke dalam satu state dengan relasi pembentukan group secara berantai : Jika p dan q indistingishable dan jika q dan r indistinguishable maka p dan r indistinguishable, dan p,q serta r indistinguishable semua berada dalam satu group.
3. sesuaikan transisi dari dan ke state-state gabungan.

Contoh

1. pasangan status indistinguishable (q_1,q_2), (q_1,q_3) dan (q_2,q_3).
2. q_1,q_2,q_3 ketiganya dapat digabung dalam satu state q_{123}
3. Menyesuaikan transisi, sehingga DFA menjadi



PR Buku Firrar Bab II nomor 3, 4, 8, 9, 12.

PERTEMUAN IV
Ekuivalensi NFA-DFA

- Ada apa dengan NFA ? konsep yang sulit diimplemen-tasikan. Komputer sepenuhnya deterministic.
- Kenapa dipelajari ? Lebih dekat ke sistem nyata
- Contoh : permainan catur, banyak alternatif pada suatu posisi tertentu -> nondeterministic
- Non deterministik dapat menyelesaikan problem tanpa backtrack, namun dapat diekuivalensikan ke DFA.

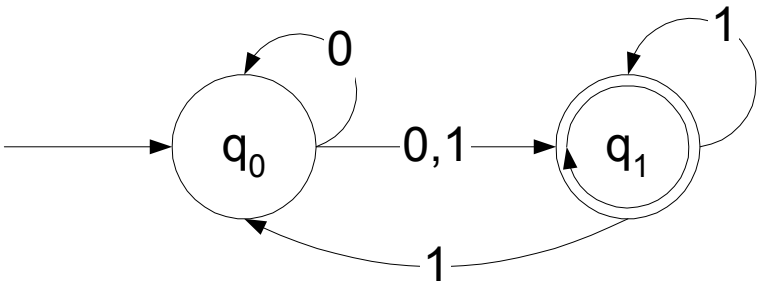
Algoritma

1. Buat semua state yang merupakan subset dari state semula. jumlah state menjadi 2^Q
2. Telusuri transisi state–state yang baru terbentuk, dari diagram transisi.
3. Tentukan state awal : $\{q_0\}$
4. Tentukan state akhir adalah state yang elemennya mengandung state akhir.
5. Reduksi state yang tak tercapai oleh state awal.

Contoh Ubahlah NFA berikut menjadi DFA

M=$\{\{q_0,q_1\}, \{0,1\}, \delta, q_0, \{q_1\}\}$ dengan tabel transisi

δ	0	1
q_0	$\{q_0,q_1\}$	q_1
q_1	{}	$\{q_0,q_1\}$

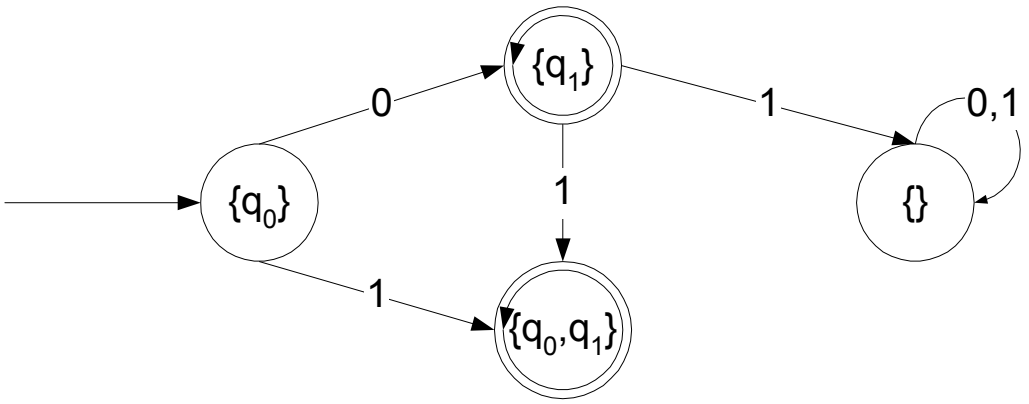


1. State yang akan dibentuk : {}, $\{q_0\}$ $\{q_1\}$,$\{q_0,q_1\}$

2. Telusuri state

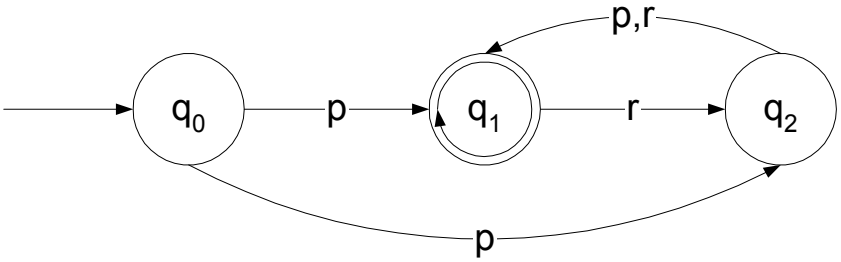
δ	0	1
{}	{}	{}
$\{q_0\}$	$\{q_0,q_1\}$	$\{q_1\}$
$\{q_1\}$	{}	$\{q_0,q_1\}$
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_1\}$

3. State awal : {q0}
4. State akhir yang mengandung q_1, yaitu $\{q_1\}$,$\{q_0,q_1\}$



Contoh : Ubahlah NFA berikut menjadi DFA
M={{q_0,q_1 ,q_2}, {p,r}, δ, q_0,{q_1}} dengan tabel transisi

δ	0	1
q_0	{q_1,q_2}	{}
q_1	{}	{q_0,q_1}
q_2	{q_1}	{q_1}

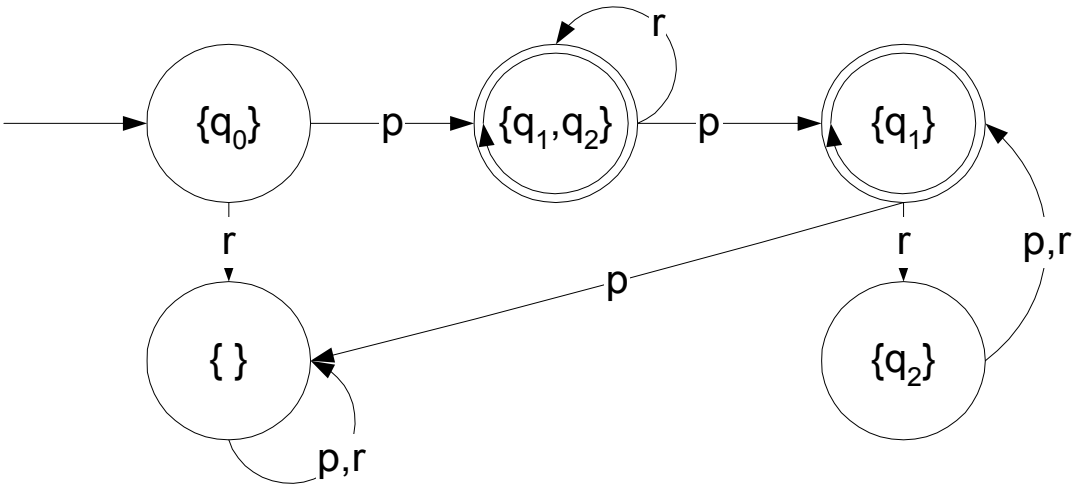


1. State yang akan dibentuk : {}, {q_0} {q_1},{q_2}, {q_0,q_1}, {q_0,q_2}, {q_1,q_2}, {q_0,q_1,q_2}

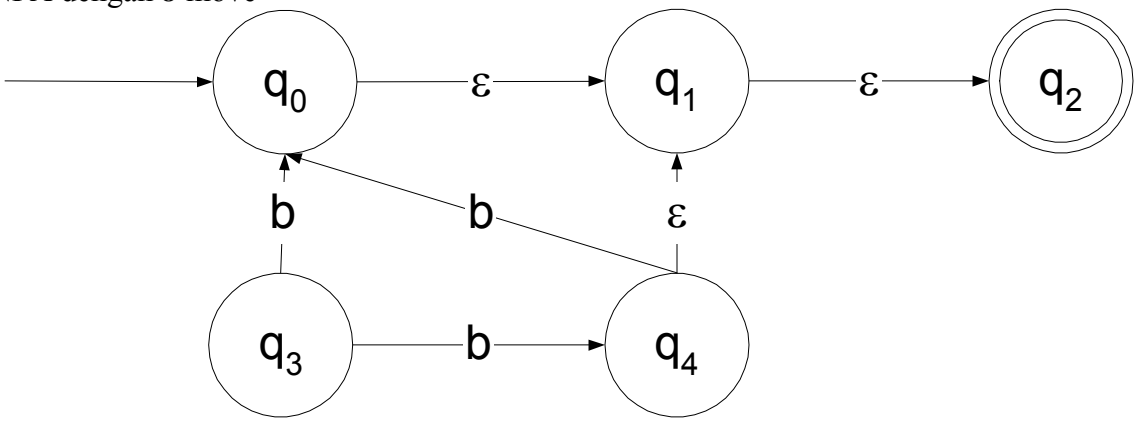
2. Telusuri state:

δ	p	r
{}	{}	{}
{q_0}	{q_1,q_2}	{}
{q_1}	{}	{q_2}
{q_2}	{q_1}	{q_1}
{q_0,q_1}	{q_1,q_2}	{q_2}
{q_0,q_2}	{q_1,q_2}	{q_1}
{q_1,q_2}	{q_1}	{q_1,q_2}
{q_0,q_1,q_2 }	{q_1,q_2}	{q_1,q_2}

3. State awal : {q_0}
4. State akhir yang mengandung q_1, yaitu {q_1},{q_1,q_2}
5. Reduksi {q_0,q_1}{q_0,q_2}{q_0,q_1,q_2 } sehingga FSA menjadi



NFA dengan ε-move



Def 1. ε-move adalah suatu transisi antara 2 status tanpa adanya input. Contoh gambar : transisi antara status q_1 ke q_3

Def 2. ε-closure adalah himpunan state yang dapat dicapai dari suatu state tanpa adanya input. Contoh gambar :

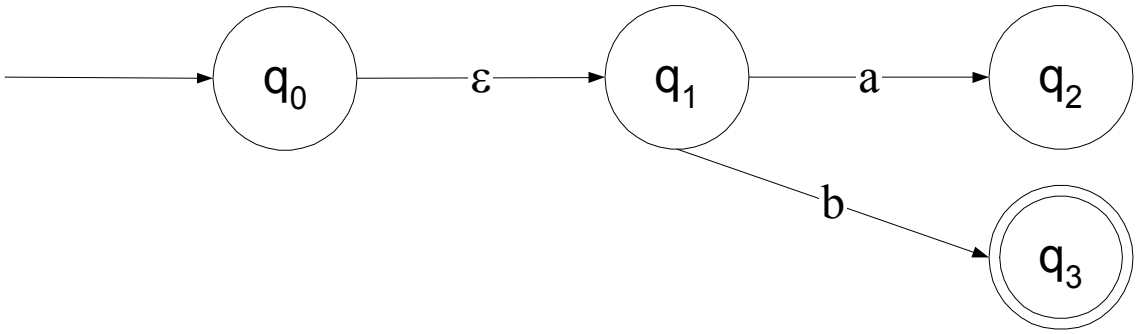
ε-closure(q_0) = [q_0,q_1,q_3]
ε-closure(q_1) = [q_1,q_3]
ε-closure(q_3) = [q_3]

Ekuivalensi NFA dengan ε-move ke NFA tanpa ε-move

1. Buat tabel transisi NFA dengan ε-move
2. Tentukan ε-closure setiap state
3. Carilah fungsi transisi /tabel transisi yang baru, rumus :
   δ'(state,input)=ε-closure(δ(ε-closure(state,input))
4. Tentukan state akhir ditambah dengan state yang ε-closure nya menuju state akhir, rumusnya

F' = F ∪ {q | (ε-closure(q) ∩ F ≠ ∅}

Contoh



Tabel transisi-nya

δ	0	1
q_0	∅	∅
q_1	q_2	q_3
q_2	∅	∅
q_3	∅	∅

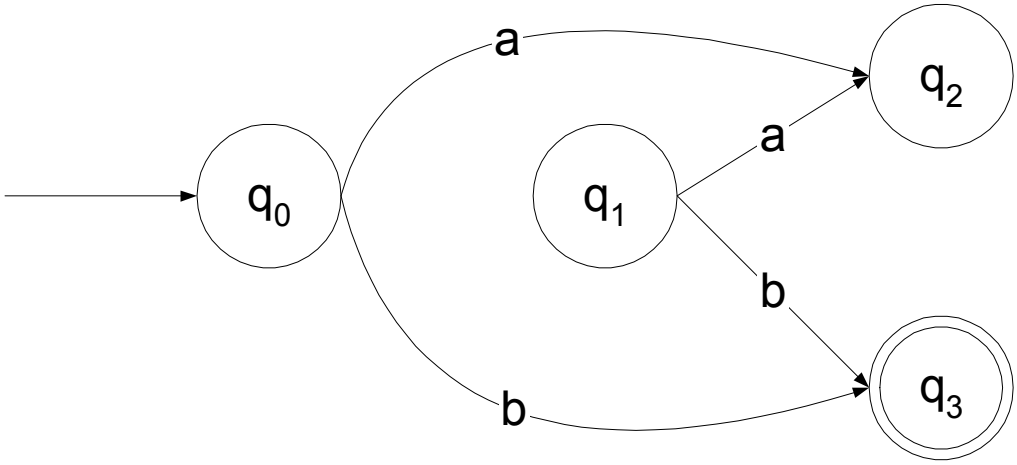
ε-closure dari FSA tersebut

ε-closure(q_0) = [q_0,q_1]
ε-closure(q_1) = [q_1]
ε-closure(q_2) = [q_2]
ε-closure(q_3) = [q_3]

Cari tabel transisi yang baru (δ') :

δ'	a	b
q_0	ε-cl(δ(ε-cl(q_0),a)) ε-cl(δ({q_0,q_1},a)) ε-cl(q_2) {q_2}	ε-cl(δ(ε-cl(q_0),b)) ε-cl(δ({q_0,q_1},b)) ε-cl(q_3) {q_3}
q_1	ε-cl(δ(ε-cl(q_1),a)) ε-cl(δ({q_1},a)) ε-cl(q_2) {q_2}	ε-cl(δ(ε-cl(q_1),b)) ε-cl(δ({q_1},b)) ε-cl(q_3) {q_3}
q_2	ε-cl(δ(ε-cl(q_2),a)) ε-cl(δ({q_3},a)) ε-cl(∅) ∅	ε-cl(δ(ε-cl(q_2),b)) ε-cl(δ({q_2},b)) ε-cl(∅) ∅
q_3	ε-cl(δ(ε-cl(q_3),a)) ε-cl(δ({q_3},a)) ε-cl(∅) ∅	ε-cl(δ(ε-cl(q_3),b)) ε-cl(δ({q_3},b)) ε-cl(∅) ∅

Hasilnya menjadi

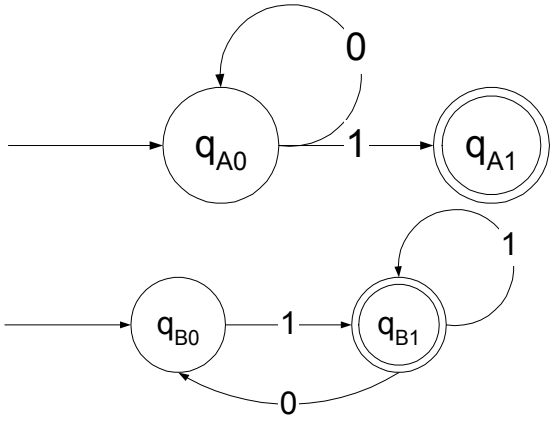


Penggabungan FSA

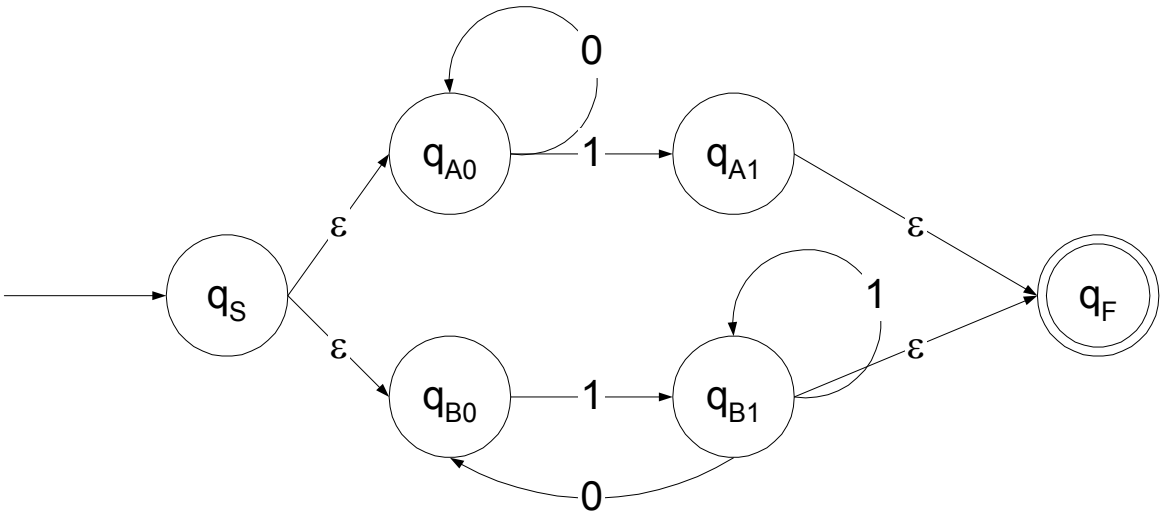
Bila diketahui L_1 adalah bahasa yang diterima oleh M1 dan L2 adalah bahasa yang diterima oleh M2 maka

1. FSA M3 yang dapat menerima L_1+L_2 dibuat dengan cara
   - Tambahkan state awal untuk M3, hubungkan dengan state awal M1 dan state awal M2 menggunakan transisi ε
   - Tambahkan state akhir untuk M3, hubungkan dengan state-state akhir M1 dan state-state akhir M2 menggunakan transisi ε
2. FSA M4 yang dapat menerima L_1L_2 dibuat dengan cara
   - State awal M1 menjadi state awal M4
   - State-state akhir M2 menjadi state-state akhir M4
   - Hubungkan state-state akhir M1 dengan state awal M2 menggunakan transisi ε

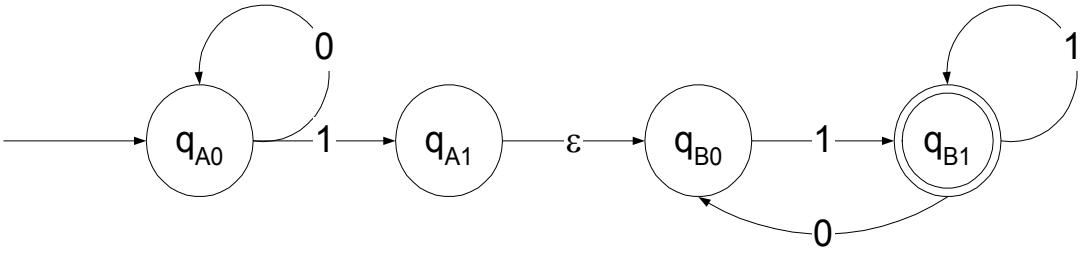
Contoh FSA M1 dan M2



FSA M3



FSA M4



PR, Buku firrar
bab 3 nomor 5 dan 6
bab 4 nomor 1 dan 4

PERTEMUAN V
Ekspressi reguler

- Bahasa disebut reguler jika terdapat FSA yang dapat menerimanya.
- Bahasa reguler dinyatakan secara sederhana dengan ekspresi reguler/regular expression (RE).
- Contoh penerapan : searching string pada file
- RE -> NFA dengan ε Move -> DFA

Definisi ekspresi reguler
Jika Σ merupakan himpunan simbol, maka

1. $\varnothing$, λ , dan a $\in\Sigma$ adalah ekspresi reguler dasar
2. jika r dan t masing masing merupakan ekspresi reguler maka komposisi berikut merupakan ekspresi reguler :

Ekspresi	Makna
r+t	himpunan string gabungan R$\cup$T
rt	operasi penyambungan string thd himpunan
r^*	Kleene closure dari R
(r)	r

Contoh ekspresi reguler

- (0+1)* : himpunan seluruh string yang dapat dibentuk dari simbol '0' dan '1'
- (0+1)*00(0+1)* : himpunan string biner yang mengandung paling sedikit satu substring '00'
- (0+1)*00 : himpunan string biner yang diakhiri dengan '00'

Bahasa Reguler
Apabila r adalah RE, maka L(r) adalah bahasa reguler yang dibentuk menggunakan ekspressi reguler r.

Contoh
Tentukan bahasa reguler yang dibentuk oleh r=(aa)*
Jawab

$L(r) = L((aa)^*)$
$= \{ \lambda, aa, aaaa, aaaaaa, ... \}$
$= \{ a^{2n} \mid n \geq 0 \}$

menyatakan himpunan string a dengan jumlah genap

Tentukan bahasa reguler yang dibentuk oleh r=(aa*)(bb)*b
Jawab

$L(r) = L((aa)^* (bb)^* b)$
$= \{ a^{2n} b^{2m+1} \mid n,m \geq 0 \}$

Tentukan ekspresi reguler pembentuk bahasa pada Σ = {0,1}, yaitu

L(r) = { w ∈ Σ* | w memiliki substring '00' }

Jawab

r = (0+1)*00(0+1)*

Tentukan ekspresi reguler pembentuk bahasa pada Σ = {a,b}, yaitu

L(r) = { $ab^n w$ | n≥ 3 , w ∈ $\{a , b\}^+$ }

Jawab

r = abbb(a+b)(a+b)*

Latihan :

1. Carilah seluruh string pada L((a+b)*b(a+ab)*) dengan panjang string kurang dari 4.

2. **Tentukan ekspresi reguler pembentuk bahasa pada Σ = {a,b,c}, yaitu**
   a. L(r) = { w ∈ Σ* | w memiliki tepat sebuah simbol 'a' }
   b. L(r) = { w ∈ Σ* | w mengandung tepat 3 buah simbol 'a'}
   c. L(r) = { w ∈ Σ* | w mengandung kemunculan masing masing simbol minimal satu kali}

3. **Tentukan ekspresi reguler pembentuk bahasa pada Σ = {0,1}, yaitu**
   a. L(r) = { w ∈ Σ* | w diakhiri dengan string 01 }
   b. L(r) ={ w ∈ Σ* | w tidak diakhiri dengan string 01 }
   c. L(r) ={ w ∈ Σ* | w mengandung simbol '0' sebanyak genap }

   d. L(r) ={ w ∈ Σ* | kemunculan string '00' pada w sebanyak kelipatan 3 }

4. **Tentukan ekspresi reguler pembentuk bahasa pada Σ = {a,b}, yaitu L(r) = { w ∈ Σ* | |w| mod 3 = 0 }**

Sifat Bahasa Reguler

- **Tertutup terhadap operasi himpunan sederhana**
  Jika L_1 dan L_2 adalah bahasa reguler, maka $L_1 \cup L_2$, $L_1 \cap L_2$, L_1L_2, ~(L_1) dan L_1^* adalah bahasa reguler juga
- Tertutup terhadap homomorphic image.
  Jika L_1 adalah bahasa reguler, maka homomorphic image h(L_1) adalah bahasa reguler juga.
  Dimisalkan Σ dan Γ adalah alfabet, maka fungsi homomorphic dinyatakan dengan
  $h : \Sigma \rightarrow \Gamma$

jika $w = a_1\ a_2 \ldots a_n$
maka $h(w) = h(a_1)\ h(a_2)\ \ldots\ h(a_n)$
Jika L adalah bahasa pada Σ maka homomorphic image bahasa L adalah

$h(L) = \{ h(w) \mid w \in L \}$

Contoh
Dimisalkan Σ = {a,b} dan Γ = {a,b,c} dan didefinisikan h(a) = ab dan h(b) =bbc
homomorphic image bahasa L = {aa,aba } adalah

h(L)= { abab, abbbcab}

Dimisalkan Σ = {a,b} dan Γ = {b,c,d} dan didefinisikan h(a) = dbcc dan h(b) =bdc
homomorphic image bahasa L(r) yang dibentuk dari ekspresi reguler

r = (a+b*)(aa)*

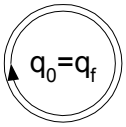
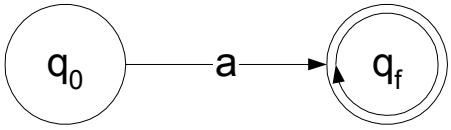
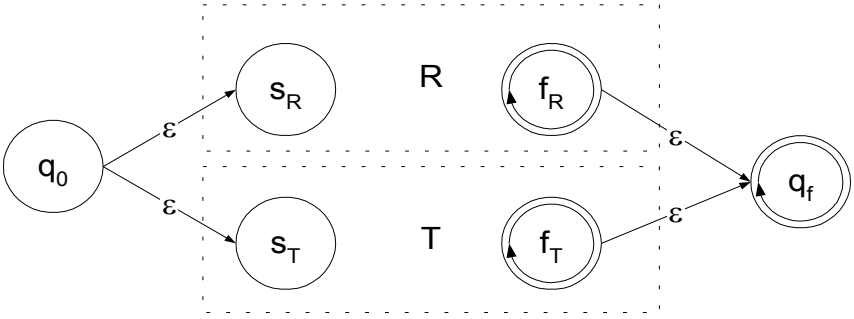
adalah h(L(r)) yang dibentuk dengan ekspresi reguler

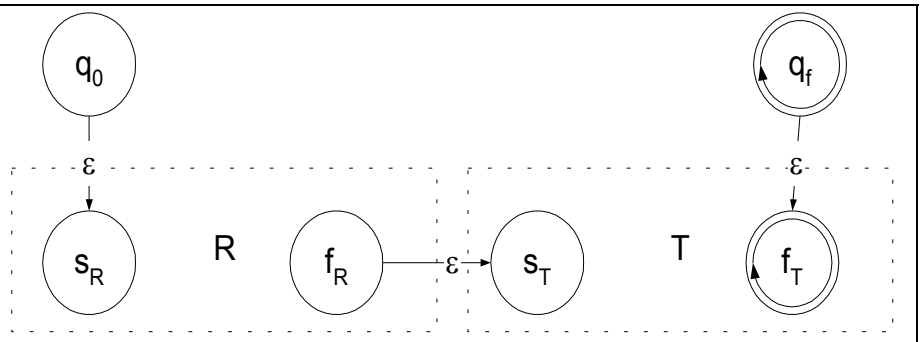
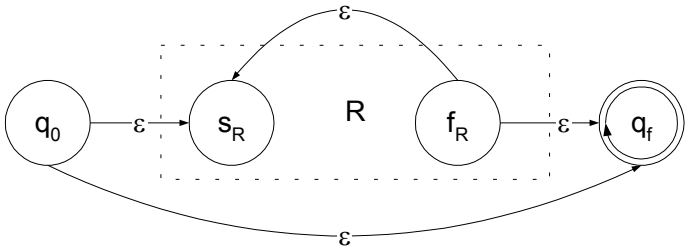
r = (dbcc + (bdc)*) (dbccdbcc)*

Hubungan RE dan NFA

- Setiap RE ada satu NFA dengan ε-move yang ekuivalen

Konversi ekspresi reguler ke FSA

Ekspresi	**FSA**
ε	
∅	
a	
r+t	

rt	
r*	

Contoh : Tentukan FSA untuk ekspresi reguler :
1. 01
2. 0+11
3. 01*+1
4. (0+1*)*

PR
Buku firrar bab V nomor 4 dan 8

PERTEMUAN VI
DFA dan Tatabahasa Reguler

Tatabahasa Linier kiri dan linier kanan

Suatu tatabahasa G (T,N,S,P) disebut linier kiri jika seluruh aturan produksinya berbentuk

$A \rightarrow xB$

$A \rightarrow x$

dengan $A, B \in N$ dan $x \in T^*$

Suatu tatabahasa G (T,N,S,P) disebut linier kiri jika seluruh aturan produksinya berbentuk

$A \rightarrow Bx$

$A \rightarrow x$

dengan $A, B \in N$ dan $x \in T^*$

Tatabahasa reguler bila bersifat linier kiri atau linier kanan.

Contoh 1

Tatabahasa G = {{S} , {a,b}, S , P } dengan aturan produksi P adalah $S \rightarrow abS \mid a$ adalah tatabahasa linier kanan /reguler

Tatabahasa G = {{S, S_1,S_2 } , {a,b}, S , P } dengan aturan produksi P adalah

$S \rightarrow S_1ab$ $S_1 \rightarrow S_1ab \mid S_2$ $S_2 \rightarrow a$

adalah tatabahasa linier kiri /reguler

Tatabahasa G = {{S, A, B} , {a,b}, S , P } dengan aturan produksi P adalah

$S \rightarrow A$ $A \rightarrow aB \mid \lambda$ $B \rightarrow Ab$

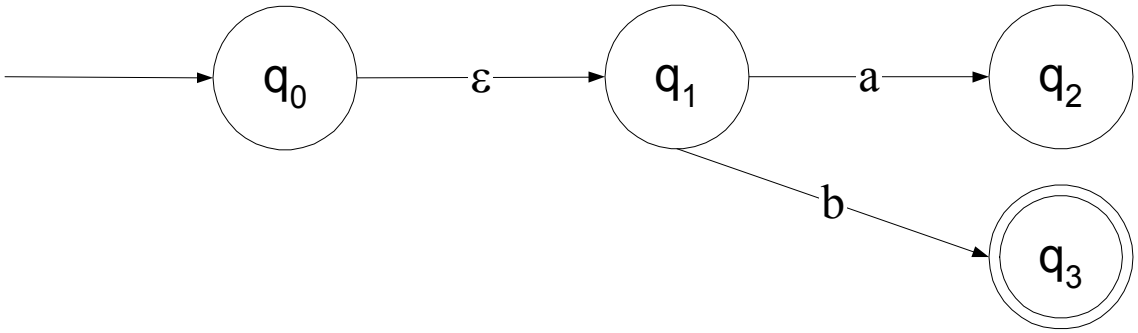
adalah bukan tatabahasa reguler

Konversi DFA ke tatabahasa linier

Setiap DFA dapat diubah menjadi tatabahasa yang memiliki aturan produksi yang linier. Aturan pengubahan ini adalah sebagai berikut :

- setiap transisi status $\delta(A,a)=B$ diubah menjadi aturan produksi $A \rightarrow aB$
- setiap status akhir P diubah menjadi aturan produksi $P \rightarrow \varepsilon$

Contoh FSA berikut



Tatabahasa linier untuk FSA tersebut yaitu G = ({a,b}, {S,S_1,S_2,S_3},S, P) dengan aturan produksi P adalah :

$S \rightarrow S_1$
$S_1 \rightarrow aS_2$
$S_2 \rightarrow bS_3$
$S_3 \rightarrow \varepsilon$

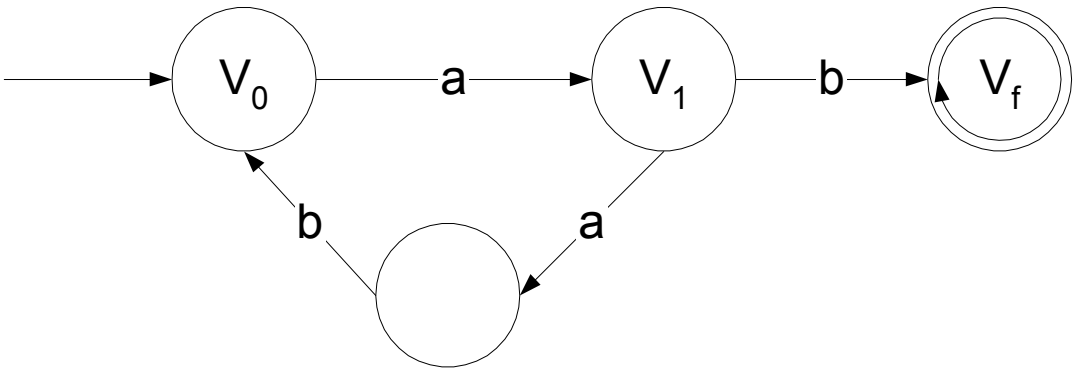
Konversi tatabahasa linier ke DFA

- setiap aturan produksi $A \rightarrow aB$ diubah menjadi transisi status $\delta(A,a)=B$
- setiap aturan produksi $A \rightarrow a$ diubah menjadi $\delta(A,a)=S_F$
- untuk $a \in T^*$ dengan $|a|>1$ dapat dibuat state tambahan
- setiap aturan produksi $A \rightarrow B$ diubah menjadi $\delta(A,\varepsilon)=B$

Contoh
tatabahasa G = ({a,b},{V_0,V_1}, V_0, P) dengan P :

$V_0 \rightarrow aV_1$
$V_1 \rightarrow abV_0 \mid b$

Mesin FSA nya menjadi



PR / Latihan buku firrar bab 6 nomor 2,3,4,5

Pertanyaan mendasar tentang bahasa reguler
1. Apakah terdapat suatu algoritma untuk menentukan diterima atau tidaknya suatu suatu string pada bahasa L ?
Jawab : YA, dengan menggunakan FSA.
2. Apakah terdapat suatu algoritma untuk menentukan suatu bahasa reguler kosong, finite atau infinite ?
Jawab : YA

- dengan DFA, jika terdapat lintasan dari simpul start ke simpul Final, maka bahasa tersebut tidak kosong.
- Cari simpul simpul yang membentuk siklus. Jika terdapat lintasan dari simpul start ke simpul Final yang melalui simpul yang membentuk siklus, maka bahasa tersebut infinite. Jika tidak, maka bahasa tersebut finite.

Penerapan ekspresi reguler

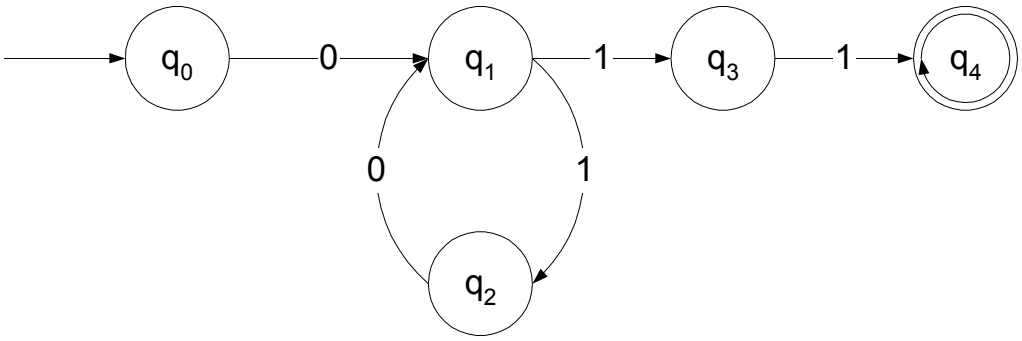
- Digunakan untuk memerinci unit-unit leksikal sebuah bahasa pemrograman (token).

contoh ekspresi reguler ‘bilangan real positif’
(0+1+...+9)(0+1+...+9)*.(0+1+...+9) (0+1+...+9)*
contoh ekspresi reguler ‘bilangan bulat’
(‘+’ + ’-‘ + λ) (0+1+...+9)(0+1+...+9)*

- Editor text

Pumping lemma

Apabila suatu bahasa merupakan bahasa reguler maka akan dapat diterima oleh mesin DFA M=(Q,Σ,δ,q_0,F), dengan sejumlah state n.
Apabila string w dengan |w| ≥ n diinputkan dalam DFA, maka pasti ada simpul k dalam DFA yang dikunjungi lebih dari satu kali.
Apabila string diantara simpul k yang sama tersebut ‘dipompa’, maka sisanya pasti masih diterima oleh DFA tersebut.
Contoh
Bahasa yang menerima ekspresi reguler 0(10)*11



- Ambil string w∈L , dengan |w|≥ n: w= 01011
  q_0 0 q_1 1 q_2 0 q_1 1 q_3 1 q_4
- simpul q_1 dikunjungi 2 kali.
- string diantara simbol q_1 tersebut ‘dipompa’ keluar
  q_0 0 q_1 1 q_3 1 q_4
- string 011 tersebut masih dapat diterima oleh FSA.

Secara formal

Misal L adalah sebuah bahasa reguler infinite, maka terdapat sebuah konstanta n dengan sifat bahwa jika w adalah sebuah string dalam L yang panjangnya lebih besar atau sama dengan n maka kita bisa menulis w=uvx sedemikian sehingga $uv^ix \in L$ untuk semua i ≥ 0. dengan |v|≥1 dan |uv|≤n .
Notasi matematisnya

$$(\forall L)(\exists n)(\forall z)\left[\begin{array}{l} z \in L \wedge |z| \geq n \Rightarrow \\ (\exists u,v,w)\left(z = uvw \wedge |uv| \leq n \wedge |v| \geq 1 \wedge (\forall i)\left(uv^i w \in L\right)\right) \end{array}\right]$$

Penjelasan

- Mengidentifikasi sifat yang harus dimiliki oleh suatu bahasa reguler.
- Cara untuk menentukan apakah sebuah bahasa tidak reguler
- Untuk memperlihatkan bahwa suatu bahasa infinite tidak reguler, maka kita tunjukkan bahwa untuk nilai n yang cukup besar, sekurang-kurangnya satu untai yang panjangnya n atau lebih besar gagal untuk dapat 'dipompa'.

Contoh :

$L = \{a^{i\wedge 2} \mid i \geq 1\}$

$\{a^1, a^4, a^9, a^{16}, ...\}$

{a , aaaa , aaaaaaaaa , aaaaaaaaaaaaaaaa, ... }

Suatu string dalam L harus mempunyai panjang yang berupa nilai kuadrat (1,4,9,16, ..., n^2, ...)

Misal bahwa L adalah bahasa reguler.

Perhatikan bahwa terdapat sebuah nilai n sedemikian sehingga $a^{n\wedge 2} \in L$,

Menurut pumping lemma dapat kita tuliskan $a^{n\wedge 2} = uvx$, sedemikian hingga

- $1 \leq |v| \leq n$
- $(\forall i)\ (uv^i w \in L)$

karena $|v| \geq 1$ maka jelas bahwa

$|uvw| < |uv^2w| < |uv^2w| < ...$

ambil i=2 maka kita dapatkan

$n^2 = |uvw| < |uv^2w| \leq n^2 + n < (n+1)^2$

Jelas

$n^2 < |uv^2w| < (n+1)^2$

Panjang $|uv^2w|$ bukan merupakan kuadrat sempurna, karena berada diantara 2 nilai kuadrat sempurna yang berurutan.

berarti $uv^2w \notin L$

Jadi disimpulkan bahwa $L = \{a^{i\wedge 2} \mid i \geq 1\}$ bukan merupakan bahasa reguler.

PERTEMUAN VII
FSA dengan Output

FSA : accepter, dapat menerima atau tidak.
FSA dengan output : transducer
1. Mesin Moore :output berasosiasi dengan state
2. Mesin Mealy :output berasosiasi dengan transisi

Mesin Moore

$M = (Q,\Sigma,\delta,S,\Delta,\lambda)$
Q : himpunan state
Σ : himpunan simbol input
δ : fungsi transisi
S : state awal $S \in Q$
Δ : himpunan output
λ : fungsi output untuk setiap **state**

Contoh mesin moore untuk memperoleh modulus 3 pada suatu bilangan biner:

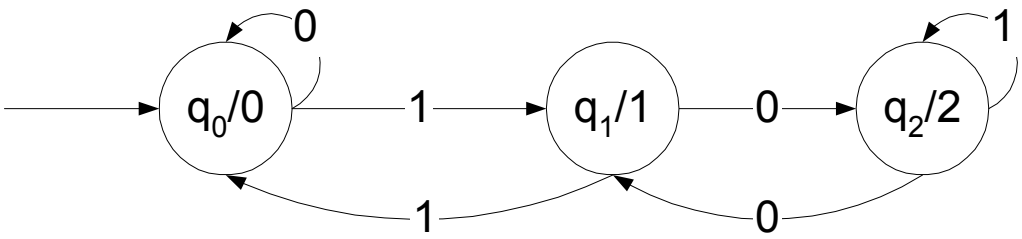
$M = (Q,\Sigma,\delta,S,\Delta,\lambda)$
$Q : q_0,q_1,q_2$
$\Sigma : [0,1]$
$S : q_0$
$\Delta : [0,1,2]$
$\lambda(q_0) = 0$
$\lambda(q_1) = 1$
$\lambda(q_2) = 2$

Prinsip:

jika i diikuti dengan 0, maka hasilnya 2i
$101_2 = 5$ $1010_2 = 2*5 = 10$
jika i diikuti dengan 1, maka hasilnya 2i+1
$101_2 = 5$ $1011_2 = 2*5+1 = 11$

jika i/3 mempunyai sisa p, maka untuk input berikutnya bernilai 0 maka
2i/3 mempunyai sisa 2p mod 3
untuk p=0 maka 2p mod 3 = 0
untuk p=1 maka 2p mod 3 = 2
untuk p=2 maka 2p mod 3 = 1
jika i/3 mempunyai sisa p, maka untuk input berikutnya bernilai 1 maka
(2i+1)/3 mempunyai sisa (2p+1) mod 3
untuk p=0 maka (2p+1) mod 3 = 1
untuk p=1 maka (2p+1) mod 3 = 0
untuk p=2 maka (2p+1) mod 3 = 2

Sehingga didapat mesin FSA sbb :



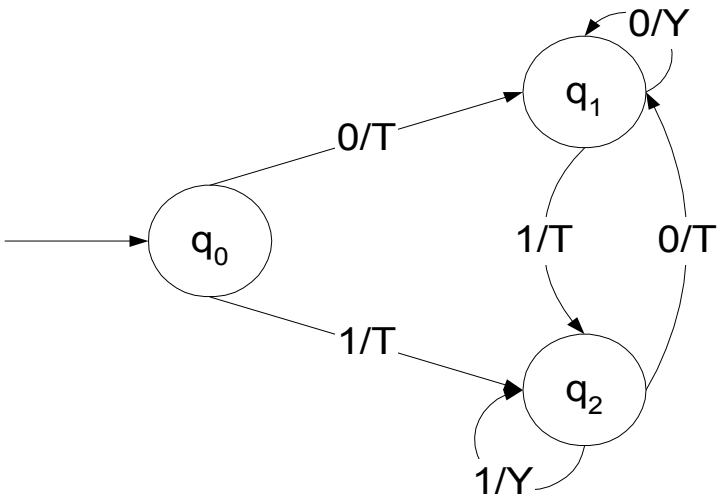
Contoh :
input 5 (101_2) , state terakhir $q_2/2$, 5 mod 3 = 2
input 10 (1010_2) , state terakhir $q_1/1$, 10 mod 3 = 1
Mesin Mealy

M = (Q,Σ,δ,S,Δ,λ)
Q : himpunan state
Σ : himpunan simbol input
δ : fungsi transisi
S : state awal S $\in$Q
Δ : himpunan output
λ : fungsi output untuk setiap **transisi**

Contoh mesin Mealy untuk mendeteksi ekspresi reguler
(0+1)*(00+11)

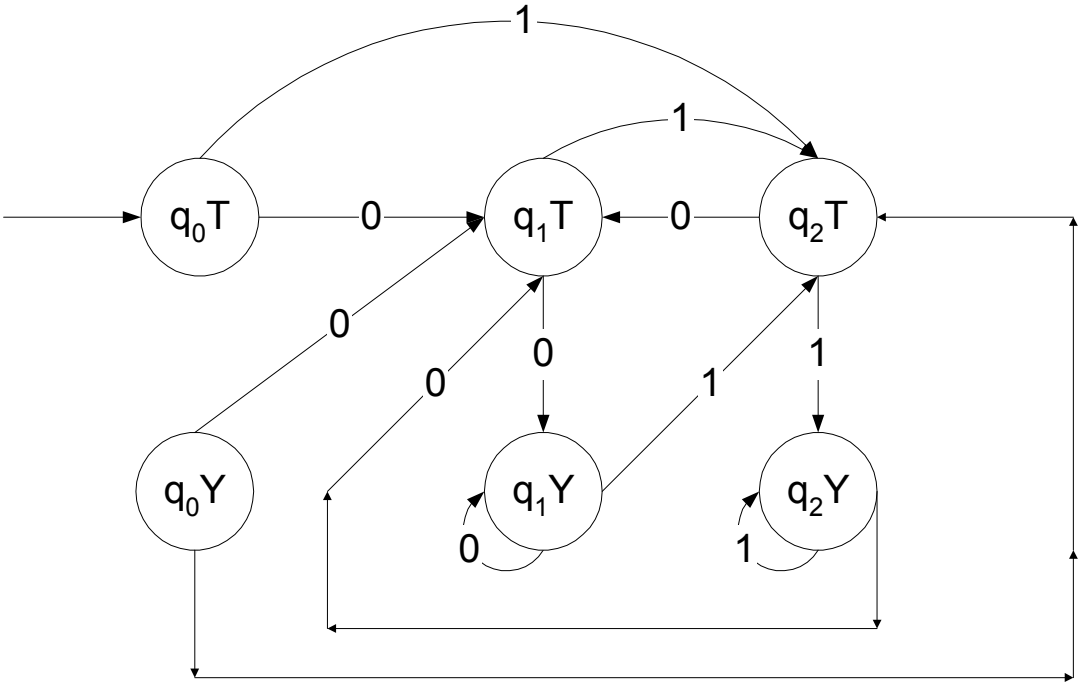
Jawab

M = (Q,Σ,δ,S,Δ,λ)
Q : q_0,q_1,q_2
Σ : [0,1]
S : q_0
Δ : [0,1,2]
λ(q_0,0) =T
λ(q_0,1) =T
λ(q_1,0) =Y
λ(q_1,1) =T
λ(q_2,0) =T
λ(q_2,1) =Y

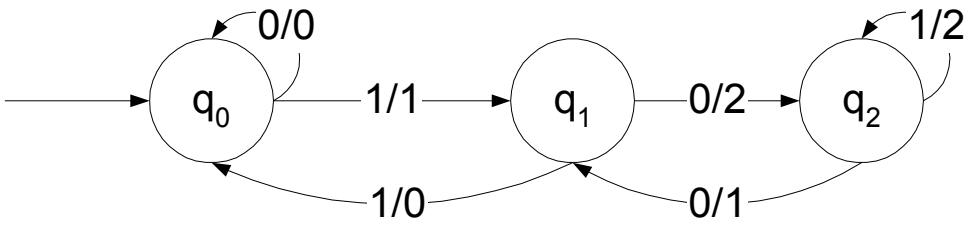


Ekuivalensi mesin Moore dengan mesin Mealy

- Mesin Moore ke mesin Mealy
  Jml state = jml state sebelum * jml output

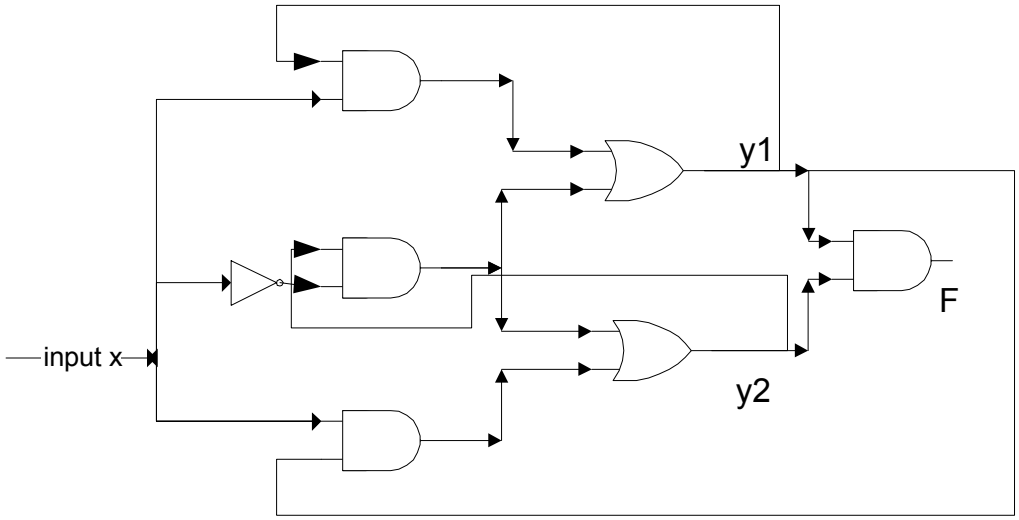


- Mesin Mealy ke mesin Moore
  Menambah label output pada transisi
  Menghapus label output pada state

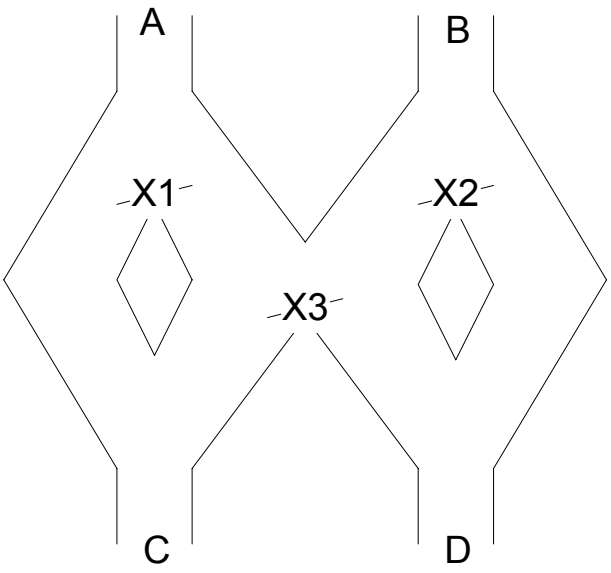


Contoh kasus

- Tentukan FSA dari rangkaian sirkuit berikut ini. Asumsi bahwa terdapat waktu yang cukup untuk perambatan sinyal menuju kondisi yang stabil.



- Kelereng dijatuhkan dari A atau B. Percabangan x1,x2 dan x3 menentukan saluran mana yang akan dilewati kelereng (kiri / kanan). Ketika percabangan dilewati, kelereng berikutnya akan melewati dengan saluran berbeda. Buatlah FSA nya



Latihan :
buku Firrar bab 7
PR.
Buatlah mesin Mealy dan Moore untuk proses membaca input (0+1)* :

- Jika input berakhir dengan 101, outputnya A
- Jika input berakhir dengan 110, outputnya A
- Jika yang lainnya , 8outputnya C

PERTEMUAN VIII
Tata Bahasa Bebas Konteks

Motivasi awal :
deskripsi bahasa alami

<kalimat> → <subjek> <predikat>
<subjek> → <kata benda>
<predikat> → <kata kerja>
<kata benda> → kucing
<kata kerja> → berlari
<kata kerja> → menyapu

Contoh kalimat yang dapat dihasilkan
kucing berlari
kucing menyapu (sintaks yes, semantik no)

Dalam tatabahasa bebas konteks

- Ruas kiri dari aturan produksi terdiri dari SATU simbol non terminal
- Ruas kanan dapat berupa string yang dibentuk dari simbol terminal dan non terminal

Contoh

$S \rightarrow aSb \mid \varepsilon$

Kalimat-kalimat yang dibangkitkan dari aturan produksi itu adalah ε,ab,aabb,aaabbb,... , $a^n b^n$

Contoh

$A \rightarrow 0A0$
$A \rightarrow 1A1$
$A \rightarrow a$

Kalimat-kalimat yang dibangkitkan dari aturan produksi itu adalah a,01a10, 1001a1001 , 110a011 $\beta a \beta^R$

Contoh

$S \rightarrow aSb \mid SS \mid \varepsilon$

Bahasa yang dihasilkan oleh tatabahasa dengan aturan produksi di atas adalah :

$L = \{w \in (a + b)^* \mid n_a(w) = n_b(w) \}$

Leftmost dan Rightmost Derivation

Suatu penguraian /penurunan dikatakan *leftmost derivation* bila setiap tahapan penurunan variabel / non terminal terkiri yang diuraikan. Apabila setiap tahapan penurunan variabel / non terminal paling kanan yang diuraikan disebut *rightmost derivation*

Contoh 1

G=({A,B,S}, {a,b},S,P} dengan aturan produksi P :

$S \rightarrow AB$
$A \rightarrow aaA \mid \lambda$

$B \rightarrow Bb \mid \lambda$

Menspesifikasikan bahasa

$L(G) = \{a^{2n}b^m \mid n \geq 0 , m \geq 0\}$

Leftmost derivation untuk menghasilkan string aab

$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaB \Rightarrow aaBb \Rightarrow aab$

Righmost derivation untuk menghasilkan string aab

$S \Rightarrow AB \Rightarrow ABb \Rightarrow aaABb \Rightarrow aaAb \Rightarrow aab$

Contoh 2

G=({A,B,S}, {a,b},S,P} dengan aturan produksi P :

$S \rightarrow aAB$

$A \rightarrow bBb$

$B \rightarrow A \mid \lambda$

Leftmost derivation untuk menghasilkan string abbbb

$S \Rightarrow aAB \Rightarrow abBbB \Rightarrow abAbB \Rightarrow abbBbbB$
$\Rightarrow abbbbB \Rightarrow abbbb$

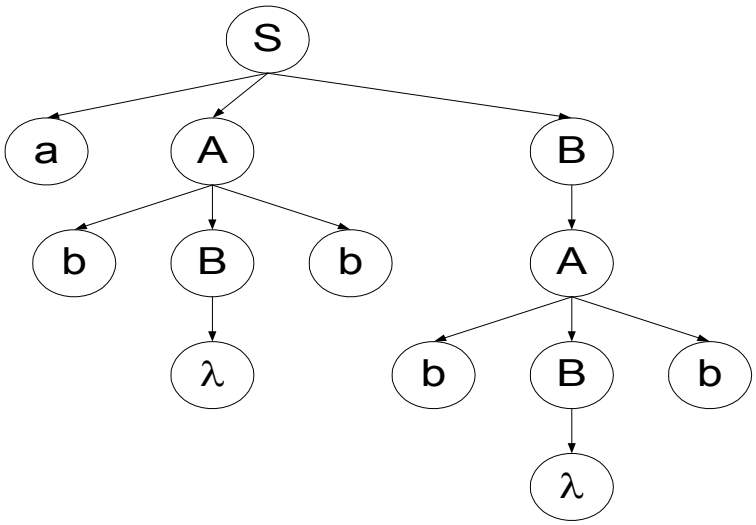
Righmost derivation untuk menghasilkan string aab

$S \Rightarrow aAB \Rightarrow aA \Rightarrow abBb \Rightarrow abAb \Rightarrow abbBbb \Rightarrow abbbb$

Pohon urai

Untuk menampilkan penguraian, dapat dilakukan dengan membentuk pohon urai (sayangnya, urutan penguraian tidak terlihat) .

Contoh pohon urai pada contoh sebelumnya :



Parsing dan Keanggotaan

Untuk menentukan apakah string w berada di L(G), dengan cara secara sistematis membangun semua kemungkinan penurunan, dan mencocokkan hasilnya apakah ada yang sama dengan string w. (disebut exhaustive search parsing)

contoh menentukan apakah string ab berada pada bahasa yang dibentuk oleh grammar dengan aturan produksi

$S \rightarrow SS \mid aSb \mid bSa \mid \lambda$

Untuk penguraian pertama

1. $S \Rightarrow SS$
2. $S \Rightarrow aSb$
3. $S \Rightarrow bSa$
4. $S \Rightarrow \lambda$

Penguraian nomor 3 dan 4 tidak perlu dilanjutkan. Penguraian 1 membentuk

Penguraian 2 membentuk

1a. $S \Rightarrow SS \Rightarrow SSS$ 2a. $S \Rightarrow aSb \Rightarrow aSSb$

1b. $S \Rightarrow SS \Rightarrow aSbS$ 2b. $S \Rightarrow aSb \Rightarrow aaSbb$

1c. $S \Rightarrow SS \Rightarrow bSaS$ 2c. $S \Rightarrow aSb \Rightarrow abSab$

1d. $S \Rightarrow SS \Rightarrow S$ **2d. $S \Rightarrow aSb \Rightarrow ab$**

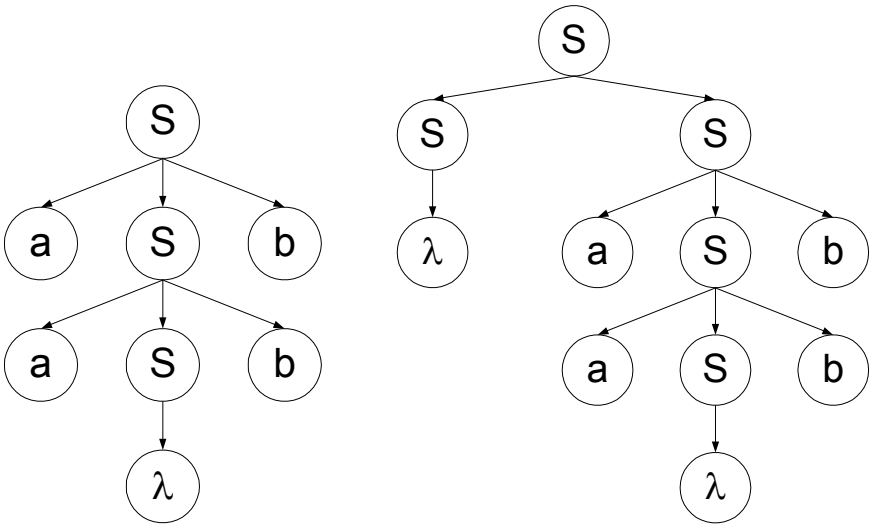
Ambiguitas pada Tatabahasa dan Bahasa

Tatabahasa bebas konteks G disebut ambigu jika terdapat beberapa $w \in L(G)$ yang mempunyai paling sedikit dua buah pohon penurunan

Contoh pada tatabahasa dengan aturan produksi

$S \rightarrow SS \mid aSb \mid \lambda$

string aabb mempunyai 2 pohon penurunan :



Pumping Lemma untuk bahasa bebas konteks

- Jika suatu rangkaian simbol /string yang cukup panjang yang merupakan sebuah bahasa bebas konteks, maka kita dapat menemukan dua substring yang jaraknya berdekatan yang jika dipompa, string baru yang diperoleh merupakan bahasa bebas konteks juga.
- Secara formal, lemma diatas dinyatakan dengan

$$(\forall L)(\exists n)(\forall z)\left[\begin{array}{l} z \in L \wedge |z| \geq n \Rightarrow \\ (\exists u,v,w,x,y)\left(\begin{array}{l} z = uvwxy \wedge |vwx| \leq n \wedge |vx| \geq 1 \Rightarrow \\ (\forall i)\left(uv^i wx^i y \in L\right) \end{array}\right) \end{array}\right]$$

- syarat " kedua lokasi berdekatan" dinyatakan dengan kondisi |vwx| ≤ n
- Jika salah satu v atau x diambil sebagai string kosong, maka lemma diatas berubah menjadi lemma untuk bahasa reguler

Contoh tatabahasa dengan aturan produksi

S $\rightarrow$uAy

A $\rightarrow$ vAx

A $\rightarrow$ w

maka aturan derivasinya

S $\Rightarrow$ uAy $\Rightarrow$ uwy

S $\Rightarrow$ uAy $\Rightarrow$ uvAxy $\Rightarrow$uvwxy

S $\Rightarrow$ uAy $\Rightarrow$ uvAxy $\Rightarrow$ uvvAxxy $\Rightarrow$uvvwxxy

sehingga untuk setiap i $\geq$0 , $uv^iwx^iy \in L$

Sifat sifat tertutup bahasa bebas konteks

- Gabungan dua CFL merupakan CFL juga
  Jika diketahui dua buah CFG $G_1= (N_1,T_1,S_1,P_1)$ dan $G_2=(N_2,T_2,S_2,P_2)$ yang menghasilkan bahasa L_1 dan L_2 , maka CFG $L_1 \cup L_2$ dapat dibentuk dengan cara :
  1. menggabungkan kedua himpunan dan menambahkan satu simbol variabel baru S
  2. menggabungkan kedua himpunan simbol terminal
  3. menggabungkan kedua himpunan aturan produksi dan menambahkan satu aturan produksi baru
     $S \rightarrow S_1|S_2$ yang digunakan untuk memilih salah satu simbol awal S_1 atau S_2 dari simbol awal baru S

  $G_3 = (N_1 \cup N_2 \cup \{S\}, T_1 \cup T_2, S, P_1 \cup P_2 \cup \{S \rightarrow S_1|S_2\})$
- Penyambungan dua CFL merupakan CFL juga
  Jika diketahui dua buah CFG $G_1= (N_1,T_1,S_1,P_1)$ dan $G_2=(N_2,T_2,S_2,P_2)$ yang menghasilkan bahasa L_1 dan L_2 , maka bahasa L_1L_2 dapat dibentuk oleh :
  $G_4 = (N_1 \cup N_2 \cup \{S\}, T_1 \cup T_2, S, P_1 \cup P_2 \cup \{S \rightarrow S_1S_2\})$
- Klosure Kleene dari CFL adalah CFL juga.
  Klosure Kleene dari tatabahasa $G=(N,T,S_1,P)$ adalah
  $G_5 = (N \cup \{S\} , T , S , P \cup \{S \rightarrow S_1S \mid \varepsilon \})$
  Latihan
  $G(L_1) = (\{S , A , B\}, \{a,b\} , S , P)$ dengan P :
  $S \rightarrow AB \mid \varepsilon$
  $A \rightarrow aB$

$B \rightarrow Sb$

$G(L_2) = (\{S , A , B\}, \{a,b\} , S , P)$ dengan P :

$S \rightarrow aaB$

$A \rightarrow bBb \mid \varepsilon$

$B \rightarrow aA$

Bagaimanakah :

a. CFG $G(L_1 \cup L_2)$

b. CFG $G(L_1L_2)$

c. CFG $G(L_1^*)$

- Bahasa bebas konteks tertutup terhadap substitusi

  Contoh

  $L_a = \{0^n1^n \mid n \geq 1 \}$ dan $L_b = \{ ww^R \mid w \in (0+2)^* \}$

  dihasilkan oleh tatabahasa G_a dengan aturan produksi

  $S_a \rightarrow 0S_a1 \mid 01$

  serta tatabahasa G_2 dengan aturan produksi

  $\mathbf{S_b \rightarrow 0S_b0 \mid 2S_b2 \mid \varepsilon}$

Didefinisikan tatabahasa G dengan aturan produksi

$S \rightarrow aSbS \mid bSaS \mid \varepsilon$

jika f adalah substitusi $f(a)= L_a$ dan $f(b) = L_b$ maka

f(L) adalah bahasa yang dihasilkan oleh tatabahasa dengan aturan produksi

$S \rightarrow S_aSS_bS \mid S_bSS_aS \mid \varepsilon$

$S_a \rightarrow 0S_a1 \mid 01$

$\mathbf{S_b \rightarrow 0S_b0 \mid 2S_b2 \mid \varepsilon}$

Tatabahasa Bebas Konteks dan Bahasa Pemrograman

- Tatabahasa bebas konteks digunakan untuk mendefinisikan sintaks bahasa pemrograman
- Menggunakan notasi BNF (Backus-Naur Form)
  - variabel / non terminal : <...>
  - terminal : tanpa tanda
  - ← diganti dengan ::=
- Contoh statemen if then else

  < if_statement> ::= if <expression>
  <then_clause>
  <else_clause>

PERTEMUAN IX
PENYEDERHANAAN
TATA BAHASA BEBAS KONTEKS

Tujuan
Melakukan pembatasan sehingga tidak menghasilkan pohon penurunan yang memiliki kerumitan yang tidak perlu atau aturan produksi yang tidak berarti.
Contoh 1:

S → AB | a
A→a

- Aturan produksi S → AB tidak berarti karena B tidak memiliki penurunan

Contoh 2 :

S→A
A→B
B→C
C→D
D → a | A

- Memiliki kelemahan terlalu panjang jalannya padahal berujung pada S → a,
- produksi D → A juga menyebabkan kerumitan.

Cara Penyederhanaan:

1. Penghilangan produksi useless (tidak berguna)
2. Penghilangan produksi unit
3. Penghilangan produksi ε

Penghilangan Produksi *Useless*

Di sini produksi *useless* didefinisikan sebagai :

- Produksi yang memuat symbol variabel yang tidak memiliki penurunan yang akan menghasilkan terminal-terminal seluruhnya.
- Produksi yang tidak akan pernah dicapai dengan penurunan apapun dari simbol awal, sehingga produksi itu redundan (berlebih)

Contoh :

S → aSa | Abd | Bde
A → Ada
B→ BBB | a

Maka

1) Simbol variabel A tidak memiliki penurunan yang menuju terminal, sehingga bisa dihilangkan
2) Konsekuensi no (1), aturan produksi S → Abd tidak memiliki penurunan

Penyederhanaan menjadi:

S→aSa | Bde
B→ BBB | a

Contoh :

S→ Aa | B
A→ab | D
B→ b | E
C→ bb
E→ aEa

Maka :

1) Aturan produksi A → D, simbol variabel D tidak memiliki penurunan.
2) Aturan produksi C → bb, Penurunan dari simbol S, dengan jalan manapun tidak akan pernah mencapai C
3) Simbol variabel E tidak memiliki aturan produksi yang menuju terminal
4) Konsekuensi no (3) Aturan produksi B → E, simbol variabel E tidak memiliki penurunan.

maka produksi yang useless:

A → D
C → bb
E → aEa
B → E

Penyederhanaannya menjadi:

S → Aa | B
A → ab
B → b

Contoh :

S → aAb | cEB
A → dBE | eeC
B → ff
C → ae
D → h

Analisa :

1) Aturan produksi S → cEB, A → dBE dapat dihilangkan (E tidak memiliki penurunan)
2) Aturan produksi D → h, redundan

Sisa aturan produksi

S → aAb
A → eeC
B → ff
C → ae

Analisis lagi

B → ff juga redundan,

Hasil penyederhanaan menjadi:

S → aAb
A → eeC
C → ae

Contoh lain lagi :

S → aB
A → bcD | dAC
B → e | Ab
C → bCb | adF | ab
F → cFB

Analisis

1) Aturan produksi A → bcD, variabel D tidak memiliki penurunan
2) Konsekuensi no (1), simbol variabel A tidak memiliki penurunan yang menuju terminal (tinggal A → dAC)
3) Konsekuensi no (2), B → Ab tidak memiliki penurunan
4) Simbol variabel F tidak memiliki penurunan yang menuju terminal
5) Konsekuensi no (4), C → adF tidak memiliki penurunan

Setelah disederhanakan menjadi:

S → aB
B → e
C → bCb | ab

Contoh lain lagi :

S → aBD
B → cD | Ab
D → ef
A → Ed
F → dc

Analisa

1) Aturan produksi A → Ed, E tidak memiliki penurunan
2) Aturan produksi F → dc, redundan

   Sisa aturan produksi:

   S → aBD
   B → cD | Ab
   D → ef

Analisa lagi

B → Ab, A tidak memiliki penurunan.

Hasil penyederhanaan:

S → aBD
B → cD
D → ef

Contoh lagi:

S → Abc | ab

A → AAA | ε

Aturan produksi setelah disederhanakan:

S → Abc | ab
A → AAA | ε

Ingat A → ε juga harus diperhitungkan

PRINSIP

Setiap kali melakukan penyederhanaan diperiksa lagi aturan produksi yang tersisa, apakah semua produksi yang useless sudah hilang.

Penghilangan Produksi Unit

- Produksi dimana ruas kiri dan kanan aturan produksi hanya berupa satu simbol variabel, misalkan: A → B, C → D.
- Keberadaannya membuat tata bahasa memiliki kerumitan yang tak perlu.
- Penyederhanaan dilakukan dengan melakukan penggantian aturan produksi unit.

Contoh:

S → Sb
S → C
C → D
C → ef
D → dd

Dilakukan penggantian berturutan mulai dari aturan produksi yang paling dekat menuju ke penurunan terminal-terminal ('=>' dibaca 'menjadi'):

- C → D => C → dd
- S → C => S → dd | ef

Sehingga aturan produksi setelah penyederhanaan:

S → Sb
S → dd | ef
C → dd
C → ef
C → dd

Contoh lain:

S → A
S → Aa
A → B
B → C
B → b
C → D
C → ab
D → b

Penggantian yang dilakukan :

- C → D => C → b
- B → C => B → b | ab, karena B → b sudah ada, maka cukup dituliskan B → ab
- A → B => A → ab | b
- S → A => ab | b

Sehingga aturan produksi setelah penyederhanaan:

S → ab | b
S → Aa
A → ab | b
B → ab
B → b
C → b
C → ab
D → b

Contoh lagi:

S → Cba | D
A → bbC
B → Sc | ddd
C → eAn | f | C
D → E | SABC
E → gh

Penggantian yang dilakukan:

- D → E menjadi D → gh
- C → C , kita hapus
- S → D menjadi S → gh | SABC

Sehingga aturan produksi setelah penyederhanaan:

S → Cba | gh | SABC
A → bbC
B → Sc | ddd
C → eA | f
D → gh | SABC
E → gh

Penghilangan Produksi ε

Produksi ε adalah produksi dalam bentuk

$$\alpha \rightarrow \varepsilon$$

atau bisa dianggap sebagai produksi kosong (empty). Penghilangan produksi ε dilakukan dengan melakukan penggantian produksi yang memuat variabel yang bisa menuju produksi ε, atau biasa disebut *nullable.*

Prinsip penggantiannya bisa dilihat kasus berikut:

S → bcAd
A → ε

A *nullable* serta A → ε satu-satunya produksi dari A, maka variabel A bisa ditiadakan, hasil penyederhanaan tata bahasa bebas konteks menjadi:

S → bcd

Tetapi bila kasusnya:

S → bcAd
A → bd | ε

A *nullable,* tapi A → ε bukan satu-satunya produksi dari A, maka hasil penyederhanaan:

S → bcAd | bcd
A → bd

Contoh lagi, terdapat tata bahasa bebas konteks:

S → Ab | Cd
A → d
C → ε

Variabel yang *nullable* adalah variabel C. Karena penurunan C → ε merupakan penurunan satu-satunya dari C, maka kita ganti S → Cd menjadi S → d. Kemudian produksi C → ε kita hapus.

Setelah penyederhanaan menjadi:

S → Ab | d
A → d

Contoh lain lagi:

S → dA | Bd
A → bc
A → ε
B → c

Variabel yang *nullable* adalah variabel A. A → ε bukan penurunan satu-satunya dari A (terdapat A → bc), maka kita ganti S → dA menjadi S → dA | d.A → ε kita hapus.

Setelah penyederhanaan :

S → dA | d | Bd
A → bc
B → c

Contoh tata bahasa bebas konteks:

S → AaCD
A → CD | AB
B → b | ε
C → d | ε
D → ε

Variabel yang *nullable* adalah variabel B, C, D. Kemudian dari A → CD, maka variabel A juga *nullable* (A → ε). Karena D hanya memilki penurunan D → ε, maka kita sederhanakan dulu:

- S → AaCD => S → AaC
- A → CD => A → C
- D → ε kita hapus

Selanjutnya kita lihat variabel B dan C memiliki penurunan ε, meskipun bukan satu-satunya penurunan, maka dilakukan penggantian:

- A → AB => A → AB | A | B
- S → AaC => S → AaC | aC | Aa | a

- B → ε dan C → ε kita hapus

  Setelah penyederhanaan:

  S → AaC | aC | Aa | a
  A → C | AB | A | B
  B → b
  C → ε

Variabel yang *nullable* adalah A, B, C. Dari S → AB, maka S juga *nullable*. Kita lakukan penggantian:

- A → aCa => A → aa
- B → bA => B → bA | b
- B → BB => B → BB | B
- A → abB => A → abB | ab
- S → AB => S → AB | A | B | ε
- C → ε, B → ε, A → ε dihapus

*Perhatikan untuk penggantian S → AB kita tetap mempertahankan S → ε, karena S merupakan simbol awal. Ini merupakan satu-satunya perkecualian produksi ε yang tidak dihapus, yaitu produksi ε yang dihasilkan oleh simbol awal.

Hasil akhir dari penyederhanaan:

S → AB | A | B | ε
A → abB | ab | aa
B → bA | b | BB | B

Contoh tata bahasa bebas konteks:

S → aAb
A → aAb | ε

Hasil penyederhanaan:

S → aAb | ab
A → aAb | ab

Contoh tata bahasa bebas konteks:

S → ABaC
A → BC
B → b | ε
C → D | ε
D → d

Hasil penyederhanaan:

S → ABaC | BaC | AaC | ABa | aC | Aa | Ba | a
A → B | C | BC
B → b
C → D
D → d

Prakteknya ketiga penyederhanaan tersebut dilakukan bersama pada suatu tata bahasa bebas konteks, yang nantinya menyiapkan tata bahasa bebas konteks tersebut untuk diubah kedalam suatu *bentuk normal Chomsky*.
Urutan penghapusan aturan produksi :

1) Hilangkan produksi ε
2) Hilangkan produksi unit
3) Hilangkan produksi *useless*

Contoh :

S → AA | C | bd
A → Bb | ε
B → AB | d
C → de

Hilangkan produksi ε, sehingga menjadi:
S → A | AA | C | bd
A → Bb
B → B | AB | d
C → de

Selanjutnya penghilangan produksi unit menjadi:

S → Bb | AA | de | bd
A → Bb
B → AB | d
C → de

Penghilangan produksi unit bisa menghasilkan produksi *useless*.
Terakhir dilakukan penghilangan produksi *useless:*
S → Bb | AA | de | bd
A → Bb
B → AB | d

Hasil akhir aturan produksi tidak lagi memiliki produksi ε, produksi unit, maupun produksi *useless*.

PERTEMUAN X
BENTUK NORMAL CHOMSKY

Pengertian 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 ε. Dengan kata lain, suatu tata bahasa bebas konteks dapat dibuat menjadi *bentuk normal Chomsky* dengan syarat tata bahasa bebas kontesk tersebut:

- Tidak memiliki produksi *useless*
- Tidak memiliki produksi unit
- Tidak memiliki produksi ε

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

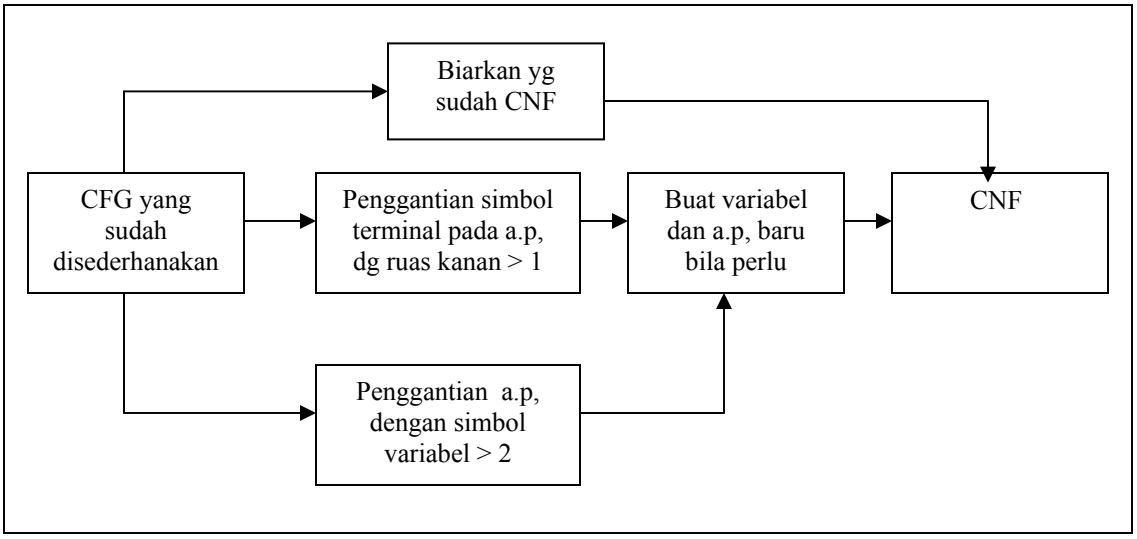
$$A \rightarrow BC$$
$$A \rightarrow b$$
$$B \rightarrow a$$
$$C \rightarrow BA \mid d$$

Pembentukan *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

Bisa dilihat tahapan-tahapan tersebut pada gambar 10.1



Tahapan-tahapan pembentukan bentuk normal Chomsky

Contoh, tata bahasa bebas konteks (kita anggap tata bahasa bebas konteks pada bab ini sudah mengalami penyederhanaan):

$$S \rightarrow bA \mid aB$$
$$A \rightarrow bAA \mid aS \mid a$$
$$B \rightarrow aBB \mid bS \mid b$$

Aturan produksi yang sudah dalam *bentuk normal Chomsky:*

$$A \rightarrow a$$
$$B \rightarrow b$$

Dilakukan penggantian aturan produksi yang belum *bentuk normal Chomsky* ('=>' bisa dibaca berubah menjadi):

$S \rightarrow bA \Rightarrow S \rightarrow P_1A$
$S \rightarrow aB \Rightarrow S \rightarrow P_1B$
$A \rightarrow bAA \Rightarrow S \rightarrow P_1AA \Rightarrow A \rightarrow P_1P_3$
$A \rightarrow aS \Rightarrow A \rightarrow P_2S$
$B \rightarrow aBB \Rightarrow B \rightarrow P_2BB \Rightarrow B \rightarrow P_2P_4$
$B \rightarrow bS \Rightarrow B \rightarrow P_1S$
Terbentuk aturan produksi dan simbol variabel baru:
$P_1 \rightarrow b$
$P_2 \rightarrow a$
$P_3 \rightarrow AA$
$P_4 \rightarrow BB$
Hasil akhir aturan produksi dalam *brntuk normal Chomsky :*

$A \rightarrow a$
$B \rightarrow b$
$S \rightarrow P_1A$
$S \rightarrow P_2B$
$A \rightarrow P_1P_3$
$A \rightarrow P_2S$
$B \rightarrow P_2P_4$
$B \rightarrow P_1S$
$P_1 \rightarrow b$
$P_2 \rightarrow a$
$P_3 \rightarrow AA$
$P_4 \rightarrow BB$

Contoh, 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:*

$S \rightarrow aB \Rightarrow S \rightarrow P_1B$
$A \rightarrow bc \Rightarrow S \rightarrow P_2P_3$
$B \rightarrow Ab \Rightarrow B \rightarrow A\ P_2$
$C \rightarrow aB \Rightarrow C \rightarrow P_1B$

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 :*

$S \rightarrow CA$
$A \rightarrow a$
$B \rightarrow BC$

$C \rightarrow b$
$S \rightarrow P_1B$
$S \rightarrow P_2P_3$
$B \rightarrow A\ P_2$
$C \rightarrow P_1B$
$P_1 \rightarrow a$
$P_2 \rightarrow b$
$P_3 \rightarrow c$

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:

$S \rightarrow aAB \Rightarrow S \rightarrow P_1P_2$
$S \rightarrow ch \Rightarrow S \rightarrow P_3P_4$
$A \rightarrow dbE \Rightarrow A \rightarrow P_5P_6$
$A \rightarrow eEC \Rightarrow A \rightarrow P_8P_9$
$B \rightarrow ff \Rightarrow B \rightarrow P_{10}P_{10}$
$C \rightarrow ADB \Rightarrow C \rightarrow AP_{11}$
$C \rightarrow aS \Rightarrow C \rightarrow P_1S$
$E \rightarrow jD \Rightarrow E \rightarrow P_{12}D$

Terbentuk aturan produksi baru:

$P_1 \rightarrow a$
$P_2 \rightarrow AB$
$P_3 \rightarrow c$
$P_4 \rightarrow h$
$P_5 \rightarrow d$
$P_6 \rightarrow P_7E$
$P_7 \rightarrow b$
$P_8 \rightarrow e$
$P_9 \rightarrow EC$

$P_{10} \rightarrow f$
$P_{11} \rightarrow DB$
$P_{12} \rightarrow j$

Hasil akhir dalam *bentuk normal Chomsky:*

$S \rightarrow CD$
$B \rightarrow DD$
$D \rightarrow i$
$S \rightarrow P_1P_2$
$S \rightarrow P_3P_4$
$A \rightarrow P_5P_6$
$A \rightarrow P_8P_9$
$B \rightarrow P_{10}P_{10}$
$C \rightarrow AP_{11}$
$C \rightarrow P_1S$
$E \rightarrow P_{12}D$
$P_1 \rightarrow a$
$P_2 \rightarrow AB$
$P_3 \rightarrow c$
$P_4 \rightarrow h$
$P_5 \rightarrow d$
$P_6 \rightarrow P_7E$
$P_7 \rightarrow b$
$P_8 \rightarrow e$
$P_9 \rightarrow EC$
$P_{10} \rightarrow f$
$P_{11} \rightarrow DB$
$P_{12} \rightarrow j$

Algoritma CYK untuk Tata Bahasa Bebas Konteks

Algoritma CYK merupakan algoritma *parsing* dan keanggotaan (*membership)* untuk tata bahasa bebas konteks. Algortima ini diciptakan oleh J. Cocke, DH. Younger, dan T. Kasami. Syarat untuk penggunaan algortima ini adalah tata bahasa harus berada dalam *bentuk normal Chomsky* . Obyektif dari algortima ini adalah untuk menunjukkan apakah suatu *string* dapat diperoleh dari suatu tata bahasa.

Algoritma CYK sebagai berikut:

```
begin
   1) for i:= 1 to n do
   2) Vi1 := {A| A → a aturan produksi dimana simbol ke- i
      adalah a };
   3) for j:= 2 to n do
```

```
   4)     for i:= 1 to (n-j+1) do
           begin
   5)           Vij:=Ø;
   6)           for k:=1 to (j - 1) do
   7)                Vij:= Vij ∪ ( A | A → BC adalah suatu
      produksi, dimana B di Vik dan C di Vi+k,j-k }
           end
end
```

Penjelasan:

- *n* = panjang untai yang akan diperiksa, missal : untuk untai ‘ada’, *n* = | ada | =3
- *i* akan menyatakan kolom ke-
- *j* akan menyatakan baris ke-
- tahapan no (1) dan (2) untuk mengisi table baris pertama kolom 1 – *n*
- no (3), interasi dari baris ke- 2 sampai *n*
- no (4), interasi untuk mengisi kolom 1 sampai (*n* – baris + *1*) pada suatu baris.
- no (5) inisialisasi V_{ij} dengan *Ø*
- no (6) dan no (7), interasi untuk memeriksa mana saja yang menjadi anggota V_{ij}

Kita lihat contoh kasus, dimana terdapat tata bahasa bebas konteks (simbol awal S):

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Periksalah apakah untai ‘baaba’ termasuk kedalam bahasa tersebut
Pertama – tama kita akan membuat tabel untuk V_{ij} ($V_{kolom,baris}$) sebagai berikut :

		b	a	a	b	a
				i →		
		1	2	3	4	5
↓	1					
	2					
	3					
	4					
	5					

Tabel diatas kita gunakan unruk mempermudah kita dalam menyelesaikan persoalan, *i* akan menyatakan kolom, *j* akan menyatakan baris.
Kita ketahui *n* = 5. Dari Algoritma langkah (1) dan (2) kita bisa mengisi baris pertama pada tabel, sebagai berikut:

- Untuk V_{11}, kita periksa variabel yang bisa menurunkan ‘b’, dari B → b kita isi V_{11}= {B}
- Untuk V_{21}, kita periksa variabel yang bisa menurunkan ‘a’, dari A → a dan C → a kita isi V_{21}{A,C}
- Untuk V_{31}, kita periksa varibel yang bisa menurunkan ‘a’, dari A → a dan C → a kita isi V_{31}={A,C}
- Untuk V_{41}, kita periksa variabel yang bisa menurunkan ‘b’, dari B → b kita isi V_{41}={B}
- Untuk V_{51}, kita periksa variabel yang bisa menurunkan’a’, dari A → a dan C → A kita isi V_{51}={A,C}

Dari hasil tersebut kita bisa tabel :

	b	a	a	b	a
			i →		
	1	2	3	4	5
1	B	A,C	A,C	B	A,C
2					
3					
4					
5					

↓

Selanjutnya kita akan mengisi baris ke-2 sampai *n* sebagai berikut
Pada baris ke -2 (*k =1)*

- Untuk V_{12}, periksa V_{ik}- $V_{i+k,\ j-k}$, berarti V_{11}-V_{21}, yaitu B-A,C, variabel yang bisa menurunkan BA atau BC adalah S dan A, maka V_{12} kita isi {S, A}
- Untuk V_{22}, periksa $V_{ik} - V_{i+k,\ j-k}$, berarti V_{21}-V_{31}, yaitu A,C-A,C, variabel yang bisa menurunkan AA, AC, CA, atau CC adalah B maka V_{22} kita isi {B}
- Untuk V_{32}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{31}-V_{41} yaitu A, C-B, variabel yang bisa menurunkan AB atau CB adalah S dan C, maka V_{12} kita isi {S, C}
- Untuk V_{42}, periksa $V_{ik\text{-}Vi+k,\ j-k}$ berarti V_{41}-V_{51}, yaitu A,C-B, variabel yang bisa menurunkan AB atau CB adalah S dan C, maka V_{12} kita isi {S,A}

Dari hasil tersebut kita bisa mengisi tabel:

	b	a	a	b	a
			i →		
	1	2	3	4	5
1	B	A,C	A,C	B	A,C
2	S,A	B	S,C	S,A	
3					
4					
5					

↓

Pada baris ke –3 (k = 1 sampai 2):

- Untuk V_{13}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{11}-V_{22} & V_{12}-V_{31}, yaitu B-B & S,A-A,C, variabel yang bisa menurunkan BB, SA,SC,AA, atau AC adalah tidak ada, maka V_{13} kita isi ∅
- Untuk V_{23}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{21}-V_{32} & V_{22}-V_{41}, yaitu A,C-S,C & B-B, variabel yang bisa menurunkan AS, AC, CS, CC, atau BB adalah B , maka V_{23} kita isi {B}
- Untuk V_{33}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{31}-V_{42} & V_{32}-$V_{51,}$ yaitu A,C-S,A & S,C-A,C variabel yang bisa menurunkan AS, AA, CS, CA, SA, SC, CA, atau CC adalah B, maka V_{33} kita isi {B}

Dari hasil tersebut kita bsa mengisi tabel:

	b	a	a	b	a
			i →		
	1	2	3	4	5
1	B	A,C	A,C	B	A,C
2	S,A	B	S,C	S,A	
3	∅	B	B		
4					
5					

Pada baris ke –4 (k = 1 sampai 3):

- Untuk V_{14}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{11}-V_{23} & V_{12}-V_{32} & V_{13}-$V_{41,}$ yaitu B-B & S,A-S,C & ∅-B, variabel yang bisa menurunkan BB, SS, SC, AS AC adalah tidak ada, maka V_{14} kita isi ∅
- Untuk V_{24}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{21}-V_{33} & V_{22}-V_{42} & V_{23}-$V_{51,}$ yaitu A,C-B & B-S,A & B-S,A & B-A,C, variabel yang bisa menurunkan AC, AB, BS, BA, BC adalah S, C, A, maka V_{24} kita isi {S,A,C}

Dari hasil tersbut kita bisa mengisi tabel:

	b	a	a	b	a
			i →		
	1	2	3	4	5
1	B	A,C	A,C	B	A,C
2	S,A	B	S,C	S,A	
3	∅	B	B		
4	∅	S,A,C			
5					

Pada baris ke –5 (k = 1 sampai 4)

- Untuk V_{15}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{11}-V_{24} & V_{12}-V_{33} & V_{13}-V_{42} & V_{14}-V_{51} yaitu B-S,A,C & S,A-B & ∅-S,A & ∅-A,C, variabel yang bisa menurunkan BA, BC, SA, SC, SB, atau AB adalah A,S,C maka V_{15} kita isi {S,A,C}

Dari hasil tersbut kita bisa mengisi tabel:

	b	A	a	b	a
			i →		
	1	2	3	4	5
1	B	A,C	A,C	B	A,C
2	S,A	B	S,C	S,A	
3	∅	B	B		
4	∅	S,A,C			
5	S,A,C				

↓

Perhatikan , syarat suatu untai dapat diturunkan dari simbol awal, V1n memuat simbol awal. Terlihat pada tabel, simbol awal S termuat di V_{15}, maka untai 'baaba' dapat diturunkan oleh tata bahasa tersebut.

Kita bisa mencoba-coba untuk membuat pohon penurunan dari untai 'baaba',
Kita lihat untuk contoh lain, terdapat tata bahasa bebas konteks:

S → AB | b
A → BA | a
B → AS | b

Periksalah apakah untai 'aaab' termasuk ke dalam bahasa tersebut

Pertama-tama kita akan membuat tabel untuk V_{ij} (Vkolom, baris) sebagai berikut:

	a	a	a	b
			i →	
	1	2	3	4
1				
2				
3				
4				

↓

Kita ketahui *n* = 4. Dari algoritma langkah (1) dan(2) kita bisa mengisi baris pertama pada tabel, sebagai berikut:

- Untuk V_{11}, kita periksa variabel yang bisa menurunkan 'a', dari A → a kita isi V_{11} = {A}
- Untuk V_{21}, kita periksa variabel yang bisa menurunkan 'a', dari A → a kita isi V_{21} = {A}
- Untuk V_{31}, kita periksa variabel yang bisa menurunkan 'a', dari A → a kita isi V_{31} = {A}
- Untuk V_{41}, kita periksa variabel yang bisa menurunkan 'b', dari B → b dan S → b kita isi V_{41} = {S,B}

Dari haisl tersebut kita bisa mengisi tabel:

	a	a	a	b
			i →	
	1	2	3	4
1	A	A	A	S,B
2				
3				
4				

j ↓

Selanjutnya kita akan mengisi baris ke –2 sampai *n* sebagai berikut:
Pada baris ke –2 ($k = 1$) :

- Untuk V_{12}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{11}-V_{21}, yaitu A-A, variabel yang bisa menurunkan AA adalah tidak ada, maka V_{12} kita isi ∅
- Untuk V_{22}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{21}-V_{31} ,yaitu A-A, variabel yang bisa menurunkan AA adalah tidak ada, maka V_{22} kita isi ∅
- Untuk V_{32}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{31}-V_{41} ,yaitu A,S-B, variabel yang bisa menurunkan AS atau AB adalah S dan B, maka V_{32} kita isi {S,B}

Dari hasil tersebut kita bisa mengisi tabel:

	a	a	a	b
			i →	
	1	2	3	4
1	A	A	A	S,B
2	∅	∅	S,B	
3				
4				

j ↓

Pada baris ke –3 ($k = 1$ sampai 2)

- Untuk V_{13}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{11}-V_{22} & V_{12}-V_{31}, yaitu A-∅ & ∅-A, variabel yang bisa menurunkannya adalah tidak ada, maka V_{13} kita isi ∅
- Untuk V_{23}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{21}-V_{32} & V_{22}-V_{41}, yaitu A-SB & ∅-SB, variabel yang bisa menurunkan AS atau AB adalah S dan B, maka V_{23} kita isi {S,B}

Dari hasil tersebut kita bisa mengisi tabel:

	a	a	a	b
			i →	
	1	2	3	4
1	A	A	A	S,B
2	∅	∅	S,B	
3	∅	S,B		
4				

j ↓

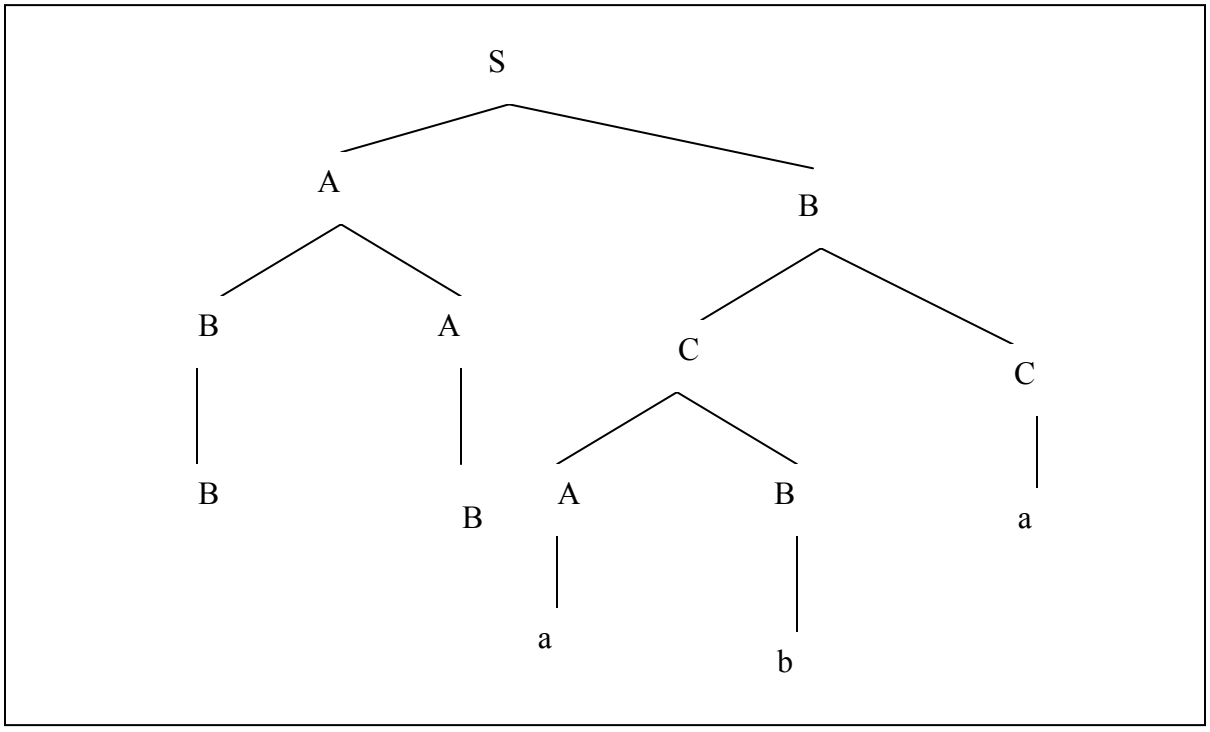
Pada baris ke –4 (k = 1 sampai 3):

- Untuk V_{14}, periksa V_{ik}-$V_{i+k,\ j-k}$, berarti V_{11}-V_{23} & V_{12}-V_{32} & V_{13}-V_{41}, yaitu A-SB & ∅-SB, variabel yang bisa menurunkan AS atau AB adalah S dan B, maka V_{14} kita isi {S,B}

Dari hasil tersebut kita bisa mengisi tabel:

		a	a	a	b
				i →	
		1	2	3	4
j ↓	1	A	A	A	S,B
	2	∅	∅	S,B	
	3	∅	S,B		
	4	S,B			

Terlihat pada tabel, simbol awal S termuat di V_{14}, maka untai ‘aaab’ dapat diturunkan oleh tata bahasa tersebut.



Pohon penurunan untuk untai ‘baaba’

PERTEMUAN XI
PENGHILANGAN REKURSIF KIRI

Aturan Produksi Rekursif

Aturan Produksi yang rekursif memilki ruas kanan (hasil produksi) yang memuat simbol variabel pada ruas kiri. Sebuah aturan produksi dalam bentuk:

$$A \rightarrow \beta A$$

merupakan aturan produksi yang rekursif kanan
$\beta=(V \cup T)^*$ atau kumpulan simbol variabel dan terminal

Contoh aturan produksi yang rekursif kanan:

$$S \rightarrow dS$$
$$B \rightarrow adB$$

Produksi dalam bentuk:

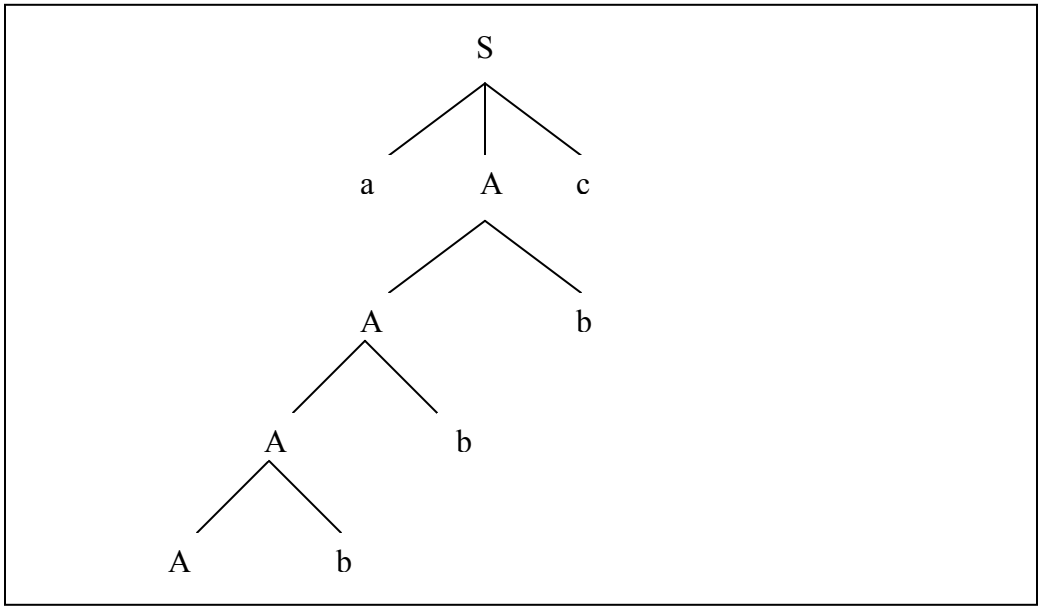
$$A \rightarrow A\beta$$

Merupakan aturan produksi yang rekursif kiri, contohnya:

$$S \rightarrow Sd$$
$$B \rightarrow Bad$$

Produksi yang rekursif kanan menyebabkan pohon penurunan tumbuh ke kanan, sebaliknya Produksi yang rekursif kiri menyebabkan pohon penurunan tumbuh ke kiri. Bisa dilihat pohon penurunanpada gambar 11.1 dari tata bahasa bebas konteks dengan aturan produksi:

$$S \rightarrow aAc$$
$$A \rightarrow Ab \mid \varepsilon$$



Gambar 11.1 *Pohon penurunan sebuah CFG yang rekursif 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 disini memungkinkan suatu tata bahasa bebas konteks nantinya diubah ke dalam *bentuk normal Greibach.*

Tahapan Penghilangan Rekursif Kiri

Langkah-langkah penghilangan rekursif kiri:

- Pisahkan aturan produksi yang rekursif kiri dan yang tidak, misal:

Aturan produksi yang rekursif kiri:

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \mid ..... A\alpha_n$$

Aturan produksi yang tidak rekursif kiri (termasuk produksi ε):

$$A \rightarrow \beta_1 \mid \beta_2 \mid \beta_3 \mid ..... \beta_m$$

- Dari situ kita bisa tentukan $\alpha_1, \alpha_2, ... \alpha_n,$ dan $\beta_1, \beta_2, ... \beta_m$ dari setiap aturan produksi yang memiliki simbol ruas kiri yang sama

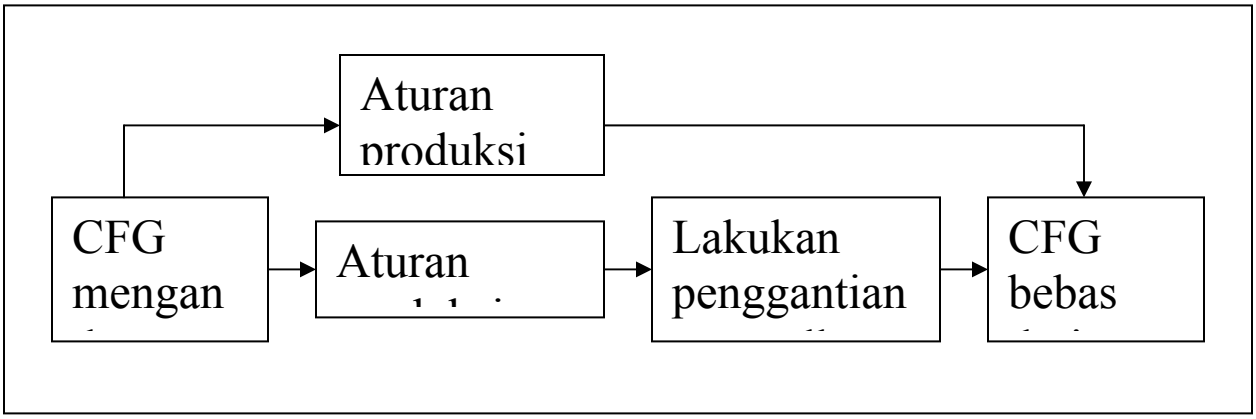
- Lakukan penggantian aturan produksi yang rekursif kiri, menjadi sebagai berikut:

1) $A \rightarrow \beta_1 Z \mid \beta_2 Z \mid .... \beta_m Z$
2) $Z \rightarrow \alpha_1 \mid \alpha_2 \mid \alpha_3 \mid .... \alpha_n$
3) $Z \rightarrow \alpha_1 Z \mid \alpha_2 Z \mid \alpha_3 Z \mid .... \alpha_n Z$

Penggantian diatas 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.

Tahapan-tahapan tersebut bisa dilihat pada Gambar berikut



Gambar 11.2 *Tahapan penghilangan 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:

$$S \rightarrow Sab \mid Sbd$$

Dari situ kita tentukan:
Untruk simbol ruas kiri S: α_1=ab, α_2=bd

Aturan produksi yang tidak rekursif kiri:

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

Dari situ kita dapatkan:
Untuk simbol ruas kiri S: β_1=aSc, β_2=dd, β_3=ff

Kita lakukan penggantian aturan produksi yang rekursif kiri:
Untuk yang memiliki simbol ruas kiri S:
S → Sab | Sbd, digantikan oleh:

i. $S \rightarrow aScZ_1 \mid dd\ Z_1 \mid ffZ_1$
ii. $Z_1 \rightarrow ab \mid bd$
iii. $Z_1 \rightarrow abZ_1 \mid bd\ Z_1$

Hasil akhir setelah penghilangan rekursif kiri adalah:

$$S \rightarrow aSc \mid dd \mid ff$$
$$S \rightarrow aScZ_1 \mid dd\ Z_1 \mid ffZ_1$$
$$Z_1 \rightarrow ab \mid bd$$
$$Z_1 \rightarrow abZ_1 \mid bd\ Z_1$$

*Pada kasus diatas S adalah satu-satunya simbol variabel yang menghasilkan produksi rekursif kiri.

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:

$$S \rightarrow Sab \mid Sb$$
$$A \rightarrow Aa$$

Dari situ kita tentukan:

Untuk simbol ruas kiri S: $\alpha_1 = ab$, $\alpha_2 = b$
Untuk simbol ruas kiri A: $\alpha_1 = a$
Aturan produksi yang tidak rekursif kiri:
$S \rightarrow cA$
$A \rightarrow a \mid bd$

Dari situ kita dapatkan

Untuk simbol ruas kiri S: $\beta_1 = cA$
Untuk simbol ruas kiri A: $\beta_1 = a$, $\beta_2 = bd$

Kita lakukan penggantian aturan produksi yang rekursif kiri:
Untuk yang memiliki simbol ruas kiri S:
$S \rightarrow Sab \mid Sb$, digantikan oleh:

i. $S \rightarrow cAZ_1$
ii. $Z_1 \rightarrow ab \mid b$
iii. $Z_1 \rightarrow abZ_1 \mid bZ_1$

Untuk yang memiliki simbol ruas kiri A :

$A \rightarrow Aa$, digantikan oleh:

i. $A \rightarrow a\ Z_2 \mid bdZ_2$
ii. $Z_2 \rightarrow a$
iii. $Z_2 \rightarrow a\ Z_2$

Hasil akhir setelah penghilangan rekursif kiri adalah:

$S \rightarrow cA$
$A \rightarrow a \mid bd$
$S \rightarrow cAZ_1$
$Z_1 \rightarrow ab \mid b$
$Z_1 \rightarrow abZ_1 \mid bZ_1$
$A \rightarrow a\ Z_2 \mid bdZ_2$
$Z_2 \rightarrow a$
$Z_2 \rightarrow a\ Z_2$

*Perhatikan bahwa penghilangan rekursif kiri memunculkan simbol variabel baru, dan aturan produksi baru yang rekursif kanan.

Contoh lain, terdapat tata bahasa bebas konteks:

$S \rightarrow Sa \mid aAc \mid c \mid \varepsilon$
$A \rightarrow Ab \mid ba$

Pertama-tama kita lakukan pemisahan aturan produksi
Aturan produksi yang rekursif kiri:

$S \rightarrow Sa$
$A \rightarrow Ab$

Dari situ kita tentukan:

Untuk simbol ruas kiri S: $\alpha_1 = a$
Untuk simbol ruas kiri A: $\alpha_1 = b$

Aturan produksi yang tidak rekursif kiri:

$S \rightarrow aAc \mid c \mid \varepsilon$
$A \rightarrow ba$

Dari situ kita dapatkan

untuk simbol ruas kiri S:$\beta_1 = aAc$, $\beta_2 = c$, $\beta_3 = \varepsilon$
untuk simbol ruas kiri A: $\beta_1 = ba$

*Perhatikan produksi ε termasuk produksi yang tidak rekursif kiri

Kita lakukan penggantian aturan produksi yang rekursif kiri:
Untuk yang memilki simbol ruas kiri S:
$S \rightarrow Sa$, digantikan oleh:

i. $S \rightarrow aAcZ_1 \mid cZ_1 \mid Z_1$
ii. $Z_1 \rightarrow a$
iii. $Z_1 \rightarrow a\ Z_1$

Untuk yang memiliki simbol ruas kiri A:
$A \rightarrow Ab$, digantikan oleh:

i. $A \rightarrow ba\ Z_2$
ii. $Z_2 \rightarrow b$

iii. $Z_2 \rightarrow bZ_2$

Hasil akhir setelah penghilangan rekursif kiri adalah:

$$S \rightarrow aAc \mid c \mid \varepsilon$$
$$S \rightarrow aAcZ_1 \mid cZ_1 \mid Z_1$$
$$A \rightarrow ba$$
$$A \rightarrow ba\ Z_2$$
$$Z_1 \rightarrow a$$
$$Z_1 \rightarrow a\ Z_1$$
$$Z_2 \rightarrow b$$
$$Z_2 \rightarrow b\ Z_2$$

PERTEMUAN 12
BENTUK NORMAL GREIBACH

Pengerian Bentuk Normal Greibach

Bentuk normal Greibach merupakan bentuk normal yang memiliki banyak konsekuensi teoritis dan prkatis. Dalam *bentuk normal Greibach* kita membatasi posisi munculnya terminal-terminal dan variabel-variabel. Suatu tata bahasa bebas konteks (CFG) dikatakan dalam *bentuk normal Greibach / Greibach Normal Form,* selanjutnya kita sebut sebagai GNF, jika setiap aturan produksinya ada dalam bentuk:

$A \rightarrow a\alpha$

a:simbol terminal (tunggal), a ε T
α: rangkaian simbol-simbol variabel (V*)

Atau dengan kata lain, suatu tata bahasa bebas konteks dalam *bentuk normal Greibach* bila hasil produksinya (ruas kanan) diawali dengan satu simbol terminal, slanjutnya bisa diikuti oleh rangkaian simbol variabel. Contoh tata bahasa bebas konteks dalam bentuk *bentuk normal Greibach:*

$S \rightarrow a \mid aAB$
$A \rightarrow aB$
$B \rightarrow cS$

Untuk dapat diubah ke dalam *bentuk normaol Greibach,* tata bahasa semula haru memenuhi syarat:

- Sudah dalam *bentuk normal Chomsky*
- Tidak bersifat rekursif kiri
- Tidak menghasilkan ε

Terdapat dua cara pembentukan *bentuk normal Greibach* , yaitu melalui substitusi dan perkalian matriks. Pada bagian berikutnya kita membahasa kedua cara tersebut.

12.2. Pembentukan *Bentuk Normal Greibach* dengan Substitusi

Secara umum langkah-langkah untuk mendapatkan *bentuk normal Greibach :*

1. Tentukan urutan simbol-simbol variabel yang ada dalam tata bahasa. Misalkan terdapat *m* variabel dengan urutan $A_1, A_2, ..., A_m$
2. Berdasarkan urutan simbol yang ditetapkan pada langkah (1) seluruh aturan produksi yang ruas kanannya diawali dengan simbol variabel dapat dituliskan dalam bentuk

$A_h \rightarrow A_i \gamma$

dimana h <> *i* (rekrusif kiri sudah dihilangkan), γ bisa berupa simbol-simbol variabel

   a. Jika $h < i$, aturan produksu ini sudah benar (tidakperlu 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

$A_h \rightarrow A_p \gamma$ (dimana $h \leq p$)

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 variabelsemula dimana 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

   $A_x \rightarrow A_y \gamma$ (di mana x < y)

   Produksi-produksi yang lain ada dalam bentuk:

   $A_x \rightarrow a \gamma$ (a = simbol terminal)

   $B_x \rightarrow \gamma$

   (B_2 = simbol variabel baru yang akan 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-2}, ... Dengan cara ini aturan produksi dalam bentuk $A_x \rightarrow A_y \gamma$ dapat diubah sehinga ruas kanannya dimulai dengan simbol terminal.
6. Produksi dalam bentuk $Bx \rightarrow \gamma$ 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 → CA
A → a | d
B → b
C → DD
D → AB

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

*Perhatikan urutan tersebut boleh anda tentukan sendiri, buatlah urutan sedemikian sehingga memudahkan untuk proses selanjutnya

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

- S → CA (sudah memenuhi aturan karena S<C)
- C → DD (sudah memenuhi karena C<D)
- D → AB (tidak memenuhi, karena D>A)

Yang belum memenuhi urutan yang telah kita tentukan adalah: D → AB, karena ruas kiri > simbol pertama pada ruas kanan. Maka kita lakukan sibstitusi pada simbol variabel A, aturan produksi menjadi:

D → aB | dB

Setelah semua aturan produksi sudah memenuhi ketentuan urutan variabel, kita lakukan substitusi mundur pada aturan produksi yang belum dalam *bentuk normal Greibach* ('=>' dibaca 'menjadi'):

- C → DD => C → aBD | dBD
- S → CA => S → aBDA | dBDA

*Perhatikan 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 disubstitusikan daripada S)

Hasil akhir aturan produksi yang sudah dalam *bentuk normal Greibach* :

S → aBDA | dBDA
A → a | d
B → b
C → aBD | dBD
D → aB | dB

*Perhatikan : setiap substitusi kita lakukan pada simbol variabel pertamapada ruas kanan (pada aturan produksi yang belum *bentuk normal Greibach* tentunya).

Prinsipnya:

- Biarkan aturan produksi yang sudah dalam *bentuk normal Greibach*
- Tentukan pengurutan simbol variabel, berdasarkan kondisi aturan produksi yang ada buatlah urutan sedemikian sehingga memudahkan untuk proses selanjutnya. Mulailah terlebih dahulu dari seimbol awal.
- Lakukan perubahan pada aturan produksi yang belum memenuhi ketentuan urutan tersebut dan bila perlu selama proses itu bisa dilakukan substitusi dan penghilangan rekursif kiri
- Lakukan substitusi mundur sedemikian rupa sehingga semua aturan produksi akan diawali dengan tepat sebuah simbol terminal. Proses substitusi mundur dimulai dari aturan produksi dengan urutan paling akhir.
- Lakukan substitusi mundur juga pada aturan produksi baru yang muncul sebagai hasil penghilangan rekursif kiri.

Contoh lain (simbol awal A):

A → BC
B → CA | b
C → AB | a

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

A → BC (sudah memenuhi karena A<B)
B → CA (sudah memenuhi karena B<C)
C → AB (padahal C > A sehingga harus diubah)

Pengubahan C → AB:
C → AB => C → BCB => C → CACB | bCB

Untuk C → CACB lakukan penghilangan rekursif kiri menjadi
- C → bCBZ1 | aZ_1
- Z_1 → ACB
- Z_1 → ACBZ_1

Kita lihat seluruh hasil produksi dari variabel C, sudah dalam *bentuk normal Greibach:*
C → bCBZ_1 | aZ_1 | bCB | a

Setelah semua aturan produksi sudah memenuhi ketentuan urutan variabel, kita laukan substitusi mundur:
B → CA => B → bCBZ_1A | aZ_1A | bCBA | aA
A → BC => A → bCBZ_1AC | aZ_1AC | bCBAC | aAC | bC

Selanjutnya lakukan pula substitusi pada aturan produksi dengan variabel baru yang terbentuk (pada contoh ini Z_1):

- Z_1 → ACB => Z_1 → bCBZ_1ACCB | aZ_1ACCB | bCBACCB | aACCB | bCCB
- Z_1 → ACBZ_1 => Z_1 → bCBZ_1ACCBZ_1 | aZ_1ACCBZ_1 | bCBACCBZ_1 | aACCBZ_1 | bCCBZ_1

Hasil akhir aturan produksi dalam bentuk *bentuk normal Greibach:*
A → bCBZ_1AC | aZ_1AC | bCBAC | aAC | bC |
B → bCBZ_1A | aZ_1A | bCBA | aA | B
C → bCBZ_1 | aZ_1 | bCB | a
Z_1 → bCBZ_1ACCB | aZ_1ACCB | bCBACCB | aACCB | bCCB
Z_1 → bCBZ_1ACCBZ_1 | aZ_1ACCBZ_1 | bCBACCBZ_1 | aACCBZ_1 | bCCBZ_1