Nama : Aktar Galih Saputra
NIM 19010309 3
Kelas 19TIA3

Teori Bahasa dan Otomata

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 menangkan 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 himpunan kalimat-kalimat Anggota bahasa bisa tak hingga kalimat
  4. Simbol berikut merupakan simbol terminal :
     * Huruf kecil
     * 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
     * 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. Secara produksi dilambangkan sebagai $\alpha \rightarrow \beta$ artinya dalam sebuah derivasi dapat diakukan penggantian simbol α dengan simbol β
  8. Derivasi adalah proses pembentukan sebuah kalimat atau sentensial. Sebuah derivasi dilambangkan sebagai $\alpha \rightarrow \beta$
  9. Sentensial adalah string yang tersusun atas simbol-simbol terminal atau simbol-simbol non terminal atau campuran keduanya.
  10. Kalimat adalah string yang tersusun atas simbol-simbol terminal. Kalimat adalah sentensial, sebaliknya belum tentu.

* Grammar

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

Vt	Himpunan simbol-simbol terminal (alfabet)
Vn	Himpunan simbol-simbol non terminal
S ∈ V	Simbol awal (start
P	Himpunan produksi

Teori Automata meliputi :

1. Teori Bahasa formal
2. Regular Expression
3. Otometa Hingga.
4. Non deterministik finite automata
5. Deterministik finite automata.
6. Konversi dari NFA ke DFA
7. Transition graph
8. mesin moore.
9. mesin mealy
10. free grammar
11. Penyederhanaan FG

a). Teori bahasa formal

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

  L = { x . xx . xxx }

  L = L + {Λ}

  L = L + θ

* 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 fungsi length untuk menghitung jumlah huruf didalam sebuah word

  length (xxxxxx) = 6

  length (7152) = 4.

  length (Λ) = 0

* Harap dipahami bahwa $x^0 = \Lambda$ dan bukannya $x^0 = 1$ seperti aljabar.
* Didefinisikan fungsi reverse, Reverse dari suatu string adalah string yang sama tetapi berurutan dari belakang, walaupun string ini bukanlah word dalam bahasa tsb

  reverse (xxx) = xxx

  reverse (xxxx) = xxxx

  reverse (145) = 541

* Palindrome adalah kata, frase, nomor atau untaian lainya dan unit yang dapat dibaca dengan cara yang sama dikedua arah. $\Sigma = \{a, b\}$

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

  maka akan diperoleh palindrome = { Λ a b aa bb aaa abb bab bbb aaaa abba ...

* Closure dimana diketahui alphabet Σ dinginkan untuk mendefinisikan sebuah language dimana string dari huruf yang ada didalam Σ adalah sebuah word termasuk non string
* Dapat dirotasikan dengan menambah sebuah asterik, sesudah nama alfabet, $\Rightarrow \Sigma^*$

b) Reguler expression.

* Reguler language adalah sebuah cara mendeterministikan language yang lebih tepat dibandingkan dengan menggunakan cara eliptas (diakhiri dengan -,,,) Contoh $L_1 = \{\Lambda\ x\ xx\ xxx\ xxxx \ldots\}$
* Dengan menerapkan closure, bila $S = \{x\}$ maka $L_1 = S^*$. dapat ditulis juga $= \{x\}^*$
* 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 $n = 0, 1, 2, 3, 9, \ldots$

  jadi x^* adalah string dari x yang banyaknya tidak dinyatakan secara pasti
* Sebuah language L yang dideterministikan 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_1 = language (xx^*)

  L_1 dapat dideskripsikan dengan notasi lain yaitu notasi $^+$

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

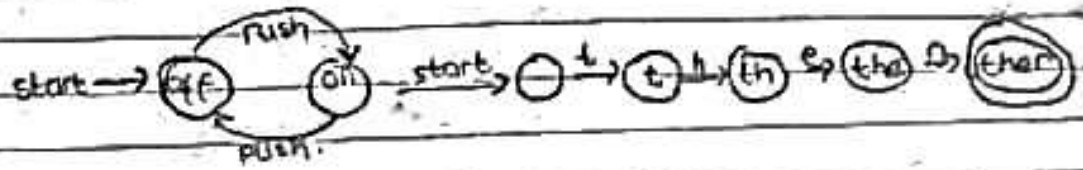
c). Otomata hingga.

Sebuah otomata hingga adalah himpunan dari tiga hal:

* Kumpulan terbatas (finite set) dari state (kondisi/keadaan) satu diantaranya akan 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 bentukan dari alphabet akan dibaca huruf demi huruf.
* Kumpulan terbatas dari transition yang menjelaskan untuk tiap state dan tiap huruf yang dibaca ke state mana perjalanan 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 komplek yang bertugas memecah teks-teks input menjadi logical unit seperti identifier, keyword dan punctuation.
  3. Perangkat lunak untuk memindai dokumen teks yang jumlah halamanya luar biasa banyak guna menemukan kesalahan kata, frase dan bentuk-bentuk lainya.

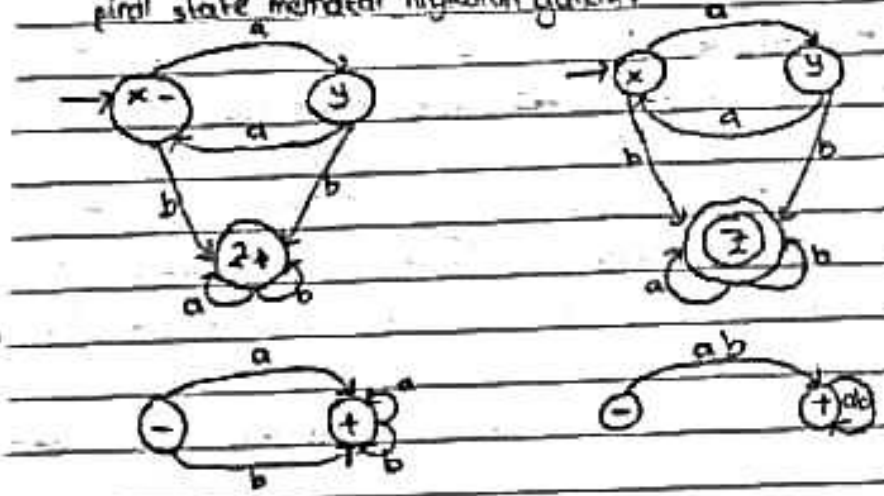
* Contoh otomata hingga.



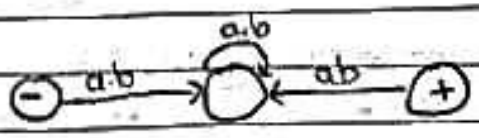
* Contoh lain, bila string abba diumpankan ke FA tersebut, hasilnya penjalanan mencapai pada state z (final state). Jadi string abba termasuk word dalam bahasa yang didefinisikan oleh otomata tersebut.
* Tidak sulit menerka word apa saja yang diterima oleh FA tersebut, yaitu stringnya harus berisi minimal sebuah b agar mencapai state z. Dari transition rule tersebut, dapat dibuatkan transition table seperti dibawah ini.

	a	b
start x	y	z
y	x	z
final q	z	z

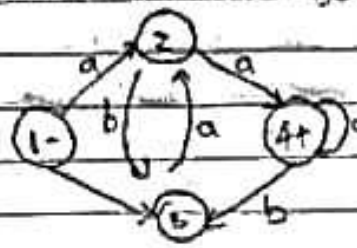
* Otomata hingga dapat juga digambarkan dalam bentuk grafis yang disebut berikut transition diagram.
* Tanda - untuk start state, + untuk final state. Bentuk lain, start state memakai panah dan final state memakai lingkaran ganda.



* 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 otomatu hingga dibawah ini:

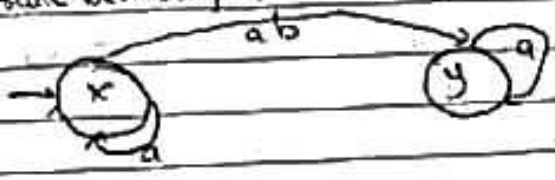


transisi bila diberi input string ababa.
* (1,a) = (2)
• (2,b) = (3)
• (3,a) = 2
• (2,b) = 3 x
• (3,a) = 2.

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

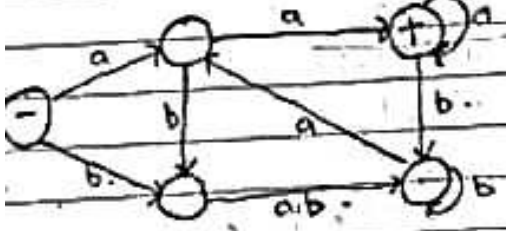
B1. Non deterministic finite automata

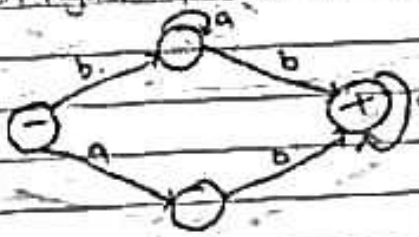
• Pada non deterministic fine automata (NFA) dari suatu state bisa terdapat 0, 1 atau lebih busur keluar (transisi) berlabel simbol input yang sama.
• Pada setiap pasangan state - input, dapat memiliki 0 (nol) atau lebih pilihan untuk state berikutnya.



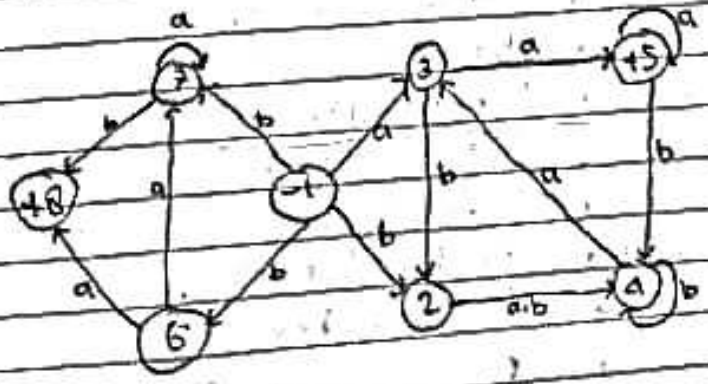
	a	b
start x	{x,y}	{y}
x or y	{y}	∅

• Contoh berikut diperhatikan sebuah NFA yang merupakan gabungan dari 2 buah FA yaitu dengan cara menumpuk start state. 2 FA berikut ini (FA_1 dan FA_2) masing-masing menerima language yang didefinisikan oleh regular expression r_1 dan r_2.





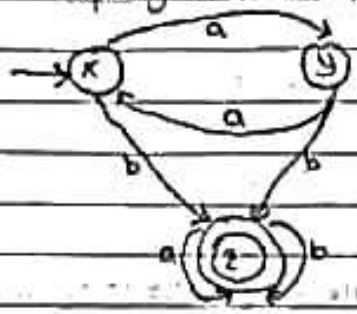
• NFA gabungan dibuat untuk bisa mendeskripsikan kedua language yang diterima oleh masing-masing FA terlihat seperti berikut ini:



1. abbab.
2. baabb.
3. babbaaa
4. aababba
5. aaabbabbaaa
6. aabbbaaabaa

E). Deterministik Finite Automata

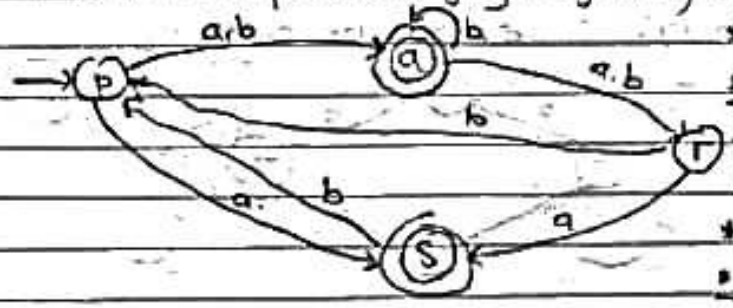
- Pada deterministic finite automata dari satu state ada tepat satu state berikutnya untuk setiap simbol masukan yang diterima. Apapun state saat itu (current state) atau masukan / inputnya, selalu dapat satu dan hanya satu state berikutnya.



	:a	b
start x	y	z
y	x	z
final z	z	z

F). 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 teracuer 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.



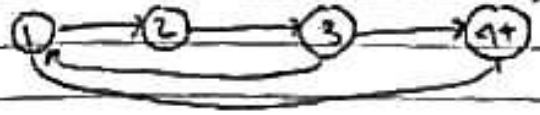
	a	b
start p	{q,s}	{q}
final q	{r}	{q,r}
r	{s}	{p}
final s	{∅}	{p}

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

6). Transtion Graph

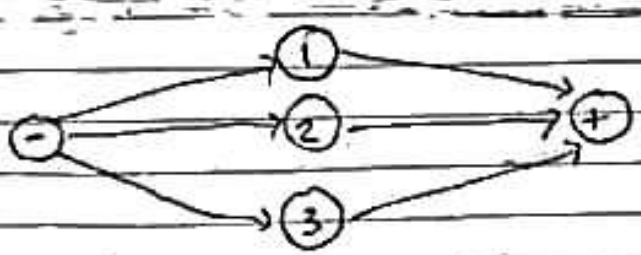
Kumpulan dari 3 hal :

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 menunjukan arah perpindahan dari satu state ke state lain bila dibaca sebuah substring dari input (termasuk juga pembacaan string Λ atau ε)

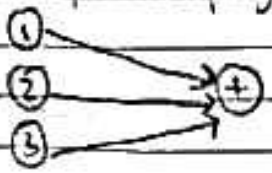


Perhatikan Transtion Graph diatas :

- String (abb) (Λ) (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 Λ 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 Λ



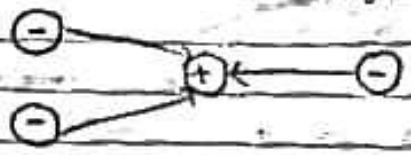
Abupun mempunyai lebih dari sebuah start state.



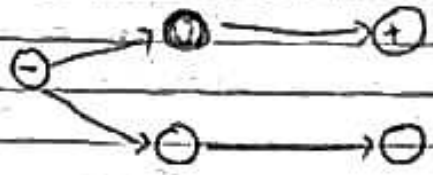
- Perlu disadari bahwa sebuah FA juga sebuah TG namun tidak sebaliknya
- Lihat TG dibawah ini
  - (-) Gambar disamping adalah sebuah TG yang tidak menerima string apapun termasuk juga Λ, karena tidak adanya final state
- Lihat mesin dibawah ini :
  - (+) Hanya menerima string Λ string apapun tidak dapat mencapai final state, karena tidak edge / busur
- Perhatikan TG dibawah ini :

mesin disamping akan menerima semua word yang diakhiri dengan huruf b regular expression untuk language ini adalah (a+b)* b.

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

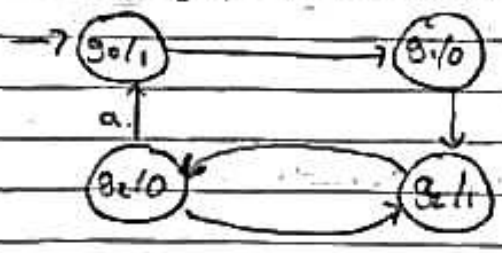


→ TG dibawah ini menerima word yang huruf awal dan akhirnya berbeda.



H). Finite Automata dengan output (Mesin Moore)

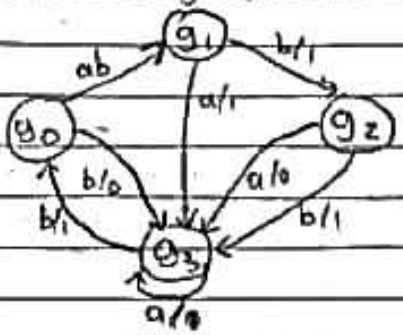
* Sebuah finite set dari stage $g_0, g_1, g_2, g_3, \ldots$ dimana g_0 adalah start state
* Alphabet Σ berisi huruf-huruf yang akan membentuk input string $\Sigma = \{a, b, c, \ldots\}$
* Alphabet dari karakter yang akan menjadi output $T = \{x, y, z, \ldots\}$
* Tabel transisi yang memperlihatkan untuk tiap state dan tiap huruf input, state apa yang akan dilakukan.
* Tabel keluaran yang memperlihatkan karakter apa dari T yang akan dihasilkan untuk tiap state yang tercapai
* Mesin moore tidak mendefinisikan 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 dikeluarkan dengan pemisah nya garis miring (/)
* Bentuk grafik dari mesin moore tersebut adalah ..



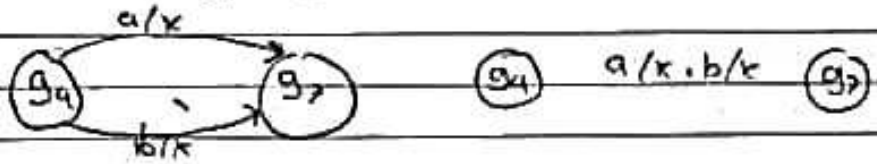
I). Finite Automata dengan output (mesin mealy)

* Sebuah finite set dari state $g_0, g_1, g_2, g_3, \ldots$ 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 yg memperlihatkan untuk tiap state dan tiap huruf input, state apa yang akan dicapai.
* Tiap edge/busur diberi label dalam bentuk i/o dimana i disebut 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
* Perbedaan dengan moore, mesin mealy tidak menghasilkan apapun pada start awal
contoh: mesin mealy apabila diberi inputan aaabb?



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



J) Context free grammer (FG)
* Sebuah alphabet ε dari huruf-huruf yang disebut terminal, dimana dan padanya akan dibentuk string yang merupakan word.
* Himpunan dari simbol-simbol yang disebut non terminal, dimana salu dari antaranya adalah s' yang berarti mulai.
* Finite set dari production dalam bentuk:
Sebuah nonterminal → finite string dari terminal dan atau non terminal.
* Dimana string dari terminal dan atau nonterminal dapat berisi
1. terminal saja 2. non terminal saja 3. kombinasi terminal dan nonterminal & empty string.
* Harus ada minimal sebuah nonterminal s terletak disebelah kiri.
* Agar tidak terjadi kekacauan antara terminal dan nonterminal, maka huruf kecil untuk terminal dan huruf kapital untuk nonterminal.

K). Penyederhanaan context free grammer
* Untuk melakukan pembabasan 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.