15/03

Teori Bahasa dan Otomata

Nama : Karlina Kusuma Ningrum
NIM : 19010310
Kelas : TI19A3

Teori Otomata adalah teori mengenai mesin-mesin abstrak, dan berkaitan erat dengan teori bahasa formal. Ada beberapa hal yang berkaitan dengan Otomata, yaitu Grammar. Grammar adalah bentuk abstrak yang dapat diterima untuk membangkitkan suatu kalimat otomata berdasarkan suatu aturan tertentu.

⇒ Konsep Dasar

1. Anggota alfabet dinamakan simbol terminal.
2. Kalimat adalah deretan hingga simbol-simbol terminal.
3. Bahasa adalah kumpulan kalimat-kalimat. Anggota bahasa bisa tak hingga kalimat.
4. Simbol-simbol berikut adalah simbol terminal :
   - huruf kecil, misalnya : a, b, c
   - simbol operator, misalnya : +, −, dan *
   - simbol tanda baca, misalnya : (,), dan ;
   - string yang tercetak tebal, misalnya : if, then, dan else
5. Simbol-simbol berikut adalah simbol non terminal / variabel :
   - huruf besar, misalnya : A, B, C
   - huruf S sebagai simbol awal
   - string yang tercetak miring, misalnya : *expr*
6. Huruf Yunani melambangkan string yang tersusun atas simbol-simbol terminal atau simbol-simbol non terminal atau campuran keduanya, misalnya : α, β, dan ε
7. Sebuah produksi dilambangkan sebagai $\alpha \rightarrow \beta$, artinya : dalam sebuah derivasi dapat dilakukan penggantian simbol α dengan simbol β.
8. Derivasi adalah proses pembentukan sebuah kalimat atau sentensial. Sebuah derivasi dilambangkan sebagai $\alpha \Rightarrow \beta$.
   a. Sentensial adalah string yang tersusun atas simbol-simbol terminal atau simbol-simbol non terminal atau campuran keduanya.
   b. Kalimat adalah string yang tersusun atas simbol-simbol terminal. Kalimat adalah merupakan sentensial, sebaliknya belum tentu.

⇒ Grammar

Grammar G didefinisikan sebagai pasangan 4 tuple : Vt, Vn, S, dan P, dan ditulis dengan sebagai G(Vt, Vn, S, P), dimana :

Vt : himpunan simbol-simbol terminal (alfabet)
Vn : himpunan simbol-simbol non terminal.
SEV : simbol awal / start
P : himpunan produksi.

Teori Automata Meliputi :

1. Teori Bahasa Formal

- Bahasa berbentuk dari kombinasi simbol-simbol dengan aturan formalnya.
- Penentuan struktur sebuah bahasa diawali dengan mengenal sebuah finite set (himpunan terbatas), dimana unit fundamentalnya disebut alphabet (Σ)
- String-string yang boleh ada didalam sebuah bahasa disebut word.
- Contoh language adalah Bahasa Indonesia. Alphabet yang biasa dipakai adalah huruf, koma, dan titik.
- Simbol alphabet tidak harus alphabet huruf latin, namun dapat berisi apa saja.
- Sebuah string dimungkinkan tidak punya alphabet. String ini disebut empty string atau null string dan dilambangkan Λ.
- Bahasa tanpa word dilambangkan dengan null set ∅.
- Perbedaan antara language tanpa word dengan word yang mempunyai Λ.

  L = {x xx xxx}

  L ≠ L + {Λ}

  L = L + ∅
- Dalam language L, dapat dilakukan operasi penggabungan (concatenation) dari word yang ada menjadi word baru.
- Tidak selalu benar bila dua word digabungkan akan membentuk sebuah word baru.
- Didefinisikan suatu fungsi length untuk menghitung jumlah huruf didalam sebuah word.

  length (xxxxxx) = 6

  length (7152) = 4

  length (Λ) = 0
- Harap dipahami bahwa x^0 = Λ dan bukannya x^0 = 1 seperti aljabar.
- Didefinisikan fungsi reverse. Reverse dari suatu string adalah string yang sama tetapi ditulis dari belakang, walaupun string ini bukanlah word dalam bahasa tersebut.

  reverse (xxx) = xxx

  reverse (xxxx) = xxxx

  reverse (145) = 541
- Palindrome adalah kata, frase, nomor atau urutan lainnya dari unit yang dapat dibaca dengan cara yang sama dikedua arah.

  Σ = {a, b}

  Palindrome = {Λ, dan semua string x sedemikian sehingga reverse (x) = x}

  Maka akan diperoleh Palindrome = [Λ a b aa bb aaa aba bab bbb aaaa abba ...]
- ~~Closure dimana diketahui alphabet~~
- Closure dimana diketahui alphabet Σ, diinginkan untuk mendefinisikan sebuah language dimana string dari huruf yang ada didalam Σ adalah sebuah word, termasuk null string
- Dapat dinyatakan dengan menambah sebuah asterisk sesudah nama alphabet → $Σ^*$

2.) Regular Expression

- Reguler language adalah sebuah cara mendefinisikan language yang lebih tepat dibandingkan dengan menggunakan cara ellipsis (diwakili dengan ...) : Contoh $L_4 = \{\Lambda, x, xx, xxx, xxxx, ...\}$
- Dengan memanfaatkan closure, bila $S = \{x\}$ maka $L_4 = S^*$, dapat ditulis juga $L_4 = \{x\}^*$
- Kleene Star tidak hanya dapat diaplikasikan untuk set namun juga langsung ke alphabet.
- Ekspresi sederhana x^* akan dipakai untuk mengekspresikan pengulangan dari x (bisa juga tidak sama sekali).

  $x^* = \Lambda$ atau x atau x^2 atau x^3 atau x^4 ...
  - x^n for $n = 0, 1, 2, 3, 4 ...$

  Jadi x^* adalah string dari x yang banyaknya tidak dinyatakan secara pasti.

- Sebuah language L yang didefinisikan dari alphabet $\Sigma = \{a, b\}$

  $L = \{a, ab, abb, abbb, abbbb, ...\}$

  L = semua word yang dimulai dengan a dan diikuti oleh sejumlah b (dan mungkin tanpa b sama sekali)

  L = language (ab^*)

- Kleene star dapat diimplementasikan pada string ab seperti :

  $(ab)^* = \Lambda$, atau ab atau abab atau ababab

  $(ab)^* \neq (ab^*)$

- L_4 = language (xx^*)

  L_4 dapat dituliskan dengan notasi lain, yaitu notasi $^+$

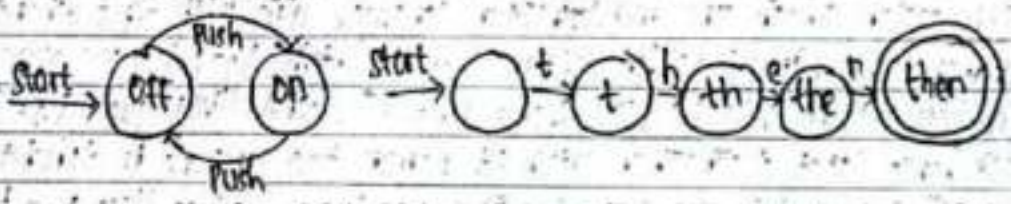
  x^+ berarti sejumlah x dalam jumlah yang selalu positif (tidak bisa 0, atau tidak bisa)

3.) Otomata Hingga

Sebuah otomata hingga adalah kumpulan dari tiga hal :

- Kumpulan terbatas (finite set) dari state (kondisi / keadaan). Satu diantaranya menjadi initial state (kondisi awal) atau start state, dan beberapa (bisa berarti tidak ada) dari antaranya dinyatakan sebagai final state.
- Himpunan alphabet Σ berisi beberapa huruf, dimana string-string berasal dari alphabet akan dibaca huruf demi huruf.
- Kumpulan terbatas dari transition yang menyatakan untuk tiap state dan tiap huruf yang dibaca ke state mana pengalaman dilanjutkan.
- Otomata hingga adalah suatu model yang dapat diterapkan pada berbagai jenis perangkat keras (hardware) dan perangkat lunak (software).
- Penerapan-penerapan dari otomata hingga adalah :
  1. Perangkat lunak yang digunakan untuk merancang dan memantau perilaku rangkaian digital.
  2. Lexical analyzer yaitu komponen kompiler yang bertugas memecah teks-teks input menjadi logical unit seperti identifiers, keyword dan punctuation.
  3. Perangkat lunak untuk mencari dokumen teks yg jumlah halamannya luar biasa banyak guna menemukan kesamaan Foto Afrize dan bentuk-bentuk lain.

• Contoh Otomata Hingga



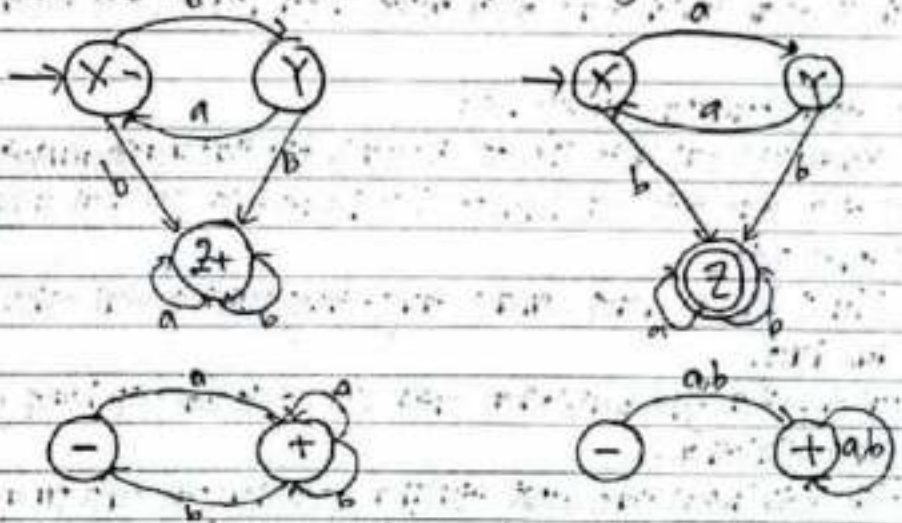
• Contoh lain, bila string akan diumpankan ke FA tersebut. Harapan pengalaman mencapai pada state 2 (final state). Jadi, string akan termasuk word dalam bahasa yang didefinisikan oleh otomata tersebut.

• Tidak sulit menerka word apa saja yang diterima oleh FA tersebut, yaitu stringnya harus berisi minimal sebuah b agar mencapai state 2. Dari transition rule tersebut, dapat dibuatkan sebuah transition table seperti dibawah ini :

	a	b
Start x	y	t
y	x	q
final q	7	z

• Otomata hingga dapat juga digambarkan dalam bentuk graf yang disebut berbentuk transition diagram.

Tanda - untuk start state dan + untuk final state. Praktik lain, start state memakai panah dan final state memakai lingkaran ganda.

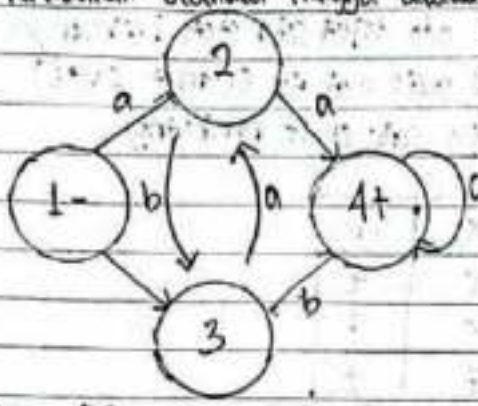


• 2 busur atau lebih yang berasal dari state yang sama dan menuju ke state yang sama pula dapat disatukan seperti gambar diatas. Jadi language yang diterima oleh mesin diatas adalah :

$(a+b)(a+b)^* = (a+b)^+$

• Ada kemungkinan sebuah otomata hingga tidak akan menerima language apapun

• Perhatikan Otomata Hingga dibawah ini :



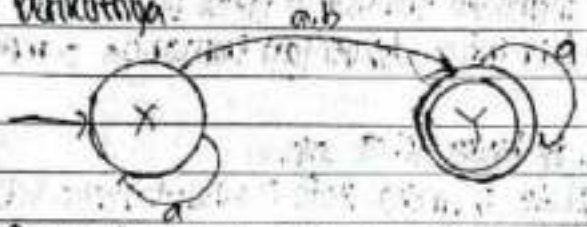
transisi bila diberi input string abaaba :
δ(1,a) = (2)
δ(2,b) = (3)
δ(3,a) = (2)
δ(2,b) = (3)
δ(3,a) = (2)

transisi bila diberi input string babbb :
δ(1,b) = (3)
δ(3,a) = (2)
δ(2,b) = (3)
δ(3,b) = (4)
δ(4,b) = (4)

4.1) Non Deterministic Finite Automata

• Pada non deterministic fine automata (NFA), dari suatu state bisa terdapat 0, 1 atau lebih busur keluar (transisi) berlabel simbol input yang sama.

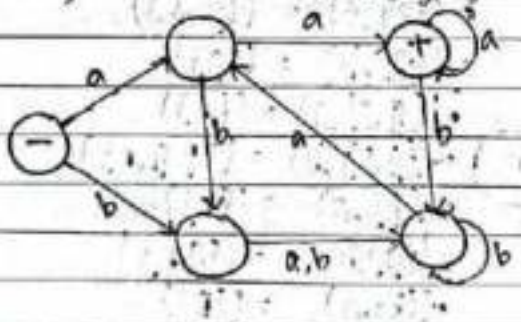
• Pada setiap pasangan state-input, dapat memiliki 0 (nol) atau lebih pilihan untuk state berikutnya.

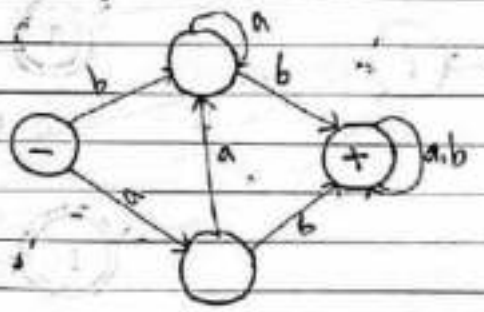


Transition table

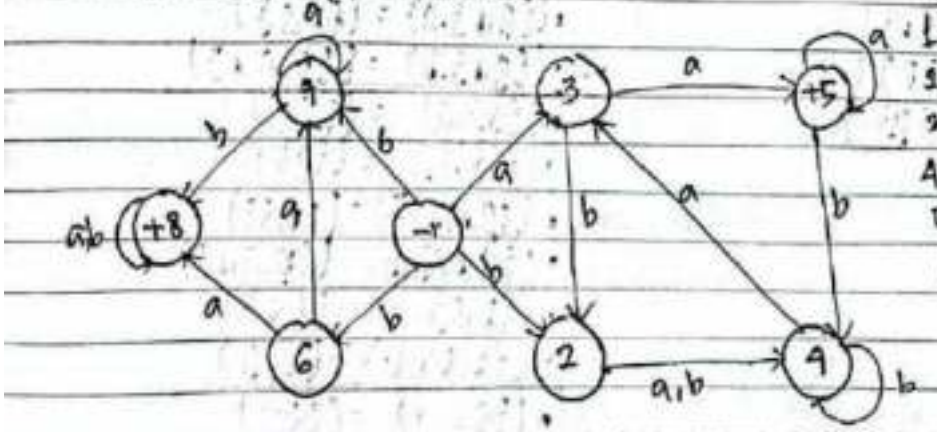
	a	b
Start x	{x,y}	{y}
Final y	{y}	∅

• Contoh berikut dapat ditemukan sebuah NFA yang merupakan gabungan dari 2 buah FA, yaitu dengan cara menambah start state. Dua FA berikut ini (FA1 dan FA2) masing-masing menerima language yang didefinisikan oleh regular expression r_1 dan r_2





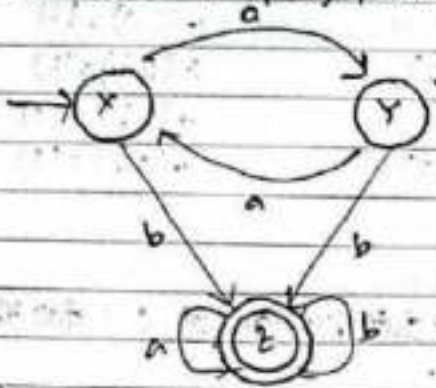
• NFA yang akan dibuat berikut bisa mendefinisikan kedua language yang diterima oleh masing-FA terlihat seperti berikut ini :



a : 1. abbab
2. baabb
3. babbaaa
4. aababba
5. aaa bbabbaaa
6. aabbbaaabaa

5) Deterministic Finite Automata

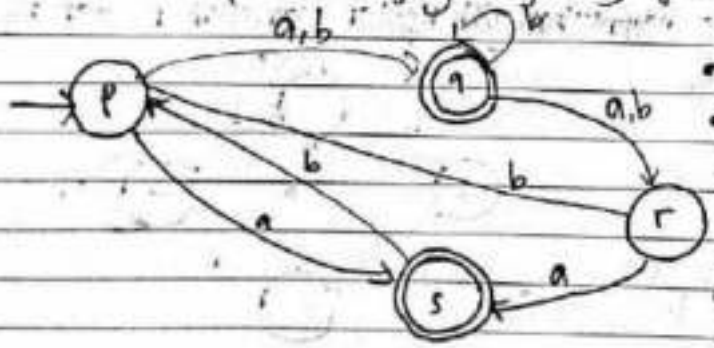
- Pada Deterministic Finite Automata, dari satu state ada tepat satu state berikutnya untuk setiap simbol masukan yang diterima. Apapun state saat itu (current state) atau masukan/inputnya, selalu dapat satu dan hanya satu state berikutnya.



	a	b
Start X	Y	Z
Y	X	Z
Final Z	Z	Z

6) Konversi dari NFA ke DFA

- Mulai dari state awal NFA, kemudian mengikuti transisinya untuk membentuk state-state baru, untuk setiap state yang terbentuk dikuti lagi transisinya sampai tercover semua.
- Jika state baru yang terbentuk sama cukup ditulis sekali saja.
- Jika state baru yang terbentuk adalah state ∅, maka state ∅ tersebut harus tetap digambarkan sebagai sebuah state.
- Semua state pada DFA yang mengandung final state NFA akan menjadi final state.

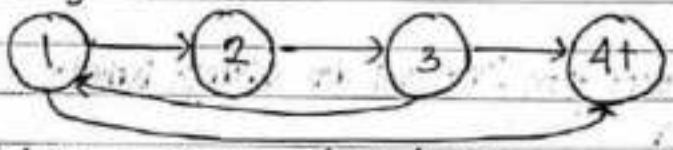


	a	b
Start p	{q,s}	{q}
Final q	{r}	{q,r}
r	{s}	{p}
Final s	∅	{p}

- ({p},a) = ({q,s})
- ({p},b) = ({q})
- ({q,s},a) = ({r})
- ({q,s},b) = ({p,q,r})
- ({q},a) = ({r})
- ({q},b) = ({q,r})
- ({r},a) = ({s})
- ({r},b) = ({p})
- ({p,q,s},a) = ({q,r,s})
- ({q,r},a) = ({r,s})
- ({q,r},b) = ({p,q,r})
- ({s},a) = (∅)
- ({s},b) = ({p})
- ({q,r,s},a) = ({r,s})
- ({q,r,s},b) = ({p,q,r})
- ({r,s},a) = ({s})
- ({r,s},b) = ({p})
- (∅,a) = (∅)
- (∅,b) = (∅)

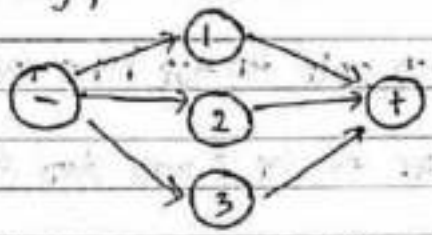
7.1 Transition Graph

- Kumpulan dari 3 hal:
  1. Finite set dari state dengan minimal satu diantaranya dinyatakan sebagai start state (-) dan beberapa (bisa juga tidak ada) dinyatakan sebagai final state (+)
  2. Sebuah alphabet Σ berisi huruf-huruf yang akan membentuk input string.
  3. Finite set dari transition yang menunjukkan arah perpindahan dari satu state ke state lain bila dibaca sebuah substring dari input (termasuk juga pembacaan string $\wedge$ atau ε)

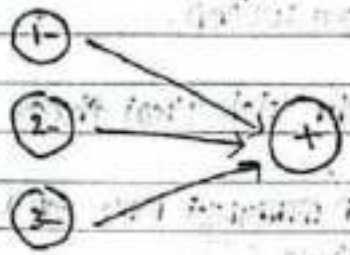


Perhatikan Transition Graph diatas:

- String (abb)(∧)(aa)(b) akan diterima oleh TG diatas.
- String abba, abbaaabba dan b adalah 3 dari beberapa string yang juga termasuk diterima oleh TG diatas.
- Sebuah edge atau busur antara state 2 dan state 3 berlabel ∧ berarti dapat berpindah dari state 2 ke state 3 tanpa harus melakukan pembacaan terhadap input.
- TG juga memungkinkan dibuat dengan lebih dari sebuah edge/busur berlabel ∧.
- TG juga memungkinkan dibuat dengan lebih dari sebuah edge/busur berlabel ∧.



- Automaton mempunyai lebih dari sebuah start state.



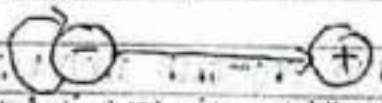
- Perlu disadari bahwa sebuah FA juga sebuah TG, namun tidak sebaliknya.
- Lihat TG dibawah ini:

Gambar disamping adalah sebuah TG yang tidak menerima string apapun termasuk juga ∧, karena tidak adanya final state.

- Lihat mesin dibawah ini:

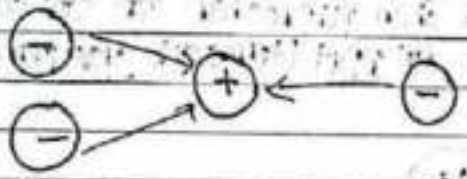
Hanya menerima string ∧. String apapun tidak dapat mencapai final state, karena tidaknya edge/busur yg menuju ke final state.

• Perhatikan TG dibawah ini :

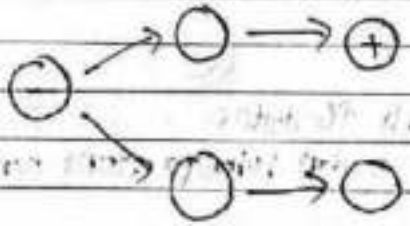


mesin diatas akan menerima semua word yang diakhiri dengan huruf b'. Reguler expression untuk language ini adalah $(a+b)^* b$.

• Mesin dibawah ini hanya menerima word Λ, baa, dan abba



• TG dibawah ini menerima word yang huruf awal dan akhirnya berbeda



8.1) Finite Automata dengan Output (Mesin Moore)

• Sebuah finite set dari stage $g_0, g_1, g_2, g_3, \ldots$ dimana g_0 adalah start state.

• Alphabet Σ berisi huruf-huruf yang akan membentuk input string
$\Sigma = \{a, b, c, \ldots\}$

• Alphabet dari karakter yang akan menjadi output
$T = \{x, y, z, \ldots\}$

• Tabel transisi yang memperlihatkan untuk tiap state dan tiap huruf input, state apa yang akan dicapai.

• Tabel keluaran yang memperlihatkan karakter apa dari T yang akan dihasilkan untuk tiap state yang tercapai.

• Mesin moore tidak mendeterminasikan language dari word yang diterima, karena tiap input yang dimasukkan akan menghasilkan suatu keluaran.

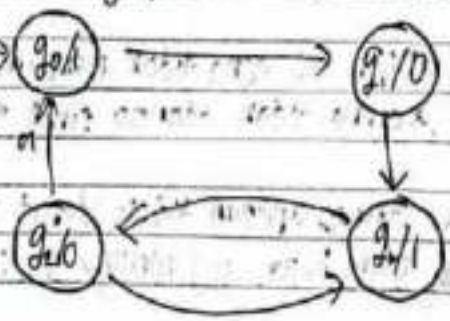
• Tidak mempunyai final state.

• Proses akan berhenti jika huruf input yang terakhir telah selesai dibaca.

• Tampilan mesin moore mirip dengan sebuah FA.

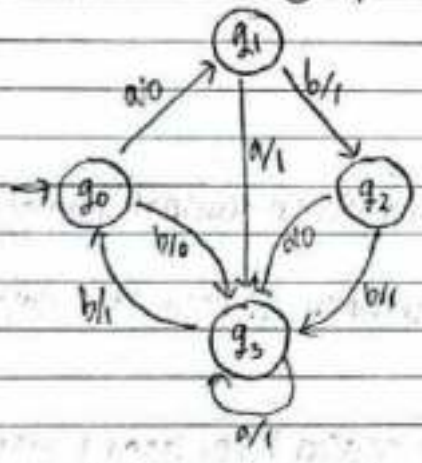
• Perbedaan terletak pada state. Sebuah state akan mempunyai nama dan karakter apa yang dihasilkan dengan pemisahnya garis miring (/).

• Bentuk grafis dari mesin moore tersebut adalah :

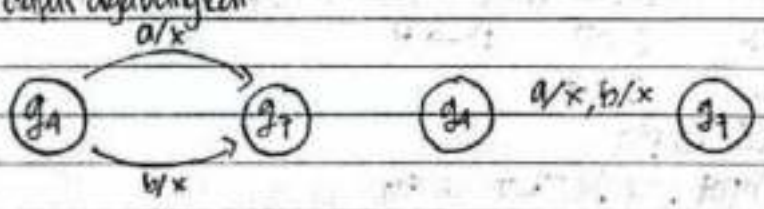


9.1) Finite Automata dengan Output (mesin Moore)

- sebuah finite set dari state $q_0, q_1, q_2, q_3, \ldots$ dimana q_0 adalah start state.
- Alphabet Σ berisi huruf-huruf yang akan membentuk input string.
  $\Sigma = \{a, b, c, \ldots\}$
- Alphabet dari karakter yang akan menjadi output $T = \{x, y, z, \ldots\}$
- Tabel transisi yang memperlihatkan untuk tiap state dan tiap huruf input, state apa yang akan dicapai.
- Tiap edge/busur dari tabel dalam bentuk i/o, dimana i adalah huruf yang akan dibaca dan o adalah karakter yang dicetak.
- Jumlah karakter yang dihasilkan akan sama banyak dengan jumlah huruf dari input.
- Mesin ini tidak mendefinisikan language yang diterima, sehingga tidak memiliki final state.
- Berbeda dengan mesin moore, mesin meany tidak menghasilkan apapun pada saat awal
- Contoh mesin meany apabila diberi inputan aabb?



- Bila terdapat 2 edge/busur yang menuju kesebuah state, maka kedua edge/busur itu dapat digabungkan



9.3) Context Free Grammer (CFG)

- Sebuah alphabet Σ dari huruf-huruf yang disebut terminal, dimana dimasukkan akan diterima string yang merupakan word.
- Himpunan dari simbol-simbol yang disebut nonterminal, dimana satu dari antaranya adalah S yang berarti mulai.
- Finite set dari production dalam bentuk:
  sebuah nonterminal → finite string dari terminal dan atau nonterminal
- Dimana string dari terminal dan atau nonterminal dapat berisi:
  * terminal saja * nonterminal saja * kombinasi terminal dan nonterminal * empty string

- Harus ada minimal sebuah nonterminal S terletak disebelah kiri.
- Agar tidak terjadi kerancuan antara terminal dan nonterminal, maka selalu digunakan huruf kecil untuk terminal dan huruf kapital untuk nonterminal.

III) Penyederhanaan Context Free Grammar

- Untuk melakukan pemotongan sehingga tidak menghasilkan pohon penurunan yang memiliki kerumitan yang tak perlu atau aturan produksi yang tidak berarti.
- Ada 3 langkah penyederhanaan :
  1. penghilangan produksi useless
  2. penghilangan produksi unit
  3. Penghilangan produksi empty

1.) Penghilangan produksi weless

- produksi yang memuat simbol variable yang tidak memiliki penurunan yang menghasilkan terminal.
- produksi yang tidak akan pernah dicapai dengan penurunan apapun dari simbol awal
- Contoh :

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

  $A \rightarrow Ada$

  $B \rightarrow BBb \mid a$

- Simbol variable A tidak memiliki penurunan yang menuju terminal, sehingga bisa dihilangkan.
- Akibatnya aturan $S \rightarrow Abd$ tidak memiliki penurunan, sehingga bisa dihilangkan juga.

2.) Penghilangan produksi Unit

- Produksi dimana ruas kiri dan kanan aturan produksi hanya berupa 1 simbol variabel, misalnya $A \rightarrow B$
- penyederhanaan ini dilakukan dengan melakukan penggantian aturan produksi unit
- Contoh : $S \rightarrow Sb$ $C \rightarrow D$ $D \rightarrow dd$

  $S \rightarrow C$ $C \rightarrow Ef$

3.) Penghilangan produksi Empty

- Produksi yang dapat menghasilkan empty
- Contoh : $S \rightarrow bcAd$

  $A \rightarrow \varepsilon$

Pada kasus ini, simbol variabel A dapat menuju empty dan hanya satu penurunan dari simbol variabel A maka simbol variabel A dapat dihilangkan.