

MAKALAH
“TEORI BAHASA DAN OTOMATA”

“Disusun sebagai tugas akhir pada mata kuliah Praktikum Teori Bahasa dan Otomata, dengan
Dosen Widya Darwin S.Pd., M.Pd.T”



Disusun Oleh :

Nama Anggota	: Mardhyah Fathania Izzati
NIM	: 21346013
Prodi	: Informatika

PROGRAM STUDI INFORMATIKA
JURUSAN TENIK ELEKRONIKA
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2023

KATA PENGANTAR

Rasa syukur kita hanya milik Allah SWT atas segala semua rahmatnya, sehingga saya dapat menyelesaikan makalah yang saya susun ini. Meskipun banyak rintangan dan hambatan yang saya alami dalam proses pekerjaan tetapi saya berhasil mengerjakan dengan baik dan tetap pada waktunya.

Dan harapan saya di sini semoga atas makalahh yang saya buat ini bisa menambah pengetahuan dan pengalaman bagi para pembaca, dan untuk kedepan nya kiat juga sama-sama memperbaiki bentuk atau menambah isi dari makalah agar semua akan lebih baik dengan sebelumnya.

Karena dari semua keterbatasan dari pengetahuan atau pun pengalaman saya, saya yakin masih banyak sekali dari kekurangan yang terdapat pada makalah ini. Oleh karena itu saya sangat berharap untuk saran dan kritik yang bisa membangun dari pembaca demi semua makalah ini akan terselesaikan dengan benar.

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI.....	ii
PENGANTAR TEORI BAHASA & OTOMATA	1
A. Tingkat Bahasa Pemrograman	1
B. Proses Kompilasi.....	1
C. Pengertian Dasar	2
D. Contoh mesin otomata sederhana.....	3
SIMBOL SRING DAN BAHASA	4
A. Pengertian simbol.....	4
B. Pengertian String	4
C. Pengertian bahasa.....	4
ATURAN PRODUKSI DAN GRAMER	5
A. Simbol terminal dan non terminal	5
B. Aturan Produksi.....	5
C. Hirarki Chomsky	6
D. Tipe 0 – Unrestricted Grammar	6
E. Tipe 1 – Context Sensitive Grammar	7
F. Tipe 2 – Context Free Grammar	7
G. Tipe 3 – Regular Grammar.....	8
FINAL STATE AUTOMATA	9
A. Pengertian FSA	9
B. Arti bentuk symbol pada graf transisi FSA.....	9
C. Pernyataan Final State Automata secara Formal.....	9
DETERMINISTIC FINAL STATE AUTOMATA	11
NON DETERMINISTIC FINITE STATE AUTOMATA	16
FINITE STATE TRANSDUCER (FST)	22
CONTAXT FREE GRAMMAR.....	24
PENGANTAR ATURAN PRODUKSI REKURSIF KIRI	27
PENGHILANGAN REKURSIF KIRI.....	28
CONTOH KASUS PENGHILANGAN REKURSIF KIRI.....	29
PENYEDERHANAAN ATURAN PRODUKSI CONTEXT FREE GRAMMAR	31
CHOMSKY NORMAL FORM (CNF).....	36

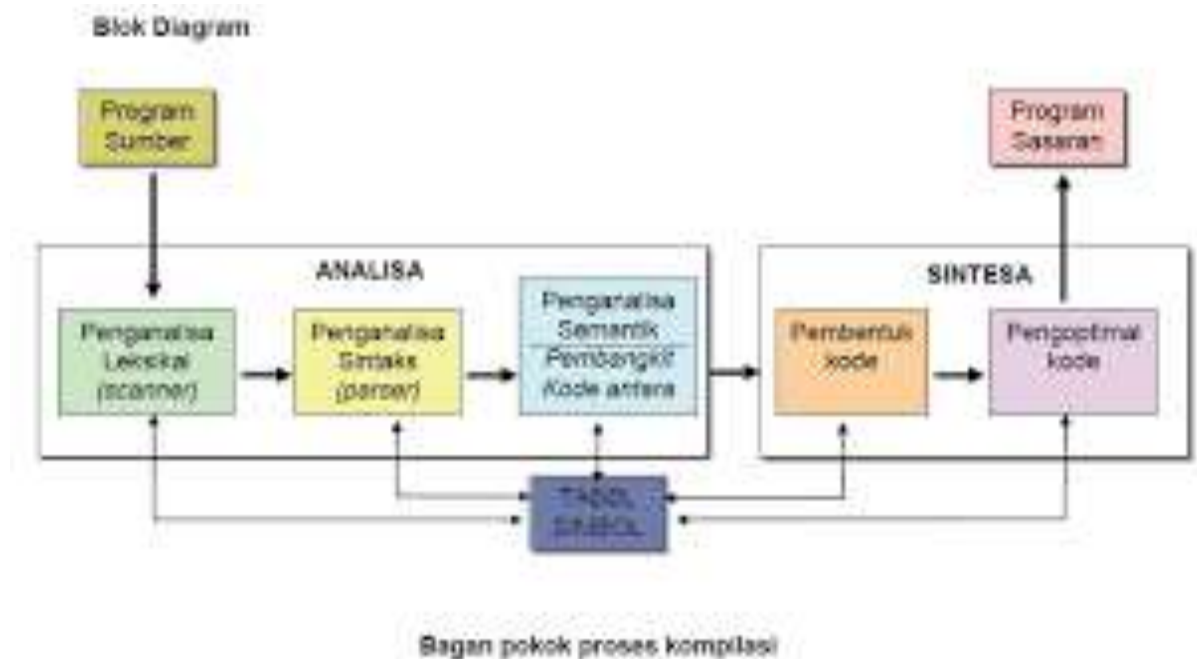
PENGANTAR TEORI BAHASA & OTOMATA

A. Tingkat Bahasa Pemrograman

Pada umumnya terdapat 3 tingkat bahasa pemrograman.

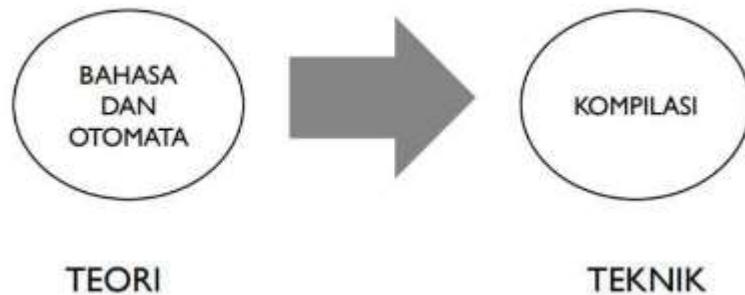
1. Bahasa Tingkat Rendah
 - Disebut rendah karena bahasa ini sulit untuk dimengerti oleh manusia
 - Contohnya adalah bahasa mesin
 - Bahasa yang dipahami oleh mesin ini berisi kode biner, yaitu kode yang terdiri dari bit bit 0 dan 1.
 - Tentu hal tersebut sangat sulit untuk dimengerti oleh manusia
2. Bahasa Tingkat Menengah
 - Bahasa pemrograman pada tingkat ini adalah bahasa yang berada satu tingkat di atas bahasa mesin, meskipun masih belum begitu mudah dimengerti
 - Contohnya adalah Assembly
3. Bahasa Tingkat Tinggi (High Level Programming Language)
 - Bahasa pemrograman jenis ini adalah bahasa pemrograman yang dapat dimengerti manusia
 - Contohnya seperti bahasa Python, C++, Java dan lainnya

B. Proses Kompilasi



C. Pengertian Dasar

Teori bahasa dan otomata adalah dasar dari teknik kompilasi, tapi apa itu teori bahasa? dan apa itu otomata?



Teori Bahasa

Teori bahasa ini membahas mengenai bahasa formal (formal language), terutama untuk pemroses teks (text processor) dan untuk kepentingan perancangan compiler. Bahasa formal ini merupakan kumpulan dari kalimat, dimana kalimat dalam sebuah bahasa dibangkitkan oleh tata bahasa.

Otomata

Otomata atau automata adalah mesin abstrak yang dapat mengenali (recognize), menerima (accept), atau membangkitkan (generate) sebuah kalimat dalam bahasa tertentu. Apa maksudnya mesin abstrak? jika kita telaah kata abstrak memiliki makna yaitu suatu bentuk yang tidak dapat didefinisikan atau memiliki bentuk yang tidak beraturan. Kata “automata” itu sendiri berasal dari bahasa Yunani yaitu automatos yang berarti “bertindak sendiri” atau “bekerja secara otomatis”.

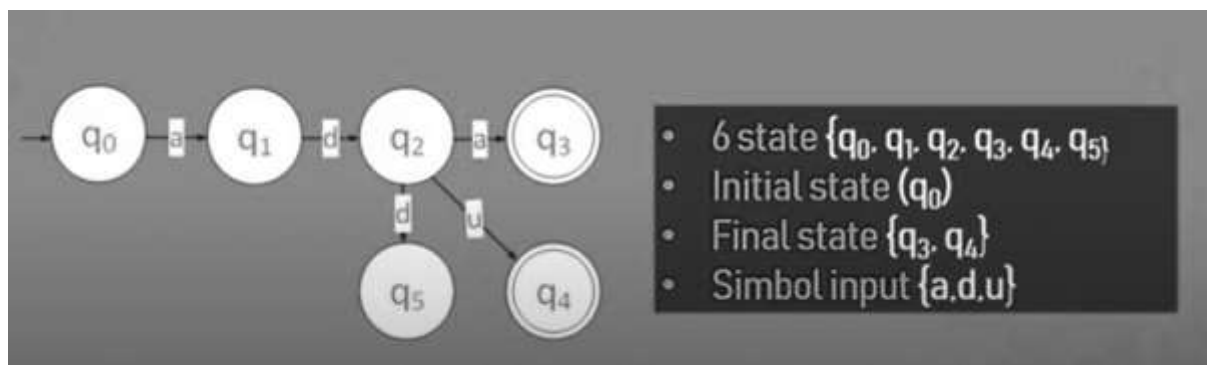
Jadi otomata atau disebut juga mesin abstrak ini menyerupai fungsi dalam matematika atau sebuah program komputer, dimana mesin abstrak ini dapat menerima input, memprosesnya dan menghasilkan output sesuai prosedur yang ditentukan.

Perlu diingat bahwa pengertian mesin bukanlah sekedar mesin yang berwujud seperti mesin elektronis / mekanis saja, melainkan segala sesuatu yang memenuhi ketiga ciri diatas (recognize, accept, generate) termasuk juga dengan perangkat lunak.

Oke mari kita coba bahas contoh dari mesin abstrak atau otomata ini. Apakah kalian tahu Mesin Minuman / makanan yang biasa disebut Vending Machine? jika kalian berpikir mengenai “loh mesin minuman gitu kan terlihat bentuknya, bukan mesin abstrak dong?”. Oke jadi gini temen temen, mesin minuman ini dibuat untuk menjual minuman secara otomatis dengan alur singkatnya seperti berikut:

- Mesin menerima pilihan minuman dari pembeli -> menerima input (accept)
- Mesin menerima uang yang dimasukkan oleh pembeli -> mengenali uang dan mengecek jumlahnya (recognize)
- Apabila uangnya sesuai maka -> mengeluarkan minuman (generate)

D. Contoh mesin otomata sederhana

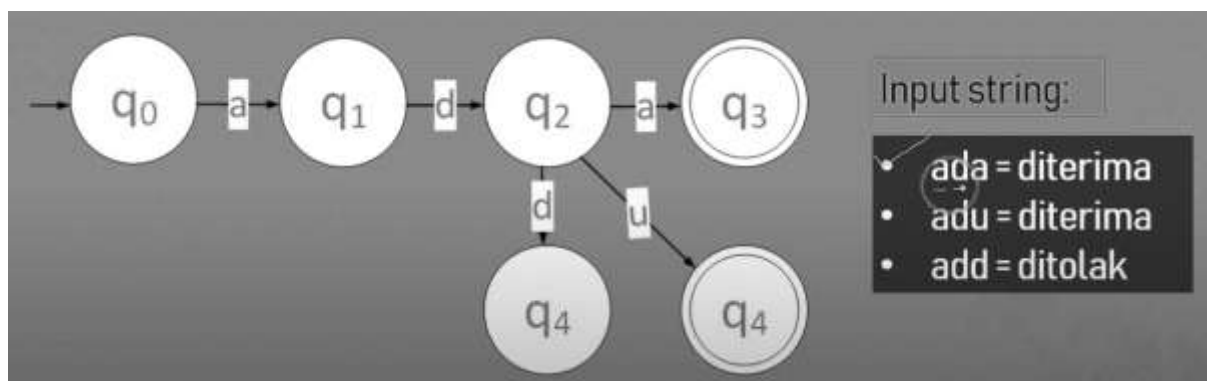


String input diterima jika mencapai final state, selain itu ditolak

Pembacaan simbol pertama dimulai dari initial state

Perpindahan state berdasarkan simbol yang dibaca

Maka hasilnya akan menjadi



SIMBOL STRING DAN BAHASA

A. Pengertian simbol

Simbol adalah sebuah entitas abstrak (seperti halnya pengertian titik dalam geometri). Sebuah huruf atau sebuah angka adalah contoh simbol. •

B. Pengertian String

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.

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 $|w| = 4$.

- String hampa adalah sebuah string dengan nol buah simbol. String hampa dinyatakan dengan simbol ϵ (atau Λ) sehingga $|\epsilon| = 0$. String hampa dapat dipandang sebagai simbol hampa karena keduanya tersusun dari nol buah simbol.

- Alfabet adalah himpunan hingga (finite set) simbol-simbol.

Suatu sistem yang meliputi pengungkapan gagasan, fakta, konsep, termasuk sekumpulan simbol-simbol dan aturan untuk melakukan manipulasinya

C. Pengertian bahasa

Bahasa adalah Suatu sistem yang meliputi pengungkapan gagasan, fakta konsep, termasuk sekumpulan simbol-simbol dan aturan untuk melakukan manipulasinya. Alfabet adalah himpunan berhingga dari simbol-simbol.

Bahasa juga disebut sebagai rangkaian simbol yang memiliki makna.

- oSebuah bahasa adalah himpunan string-string dari simbol-simbol untuk suatu alfabet.

- oKarena sebuah bahasa adalah kumpulan dari string-string, maka kita bisa mempunyai bahasa yang tidak mempunyai string-string, yaitu Bahasa kosong yang dinotasikan seperti menuliskan notasi himpunan kosong.

- oInput pada mesin otomatis dianggap sebagai Bahasa yang harus dikenali oleh mesin.

- oMesin akan mengindikasikan apakah suatu Bahasa dapat diterima atau tidak

ATURAN PRODUKSI DAN GRAMER

A. Simbol terminal dan non terminal

Apa itu simbol terminal?

- simbol terminal adalah simbol yang tidak dapat diturunkan lagi
- arti bisa **diturunkan** yaitu misalkan simbol A yang dapat diturunkan menjadi bc

yang termasuk simbol terminal:

- huruf kecil alfabet: a,b,c
- simbol operator: + (tambah), – (kurang)
- simbol tanda baca: , (koma) ! (tanda seru)
- string yang tercetak tebal, misalnya: if, then dan else

Apa itu simbol NON terminal (Variabel)?

- adalah simbol yang masih bisa diturunkan menjadi simbol terminal atau non terminal lainnya

yang termasuk simbol NON terminal (variabel):

- huruf besar alfabet: A, B, C
- huruf S sebagai simbol awal
- String yang tercetak miring misal expr dan stmt

B. Aturan Produksi

Dalam teori bahasa dan otomata, aturan produksi dinyatakan dalam:

$\alpha \rightarrow \beta$

(dibaca: alpha menghasilkan atau menurunkan beta)

- Dengan menerapkan aturan produksi, suatu tata bahasa bisa menghasilkan sejumlah string.
- Himpunan semua string tersebut adalah bahasa yang didefinisikan oleh tata bahasa tersebut.

contoh aturan produksi:

$T \rightarrow a$

(dibaca: T menghasilkan a)

Contoh lain:

$E \rightarrow T \mid T + E$

(dibaca: E menghasilkan T atau E menghasilkan T + E)

C. Hirarki Chomsky

Grammar

- Grammar atau tata bahasa didefinisikan secara formal sebagai kumpulan dari himpunan himpunan variabel, simbol simbol terminal, simbol awal, yang dibatasi oleh aturan aturan produksi.

4 tingkatan tata bahasa menurut Chomsky:

1. Tipe 0 – Unrestricted Grammar
2. Tipe 1 – Context Sensitive Grammar
3. Tipe 2 – Context Free Grammar
4. Tipe 3 – Regular Grammar

D. Tipe 0 – Unrestricted Grammar

- Aturan produksinya tidak memiliki batasan
- Dalam aturan $\alpha \rightarrow \beta$ pada tipe 0:
 - α adalah string terminal dan non-terminal dengan setidaknya 1 non-terminal
 - α tidak boleh kosong
 - β adalah rangkaian simbol terminal dan non – terminal
 - α adalah $(V + T)^*V(V+T)^*$
 - β adalah $(V+T)^*$
 - note:
 - (tanda plus '+' dibaca atau)
 - (tanda * (asterisk) yang berarti bisa tidak muncul atau bisa juga muncul hingga berkali kali $(0 - n)$)
- Tipe ini menghasilkan bahasa yang dikenali oleh mesin Turing
- Bahasa ini juga dikenal dengan nama “Recursively Enumerable Languages”.

jadi aturan yang perlu diingat adalah:

- α adalah $(V + T)^*V(V+T)^*$
- β adalah $(V+T)^*$

Contoh penerapan aturan tipe 0:

- $S \rightarrow ACaB$ (Terpenuhi, karena di ruas kiri ada non terminal, di ruas kanan ada terminal dan non-terminal)
- $Bc \rightarrow acB$ (Terpenuhi, karena di ruas kiri ada non terminal, di ruas kanan ada terminal dan non-terminal)

- $CB \rightarrow DB$ (Terpenuhi)
- $aD \rightarrow Db$ (Terpenuhi)
- $Sab \rightarrow ba$ (Terpenuhi)
- $A \rightarrow S$ (Terpenuhi)

E. Tipe 1 – Context Sensitive Grammar

- Aturan produksinya sama dengan tipe 0 namun dibatasi dengan aturan $|a| \leq |b|$ (jumlah simbol di ruas kiri harus lebih kecil atau sama dengan jumlah simbol di ruas kanan)
- Aturan $S \rightarrow \epsilon$ dibolehkan jika S tidak muncul pada ruas kanan setiap aturan
- Menghasilkan bahasa yang dikenali oleh Linear Bound Automata

jadi aturan yang perlu diingat adalah:

- $|a| \leq |b|$
- a adalah $(V+T)^*V(V+T)^*$
- b adalah $(V+T)^*(V+T)(V+T)^*$

Contoh penerapan:

- $S \rightarrow AB$ (Terpenuhi)
- $AB \rightarrow abcd$ (Terpenuhi, ukuran ruas kiri lebih kecil dari ruas kanan)
- $B \rightarrow b$ (Terpenuhi)
- $aD \rightarrow Db$ (Terpenuhi, ukuran kedua ruas sama)
- $aB \rightarrow aa \mid aaAA$ (Terpenuhi)
- $Ac \rightarrow Bbcc$ (Terpenuhi)

F. Tipe 2 – Context Free Grammar

- Aturan produksi ini sama dengan tipe 0 namun a sisi kiri hanya boleh memiliki 1 variabel non terminal ($|a| = 1$) dan b tidak memiliki batasan
- a adalah sebuah simbol non terminal
- b dapat berupa rangkaian atau simbol terminal, non terminal atau epsilon
- tipe ini menghasilkan bahasa yang dikenali oleh Non Deterministic Push Down Automata

jadi aturan yang perlu diingat adalah:

- $|a| = 1$
- a adalah V
- b adalah $(V+T)^*$

contoh:

- $S \rightarrow X a$ (Terpenuhi, di ruas kiri hanya ada satu non terminal yaitu S dan di ruas kanan boleh terminal atau non terminal)
- $X \rightarrow a$ (Terpenuhi)
- $X \rightarrow aX$ (Terpenuhi)
- $X \rightarrow abc$ (Terpenuhi)
- $X \rightarrow \epsilon$ (Terpenuhi)
- $S \rightarrow AB$ (Terpenuhi)
- $A \rightarrow a$ (Terpenuhi, di ruas kiri hanya ada satu non terminal dan di ruas kanan ada satu terminal)
- $B \rightarrow b$ (Terpenuhi)

G. Tipe 3 – Regular Grammar

- Aturan $S \rightarrow \epsilon$ dibolehkan jika S tidak muncul pada ruas kanan setiap aturan
- α adalah sebuah simbol non terminal
- β adalah simbol terminal atau simbol terminal dengan sebuah simbol variabel yang jika ada terletak pada posisi paling kanan
- Menghasilkan bahasa yang dikenali oleh Finite State Automata

jadi perlu diingat:

- α adalah V
- β adalah T^* atau T^*V

Contoh:

- $X \rightarrow \epsilon$
- $X a \mid aY$
- $Y \rightarrow b$
- $S \rightarrow abB$
- $B \rightarrow cd$

FINAL STATE AUTOMATA

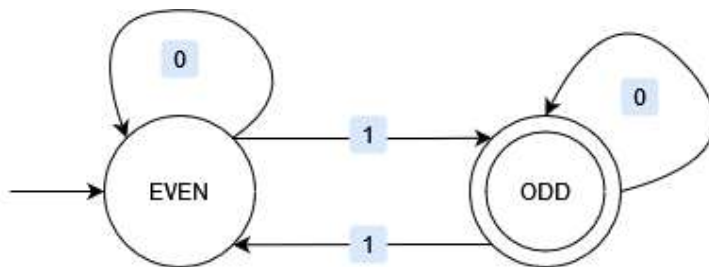
A. Pengertian FSA

- FSA adalah mesin abstrak berupa sistem model matematika dengan masukan dan keluaran diskrit yang dapat mengenali bahasa paling sederhana
- mekanisme kerja FSA dapat diterapkan pada analisis leksikal, text editor, protocol dan komunikasi jaringan

contoh analisis leksikal:

- analisis leksikan adalah salah satu bagian dari proses penerjemahan
- code yang dituliskan akan dianalisis seperti yang mana variabel, operator, keyword

B. Arti bentuk symbol pada graf transisi FSA



Keterangan gambar:

- Initial state ditandai dengan busur tanpa asal
- Lingkaran menyatakan state
- Label pada lingkaran adalah nama state, yaitu EVEN dan ODD
- Busur adalah transisi / arah perpindahan state
- Label pada busur adalah simbol input, yaitu 0 dan 1
- Lingkaran ganda menyatakan final state, yaitu ODD

C. Pernyataan Final State Automata secara Formal

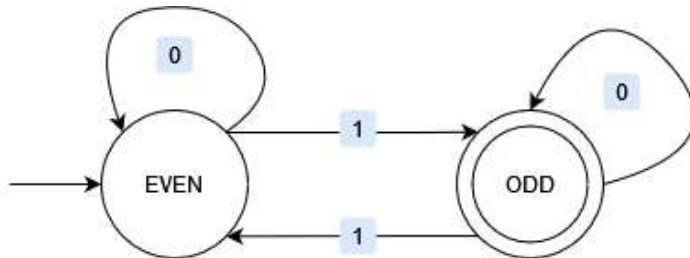
FSA dapat dinyatakan dalam 5 tupel atau $M=(Q, \Sigma, \delta, q_0, F)$ dimana:

- Q adalah Himpunan state atau kedudukan
- Σ (dibaca: sigma) adalah Himpunan symbol input / masukan / abjad
 - Jika pada contoh sebelumnya, terdapat input / simbol yaitu 0 dan 1
- δ (dibaca: delta) adalah Fungsi transisi
- q_0 (atau bisa juga S) adalah State awal $q_0 \in Q$
- F adalah Himpunan State akhir (final) $F \subseteq Q$

Catatan:

- Jumlah state akhir bisa lebih dari satu

Contoh:



- $Q = \{ \text{ODD}, \text{EVEN} \}$
- $\Sigma = \{0, 1\}$
- $q_0 = \text{EVEN}$
- $F = \{ \text{ODD} \}$
- δ (Transisi):
 - $\delta(\text{EVEN}, 0) = \text{EVEN}$ State EVEN menerima inputan 0 ke EVEN
 - $\delta(\text{EVEN}, 1) = \text{ODD}$
 - $\delta(\text{ODD}, 0) = \text{ODD}$
 - $\delta(\text{ODD}, 1) = \text{EVEN}$

catatan: Final state $F = \{ \text{ODD} \}$ berupa himpunan karena final state dapat berjumlah lebih dari satu

Jadi aturan ini menyesuaikan model mesinnya

Contoh ketika mesin digunakan:

- Ketika mendapat input 1101, maka urutan state yang terjadi adalah
EVEN1ODD1EVEN0EVEN1ODD
- berakhir dengan state ODD, maka 1011 diterima mesin
- Karena diterima maka dapat diketahui bahwa jumlah bit 1 nya adalah ganjil

Contoh Lain:

- Ketika mendapat input 101, maka urutan state yang terjadi adalah
EVEN1ODD0ODD1EVEN
- berakhir dengan state EVEN, maka 101 ditolak mesin
- Karena ditolak maka dapat diketahui nilai bit nya genap

Berdasarkan pendefinisian kemampuan merubah statenya, FSA dikelompokkan kedalam 2 jenis, yaitu:

- Deterministic FSA
- Non Deterministic FSA

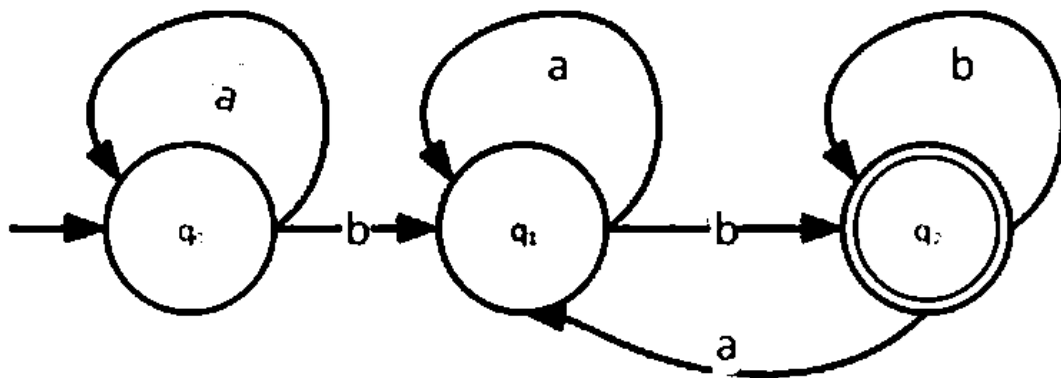
DETERMINISTIC FINAL STATE AUTOMATA

Berdasarkan pendefinisian kemampuan merubah statenya, FSA dikelompokkan ke dalam dua jenis

- Deterministic FSA
- Non- Deterministic FSA

Contoh 1:

Pada DFA, dari suatu state hanya ada tepat satu state berikutnya untuk setiap simbol masukan yang diterima



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

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

$S = q_0$

$F = \{q_2\}$

Fungsi Transisi

$\delta(q_0, a) = q_0$	$\delta(q_0, b) = q_1$
$\delta(q_1, a) = q_1$	$\delta(q_1, b) = q_2$
$\delta(q_2, a) = q_1$	$\delta(q_2, b) = q_2$

Tabel transisi

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_2

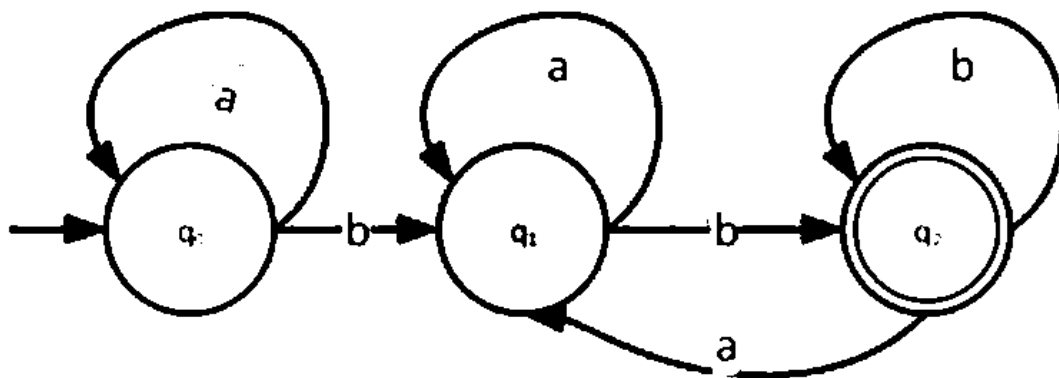
Kapan suatu string input dinyatakan diterima?

Suatu string x dinyatakan diterima, Bila $\delta(S, x)$ berada pada state akhir.

Bila M adalah sebuah bahasa FSA

$M = (Q, \Sigma, \delta, S, F)$ menerima bahasa yang disebut $L(M)$ yang merupakan himpunan $\{ x \mid \delta(S, x) \text{ di dalam } F \}$

Contoh 2:

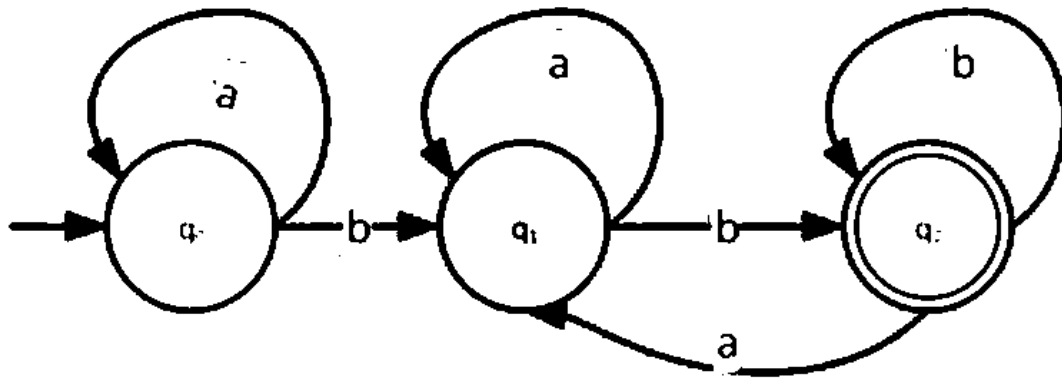


Jika pada gambar contoh 1 kita inputkan string 'abb'. Maka:

$$\delta(q_0, abb) = \delta(q_0, bb) = \delta(q_1, b) = q_2$$

Karena q_2 adalah state akhir maka 'abb' berada dalam $L(M)$

Contoh 3:

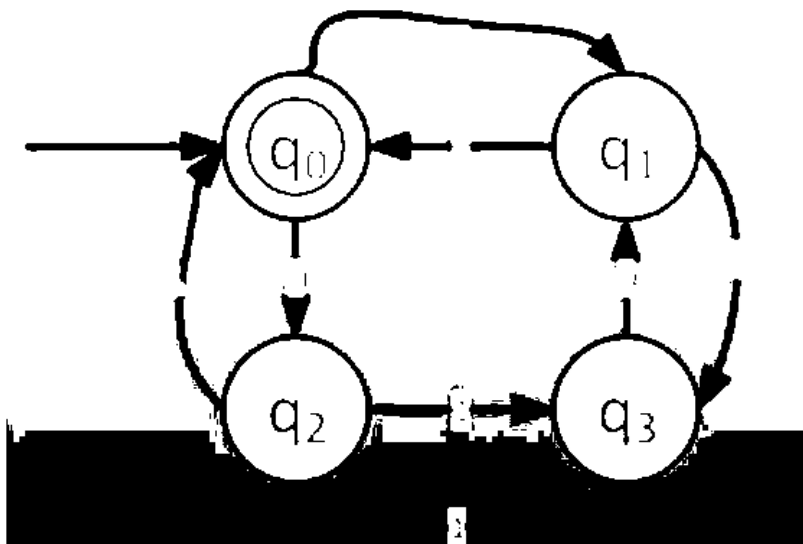


Jika pada gambar contoh 1 inputkan string 'baba'. Maka;

$$\delta(q_0, \text{baba}) = \delta(q_1, \text{aba}) = \delta(q_1, \text{ba}) = \delta(q_2, \text{a}) = q_1$$

Karena q1 bukan state akhir maka 'baba' tidak berada dalam $L(M)$

Contoh 4 – Pengecekan bit 0 dan 1 ganap



Tabel transisi

δ	0	1
q0	q2	q1
q1	q3	q0
q2	q0	q3
q3	q1	q2

Konfigurasi NFA berikut :

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

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

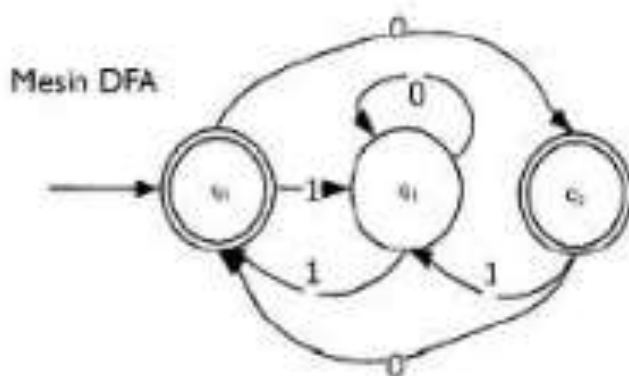
$S = q_0$

$F = \{q_0\}$

Secara formal dinyatakan sebagai berikut:

String	Status
0011	Diterima
00111	Ditolak
01010	Ditolak
010111	Diterima

Contoh 5



Tabel transisi:

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

Secara formal dinyatakan sebagai berikut:

String	Status
01	Ditolak
10	Ditolak
1110	Ditolak
11	Diterima
101	Diterima
1100101	Diterima
0100011101	Ditolak

NON DETERMINISTIC FINITE STATE AUTOMATA

Kemungkinan transisinya ke lebih dari satu state. Dari suatu state bisa terdapat 0, 1 atau lebih transisi dengan label input yang sama. Perubahan state dapat terjadi secara spontan tanpa input (transisi kosong).

Kemungkinan transisinya ke lebih dari satu state. Dari suatu state bisa terdapat 0, 1 atau lebih transisi dengan label input yang sama. Perubahan state dapat terjadi secara spontan tanpa input (transisi kosong).

Salah satu tracing-nya berakhir di state akhir, atau himpunan state setelah membaca string tersebut mengandung state akhir

FSA dinyatakan oleh 5 tuple :

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

FSA dinyatakan oleh 5 tuple :

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

FSA dinyatakan oleh 5 tuple : $M = (Q, \Sigma, \delta, S, F)$ dimana:

Q = Himpunan state / kedudukan

Σ = Himpunan symbol input

δ = Fungsi Transisi

q_0 = State awal q_0 ,

F = himpunan state akhir

* Catatan : Jumlah state akhir bisa lebih dari satu

- **NFA Dengan E-MOVE**

Pada NFA dengan e-move dibolehkan berpindah state (tanpa membaca input)

- **E-closure untuk NFA dengan e-move**

* E-closure adalah himpunan state-state yang dapat dicapai dari suatu state tanpa membaca input

* E-closure (q_0)

berarti bahwa himpunan yang dapat dicapai dari q_0 tanpa membaca input

Contoh 1 :

FSA dinyatakan oleh 5 tuple : $M = (Q, \Sigma, \delta, S, F)$ dimana:

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

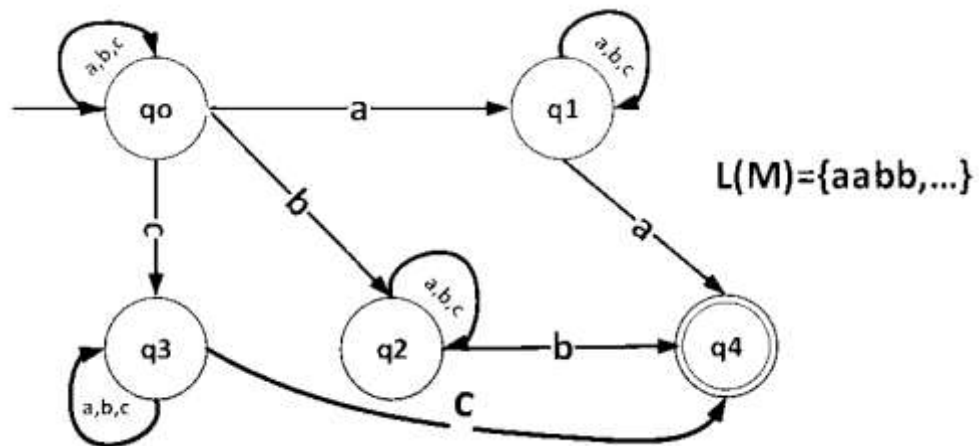
$$\Sigma = \{a, b, c\}$$

$S = q_0$

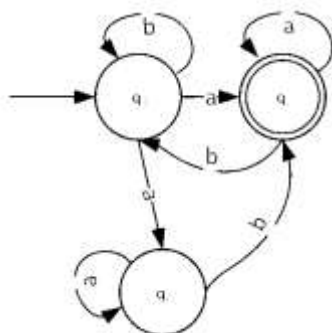
$F = \{q_4\}$

Tabel Transisi :

δ	a	b	c
q0	{q0,q1}	{q0,q2}	{q0,q3}
q1	{q1,q4}	{q1}	{q1}
q2	{q2}	{q2,q4}	{q2}
q3	{q3}	{q3}	{q3,q4}
q4	\emptyset	\emptyset	\emptyset



Contoh 2 :



Gambar tersebut termasuk NFA karena jika q_0 menerima inputan 'a' maka akan berpindah ke q_1 atau q_2 . Tidak tepat satu berikutnya

δ	a	b
q_0	$\{q_1, q_2\}$	q_0
q_1	q_1	q_0
q_2	q_2	q_1

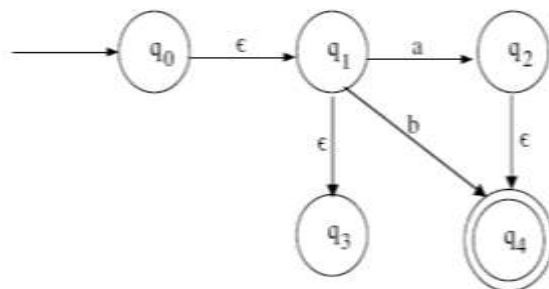
Pada NFA, String diterima jika setidaknya ada 1 dari semua kemungkinan transisi berakhir pada sebuah final state. Harus mencoba semua kemungkinan.

Pada NFA boleh terdapat transisi kosong. Simbol ϵ berarti tanpa masukan atau disebut transisi kosong. Terjadi perubahan state secara spontan

Pada NFA, String diterima jika setidaknya ada 1 dari semua kemungkinan transisi berakhir pada sebuah final state. Harus mencoba semua kemungkinan. Pada NFA boleh terdapat transisi kosong. Simbol ϵ berarti tanpa masukan atau disebut transisi kosong. Terjadi perubahan state secara spontan

- **Ekuivalensi NFA dengan e-move**

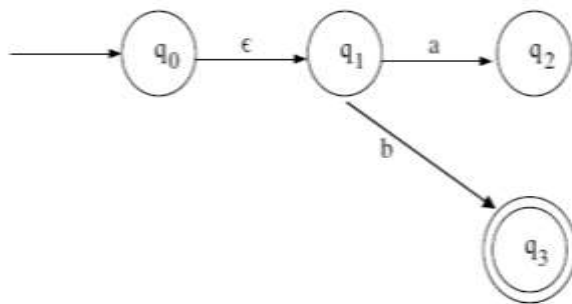
Kita dapat membuat suatu NFA tanpa e-move dari NFA dengan e-move yang ekuivalen



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

- ϵ – Closure (q_0) = { q_0, q_1, q_3 }
 - ϵ – Closure (q_1) = { q_1, q_3 }
 - ϵ – Closure (q_2) = { q_2, q_4 }
 - ϵ – Closure (q_3) = { q_3 }
 - ϵ – Closure (q_4) = { q_4 }
- **Langkah Ekuivalensi NFA dengan e-move ke NFA tanpa e-move**

* Buatlah tabel transisi dari NFA dengan e-move



Tabel Transisi

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

* Tentukan ϵ -closure untuk setiap *state*

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 }

* Carilah setiap fungsi transisi hasil dari pengubahan NFA ϵ – move ke NFA tanpa ϵ – move. Fungsi transisi itu ditandai dengan simbol δ'

$$\begin{aligned}\delta^*(q_0, a) &= \epsilon_cl(\delta(\epsilon_cl(q_0), a)) \\ &= \epsilon_cl(q_2) \\ &= \{q_2\}\end{aligned}$$

$$\begin{aligned}\delta^*(q_0, b) &= \epsilon_cl(\delta(\epsilon_cl(q_0), b)) \\ &= \epsilon_cl(q_3) \\ &= \{q_3\}\end{aligned}$$

$$\begin{aligned}\delta^*(q_1, a) &= \epsilon_cl(\delta(\epsilon_cl(q_1), a)) \\ &= \epsilon_cl(q_2) \\ &= \{q_2\}\end{aligned}$$

$$\begin{aligned}\delta^*(q_1, b) &= \epsilon_cl(\delta(\epsilon_cl(q_1), b)) \\ &= \epsilon_cl(q_2) \\ &= \{q_3\}\end{aligned}$$

$$\begin{aligned}\delta^*(q_2, a) &= \epsilon_cl(\delta(\epsilon_cl(q_2), a)) \\ &= \epsilon_cl(\emptyset) \\ &= \emptyset\end{aligned}$$

$$\begin{aligned}\delta^*(q_2, b) &= \epsilon_cl(\delta(\epsilon_cl(q_2), b)) \\ &= \epsilon_cl(\emptyset) \\ &= \emptyset\end{aligned}$$

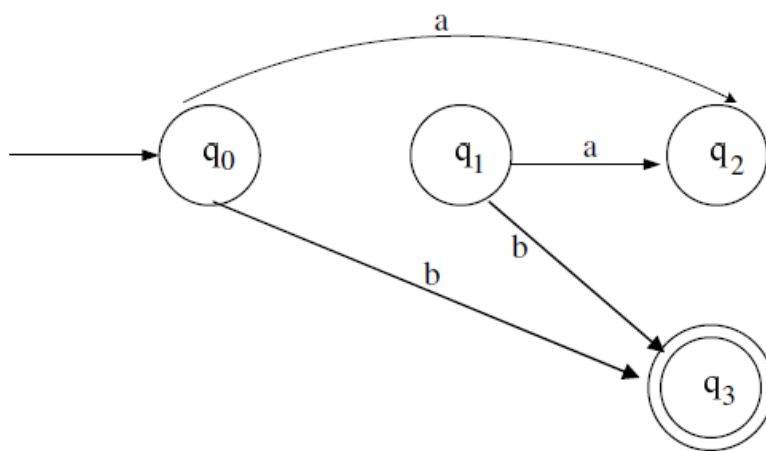
$$\begin{aligned}\delta^*(q_3, a) &= \epsilon_cl(\delta(\epsilon_cl(q_3), a)) \\ &= \epsilon_cl(\emptyset) \\ &= \emptyset\end{aligned}$$

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

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

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

* Kemudian, tentukanlah himpunan state akhir untuk NFA tanpa ϵ – move ini.
Himpunan state akhir semula adalah $\{q_3\}$. Karena tidak ada state lain yang ϵ –
closurenya memuat q_3 , maka himpunan state akhir sekarang tetap $\{q_3\}$. Sehingga
diperoleh diagram transisi sebagai berikut.



FINITE STATE TRANSDUCER (FST)

1. JENIS FINITE STATE TRANSDUCER (FST)

- Moore Machine
- Mealy Machine

2. Karakteristik FST

- FST terdiri dari dari sejumlah state tanpa final state
- (Mealy) -> State tersebut dihubungkan oleh transisi yang diberi label pasangan input/output
- (Moore) -> Output berasosiasi dengan state
- FST dimulai dengan initial state yang ditentukan dan melompat ke state yang berbeda, tergantung pada nilai input, sambil menghasilkan output sesuai dengan tabel transisinya

- **Moore Machine**

Pada mesin moore, output akan berasosiasi state.

Didefinisikan dengan 6 tuple

- Q =Himpunan state
- Σ =Himpunan symbol input
- δ =Fungsi transisi
- S =State awal S , dimana $S \in Q$
- Δ =Himpunan output
- ω =fungsi output untuk setiap state

Contoh 1

Kita akan mencari nilai sisa pembagian (modulus) suatu bilangan dengan 3.

Dimana input dinyatakan dalam biner.

Konfigurasi :

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

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

$S = q_0$

$\Delta = \{0, 1, 2\}$ (untuk outputnya pada kasus mod dengan 3 maka sisanya

kemungkinan

adalah

0,1,2)

$\lambda(q_0) =$

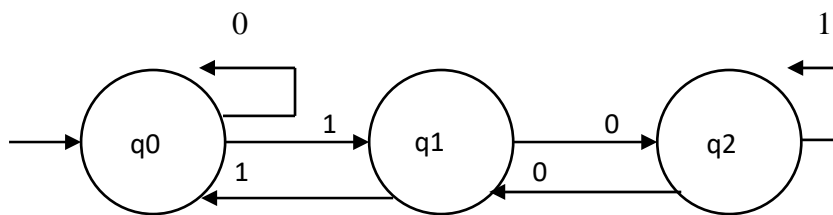
0 $\lambda(q_1)$

= 1

$\lambda(q_2) =$

2

Gambar Mesin Moore untuk modulus 3 :



Pembuktian :

- $5 \bmod 3 = ?$

input 5 dalam biner 101

bila kita masukkan 101 kedalam mesin, urutan state yang dicapai adalah :

q_0, q_1, q_2, q_2

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 1010 kedalam mesin,

urutan state yang dicapai adalah : q_0, q_1, q_2, q_2, q_1

State terakhir yang dicapai adalah q_1 , $\lambda(q_1) = 1$ Maka 10

$\bmod 3 = 1$

- $12 \bmod 3 = ?$

input 12 dalam biner 1100 bila kita masukkan 1100 kedalam mesin,

urutan state yang dicapai adalah : q_0, q_1, q_0, q_0

State terakhir yang dicapai adalah q_0 , $\lambda(q_0) = 0$ Maka 12

$\bmod 3 = 0$

CONTEXT FREE GRAMMAR

Tata Bahasa Bebas Konteks atau CFG

•CFG atau Context Free Grammar adalah tata bahasa formal dimana setiap aturan produksi adalah dalam bentuk $a \rightarrow B$, dimana a adalah pemproduksi, dan B adalah hasil produkai

*Batasan Aturan Produksi

- Ruas kiri adalah sebuah simbol variabel
- Ruas kanan bisa berupa terminal, variable aturan E
- $a \in N$ (Non-terminal)
- $B \in (T \cup N)$ (Terminal atau Non-terminal)
- B boleh berisi E

*Contoh aturan Produksi CFG

- $X \rightarrow bY|Za$
- $Y \rightarrow aY|B$
- $Z \rightarrow bZ|E$
- $B \rightarrow CDefg$
- $D \rightarrow BcDe$

CFG VS RE

•Keduanya merupakan suatu cara untuk menunjukkan bagaimana menghasilkan suatu untai-untai dalam sebuah bahasa

•Pada saat menurunkan suatu string, symbol-symbol variable akan mewakili bagian-bagian yang belum diturunkan dari string tersebut

•Pada ER, bagian yang belum diturunlan selalu berada diujung

•Pada CFG, bagian yang belum diturunlan bisa berada dimana saja

Contoh aturan produksi ER

$S \rightarrow aE$

$E \rightarrow A|B$

$A \rightarrow aA|b$

$B \rightarrow aE|b$

Contoh aturan produksi CFG

$X \rightarrow bY | Za$

$Y \rightarrow aY | B$

$Z \rightarrow bZ | E$

$B \rightarrow CDefg$

*Parsing

Pohon atau tree adalah sebuah graph terhubung tidak sirkuler yang memiliki satu simpul(node)/ vertex yang disebut akar (root) dan dari situ kita memiliki lintasan setiap simpul

•Derivation Tree (Pohon Penurunan)

Derivation tree berguna untuk menggambarkan bagaimana memperoleh suatu string dengan cara menurunkan symbol-symbol variable menjadi symbol-symbol terminal. Hingga tidak ada lagi symbol variable yang bisa diturunkan

•Contoh:pohon penurunan"aabbb"

Aturan produksi

$S \rightarrow AB$

$A \rightarrow aA | a$

$B \rightarrow bB | b$

Hasil penurunan

$S \rightarrow AB \rightarrow aAV \rightarrow aaB \rightarrow aabB \rightarrow aabb$

•Proses penuruan / parsing

1.Lefmost Derivation

Simbol variabel yang paling kiri diturunkan terlebih dahulu

2.Reightmos

Simbol variable yang diturunkan terlebih dahulu

•Contoh proses penurunn / parsing

Lefmost Derivation

$S \rightarrow aAS \rightarrow asbAS \rightarrow aabAS \rightarrow aabbaS \rightarrow aabbba$

Reightmost Derivation

$S \rightarrow aAS \rightarrow aAa \rightarrow aSbAa \rightarrow aSbbbaa \rightarrow aabbbaa$

Ambigiutas

Ambiguitas adalah kedwianartian atau makna ganda

Ambiguitas terjadi jika terdapat lebih dari satu pohon penurunan yang berbeda untuk memperoleh suatu untai

Contoh ambiguitas

Aturan produksi

$$S \rightarrow A|B$$
$$A \rightarrow a$$
$$B \rightarrow a$$

String yang dicari "a"

Cara 1

$$S \rightarrow A \rightarrow a$$

Cara 2

$$S \rightarrow B \rightarrow a$$

Aturan produksi

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

String yang dicari "abaca"

Cara

$$S \rightarrow SbS \rightarrow SbScS \rightarrow SbSca \rightarrow Sbaca \rightarrow abaca$$

Kesimpulan

Ambiguitas dapat menimbulkan masalah pada bahasa-bahasa tertentu, baik pada bahasa alami maupun pada bahasa pemrograman

Bila suatu struktur bahasa memiliki lebih dari satu dekomposisi, dan susunannya akan menentukan arti, maka artinya menjadi ambigu

PENGANTAR ATURAN PRODUKSI REKURSIF KIRI

Aturan Produksi yang rekursif memiliki ruas kanan (hasil produksi) yang memuat simbol variabel pada ruas kiri. Terdapat rekursif kanan dan rekursif kiri.

* Sebuah aturan produksi dalam bentuk:

$$A \rightarrow \beta A$$

* Merupakan aturan produksi yang rekursif kanan:

$$\beta(V \cup T)^*$$

atau kumpulan symbol variabel dan terminal

* Contoh aturan produksi yang rekursif kanan:

$$S \rightarrow dS$$

$$B \rightarrow adB$$

Aturan Produksi Rekursif Kiri

* Sebuah aturan 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 ke kiri menyebabkan pohon penurunan tumbuh ke kiri. Bisa dilihat pada pohon penurunan dari tata bahasa bebas konteks dengan aturan produksi:

$$S \rightarrow aAc$$

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

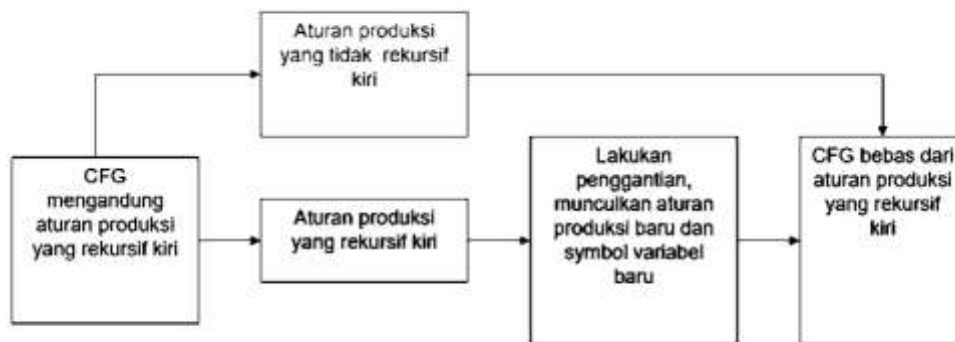


PENGHILANGAN REKURSIF KIRI

1. Tujuan Rekursif Kiri Dihapus

- Dalam banyak penerapan tata bahasa rekursif kiri tak diinginkan. Untuk menghindari penurunan yang bisa mengakibatkan loop
- hilangkan sifat rekursif kiri dari aturan produksi
- Penghilangan rekursif kiri memungkinkan suatu tata bahasa bebas konteks diubah ke dalam bentuk normal Greibach

2. Tahapan Penghilangan Rekursif Kiri



1) Pisahkan aturan produksi yang rekursif kiri dan yang tidak

- Aturan produksi yang rekursif kiri :
$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \mid \dots \mid A\alpha_n$$
- Aturan produksi yang tidak rekursif kiri :
$$A \rightarrow \beta_1 \mid \beta_2 \mid \beta_3 \mid \dots \mid \beta_m$$

2) Tentukan simbol-simbol $\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n$ dan $\beta_1, \beta_2, \beta_3, \dots, \beta_m$ dari setiap aturan produksi yang memiliki simbol ruas kiri yang sama

3) Lakukan penggantian aturan produksi yang rekursif kiri menjadi sebagai berikut:

$$A \rightarrow \beta_1 Z \mid \beta_2 Z \mid \beta_3 Z \mid \dots \mid \beta_m Z \quad Z \rightarrow \alpha_1 \mid \alpha_2 \mid \alpha_3 \mid \dots, \alpha_n \quad Z \rightarrow \beta_1 \mid \beta_2 \mid \beta_3 \mid \dots \mid \beta_m$$

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

4) Hasil akhir berupa aturan produksi pengganti ditambah dengan aturan produksi semula yang tidak rekursif kiri

CONTOH KASUS PENGHILANGAN REKURSIF KIRI

- Contoh 12.1

Hilangkan rekursif kiri dari aturan produksi berikut!

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

Penyelesaian:

Aturan produksi rekursif kiri : $S \rightarrow Sab \mid Sbd$ $\alpha_1 = ab$; $\alpha_2 = bd$

Aturan produksi tidak rekursif kiri : $S \rightarrow aSc \mid dd \mid ff$ $\beta_1 = aSc$; $\beta_2 = dd$; $\beta_3 = ff$

Aturan produksi rekursif kiri, $S \rightarrow Sab \mid Sbd$ digantikan oleh:

$$S \rightarrow aScZ_1 \mid dd Z_1 \mid ff Z_1 \quad Z_1 \rightarrow ab \mid bd \quad Z_1 \rightarrow abZ_1 \mid bdZ_1$$

Hasil akhir setelah penghilangan rekursif kiri adalah : $S \rightarrow aSc \mid dd \mid ff$

- Contoh 12.2

Hilangkan rekursif kiri dari aturan produksi berikut!

$$S \rightarrow Sab \mid Sb \mid cA \quad A \rightarrow Aa \mid a \mid bd$$

Penyelesaian:

- Aturan produksi rekursif kiri, $S \rightarrow Sab \mid Sb$ $\alpha_1 = ab$; $\alpha_2 = b$

- Aturan produksi rekursif kiri, $A \rightarrow Aa$ $\alpha_1 = a$

- Aturan produksi tidak rekursif kiri, $S \rightarrow cA$ $\beta_1 = cA$

- Aturan produksi tidak rekursif kiri, $A \rightarrow a \mid bd$ $\beta_1 = a$; $\beta_2 = bd$

Pergantian aturan produksi rekursif kiri:

$S \rightarrow Sab \mid Sb$ digantikan oleh:

➤ $S \rightarrow caZ_1$

➤ $Z_1 \rightarrow ab \mid b$

➤ $Z_1 \rightarrow abZ_1 \mid bZ_1$

$A \rightarrow Aa$ digantikan oleh :

➤ $A \rightarrow aZ_2 \mid bdZ_2$

➤ $Z_2 \rightarrow a$

➤ $Z_2 \rightarrow aZ_2$

Hasil akhir setelah penghilangan rekursif kiri adalah:

$$S \rightarrow cA$$

$$A \rightarrow a \mid bd$$

$$S \rightarrow caZ1$$

$$Z1 \rightarrow ab \mid b$$

$$Z1 \rightarrow abZ1 \mid bZ1$$

$$A \rightarrow aZ2 \mid bdZ2$$

$$Z2 \rightarrow a$$

$$Z2 \rightarrow aZ2$$

PENYEDERHANAAN ATURAN PRODUKSI CONTEXT FREE GRAMMAR

1. Tujuan Penyederhanaan

Melakukan pembatasan, sehingga tidak dihasilkan pohon penurunan yang memiliki kerumitan yang tak perlu atau aturan produksi yang tidak berarti

Misal

Diketahui suatu tata bahasa konteks:

$S \rightarrow AB \mid a$

$A \rightarrow a$

Kelemahan : Aturan produksi $S \rightarrow AB$ tidak berarti karena B tidak memiliki penurunan

2. Teknik Penyederhanaan

a. Penghilangan Produksi ϵ

Produksi ϵ (Empty) adalah produksi dalam bentuk $\alpha \rightarrow \epsilon$ atau bisa dianggap sebagai produksi kosong. Penghilangan produksi ϵ dilakukan dengan melakukan penggantian produksi yang memuat variable yang menuju produksi ϵ , atau biasa disebut nullable.

Contoh 1 :

Diketahui tata bahasa bebas konteks sebagai berikut :

$S \rightarrow dA \mid Bd$

$A \rightarrow bc$

$A \rightarrow \epsilon$

$B \rightarrow c$

Sehingga aturan produksi setelah penyederhanaan : $S \rightarrow dA \mid d \mid Bd$

$A \rightarrow bc$

$B \rightarrow c$

Contoh 2 :

$S \rightarrow bcAd$

$A \rightarrow \epsilon$

Pada kasus 1, A nullable serta $A \rightarrow \epsilon$ merupakan satu-satunya produksi dari A maka variable A bias ditiadakan.

Maka hasil penyederhanaan adalah : $S \rightarrow bcd$

Kasus 2 $S \rightarrow bcAd$

$A \rightarrow bd \mid \varepsilon$

Pada kasus 2, A nullable, tapi $A \rightarrow \varepsilon$ bukan satu-satunya produksi dari A. Maka hasil penyederhanaan adalah : $S \rightarrow bcAd \mid bcd A \rightarrow bd$

b. Penghilangan Produksi Unit

Produksi unit adalah produksi yang ruas kiri dan kanan aturan produksinya hanya berupa satu simbol variable. ($\alpha = 1$ N/V dan $\beta = 1$ N/V)

Dengan adanya bentuk produksi unit ini membuat tata bahasa memiliki kerumitan yang tidak perlu atau menambah panjang penurunan.

Penyederhanaan ini dilakukan dengan melakukan penggantian aturan produksi unit.

Contoh 1 :

Diketahui tata bahasa bebas konteks sebagai berikut :

$S \rightarrow Sb$

$S \rightarrow C$

$C \rightarrow D$

$C \rightarrow ef$

$D \rightarrow dd$

Sehingga aturan produksi setelah penyederhanaan :

$S \rightarrow Sb \mid dd \mid ef$

$C \rightarrow dd \mid ef$

$D \rightarrow dd$

Contoh 2 :

Diketahui tata bahasa bebas konteks sebagai berikut :

$S \rightarrow A$

$S \rightarrow Aa$

$A \rightarrow B$

$B \rightarrow C$

$B \rightarrow b$

$C \rightarrow D$

$C \rightarrow ab$

$D \rightarrow b$

Sehingga aturan produksi setelah penyederhanaan :

$S \rightarrow A \Rightarrow S \rightarrow ab \mid b$
 $S \rightarrow Aa$
 $A \rightarrow B \Rightarrow A \rightarrow ab \mid b \quad B \rightarrow ab \quad B \rightarrow b$
 $C \rightarrow b$
 $C \rightarrow ab$
 $D \rightarrow b$

c. Penghilang Produksi Useless

Produksi yang memuat simbol variable yang tidak memiliki penurunan yang akan menghasilkan terminal-terminal seluruhnya (menuju terminal), produksi ini tidak berguna karena bila diturunkan tidak akan pernah selesai (masih ada simbol variable tersisa).

Produksi yang tidak akan pernah dicapai dengan penurunan apapun dari simbol awal, sehingga produksi itu redundan (berlebih).

Contoh 1 :

Diketahui tata bahasa bebas konteks sebagai berikut

$S \rightarrow aSa \mid Abd \mid Bde$

$A \rightarrow Ada \quad B \rightarrow BBB \mid a$

Maka tata bahasa hasil penyederhanaan adalah :

$S \rightarrow aSa \mid Bde$

$B \rightarrow BBB \mid a$

Contoh 2 :

Diketahui tata bahasa bebas konteks :

$S \rightarrow Aa \mid B$

$A \rightarrow ab \mid D$

$B \rightarrow b \mid E$

$C \rightarrow bb$

$E \rightarrow aEa$

Maka tata bahasa hasil penyederhanaan menjadi :

$S \rightarrow Aa \mid B$

$A \rightarrow ab$

$B \rightarrow b$

3. Gabungan Useless, Unit & ϵ

Urutannya sebagai berikut :

- a. Hilangkan produksi ε
- b. Hilangkan produksi unit
- c. Hilangkan produksi useless

Contoh 1 :

Hilangkan produksi useless, unit dan empty dari tata bahasa bebas konteks berikut :

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

$A \rightarrow aB \mid \varepsilon$

$B \rightarrow Aa$

$C \rightarrow cCD$

$D \rightarrow ddd$

- a. Hilangkan produksi ε

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

$A \rightarrow aB \mid \varepsilon$

$B \rightarrow Aa$

$C \rightarrow cCD$

$D \rightarrow ddd$

Hasilnya

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

$A \rightarrow aB$

$B \rightarrow Aa \mid a$

$C \rightarrow cCD$

$D \rightarrow ddd$

- b. Hilangkan produksi unit

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

$A \rightarrow aB$

$B \rightarrow Aa \mid a$

$C \rightarrow cCD$

$D \rightarrow ddd$

Hasilnya

$S \rightarrow a \mid aA \mid Aa \mid cCD$

$A \rightarrow aB$

$B \rightarrow Aa \mid a$

$C \rightarrow cCD$

$D \rightarrow ddd$

c. Hilangkan produksi useless

$S \rightarrow a \mid aA \mid Aa \mid \epsilon$ ~~CD~~

$A \rightarrow aB$

$B \rightarrow Aa \mid a$

~~$C \rightarrow \epsilon CD$~~

~~$D \rightarrow ddd$~~

Hasilnya

$S \rightarrow a \mid aA \mid Aa$

$A \rightarrow aB$

$B \rightarrow Aa \mid a$

CHOMSKY NORMAL FORM (CNF)

Merupakan salah satu bentuk normal yang sangat berguna untuk tata Bahasa bebas konteks (CFG)

Chomsky Normal Form (CNF) dapat dibentuk dari CFG yang telah mengalami penyederhanaan yaitu penghilangan produksi useless, unit dan kosong.

1. Bentuk Normal Chomsky

Syarat Konversi CFD ke CNF

Tidak memiliki produksi useless

Tidak memiliki Produksi unit

Tidak memiliki Produksi ϵ (Kosong)

Aturan Produksi CNF

Ruas kanan hanya boleh berupa sebuah simbol terminal atau dua buah simbol variable.

Jika terdapat lebih dari satu simbol terminal maka harus dilakukan penggantian dan juga jika terdapat lebih dari dua buah simbol variable maka harus dilakukan perubahan.

$\alpha \rightarrow \beta$

$\alpha = 1$ Non-Terminal

$\beta = 1$ Terminal atau 2 Non-Terminal

2. Pembentukan Bentuk Normal Chomsky

Langkah Secara Umum

- Biarakan aturan produksi yang telah dalam bentuk CNF
- Lakukan penggantian aturan produksi yang ruas kanannya memuat symbol terminal dan panjang ruas kanan >1
- Lakukan penggantian aturan produksi yang ruas kanannya memuat >2 symbol variable
- 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 symbol-simbol variable baru.

Tahapan Pembentukan CNF



Contoh 1 :

Perhatikan aturan produksi CFG berikut, diasumsikan telah disederhanakan :

$$S \rightarrow bA \mid aB$$

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

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

Ubahlah ke dalam bentuk normal Chomsky !!!

Jawab :

Aturan produksi yang sudah dalam bentuk normal Chomsky:

$$A \rightarrow aB \rightarrow bb.$$

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

$$S \rightarrow bA \Rightarrow S \rightarrow P1A$$

$$S \rightarrow aB \Rightarrow S \rightarrow P2B$$

$$A \rightarrow bAA \Rightarrow A \rightarrow P1AA \Rightarrow A \rightarrow P1P3$$

$$A \rightarrow aS \Rightarrow A \rightarrow P2S$$

$$B \rightarrow aBB \Rightarrow B \rightarrow P2BB \Rightarrow B \rightarrow P2P4$$

$$B \rightarrow bS \Rightarrow B \rightarrow P1Sc.$$

Terbentuk aturan produksi dan simbol variabel baru:

$$P1 \rightarrow b$$

$$P2 \rightarrow a$$

$$P3 \rightarrow AA$$

$$P4 \rightarrow BBd.$$

Hasil akhir aturan produksi dalam bentuk normal Chomsky :

$$A \rightarrow a$$

$$B \rightarrow b$$

$$S \rightarrow P1A$$

$$S \rightarrow P2B$$

$$A \rightarrow P1P3$$

$$A \rightarrow P2S$$

$$B \rightarrow P2P4$$

$$B \rightarrow P1S$$

$$P1 \rightarrow b$$

$$P2 \rightarrow a$$

$$P3 \rightarrow AA$$

$$P4 \rightarrow BB$$

Contoh 2 :

Berikut adalah tata bahasa bebas konteks yang sudah disederhanakan.

$$S \rightarrow aB \mid CA$$

$$A \rightarrow a \mid bc \quad B \rightarrow BC \mid Ab$$

$C \rightarrow aB \mid b$ Dari aturan produksi diatas, aturan produksi yang sudah dalam bentuk normal Chomsky (CNF) adalah: $S \rightarrow CA$

$$A \rightarrow a$$

$$B \rightarrow BC$$

$$C \rightarrow b$$

Aturan produksi yang belum dalam CNF adalah:

$$S \rightarrow aB \Rightarrow S \rightarrow P1B$$

$$A \rightarrow bc \Rightarrow A \rightarrow P2P3$$

$$B \rightarrow Ab \Rightarrow B \rightarrow AP2$$

$$C \rightarrow aB \Rightarrow C \rightarrow P1B$$

Terbentuk aturan produksi baru:

$$P1 \rightarrow a$$

$$P2 \rightarrow b$$

$$P3 \rightarrow c$$

Hasil akhir aturan produksi dalam CNF adalah:

$S \rightarrow CA$
 $A \rightarrow a$
 $B \rightarrow BC$
 $C \rightarrow b$
 $S \rightarrow P1B$
 $A \rightarrow P2P3$
 $B \rightarrow AP2$
 $C \rightarrow P1B$
 $P1 \rightarrow a$
 $P2 \rightarrow b$
 $P3 \rightarrow c$

3. Algoritma CYK untuk CFG

Algoritma the Cocke-Younger-Kasami algorithm (CYK) merupakan algoritma parsing dan keanggotaan (Membership) untuk tata Bahasa Bebas Konteks. Algoritma ini diciptakan oleh J.Cocke.DH.Younger. dan T.Kasami. Syarat untuk menggunakan algoritma ini adalah tata bahasa harus sudah dalam bentuk Normal Chomsky

Simbol Algoritma CYK

n = panjang string yang akan diperiksa, misalnya 'ada', $n = \text{length} = 3$

I menyatakan kolom ke-

J menyatakan baris ke-

Tahapan no (2) dan (3) untuk mengisi tabel baris pertama kolom 1- n

No (4), iterasi dari baris ke-2 sampai no (4)

No (5), iterasi untuk mengisi kolom 1 sampai (n -baris+1) pada suatu baris

No(7), inisialisasi V_{ij} dengan \emptyset

No (8) dan (9), iterasi untuk memeriksa mana saja yang menjadi anggota V_{ij}

Contoh 1 :

Diketahui tata bahasa bebas konteks:

$S \rightarrow AB \mid BC$
 $A \rightarrow BA \mid a$
 $B \rightarrow CC \mid b$
 $C \rightarrow AB \mid a$

Periksa, apakah untai 'baaba' termasuk ke dalam bahasa tersebut.

Penyelesaian:

Syarat agar sebuah untai termasuk ke dalam tata bahasa tertentu adalah $V1n$ harus memuat simbol awal

$i \rightarrow (1 - 5)$

$j \rightarrow 1 - 5$

Hasil pada 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	Ø	B	B		
4	Ø	S, A, C			
5	S, A, C				