TEORI BAHASA DAN AUTOMATA

“RANGKUMAN ”

Disusun Oleh :

Nama	: Nafisa Aura Dwiyanti
Nim	: 21346016
Dosen Pengampu	: Widya Darwin

PROGRAM STUDI INFORMATIKA

JURUSAN TEKNIK ELEKTRONIKA

FAKUTAS TEKNIK

UNIVERSITAS NEGERI PADANG

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 Teori Bahasa Dan Automata yang berjudul "Kumpulan Teori Bahasa dan Automata" 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.

Makalah ini dibuat sebagai salah satu syarat untuk memenuhi tugas mata kuliah Teori Bahasa Dan Automata. Kami menyadari bahwa dalam penyusunan makalah ini masih terdapat banyak kekurangan dan jauh dari kata sempurna. Oleh karena itu, Kami mengharapkan kritik dan saran untuk menyempurnakan makalah ini dan dapat menjadi acuan dalam Menyusun makalah-makalah selanjutnya. Kami juga memohon maaf apabila dalam penulisan makalah ini terdapat kesalahan kata - kata, pengetikan dan kekeliruan, sehingga membingungkan pembaca dalam memahami maksud penulis.

Adapun makalah ini penulis rangkum dari beberapa sumber yang dapat dipercaya yang sajian penulisnya, dengan harapan makalah ini dapat menambah pengetahuan kita tentang Pendidikan Di Era Teknologi Informasi Dan Komunikasi. Demikian yang dapat kami sampaikan. Akhir kata, semoga makalah ini dapat menambah wawasan bagi kita semua.

Painan, Juni 2023

Nafisa Aura Dwiyanti

DAFTAR ISI

BAB I

PENDAHULUAN

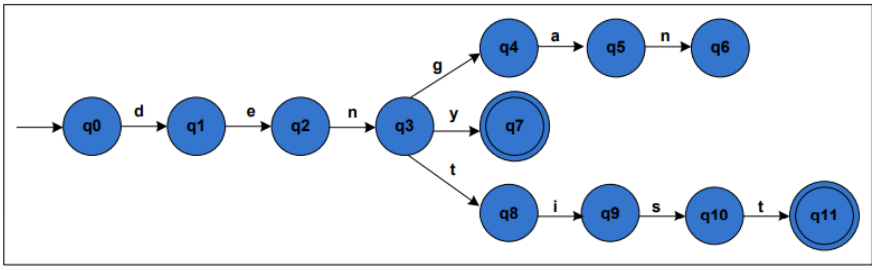
A. PENGANTAR TEORI BAHASA DAN AUTOMATA

Ilmu komputer memiliki dua komponen utama yaitu model dan gagasan tentang komputasi dan teknik rekayasa untuk perancangan sistem komputasi (perangkat keras dan perangkat lunak). Teori bahasa dan otomata masuk kedalam komponen utama dalam ilmu komputer. Teori bahasa dan otomata diterapkan pada beberapa perancangan digital seperti pada pmbuatan bahas apemrograman dan kompilator. Sedangkan tujuan mempelajari Teori Bahasa dan Otomata sendiri yaitu mengajarkan dasar-dasar teori bahasa formal dan modelmodel mesin matematis yang menggambarkan prinsip kerja komputer.

Otomata merupakan suatu bentuk (model matematika) yang memiliki fungsi-fungsi dari komputer digital yaitu menerima input, menghasilkan output, bisa memiliki penyimpanan sementara dan mampu membuat keputusan dalam mentransformasikan input ke output. Input pada mesin otomata dianggap sebagai bahasa yang harus dikenali oleh mesin. Selanjutnya mesin otomata akan membuat keputusan yang mengindikasikan apakah input ini diterima atau tidak. Otomata bisa digunakan untuk memodelkan hardware.

String sendiri merupakan suatu kata (gabungan dari huruf) dalam bahasa, contoh string adalah ‘Tanjungpinang‘. String juga dapat bernilah kosong dan biasa dilambangkan menggunakan simbol ε.

Contoh penerapan mesin otomata, misalnya kita akan memodelkan mesin otomata yang hanya dapat menerima inputan dalam bahasa inggris. Apabila digambarkan pemodelan bahasanya akan menjadi :



Pada gambar diatas merupakan mesin otomata yang hanya dapat menerima inputan berupa bahasa inggris, jika mesin mendapatkan string inputan :

- Ditolak
- Diterima
- Diterima

suatu string dikatakan diterima apabila mencapai state akhir(lingkaran ganda) untuk kasus ini ada pada q7 dan q11. State awal selalu diawali oleh panah tanpa inputan (state q0).

BAB II
PEMBAHASAN

A. Tata Bahasa (Grammar) pada hirarki Chomsky

Tata Bahasa (grammar) didefinisikan sebagai kumpulan dari himpunan-himpunan variable, simbol-simbol terminal, simbol awal yang dibatasi oleh aturan produksi. Pada tahun 1959, Noam Chomsky menggolongkan tingkatan bahasa menjadi empat yang disebut sebagai Hirarki Chomsky. Penggolongan Hirarki Chomsky :

BAHASA	MESIN OTOMATA	BATASAN ATURAN PRODUKSI
Regular/Tipe 3	Finite State Automata (FSA) : • Deterministic Finite Automata (DFA) • Non-Deterministic Finite Automata (NFA)	• α merupakan sebuah simbol variable • β maksimal memiliki sebuah simbol variable yang bila ada terletak diposisi paling kanan
Bebas Konteks	Push Down Automata (PDA)	α berupa sebuah simbol variable
Context Sensitive/Tipe1	Linier Bounded Automata	$\|\alpha\| \leq \|\beta\|$
Natural Language/Tipe 0	Mesin Turing	Tidak ada batasan

Aturan produksi merupakan pusat dari tata bahasa yang menspesifikasikan bagaimana suatu tata bahasa melakukan transformasi dari suatu string ke bentuk lainnya dan melalui aturan produksi tersebut didefinisikan suatu bahasa yang berhubungan dengan tata bahasa tersebut. Suatu aturan produksi dapat dinyatakan dalam bentuk $\alpha \rightarrow \beta$ (dibaca : α menghasilkan β atau dibaca : α menurunkan β). Suatu α menyatakan simbol-simbol pada ruas kiri aturan produksi. Sedangkan β menyatakan simbol-simbol pada ruas kanan aturan produksi. Beberapa istilah dalam aturan produksi :

- Simbol variable/nonterminal
  simbol yang masih dapat diturunkan, biasanya dilambangkan menggunakan abjad capital.
- Simbol terminal
  simbol yang tidak dapat diturunkan kembali, biasanya dilambangkan menggunakan abjad non-capital (huruf kecil).

B. Memodelkan Finite Automata (FA)

Finite State Automata merupakan materi fundamental yang dipelajari pada jurusan ilmu komputer atau teknik informatika, khususnya pada mata kuliah teori bahasa dan otomata. Finite State Automata dapat didefinisikan sebagai model matematika untuk sistem apa pun yang memiliki jumlah state kondisional terbatas.

Finite State Automata atau Finite State Machine adalah mesin abstrak yang memiliki lima elemen atau tuple. Kelima elemen tersebut meliputi **input**, **output**, **himpunan state**, **relasi state**, dan **relasi output**.

Finite State Automata (FSA) berupa sistem model matematika dengan masukan dan keluaran diskrit yang dapat mengenali bahasa paling sederhana (bahasa reguler) dan dapat diimplementasikan secara nyata.

FSA dapat menerima input dan mengeluarkan output yang memiliki state yang berhingga banyaknya. Selain itu, FSA memiliki sekumpulan aturan untuk berpindah dari satu state ke state lain berdasarkan input dan fungsi transisi yang diterapkan. Perlu diketahui bahwa sistem Finite State Automata hanya dapat mengingat state terkini karena tidak memiliki tempat penyimpanan/memory.

Finite State Automata pada dasarnya digunakan untuk mengenali pola. Dibutuhkan string simbol sebagai input dan statusnya berubah sesuai dengan input tersebut. Ketika ditemukan simbol input yang diinginkan, maka terjadi transisi.

Pada saat transisi, automata dapat berpindah ke keadaan berikutnya atau tetap dalam keadaan yang sama. Finite State Automata memiliki dua status, status **Terima** atau status **Tolak**. Ketika string input berhasil diproses, dan automata mencapai state akhir, maka statusnya adalah Terima, demikian juga sebaliknya.

Finite State Automata dapat didefinisikan dengan persamaan berikut:

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

- **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, $F \subseteq Q$

a. Karakteristik Finite State Automata

Finite State Automata memiliki beberapa karakteristik berikut:

1. Setiap Finite Automata memiliki keadaan dan transisi yang terbatas.
2. Transisi dari satu keadaan ke keadaan lainnya dapat bersifat deterministik atau non-deterministik.
3. Setiap Finite Automata selalu memiliki keadaan awal.
4. Finite Automata dapat memiliki lebih dari satu keadaan akhir.
5. Jika setelah pemrosesan seluruh string, keadaan akhir dicapai, artinya otomata menerima string tersebut.

b. Cara kerja Finite State Automata

Finite State Automata bekerja dengan cara mesin membaca memori masukan berupa tape yaitu 1 karakter tiap saat (dari kiri ke kanan) menggunakan head baca yang dikendalikan oleh kotak kendali state berhingga dimana pada mesin terdapat sejumlah state berhingga.

Finite Automata selalu dalam kondisi yang disebut state awal (initial state) pada saat Finite Automata mulai membaca tape. Perubahan state terjadi pada mesin ketika sebuah karakter berikutnya dibaca.

Ketika head telah sampai pada akhir tape dan kondisi yang ditemui adalah state akhir, maka string yang terdapat pada tape dikatakan diterima Finite Automata (String-string merupakan milik bahasa bila diterima Finite Automata bahasa tersebut).

c. Jenis-Jenis Finite State Automata

State Finite Automata dapat dibagi menjadi dua jenis, yaitu:

- Deterministic Finite Automata (DFA)
- Non-deterministic Finite Automata (NFA)

Kedua finite automata di atas mampu mengenali himpunan reguler secara presisi. Dengan demikian kedua finite automata itu dapat mengenali string-string yang ditunjukkan dengan ekspresi reguler secara tepat.

DFA dapat menuntun recognizer(pengenal) lebih cepat dibanding NDFA. Namun demikian, DFA berukuran lebih besar dibanding NDFA yang ekivalen dengannya. Lebih mudah membangun NDFA dibanding DFA untuk suatu bahasa, namun lebih mudah mengimplementasikan DFA dibanding NDFA.

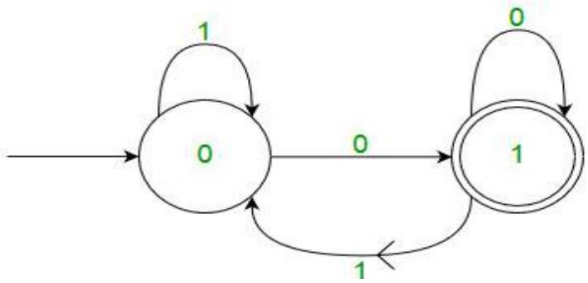
- Deterministic Finite Automata (DFA)

  Dalam Deterministic Finite Automata (DFA), untuk karakter input tertentu, mesin hanya dapat menuju satu state dan fungsi transisi dipakai pada setiap state untuk setiap simbol input. Selain itu, pada DFA, perpindahan state null (atau ε) tidak diperbolehkan, artinya, DFA tidak dapat mengubah state tanpa karakter input sama sekali.

  Deterministic Finite Automata (DFA) terdiri atas 5 tuple, yakni: **{Q, Σ, q, F, δ}**

  - **Q** : himpunan semua state
  - **Σ** : himpunan simbol input (simbol yang digunakan oleh mesin untuk mengambil nilai input)
  - **q** : initial state (state atau keadaan awal dari mesin)
  - **F** : himpunan state akhir
  - **δ** : fungsi transisi, yang didefinisikan δ : Q X Σ --> Q.

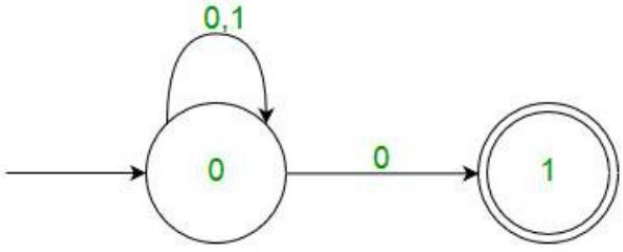
  Sebagai contoh berikut adalah DFA Σ = {0, 1}yang menerima semua string yang diakhiri dengan 0.



- Nondeterministic Finite Automata (NFA)

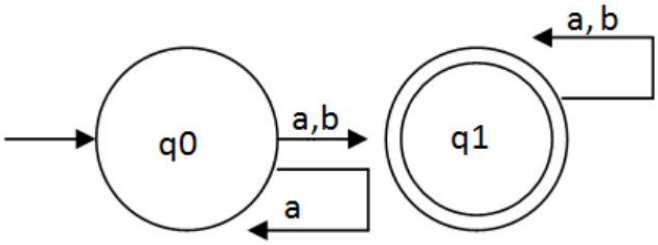
  Pada dasarnya beberapa hal di atas tidak membuat NFA lebih unggul dari DFA. Jika kita membandingkan keduanya dalam hal kekuatan, keduanya setara.

  Karena fitur tambahan di atas, NFA memiliki fungsi transisi yang berbeda, selebihnya sama dengan DFA. Fungsi transisi pada NFA dapat didefinisikan sebagai berikut: **δ: Q X (Σ U ε) --> 2 ^ Q.**



B. Konversi NFA ke DFA

Konversi dari NonDeterministic Finite Automata (NFA) ke Deterministic Finite Automata (DFA). Contoh
Buatlah DFA yang Equevalen dengan NFA berikut ini :



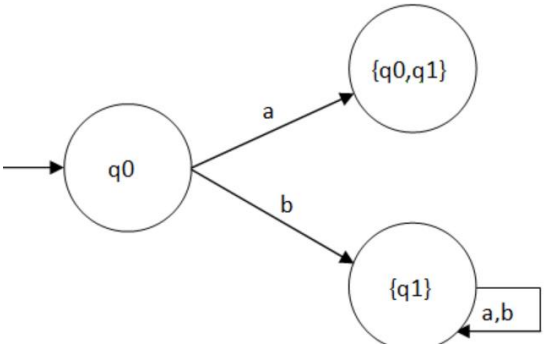
Konfigurasi NFA secara formal adalah sebagai berikut :
Q = {q0, q1 }
S = {a, b}
S = q0
F = {q1}

Fungsi-fungsi transisinya sebagai berikut :
d (q0, a) = {q0,q1},
d (q0, b) = q1,
d (q1, a) = q1,
d (q1, b) = q1,

Jika disajikan dalam tabel transisi :

d	a	b
q0	{q0,q1}	{q1}
q1	{q1}	{q1}

Berdasarkan tabel transisi pada NFA, kita gambarkan diagram transisi DFA nya terlebih dahulu .



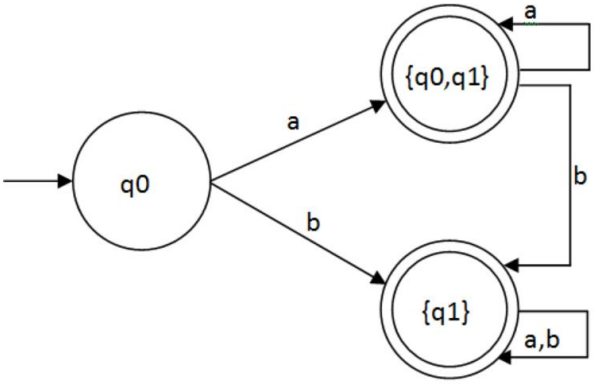
Kemudian tentukan arah dua busur keluar untuk State {q0,q1} yaitu busur arah ‘a’ dan ‘b’. Hal ini karena untuk DFA masing masing state harus memiliki pasangan busur, dalam soal ini busur ‘a’ dan ‘b’.

Busur arah ‘a’ :
d ({q0,q1}, a) = {q0,a} È {q1,a}
= {q0,q1} È {q1}
= {q0,q1}

Busur arah ‘b’ :
d ({q0,q1}, b) = {q0,b} È {q1,b}
= {q1} È {q1}
= {q1}

Selanjutnya menentukan state akhir, yaitu kita ingat bahwa F = {q1} ketika masih NFA maka himpunan state akhir (F) sekarang adalah semua yang mengandung state q1.
Maka, F = {{q1}, {q0, q1}}

Gambar Diagram Transisi Akhir setelah di konversi ke DFA

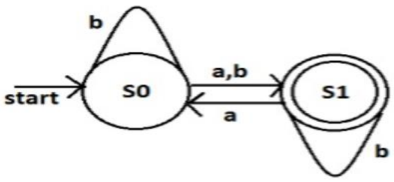


C. Mengubah FA Menjadi DFA

Dari suatu mesin Non Deterministic Finite Automata (NFA) dapat dikonversi atau dibuat menjadi suatu mesin Deterministic Finite Automata (DFA) yang memiliki kemampuan menerima Bahasa yang sama (ekuivalen). Berikut contoh konversi dari Non Deterministic Finite Automata menjadi Deterministic Finite Automata:

- Berikut table transisi dan diagram transisi untuk NFA :

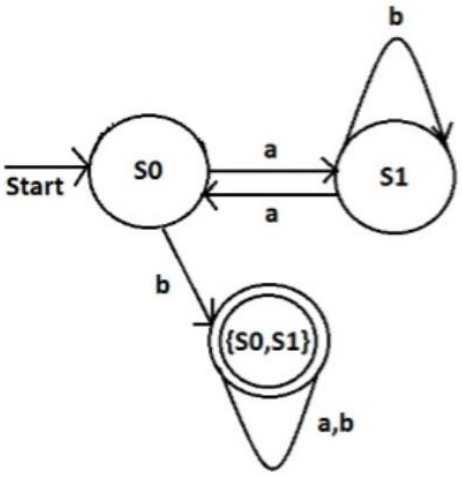
State	a	b
S0	S1	{S0,S1}
S1*	S0	S1



Dapat dilihat dari gambar di atas ada suatu state yang diberi inputan menuju ke beberapa state, yaitu S0 yang diberi inputan b bisa menuju ke S0 dan S1

- Lalu dikonversi menjadi DFA, dengan cara membuat state baru berupa gabungan dari S0 dan S1.
- Lalu untuk state {S0,S1} jika diberi inputan a maka hasilnya adalah gabungan dari hasil S0 dan S1 yang diberi inputan a yaitu {S0,S1}
- Lalu untuk state {S0,S1} jika diberi inputan b maka hasilnya adalah gabungan dari hasil S0 dan S1 yang diberi inputan b yaitu {S0,S1}
- Berikuta table transisi dan diagram transisi untuk DFA nya :

State	a	b
S0	S1	{S0,S1}
S1*	S0	S1
{S0,S1}*	{S0,S1}	{S0,S1}



D. Menggabungkan FA menggunakan operasi Union, Konkatenasi, dan operasi Kleene-Star dan dapat memodelkan format teks tertentu menggunakan Ekspresi Reguler.

Operasi *UNION* — menggunakan simbol ∪ sebagai operatornya — menggabungkan *tuple* dua buah relasi atau lebih. Sebagai contoh, ekspresi aljabar relasional **R∪S** menggabungkan i *tuple* relasi R dengan j *tuple* relasi S, tanpa duplikasi *tuple*, sehingga diperoleh maksimum (i+j) *tuple*.

Sintaks *query* untuk operasi *union* adalah sebagai berikut.

```
SELECT DaftarEkspresi FROM tabel
UNION
SELECT DaftarEkspresi FROM tabel
```

Bila diperlukan, pernyataan **SELECT** boleh dikombinasikan dengan operasi lain: proyeksi, seleksi, produk Cartesian, inner join, outer join, dsb. Daftar ekspresi dapat berupa kolom tabel, konstanta, perhitungan, fungsi skalar, fungsi agregasi. Yang penting, kolom dalam setiap pernyataan **SELECT** harus *union compatible* yaitu:

• jumlah kolom sama

• urutan kolom sama dengan tipe data sama atau kompatibel.

Operator **UNION** menghasilkan gabungan baris unik terurut dari dua tabel atau lebih. Data diurutkan berdasarkan *primary key* (bila ada) atau berdasarkan gabungan kolom yang ada di pernyataan **SELECT** pertama.

Operator **UNION ALL** menghasilkan gabungan semua baris dari dua tabel atau lebih tanpa pengurutan.

E. Konversi Ekspresi Reguler ke FA

Ekspresi Regular adalah menggambarkan bahasa regular, contoh nya yaitu :
Regular expression atau ekspresi reguler adalah model yang menyatakan aturan penurunan/penyusunan suatu bahasa reguler. Regular expression dapat dibentuk dari sebuah FA (hasil terjemahan sebuah FA) atau sebaliknya, regular expression menyatakan spesifikasi sebuah bahasa yang kemudian dapat disusun FA-nya.

Contoh: [a..z]([a..z]+[0..9])*@[a..z]*.[a..z]* ◊ regular expression untuk menyatakan sebuah alamat email ◊ dapat digunakan untuk mem-validasi sebuah alamat email.
Definisi ekspresi reguler Jika Σ merupakan himpunan simbol,

maka 1. Ø , λ, dan a ∈Σ 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∪T
rt	operasi penyambungan string thd himpunan
r^*	Kleene closure dari R
(r)	r

Contoh ekspresi reguler

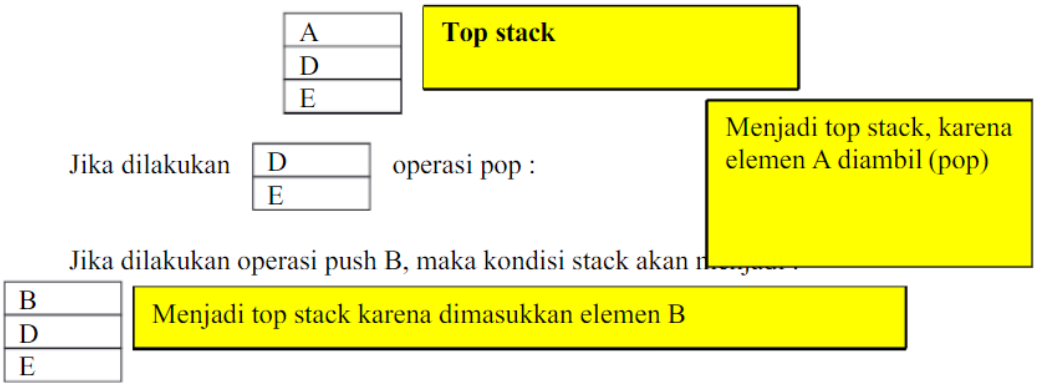
- (0+1)* : himpunan seluruh string yang dapat dibentuk dari simbol '0' dan '1'
- (0+1)*00(0+1)* : himpunan string biner yang mengandung paling sedikit satu substring '00'
- (0+1)*00 : himpunan string biner yang diakhiri dengan '00'

F. Pushdown Automata

Push Down Automata (PDA) merupakan mesin otomata dari bahasa bebaskonteks. PDA di gambarkan sebagai tempat penyipanan yang tidak terbatas berupa stack / tumpukan.

Stack ialah kumpulan dari elemen-elemen sejenis dengan sifat penambahan elemen dan pengambilan elemen melalui suatu tempat yang disebut top of stack (puncak stack). Prinsip pada stack adalah LI□!. Pengambilan elemen dari stack dinyatakan dengan operasi pop " sedang memasukkan elemen ke dalam stack denganoperasi push.

Contoh stack :



Definisi PDA adalah pasangan 7 tuple :

$M = (Q, \Sigma, q_0, F, \delta, \Gamma, Z_0)$, dimana :
Q : himpunan hingga state,
Σ : alfabet input,
Γ : alfabet/simbol *stack*,

q_0 : state awal, $q_0 \in Q$

Z_0 : simbol awal *stack*, $Z_0 \in \Gamma$
F : himpunan state penerima, $F \subseteq Q$

δ : fungsi transisi , $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$ (himpunan bagian dari $Q \times \Gamma^*$)

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$. Push/insert

$\delta(q_0, a, A) = (q_1, \varepsilon)$. Pop /delete

untuk state $q \in Q$, simbol input $a \in \Sigma$, dan symbol stack $X \in \Gamma$, $\delta (Q, a, X) = (p, \alpha)$ berarti $ PDA bertransisi ke state p dan mengganti X pada stack dengan string α.

FA tidak mampu mengenali bahasa yang memiliki karakteristik x n y n. Hal ini dikarenakan FA tidak memiliki kemampuan untuk “mengingat” jumlah kemunculan x untuk digunakan pada saat memeriksa jumlah kemunculan y. Kita akan tambahkan kemampuan “mengingat” berbentuk stack pada mesin modifikasi FA sehingga dapat memecahkan permasalahan ini. Mesin ini dikenal dengan Pushdown Automata.

Dengan adanya pengingat, maka mesin harus dapat melakukan beberapa hal pada saatmelakukan transisi :1. Membaca simbol input,2. Mem-pop simbol,3. Mem-push simbol,4. Bertransisi ke state lain.Proses-proses ini bisa kita notasikan ke dalam bentuk (p, x, s; q, y) dimana :- p menyatakan current state,- x menyatakan simbol yang dibaca,- s menyatakan simbol yang di pop,- q menyatakan new state / next state,- y menyatakan simbol yang di push.Variasi dari transisi, memungkinkan mesin untuk membaca, mem-pop, ataupunmempush string kosong.

$(p , \lambda , \lambda; q , \lambda)$ yang berarti dari state p, kita akan berpindah ke state q dengan membaca, mem-pop, mem-push string kosong.Karena pada pushdown automata memiliki kemampuan transisi (p, x, s; q, y), makakita harus membuat perubahan pada diagram transisi, sehingga kemampuan itu dapatdiakomodir.State p adalah current state, dan state q adalah next state. Hal ini bisa digambarkandengan dua node yang memiliki busur berarah.

Sedang x, s; y bisa kita jadikan sebagai labelpada busur berarah kita.Dari gambar di bawah, state 1 dan 4 adalah final state. State 1 adalah initial state. Daristate 1 pada saat membaca string kosong, maka akan mempop string kosong, dan mempush #baru kemudian pindah ke state 2. Definisi Formal PDA. PDA dapat didefinisikan secara formal menjadi sextuple dalam bentuk (S, Σ, Γ, T, i,F)dimana :S adalah himpunan finite state,Σ adalah machine alphabet, Γ adalah himpunan stack symbol, T adalah himpunan transisi,i adalah initial state (bagian dari S),F adalah himpunan accepted state PDA sebagai penerima Bahasa.

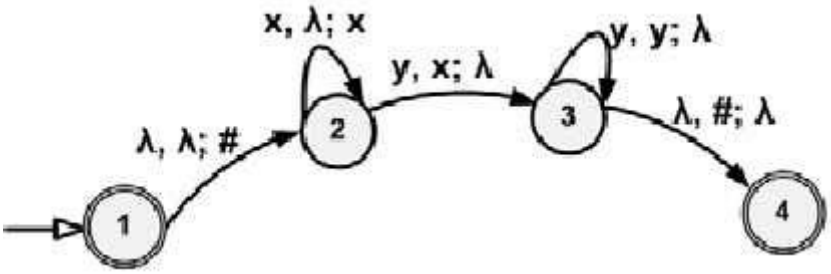
Cara hampir sama dengan di FA.Tempatkan string di tape, mesin akan membaca dari yang paling kiri. Mesin akanmulai di initial state dengan stack kosong. Mesin akan menerima jika stringmemungkinkan mesin menjangkau accepted state setelah membaca semua string.Bahasa yang diterima oleh mesin PDA M akan membentuk bahasa L(M). BahasaL(M) adalah himpunan semua string yang diterima oleh mesin M kita (bukan beberapa).

Salah satu bentuk penting dari mesin ini adalah transisi (p, x, λ; q, λ). Bentuk ini adalah

bentuk lain dari FA. Salah satu kelebihan lain adalah PDA juga bisamembaca bahasa xn yn. Sepintas kedua mesin ini sama. Tapi dimesin kedua, akan mengenali pola xm yn ,dengan m >= n. Lihat bahwa mesin 2 tidak mewajibkan stack diakhir harus kosong. Kita akanmenfokuskan diri di PDA dengan hasil akhir stack adalah kosong, karena ada teorema : “Untuk setiap PDA yang menerima string tanpa mengosongkan stack, maka ada PDA yangmenerima bahasa yang sama dengan mengosongkan stack”.

PDA yang kita pelajari juga NonDeterministik.Langkah Mengubah Menjadi PDAyang Mengosongkan Stack Hilangkan tanda initial state dari mesin, dan buat satu initial state dengan transisi dari initial

state baru ke initial lama dengan mempush simbol “#”. Hilangkan status accepted di setiap



accepted state di mesin M. Perkenalkan sebuah state p, yang memungkinkan semua oldaccepted state berpindah ke p tanpa membaca, mempop atau mempush apapun. Untuk setiap simbol x di Γ (kecuali “#”) perkenalkan transisi (p, λ, x; p, λ) Satu accepted state q dan buat transisi

(p, λ, #; q, λ).