RESUME

“TEORI BAHASA DAN AUTOMATA”

“Disusun sebagai tugas akhir pada mata kuliah Perancangan dan Analisis Algoritma , dengan Dosen IIP S.Pd., M.Pd.T”

Disusun Oleh :

Putri Maharani || 21346018

PROGRAM STUDI S1 TEKNIK INFORMATIKA

JURUSAN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

TAHUN 2023

KATA PENGANTAR

Puji syukur kita hanturkan kehadiran Tuhan Yang Maha Esa, yang telah melimpahkan rahmat dan karunianya kepada kita, sehingga kita dapat menyusun dan menyajikan makalah Perancangan dan Analisis Algoritma ini dengan tepat waktu. Tak lupa pula kita mengucapkan terima kasih kepada berbagai pihak yang telah memberikan dorongan dan motivasi. Sehingga makalah ini dapat tersusun dengan baik.

Teori bahasa dan automata membahas konsep-konsep yang mendasari pengembangan bahasa pemrograman, kompilasi, pengenalan pola, kecerdasan buatan, dan bidang lainnya yang berkaitan dengan komputasi. Dalam kata pengantar ini, saya akan memberikan gambaran umum tentang topik-topik yang akan Anda temui dalam materi ini.

Pertama, kita akan mempelajari dasar-dasar bahasa formal dan bahasa alami. Bahasa formal adalah bahasa yang digunakan untuk mendefinisikan aturan-aturan dan struktur dalam komputasi. Kita akan menjelajahi tipe-tipe bahasa formal, termasuk bahasa regulern, bahasa konteks-bebas, dan bahasa bergantung konteks. Selain itu, kita akan membahas struktur bahasa alami dan bagaimana bahasa-bahasa ini dapat dianalisis dan dipahami oleh mesin.

Selanjutnya, kita akan mempelajari automata, yang merupakan model matematis dari mesin komputasi. Automata dapat digunakan untuk memahami dan menganalisis bahasa-bahasa formal. Kita akan belajar tentang automata-finite state automata, pushdown automata, dan mesin Turing. Kami akan mempelajari kemampuan dan batasan dari setiap jenis automata ini dalam mengenali dan memproses bahasa formal.

Selain itu, kita akan mempelajari topik-topik lain yang berkaitan dengan teori bahasa dan automata, seperti bahasa Chomsky, teori komputabilitas, dan kompleksitas algoritma. Kami juga akan melihat aplikasi praktis dari konsep-konsep ini dalam komputasi sehari-hari dan pengembangan perangkat lunak.

Materi ini dirancang untuk memberikan pemahaman yang kokoh tentang teori bahasa dan automata kepada pembaca, baik yang baru memasuki dunia komputasi maupun yang ingin memperdalam pemahaman mereka. Saya berharap bahwa materi ini akan memberikan pemahaman yang jelas dan memuaskan tentang topik yang menarik ini.

Sago, 01 juni 2023

Putri Maharani

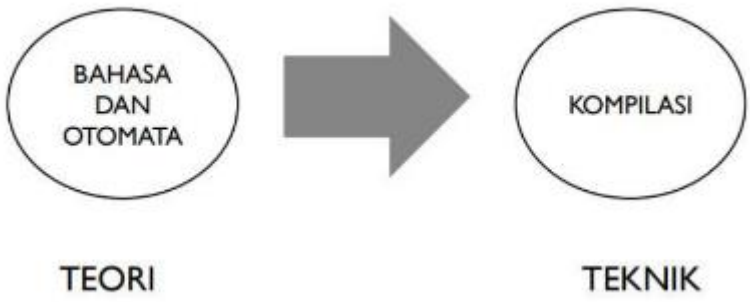
DAFTAR ISI

Praktikum Artificial Intelligence

PENGANTAR TEORI BAHASA & OTOMATA

A. Tingkat bahasa pemograman.

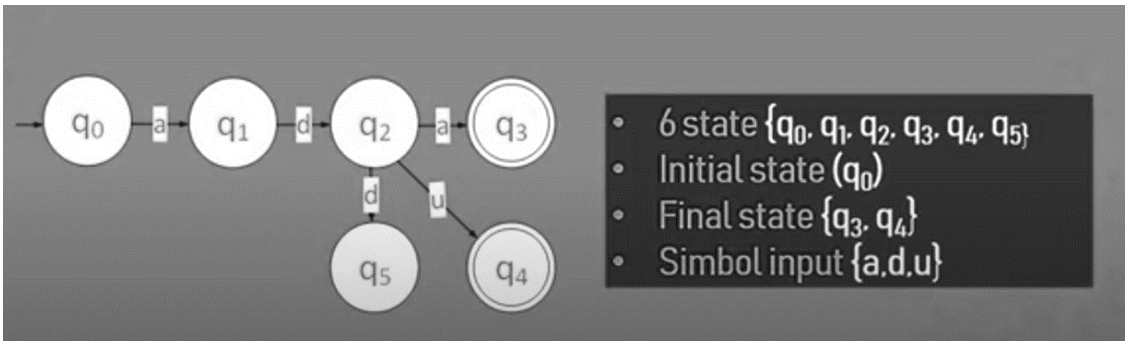
Proses Kompilasi



Teori bahasa dan automata adalah dasar dari teknologi penerjemahan
Mesin yang hanya mengenal bahasa mesin dapat memahami bahasa pemrograman tingkat lanjut karena ada juru bahasa/kompiler.
Teori bahasa berurusan dengan bahasa formal. Salah satunya menguntungkan penerjemah
Kamus bahasa adalah sistem yang mencakup pengungkapan ide, fakta, dan konsep, termasuk seperangkat simbol dan aturan untuk memanipulasinya. Bahasa juga dapat digambarkan sebagai seperangkat simbol yang memiliki makna.
Otomat adalah sistem yang terdiri dari sejumlah negara bagian yang terbatas, di mana negara bagian menentukan data input. Automata juga dianggap sebagai mesin otomatis (bukan mesin fisik), yang merupakan model matematis dari sistem yang menerima input dan menghasilkan output dan terdiri dari sejumlah keadaan yang

terbatas. Input yang dimasukkan pada mesin dianggap sebagai bahasa yang harus dikenali oleh mesin. Mesin menunjukkan apakah ucapan tersebut dapat diterima atau tidak.
Hubungan antara ucapan dan otomat adalah bahwa otomat mengambil ucapan sebagai masukan, setelah itu otomat membuat keputusan yang menunjukkan apakah akan menerima masukan atau tidak.

Contoh mesin otomata sederhana

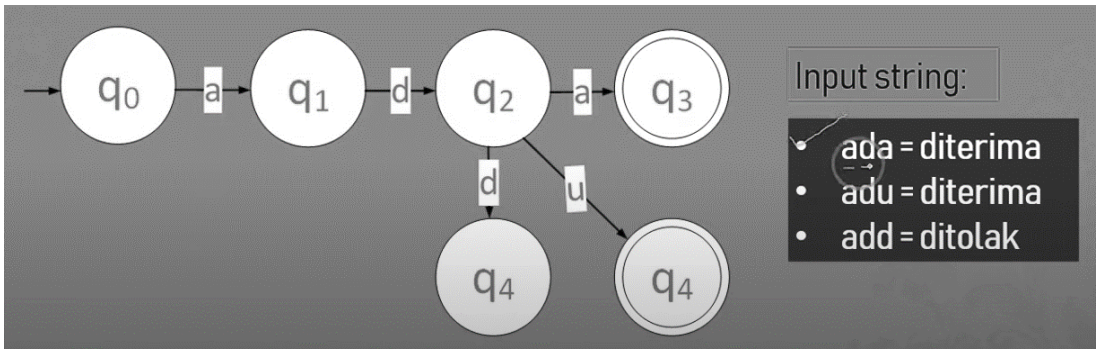


String input diterima jika mencapai final state, selain itu ditolak

Pembacaan simbol pertama dimulai dari initial state

Perpindahan state berdasarkan simbol yang dibaca

Maka hasilnya akan menjadi



SIMBOL STRING DAN BAHASA

- Simbol adalah entitas abstrak (seperti konsep titik dalam geometri). Contoh symbol adalah huruf atau angka.
- String adalah kumpulan karakter yang terbatas. Misalnya, jika a, b, dan c adalah tiga simbol, maka abcb adalah string yang tersusun dari ketiga simbol tersebut.
  Jika w adalah string, panjang string didefinisikan sebagai $|w|$ dan didefinisikan sebagai jumlah simbol yang membentuk string. Misalnya, jika w = abcb maka $|w| = 4$.
- String kosong adalah string dengan simbol null. String kosong dilambangkan dengan simbol ε (atau ^), jadi $|\varepsilon| = 0$. String kosong dapat dianggap sebagai simbol kosong karena keduanya terdiri dari simbol nol.
- Alfabet adalah hinpunan hingga (finite set) simbol-simbol.

  Suatu Entitas Abstark yang tidak didefinisikan secara formal

Bahasa adalah Suatu sistem yang meiliputi pengekspresian gagasan, fakta konsep,termasuk sekumpulan simbol-simbol dan aturan untuk melakukan manipulasinya.Alphabet adalah himpunan berhingga dari simbol-simbol.

Bahasa juga disebut sebagai seperangkat simbol yang memiliki makna.
- Bahasa adalah sekumpulan string abjad.
- Karena bahasa adalah kumpulan string, kita dapat memiliki bahasa yang tidak memiliki string, yaitu bahasa kosong, dinotasikan seperti saat Anda menulis notasi himpunan kosong.
- Input mesin otomatis dianggap sebagai bahasa yang harus dikenali oleh mesin.
- Perangkat akan menunjukkan apakah bahasa tersebut dapat diterima atau tidak

ATURAN PRODUKSI DAN GRAMER

Aturan produksi dan tata bahasa (grammar) merupakan konsep penting dalam linguistik dan pengolahan bahasa alami. Aturan produksi menggambarkan bagaimana unit-unit bahasa dapat dikombinasikan untuk membentuk struktur yang sah dalam suatu bahasa. Di bawah ini, saya akan menjelaskan lebih lanjut mengenai aturan produksi dan tata bahasa.

Aturan Produksi: Aturan produksi adalah himpunan aturan yang mendefinisikan cara menghasilkan kalimat yang sah dalam suatu bahasa. Aturan produksi terdiri dari simbol-simbol dan kaidah-kaidah yang menjelaskan bagaimana simbol-simbol tersebut dapat digabungkan. Aturan produksi biasanya terdiri dari dua bagian: simbol non-terminal dan simbol terminal.

- **Simbol Non-Terminal**: Simbol-simbol yang dapat diperluas atau digantikan oleh simbol-simbol lain dalam aturan produksi. Simbol non-terminal sering kali merupakan frasa atau konstituen dalam tata bahasa. Contoh simbol non-terminal adalah NP (Noun Phrase) atau VP (Verb Phrase).
- **Simbol Terminal**: Simbol-simbol yang tidak dapat diperluas lagi dan merupakan unit-unit terkecil dalam tata bahasa. Simbol terminal biasanya merupakan kata-kata dalam suatu bahasa. Contoh simbol terminal adalah kata benda (noun) atau kata kerja (verb).

  Contoh sederhana aturan produksi dalam tata bahasa menggunakan notasi BNF:

  S -> NP VP NP -> Det N VP -> V NP Det -> 'the' N -> 'cat' V -> 'is'

  Aturan produksi di atas menjelaskan bahwa sebuah kalimat (S) terdiri dari frasa nomina (NP) yang diikuti oleh frasa verba (VP). Kemudian, frasa nomina (NP) terdiri dari sebuah penentu (Det) diikuti oleh sebuah nomina (N), dan frasa verba (VP) terdiri dari sebuah verba (V) diikuti oleh sebuah frasa nomina (NP). Simbol terminal seperti 'the', 'cat', dan 'is' adalah kata-kata konkret yang tidak dapat diperluas lagi.

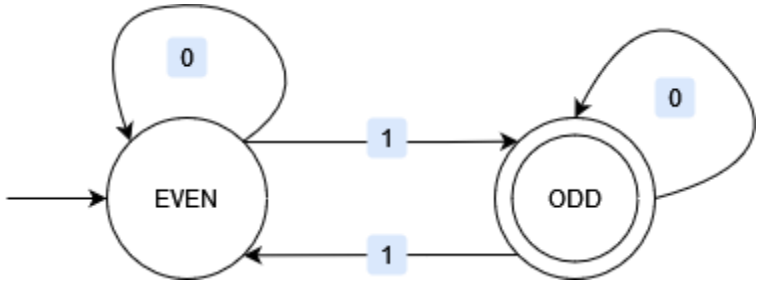
- **Tata Bahasa (Grammar)**: Tata bahasa (grammar) adalah sistem aturan yang mengatur bagaimana kata-kata, frasa, dan kalimat dapat dikombinasikan untuk membentuk struktur yang sah dalam suatu bahasa. Tata bahasa meliputi aturan sintaksis, morfologi, fonologi, dan semantik.
- **Sintaksis**: Mempelajari aturan-aturan yang mengatur struktur dan urutan kata dalam kalimat. Sintaksis menjelaskan hubungan antara kata-kata, frasa, dan kalimat dalam suatu bahasa.
- **Morfologi**: Mempelajari struktur internal kata, seperti afiksasi, infleksi, dan pembentukan kata. Morfologi berfokus pada bagaimana kata-kata dapat mengalami perubahan bentuk untuk mengungkapkan makna.
- **Fonologi**: Mempelajari sistem bunyi dalam suatu bahasa, termasuk aturan-aturan tentang pengucapan, penggabungan, dan pemenggalan kata.

- **Semantik**: Mempelajari makna kata-kata dan struktur kal

FINAL STATE AUTOMATA

- 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

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

Pernyataan Final State Automata secara Formal

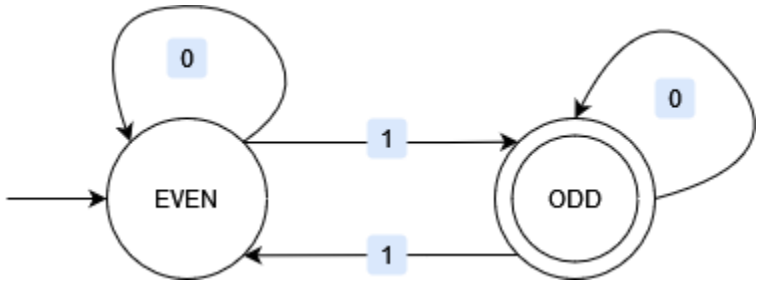
FSA dapat dinyatakan dalam 5 tupel atau M=(Q, Σ, δ, q0, F) dimana:

- 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
- q0 (atau bisa juga S) adalah State awal q0. dimana q0 Q
- F adalah Himpunan State akhir (final) F ⊆ Q

Catatan:

- Jumlah state akhir bisa lebih dari satu

Contoh:



- Q = {ODD, EVEN}
- Σ = {0, 1}
- q0 = EVEN
- F = {ODD}
- δ (Transisi):
    - δ(EVEN, 0) = EVEN State EVEN menerima inputan 0 ke EVEN
    - δ(EVEN, 1) = ODD
    - δ(ODD, 0) = ODD
    - δ(ODD, 1) = EVEN

catatan: Final state F = {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

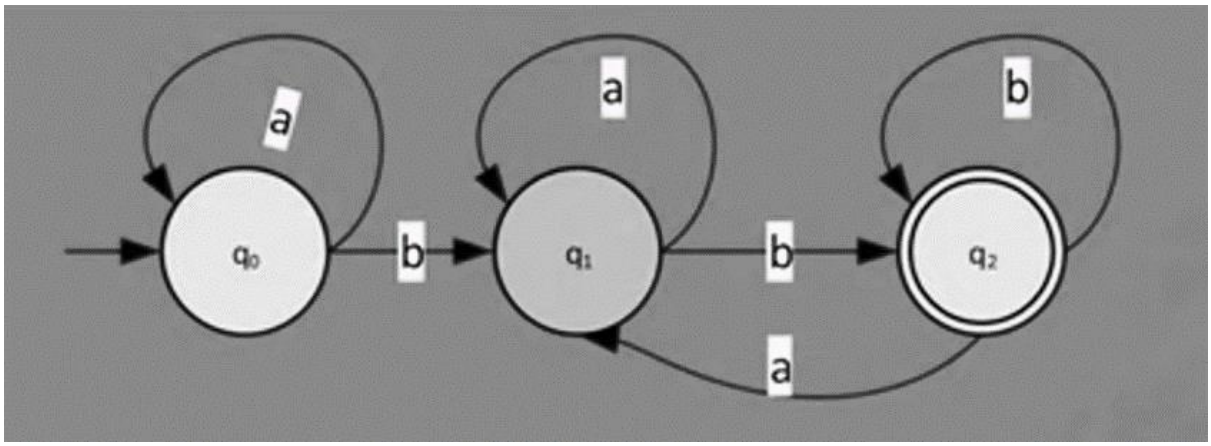
DETERMINISTIC FINAL STATE AUTOMATA

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 = {q0, q1, q2}

∑ = { a,b }

S = q0

F = {q2}

Fungsi Transisi

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

Tabel transisi

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

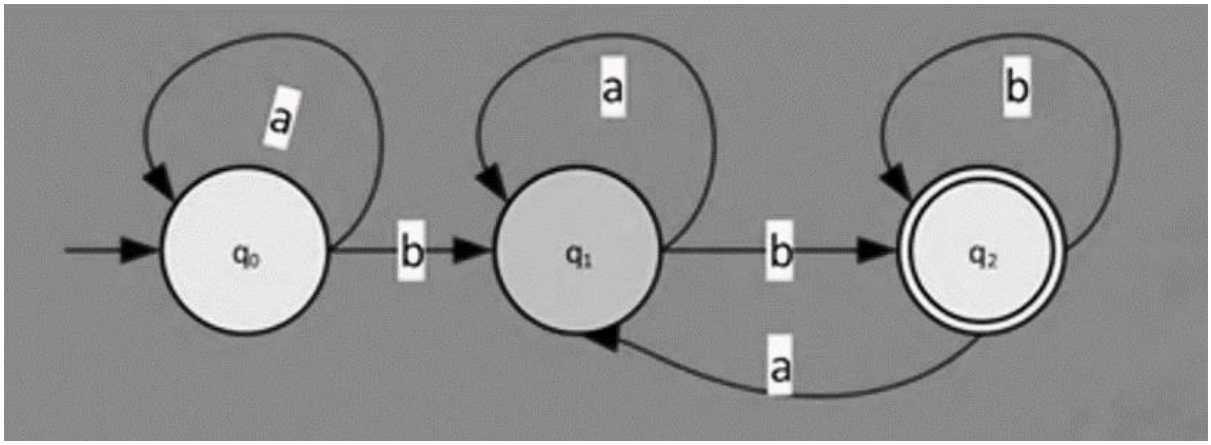
Kapan suatu string input dinyatakan diterima?

Suatu string x dinyatakan diterima,Bila δ (S,x) berada pada state akhir.

Bila M adalah sebuah bahasa FSA

M = (Q, ∑, δ, S, F) menerima bahasa yang disebut L(M) yang merupakan himpunan { x | δ(S,x) di dalam F }

Contoh 2:

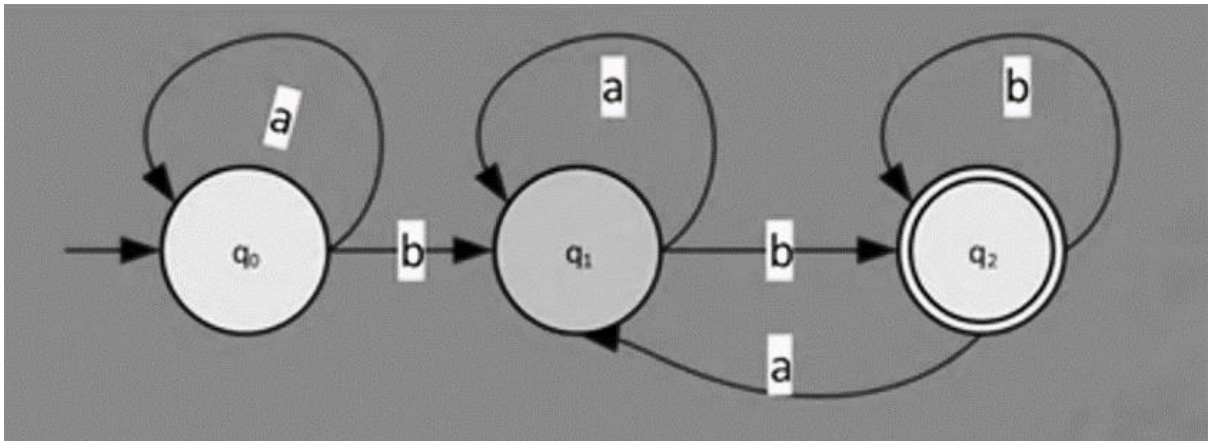


Jika pada gambar contoh 1 kita inputkan string ‘abb’. Maka:

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

Karena q2 adalah state akhir maka ‘abb’ berada berada dalam L(M}

Contoh 3:

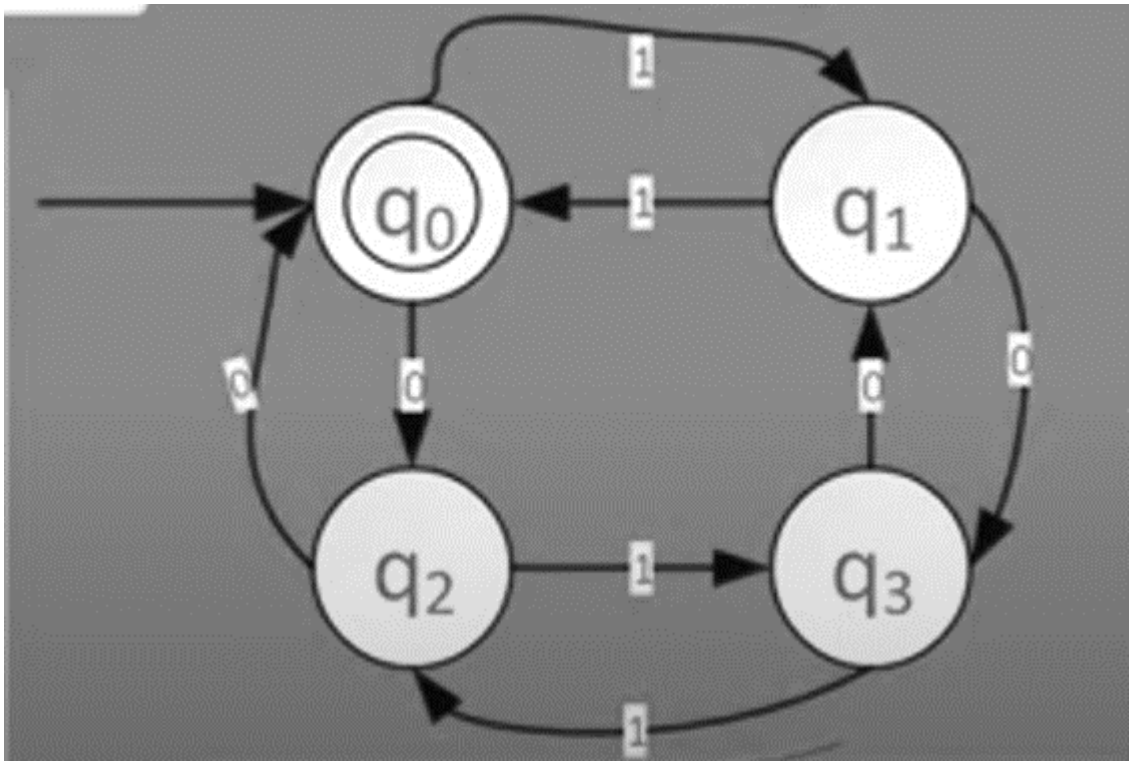


Jika pada gambar contoh 1 inputkan string ‘baba’. Maka;

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

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 = {q0, q1, q2,q3}

∑ = { 0,1 }

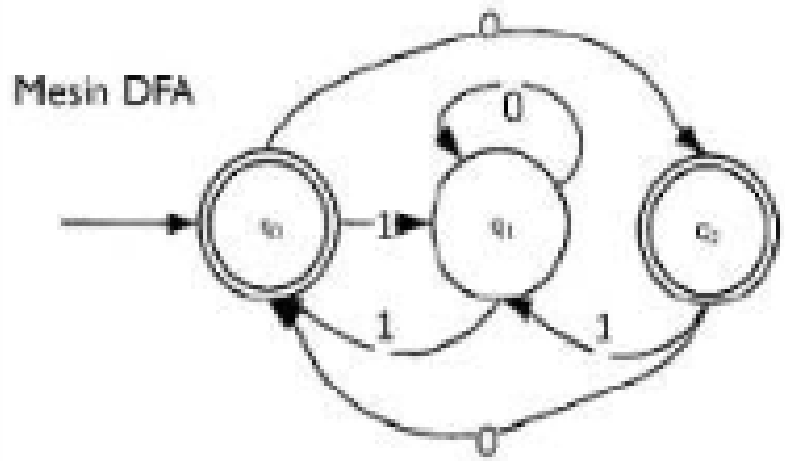
S = q0

F = {q0}

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 dinyatakana 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 statebisa terdapat 0, 1 atau lebih transisi dengan label input yang sama.Perubahan state dapat terjadi secara spontan tanpa input (transisikosong).

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 , Σ , δ , S , F) dimana:

Q = Himpunan state / kedudukan

Σ = Himpunan symbol input

δ = Fungsi Transisi

q0 = State awal q0,

F = himpunan state akhir

* Catatann : 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 (q0)

  berarti bahwa himpunan yang dapat dicapai dari q0 tanpa membaca input

Contoh 1 :

FSA dinyatakan oleh 5 tuple :M = (Q , Σ , δ , S , F) dimana:

Q = {q0 , q1 , q2 , q3 , q4 }

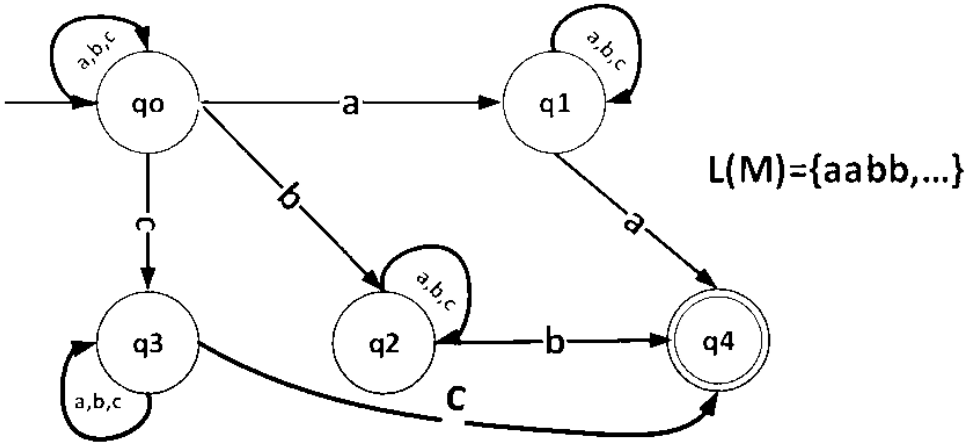
Σ = {a,b,c}

S = q0

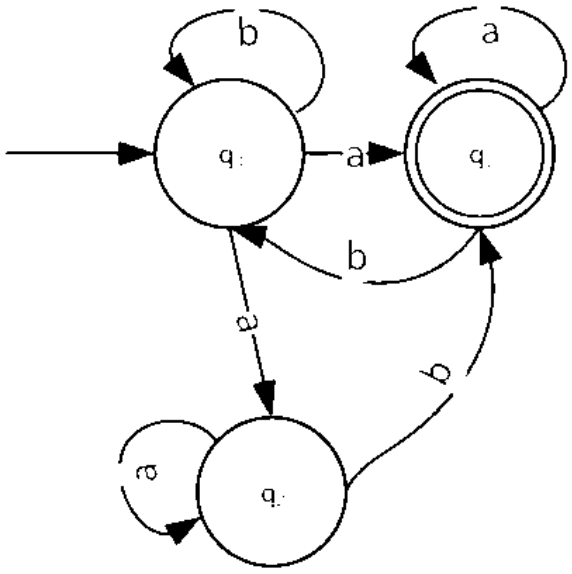
F = {q4}

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	ø	ø	ø



Contoh 2 :

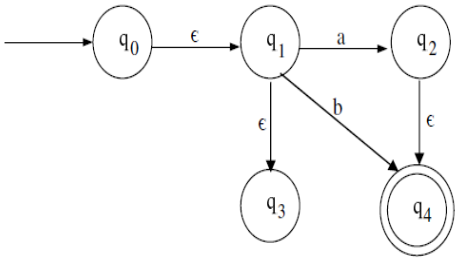


Gambar tersebut termasuk NFA karena jika q0 manerima inputan ‘a’ maka akan berpindah ke q1 atau q2. Tidak tepat satu berikutnya

δ	a	b
q_0	{q1, q2}	q_o
q_1	q_1	q_0
q_2	q_2	q_1

Pada NFA, String diterima jika setidaknya ada 1 dari semuakemungkinan transisi berakhir pada sebuah final sate. Harus mencobasemua kemungkinan.Pada NFA boleh terdapat transisi kosong. Simbol ε berarti tanpamasukan 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

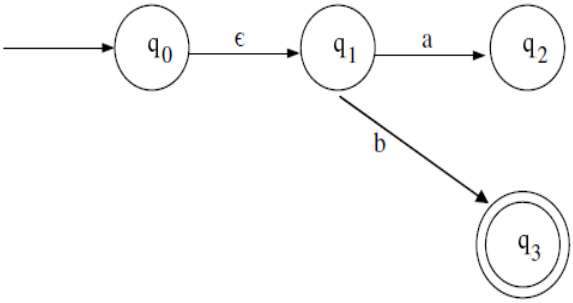


Dari gambar di atas, kita ketahui Є – Closure untuk setiap *state adalah sebagai berikut.*

- Є – Closure (q0) = { q0, q1, q3 }
- Є – Closure (q1) = { q1, q3 }
- Є – Closure (q2) = { q2, q4 }
- Є – Closure (q3) = { q3 }
- Є – Closure (q4) = { q4 }

➢ **Langkah Ekuivalensi NFA dengan e-move ke NFA tanpa e-move**

***** Buatlah tabel tarnsisi dari NFA dengan e-move



Tabel Transisi

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

* Tentukan ϵ-closure untuk setiap *state*

Є – Closure (q0) = { q0, q1 }

Є – Closure (q1) = { q1 }

Є – Closure (q2) = { q2 }

Є – Closure (q3) = { q3 }

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

$$\delta'(q_0, a) = \epsilon\_cl(\delta(\epsilon\_cl(q_0), a)) = \epsilon\_cl(q_2) = \{ q_2 \}$$

$$\delta'(q_0, b) = \epsilon\_cl(\delta(\epsilon\_cl(q_0), b)) = \epsilon\_cl(q_3) = \{ q_3 \}$$

$$\delta'(q_1, a) = \epsilon\_cl(\delta(\epsilon\_cl(q_1), a)) = \epsilon\_cl(q_2) = \{ q_2 \}$$

$$\delta'(q_1, b) = \epsilon\_cl(\delta(\epsilon\_cl(q_1), b)) = \epsilon\_cl(q_2) = \{ q_3 \}$$

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

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

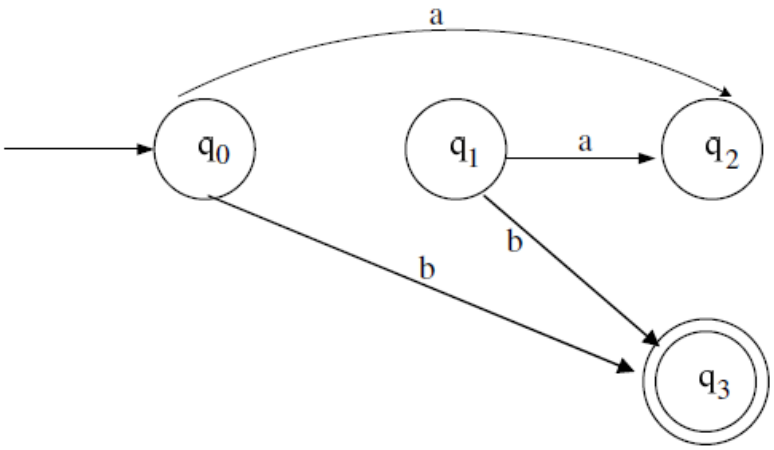
$$\delta'(q_3, a) = \epsilon\_cl(\delta(\epsilon\_cl(q_3), a)) = \epsilon\_cl(\emptyset) = \emptyset$$

$$\delta'(q_3, b) = \epsilon\_cl(\delta(\epsilon\_cl(q_3), b)) = \epsilon\_cl(\emptyset) = \emptyset$$

* 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	Ø	Ø
q_3	Ø	Ø

* Kemudian, tentukanlah himpunan state akhir untuk NFA tanpa ϵ – move ini. Himpunan state akhir semula adalah {q3}. Karena tidak ada state lain yang ϵ – closurenya memuat q3 , maka himpunan state akhir sekarang tetap {q3}. Sehingga diperoleh diagram transisi sebagai berikut.



FINITE STATE TRANSDUCER (FST)

Input : Input berupa deretan karakter dan string
Output : Output berupa diterima atau tidak diterimanya suatu inputan

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.
Didefenisikan dengan 6 tupel
•Q =Himpunan state
•£ =Himpunan symbol input
• =Fungsi transisi
•S=State awal S, dimana S Q
•Δ=Himpunan output
• =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 = {q0, q1, q2}

Σ = {0, 1} (input dalam biner)

S = q0

Δ = {0, 1, 2} (untuk outputnya pada kasus mod dengan 3 maka sisanya kemungkinan

adalah

0,1,2)

λ(q0) =

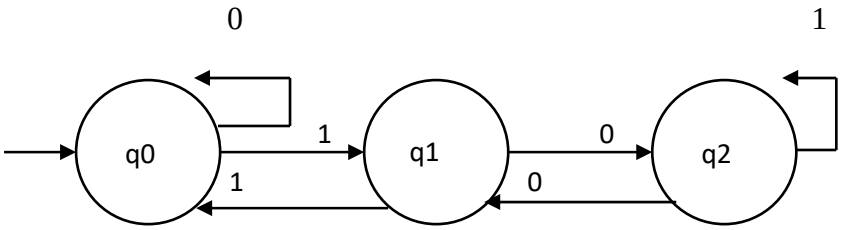
0 λ(q1)

= 1

λ(q2) =

2

Gambar Mesin Moore untuk modulus 3 :



Pembuktian :

- 5 mod 3 = ?
  input 5 dalam biner 101

bila kita masukkan 101 kedalam mesin, urutan state yang dicapai adalah :
q0, q1, q2, q2

State terakhir yang dicapai adalah q2, λ(q2) = 2 Maka 5
mod 3 = 2

- 10 mod 3 = ?

input 10 dalam biner 1010 bila kita masukkan 1010 kedalam mesin,
urutan state yang dicapai adalah : q0, q1, q2, q2, q1

State terakhir yang dicapai adalah q1, λ(q1) = 1 Maka 10
mod 3 = 1

- 12 mod 3 = ?

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

urutan state yang dicapai adalah : q0, q1, q0, q0

State terakhir yang dicapai adalah q0, λ(q0) = 0 Maka 12

mod 3 = 0

Contaxt Free Grammar

Tata Bahasa Bebas Komteks atau CFG

•CFG atau Contaxt Free Grammar adalah tata bahasa formal dimana setiap aturan produksi adalah dalam bentuk a→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 E N (Non-terminal)

•B E (TUN) (Terminal atau Non-terminal)

•B boleh berisi E

*Contoh aturan Produksi CFG

•X→bY|Za

•Y→aY|B

•Z→bZ|E

•B→CDefg

•D→BcDe

CFG VS RE

•Keduanya merupakan suatu cara untuk menunjukan 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

•PadavCFG,bagian yang belum diturunlan bisa berada dimana saja

Contoh aturan produksi ER

S→aE

E→A|B

A→aA|b

B→aE|b

Contoh aturan produksi CFG

X→bY|Za

Y→aY|B

Z→bZ|E

B→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→AB

A→aA|a

B→bB|b

Hasil penurunan

S→AB→aAV→aaB→aabB→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→aAS→asbAS→aabAS→aabbaS→aabbaa

Reightmost Derivation

S→aAS→aAa→aSbAa→aSbbaa→aabbaa

Ambigiutas

Ambigius adalah kedwiartian atau maksa ganda

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

Contoy ambigius

Aturan produksi

S→A|B

A→a

B→a

String yang dicari "a"

Cara 1

S→A→a

Cara 2

S→B→a

Aturan produksi

S→SbS

S→ScS

S→a

String yang dicari'abaca"

Cara

S→SbS→SbScS→SbSca→Sbaca→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 suatu dekomposisi,dan susunannya akan menentukan arti, mla 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:

$$A \lozenge \beta A$$

* Merupakan aturan produksi yang rekursif kanan:

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

atau kumpulan symbol variabel dan terminal

* Contoh aturan produksi yang rekursif kanan:

$$S \lozenge dS$$

$$B \lozenge adB$$

Aturan Produksi Rekursif Kiri

* Sebuah aturan produksi dalam bentuk:

$$A \lozenge A\beta$$

* Merupakan aturan produksi yang rekursif kiri, contohnya:

$$S \lozenge Sd$$

$$B \lozenge Bad$$

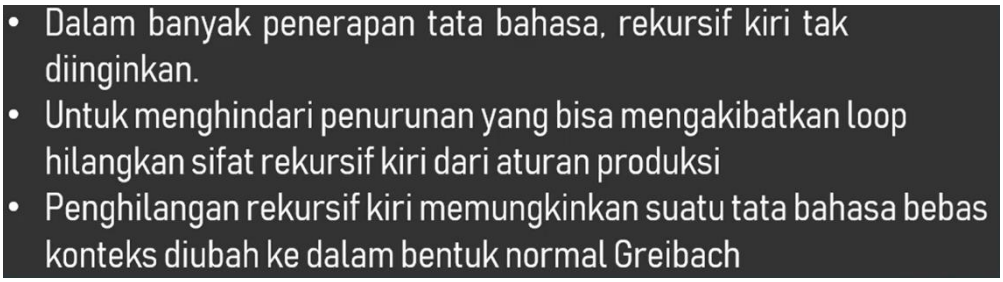
* Produksi yang rekursif kanan menyebabkan pohon penurunan tumbuh ke kanan, sebaliknya produksi yang rekursif ke kiri menyebabkan pohon penurunan tumbuh ke kiri. Bisa dilihat pada pohon penurunan dari tata bahasa bebas konteks dengan aturan produksi:

$$S \lozenge aAc$$

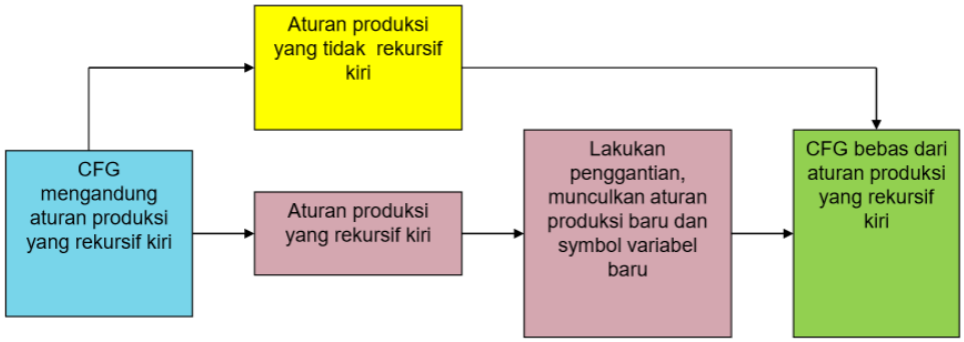
$$A \lozenge Ab \mid \varepsilon$$

Penghilangan Rekursif Kiri

1. Tujuan Rekursif Kiri Dihapus



2. Tahapan Penghilangan Rekursif Kiri



a) Pisahkan aturan produksi yang rekursif kiri dan yang tidak
   - ❖ Aturan produksi yang rekursif kiri :
     $A \rightarrow A\alpha1 \mid A\alpha2 \mid A\alpha3 \mid \ldots \mid A\alpha n$
   - ❖ Aturan produksi yang tidak rekursif kiri :
     $A \rightarrow \beta1 \mid \beta2 \mid \beta3 \mid \ldots \mid \beta m$

b) **Tentukan simbol-simbol $\alpha1, \alpha2, \alpha3, \ldots, \alpha n$ dan $\beta1, \beta2, \beta3, \ldots, \beta m$** dari setiap aturan produksi yang memiliki simbol ruas kiri yang sama

c) Lakukan penggantian aturan produksi yang rekursif kiri menjadi sebagai berikut:
   $A \rightarrow \beta1\ Z \mid \beta2\ Z \mid \beta3\ Z \mid \ldots \mid \beta m\ Z$ $Z \rightarrow \alpha1 \mid \alpha2 \mid \alpha3 \mid \ldots, \alpha n$ $Z \rightarrow \beta1 \mid \beta2 \mid \beta3 \mid \ldots \mid \beta m$

   Penggantian diatas dilakukan untuk setiap aturan produksi dengan symbol ruas kiri yang sama. Bisa muncul symbol variabel baru Z1, Z2 dan seterusnya sesuai banyaknya variabel yang menghasilkan produksi yang rekursif kiri.

d) 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 : $S \rightarrow Sab \mid Sbd$ $\alpha1 = ab$; $\alpha2 = bd$
  Aturan produksi tidak rekursif kiri : $S \rightarrow aSc \mid dd \mid ff$ $\beta1 = aSc$; $\beta2 = dd$; $\beta3 = ff$

  Aturan produksi rekursif kiri, $S \rightarrow Sab \mid Sbd$ digantikan oleh:
  $S \rightarrow aScZ1 \mid dd\ Z1 \mid ff\ Z1$ $Z1 \rightarrow ab \mid bd\ Z1 \rightarrow abZ1 \mid bdZ1$
  Hasil akhir setelah penghilangan rekursif kiri adalah : $S \rightarrow aSc \mid dd \mid ff$

- Contoh 12.2
  Hilangkan rekursif kiri dari aturan produksi berikut!
  $S \rightarrow Sab \mid Sb \mid cA$ $A \rightarrow Aa \mid a \mid bd$
  Penyelesaian:
  - Aturan produksi rekursif kiri, $S \rightarrow Sab \mid Sb$ $\alpha1 = ab$; $\alpha2 = b$
  - Aturan produksi rekursif kiri, $A \rightarrow Aa$ $\alpha1 = a$
  - Aturan produksi tidak rekursif kiri, $S \rightarrow cA$ $\beta1 = cA$
  - Aturan produksi tidak rekursif kiri, $A \rightarrow a \mid bd$ $\beta1 = a$; $\beta2 = bd$

  Pergantian aturan produksi rekursif kiri:

S →Sab | Sb digantikan oleh:

- S → caZ1
- Z1 → ab | b
- Z1 → abZ1 | bZ1

A →Aa digantikan oleh :

- A → aZ2 | bdZ2
- Z2 → a
- Z2 → aZ2

Hasil akhir setelah penghilangan rekursif kiri adalah:

S → cA

A → a | bd

S → caZ1

Z1 → ab | b

Z1 → abZ1 | bZ1

A → aZ2 | bdZ2

Z2 → a

Z2 → aZ2

Pengantar penyederhanaan aturan produksi Context Free Grammar

CFG atau Context Free Grammar adalah tata bahasa formal di mana setiap aturan produksi adalah dalam bentuk A → B di mana A adalah pemproduksi, dan B adalah hasil produksi. Batasannya hanyalah ruas kiri adalah sebuah simbol variabel. Dan pada ruas kanan bisa berupa terminal, symbol, variable ataupun ε, Contoh aturan produksi yang termasuk CFG adalah seperti berikut ini:

- X → bY | Za
- Y → aY | b
- Z → bZ | ε

CFG adalah tata bahasa yang mempunyai tujuan sama seperti halnya tata bahasa regular yaitu merupakan suatu cara untuk menunjukkan bagaimana menghasilkan suatu untai-untai dalam sebuah bahasa.

CFG perlu disederhankan dengan tujuan untuk melakukan pembatasan sehingga tidak menghasilkan pohon penurunan yang memiliki kerumitan yang tak perlu atau aturan produksi tak berarti. Berikut merupakan langkah-langkah dalam melakukan penyederhanaan CFG:

- Eliminasi ε-production
- Eliminasi unit production
- Eliminasi useless symbol

Berikut contoh Penyederhanaan CFG:

A→ cAB | ab
B→ BaC | C
C → bC | ε

<u>Step 1 – Eliminasi ε-production</u>

- Hilangkan semua hasil produksi yang ε
- Jika X → ε, maka X adalah nullable
- Jika Y → X, maka Y adalah nullable
- Jika Z → Xa, maka Z → a

Result :

A→cAB|ab|cA
B→BaC|C|Ba|aC|a
C → bC| b

<u>Step 2 – Eliminasi unit production</u>

- Jika ada hasil produksi yang terdiri dari 1 variable, maka hasil produksi tersebut disubstitusi dengan hasil produksi dari grammar dimana variable tersebut menjadi pemproduksiResult :
  A→cAB|ab|cA
  B → BaC| bC | b | Ba | aC | a
  C → bC| b

<u>Step 3 – Eliminasi useless symbol</u>

- Uji Generate
  Jika X → YZ, Y → aa, Z → b, maka X,Y,Z lolos uji generate
  Jika M → PQ, P → aa, Q → QQ, maka M dan Q tidak lolos uji generate
- Uji Reachable
  Jika S → TU dimana S adalah start symbol maka T dan U lolos uji reachable

Final Result :

A→cAB|ab|cA
B → BaC| bC | b | Ba | aC | a
C → bC| b

* Urutan penghapusan aturan produksi :
Hilangkan produksi ε
Hilangkan produksi unit
Hilangkan produksi useless

CNF (Chomsky Normal Form)

1. **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

   Contoh :
   Contoh :
   S ◊ AA | C | bd
   A ◊ Bb | ε
   B ◊ AB | d
   C ◊ de
   1. Hilangkan produksi ε,sehingga menjadi:
   S ◊ A | AA | C | bd
   A ◊ Bb
   B ◊ B | AB | d
   C ◊ de
   2. Selanjutnya penghilangan produksi unit menjadi:
   S ◊ Bb | AA | de | bd
   A ◊ Bb
   B ◊ AB | d
   C ◊ de
   Penghilangan produksi unit bisa menghasilkan produksi useless.
   3. 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.

CFG dengan CNF

Bentuk normal Chomsky (Chomsky Normal Form, CNF) adalah grammar bebas konteks (CFG) dengan setiap produksinya berbentuk :

$A \rightarrow BC$ atau $A \rightarrow a$.

Transformasi CFG ke CNF adalah transformasi berikut :

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

$A \rightarrow ß$,
dimana ß
$\in (V \mid V)^*$
$A \rightarrow ß$,
dimana ß
$\in (V \mid V)^*$
$A \rightarrow BC$,
atau
$A \rightarrow a$
$A \rightarrow BC$,
atau
$A \rightarrow a$