TEORI BAHASA DAN AUTOMATA

I. PENDAHULUAN

Teori Bahasa

Teori bahasa membicarakan bahasa formal (*formal language*), terutama untuk kepentingan perancangan kompilator (*compiler*) dan pemroses naskah (*text processor*). Bahasa formal adalah kumpulan *kalimat*. Semua kalimat dalam sebuah bahasa dibangkitkan oleh sebuah tata bahasa (*grammar*) yang sama. Sebuah bahasa formal bisa dibangkitkan oleh dua atau lebih tata bahasa berbeda. Dikatakan bahasa formal karena grammar diciptakan mendahului pembangkitan setiap kalimatnya. Bahasa manusia bersifat sebaliknya; grammar diciptakan untuk meresmikan kata-kata yang hidup di masyarakat. Dalam pembicaraan selanjutnya 'bahasa formal' akan disebut 'bahasa' saja.

Automata

Automata adalah mesin abstrak yang dapat mengenali (*recognize*), menerima (*accept*), atau membangkitkan (*generate*) sebuah kalimat dalam bahasa tertentu.

Beberapa Pengertian Dasar

- Simbol adalah sebuah entitas abstrak (seperti halnya pengertian *titik* dalam geometri). Sebuah huruf atau sebuah angka adalah contoh simbol.
- String adalah deretan terbatas (*finite*) simbol-simbol. Sebagai contoh, jika *a*, *b*, dan *c* adalah tiga buah simbol maka *abcb* adalah sebuah string yang dibangun dari ketiga simbol tersebut.
- Jika *w* adalah sebuah string maka panjang string dinyatakan sebagai $|w|$ dan didefinisikan sebagai cacahan (banyaknya) simbol yang menyusun string tersebut. Sebagai contoh, jika $w = abcb$ maka $|w| = 4$.
- String hampa adalah sebuah string dengan nol buah simbol. String hampa dinyatakan dengan simbol ε (atau ^) sehingga $|\varepsilon| = 0$. String hampa dapat dipandang sebagai simbol hampa karena keduanya tersusun dari nol buah simbol.
- Alfabet adalah hinpunan hingga (*finite set*) simbol-simbol

Operasi Dasar String

Diberikan dua string : $x = abc$, dan $y = 123$

- Prefik string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *nol* atau lebih simbol-simbol paling belakang dari string *w* tersebut.
  Contoh : *abc*, *ab*, *a*, dan ε adalah semua Prefix(*x*)
- ProperPrefix string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *satu* atau lebih simbol-simbol paling belakang dari string *w* tersebut.
  Contoh : *ab*, *a*, dan ε adalah semua ProperPrefix(*x*)
- Postfix (atau Sufix) string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *nol* atau lebih simbol-simbol paling depan dari string *w* tersebut.
  Contoh : *abc*, *bc*, *c*, dan ε adalah semua Postfix(*x*)
- ProperPostfix (atau PoperSufix) string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *satu* atau lebih simbol-simbol paling depan dari string *w* tersebut.
  Contoh : *bc*, *c*, dan ε adalah semua ProperPostfix(*x*)
- Head string *w* adalah simbol paling depan dari string *w*.
  Contoh : *a* adalah Head(*x*)

- Tail string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan simbol paling depan dari string *w* tersebut.
  Contoh : *bc* adalah Tail(*x*)
- Substring string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *nol* atau lebih simbol-simbol paling depan dan/atau simbol-simbol paling belakang dari string *w* tersebut.
  Contoh : *abc*, *ab*, *bc*, *a*, *b*, *c*, dan ε adalah semua Substring(*x*)
- ProperSubstring string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *satu* atau lebih simbol-simbol paling depan dan/atau simbol-simbol paling belakang dari string *w* tersebut.
  Contoh : *ab*, *bc*, *a*, *b*, *c*, dan ε adalah semua Substring(*x*)
- Subsequence string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *nol* atau lebih simbol-simbol dari string *w* tersebut.
  Contoh : *abc*, *ab*, *bc*, *ac*, *a*, *b*, *c*, dan ε adalah semua Subsequence(*x*)
- ProperSubsequence string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *satu* atau lebih simbol-simbol dari string *w* tersebut.
  Contoh : *ab*, *bc*, *ac*, *a*, *b*, *c*, dan ε adalah semua Subsequence(*x*)
- Concatenation adalah penyambungan dua buah string. Operator concatenation adalah *concate* atau tanpa lambang apapun.
  Contoh : concate(*xy*) = *xy* = *abc123*
- Alternation adalah pilihan satu di antara dua buah string. Operator alternation adalah *alternate* atau $|$.
  Contoh : alternate(*xy*) = $x \mid y$ = *abc* atau *123*
- Kleene Closure : $x^* = \varepsilon \mid x \mid xx \mid xxx \mid \ldots = \varepsilon \mid x \mid x^2 \mid x^3 \mid \ldots$
- Positive Closure : $x^+ = x \mid xx \mid xxx \mid \ldots = x \mid x^2 \mid x^3 \mid \ldots$

Beberapa Sifat Operasi

- Tidak selalu berlaku : *x* = Prefix(*x*)Postfix(*x*)
- Selalu berlaku : *x* = Head(*x*)Tail(*x*)
- Tidak selalu berlaku : Prefix(*x*) = Postfix(*x*) atau Prefix(*x*) ≠ Postfix(*x*)
- Selalu berlaku : ProperPrefix(*x*) ≠ ProperPostfix(*x*)
- Selalu berlaku : Head(*x*) ≠ Tail(*x*)
- Setiap Prefix(*x*), ProperPrefix(*x*), Postfix(*x*), ProperPostfix(*x*), Head(*x*), dan Tail(*x*) adalah Substring(*x*), tetapi tidak sebaliknya
- Setiap Substring(*x*) adalah Subsequence(*x*), tetapi tidak sebaliknya
- Dua sifat aljabar concatenation :
  - Operasi concatenation bersifat asosiatif : $x(yz) = (xy)z$
  - Elemen identitas operasi concatenation adalah ε : $\varepsilon x = x\varepsilon = x$
- Tiga sifat aljabar alternation :
  - Operasi alternation bersifat komutatif : $x \mid y = y \mid x$
  - Operasi alternation bersifat asosiatif : $x \mid (y \mid z) = (x \mid y) \mid z$
  - Elemen identitas operasi alternation adalah dirinya sendiri : $x \mid x = x$
- Sifat distributif concatenation terhadap alternation : $x\,(y \mid z) = xy \mid xz$
- Beberapa kesamaan :
  - Kesamaan ke-1 : $(x^*)^* = (x^*)$
  - Kesamaan ke-2 : $\varepsilon \mid x^+ = x^+ \mid \varepsilon = x^*$
  - Kesamaan ke-3 : $(x \mid y)^* = \varepsilon \mid x \mid y \mid xx \mid yy \mid xy \mid yx \mid \ldots$ = semua string yang merupakan concatenation dari nol atau lebih *x*, *y*, atau keduanya.

II. GRAMMAR DAN BAHASA

Konsep Dasar

1. Dalam pembicaraan grammar, anggota alfabet dinamakan simbol terminal atau token.
2. Kalimat adalah deretan hingga simbol-simbol terminal.
3. Bahasa adalah himpunan kalimat-kalimat. Anggota bahasa bisa tak hingga kalimat.
4. Simbol-simbol berikut adalah simbol terminal :
   - huruf kecil awal alfabet, 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 :
   - huruf besar awal alfabet, misalnya : A, B, C
   - huruf S sebagai simbol awal
   - string yang tercetak miring, misalnya : *expr* dan *stmt*.
6. Huruf besar akhir alfabet melambangkan simbol terminal atau non terminal, misalnya : X, Y, Z.
7. Huruf kecil akhir alfabet melambangkan string yang tersusun atas simbol-simbol terminal, misalnya : x, y, z.
8. Huruf yunani melambangkan string yang tersusun atas simbol-simbol terminal atau simbol-simbol non terminal atau campuran keduanya, misalnya : α, β, dan γ.
9. Sebuah produksi dilambangkan sebagai $\alpha \rightarrow \beta$, artinya : dalam sebuah derivasi dapat dilakukan penggantian simbol α dengan simbol β.
10. Simbol α dalam produksi berbentuk $\alpha \rightarrow \beta$ disebut ruas kiri produksi sedangkan simbol β disebut ruas kanan produksi.
11. Derivasi adalah proses pembentukan sebuah kalimat atau sentensial. Sebuah derivasi dilambangkan sebagai : $\alpha \Rightarrow \beta$.
12. Sentensial adalah string yang tersusun atas simbol-simbol terminal atau simbol-simbol non terminal atau campuran keduanya.
13. Kalimat adalah string yang tersusun atas simbol-simbol terminal. Jelaslah bahwa kalimat adalah kasus khusus dari sentensial.
14. Pengertian terminal berasal dari kata *terminate* (berakhir), maksudnya derivasi berakhir jika sentensial yang dihasilkan adalah sebuah kalimat (yang tersusun atas simbol-simbol terminal itu).
15. Pengertian non terminal berasal dari kata *not terminate* (belum/tidak berakhir), maksudnya derivasi belum/tidak berakhir jika sentensial yang dihasilkan mengandung simbol non terminal.

Grammar dan Klasifikasi Chomsky

Grammar G didefinisikan sebagai pasangan 4 tuple : V_T, V_N, S, dan Q, dan dituliskan sebagai G(V_T, V_N, S, Q), dimana :

V_T : himpunan simbol-simbol terminal (atau himpunan token -token, atau alfabet)

V_N : himpunan simbol-simbol non terminal

$S \in V_N$: simbol awal (atau simbol start)

Q : himpunan produksi

Berdasarkan komposisi bentuk ruas kiri dan ruas kanan produksinya ($\alpha \rightarrow \beta$), Noam Chomsky mengklasifikasikan 4 tipe grammar :

1. Grammar tipe ke-0 : Unrestricted Grammar (UG)
   Ciri : $\alpha, \beta \in (V_T \mid V_N)^*, \mid \alpha \mid > 0$
2. Grammar tipe ke-1 : Context Sensitive Grammar (CSG)
   Ciri : $\alpha, \beta \in (V_T \mid V_N)^*, 0 < \mid \alpha \mid \leq \mid \beta \mid$
3. Grammar tipe ke-2 : Context Free Grammar (CFG)
   Ciri : $\alpha \in V_N, \beta \in (V_T \mid V_N)^*$
4. Grammar tipe ke-3 : Regular Grammar (RG)
   Ciri : $\alpha \in V_N, \beta \in \{V_T, V_T V_N\}$ atau $\alpha \in V_N, \beta \in \{V_T, V_N V_T\}$
   Mengingat ketentuan simbol-simbol (hal. 3 no. 4 dan 5), ciri-ciri RG sering dituliskan sebagai : $\alpha \in V_N, \beta \in \{a, bC\}$ atau $\alpha \in V_N, \beta \in \{a, Bc\}$

Tipe sebuah grammar (atau bahasa) ditentukan dengan aturan sebagai berikut :

> *A language is said to be type-i (i = 0, 1, 2, 3) language if it can be specified by a type-i grammar but can't be specified any type-(i+1) grammar.*

Contoh Analisa Penentuan Type Grammar

1. Grammar G_1 dengan $Q_1 = \{S \rightarrow aB, B \rightarrow bB, B \rightarrow b\}$. Ruas kiri semua produksinya terdiri dari sebuah V_N maka G_1 kemungkinan tipe CFG atau RG. Selanjutnya karena semua ruas kanannya terdiri dari sebuah V_T atau string $V_T V_N$ maka G_1 adalah RG.
2. Grammar G_2 dengan $Q_2 = \{S \rightarrow Ba, B \rightarrow Bb, B \rightarrow b\}$. Ruas kiri semua produksinya terdiri dari sebuah V_N maka G_2 kemungkinan tipe CFG atau RG. Selanjutnya karena semua ruas kanannya terdiri dari sebuah V_T atau string $V_N V_T$ maka G_2 adalah RG.
3. Grammar G_3 dengan $Q_3 = \{S \rightarrow Ba, B \rightarrow bB, B \rightarrow b\}$. Ruas kiri semua produksinya terdiri dari sebuah V_N maka G_3 kemungkinan tipe CFG atau RG. Selanjutnya karena ruas kanannya mengandung string $V_T V_N$ (yaitu bB) dan juga string $V_N V_T$ (Ba) maka G_3 bukan RG, dengan kata lain G_3 adalah CFG.
4. Grammar G_4 dengan $Q_4 = \{S \rightarrow aAb, B \rightarrow aB\}$. Ruas kiri semua produksinya terdiri dari sebuah V_N maka G_4 kemungkinan tipe CFG atau RG. Selanjutnya karena ruas kanannya mengandung string yang panjangnya lebih dari 2 (yaitu aAb) maka G_4 bukan RG, dengan kata lain G_4 adalah CFG.
5. Grammar G_5 dengan $Q_5 = \{S \rightarrow aA, S \rightarrow aB, aAb \rightarrow aBCb\}$. Ruas kirinya mengandung string yang panjangnya lebih dari 1 (yaitu aAb) maka G_5 kemungkinan tipe CSG atau UG. Selanjutnya karena semua ruas kirinya lebih pendek atau sama dengan ruas kananya maka G_5 adalah CSG.
6. Grammar G_6 dengan $Q_6 = \{aS \rightarrow ab, SAc \rightarrow bc\}$. Ruas kirinya mengandung string yang panjangnya lebih dari 1 maka G_6 kemungkinan tipe CSG atau UG. Selanjutnya karena terdapat ruas kirinya yang lebih panjang daripada ruas kananya (yaitu SAc) maka G_6 adalah UG.

Derivasi Kalimat dan Penentuan Bahasa

Tentukan bahasa dari masing-masing gramar berikut :

1. G_1 dengan $Q_1 = \{1.\ S \rightarrow aAa,\ 2.\ A \rightarrow aAa,\ 3.\ A \rightarrow b\}$.

   Jawab :

   Derivasi kalimat terpendek :

   $S \Rightarrow aAa$ (1)
   $\Rightarrow aba$ (3)

   Derivasi kalimat umum :

   $S \Rightarrow aAa$ (1)
   $\Rightarrow aaAaa$ (2)
   …
   $\Rightarrow a^n Aa^n$ (2)
   $\Rightarrow a^n ba^n$ (3)

   Dari pola kedua kalimat disimpulkan : $L_1(G_1) = \{ a^n ba^n \mid n \geq 1\}$

2. G_2 dengan $Q_2 = \{1.\ S \rightarrow aS,\ 2.\ S \rightarrow aB,\ 3.\ B \rightarrow bC,\ 4.\ C \rightarrow aC,\ 5.\ C \rightarrow a\}$.

   Jawab :

   Derivasi kalimat terpendek :

   $S \Rightarrow aB$ (2)
   $\Rightarrow abC$ (3)
   $\Rightarrow aba$ (5)

   Derivasi kalimat umum :

   $S \Rightarrow aS$ (1)
   …
   $\Rightarrow a^{n-1}S$ (1)
   $\Rightarrow a^n B$ (2)
   $\Rightarrow a^n bC$ (3)
   $\Rightarrow a^n baC$ (4)
   …
   $\Rightarrow a^n ba^{m-1}C$ (4)
   $\Rightarrow a^n ba^m$ (5)

   Dari pola kedua kalimat disimpulkan : $L_2(G_2) = \{ a^n ba^m \mid n \geq 1, m \geq 1\}$

3. G_3 dengan $Q_3 = \{1.\ S \rightarrow aSBC,\ 2.\ S \rightarrow abC,\ 3.\ bB \rightarrow bb,$
   $4.\ bC \rightarrow bc,\ 5.\ CB \rightarrow BC,\ 6.\ cC \rightarrow cc\}$.

   Jawab :

   Derivasi kalimat terpendek 1:

   $S \Rightarrow abC$ (2)
   $\Rightarrow abc$ (4)

   Derivasi kalimat terpendek 2 :

   $S \Rightarrow aSBC$ (1)
   $\Rightarrow aabCBC$ (2)
   $\Rightarrow aabBCC$ (5)
   $\Rightarrow aabbCC$ (3)
   $\Rightarrow aabbcC$ (4)
   $\Rightarrow aabbcc$ (6)

   Derivasi kalimat terpendek 3 :

   $S \Rightarrow aSBC$ (1)
   $\Rightarrow aaSBCBC$ (1)
   $\Rightarrow aaabCBCBC$ (2)
   $\Rightarrow aaabBCCBC$ (5)
   $\Rightarrow aaabBCBCC$ (5)
   $\Rightarrow aaabBBCCC$ (5)
   $\Rightarrow aaabbBCCC$ (3)
   $\Rightarrow aaabbbCCC$ (3)
   $\Rightarrow aaabbbcCC$ (4)
   $\Rightarrow aaabbbccC$ (6)
   $\Rightarrow aaabbbccc$ (6)

   Dari pola ketiga kalimat disimpulkan : $L_3(G_3) = \{ a^n b^n c^n \mid n \geq 1\}$

Menentukan Grammar Sebuah Bahasa

1. Tentukan sebuah gramar regular untuk bahasa $L_1 = \{ a^n \mid n \geq 1\}$

   Jawab :

   $Q_1(L_1) = \{S \rightarrow aS \mid a\}$

2. Tentukan sebuah gramar bebas konteks untuk bahasa :

   L_2 : himpunan bilangan bulat non negatif ganjil

   Jawab :

   Langkah kunci : digit terakhir bilangan harus ganjil.

   Buat dua buah himpunan bilangan terpisah : genap (G) dan ganjil (J)

   $Q_2(L_2) = \{S \rightarrow J \mid GS \mid JS,\ G \rightarrow 0 \mid 2 \mid 4 \mid 6 \mid 8,\ J \rightarrow 1 \mid 3 \mid 5 \mid 7 \mid 9\}$

3. Tentukan sebuah gramar bebas konteks untuk bahasa :

   L_3 = himpunan semua identifier yang sah menurut bahasa pemrograman Pascal dengan batasan : terdiri dari simbol huruf kecil dan angka, panjang identifier boleh lebih dari 8 karakter

   Jawab :

   Langkah kunci : karakter pertama identifier harus huruf.

   Buat dua buah himpunan bilangan terpisah : huruf (H) dan angka (A)

   $Q_3(L_3) = \{S \rightarrow H \mid HT,\ T \rightarrow AT \mid HT \mid H \mid A,\ H \rightarrow a \mid b \mid c \mid \ldots,\ A \rightarrow 0 \mid 1 \mid 2 \mid \ldots\}$

4. Tentukan gramar bebas konteks untuk bahasa $L_4(G_4) = \{a^n b^m \mid n,m \geq 1,\ n \neq m\}$

   Jawab :

   Langkah kunci : sulit untuk mendefinisikan $L_4(G_4)$ secara langsung. Jalan keluarnya adalah dengan mengingat bahwa $x \neq y$ berarti $x > y$ atau $x < y$.

   $L_4 = L_A \cup L_B,\ L_A = \{a^n b^m \mid n > m \geq 1\},\ L_B = \{a^n b^m \mid 1 \leq n < m\}$.

   $Q_A(L_A) = \{A \rightarrow aA \mid aC,\ C \rightarrow aCb \mid ab\},\ Q(L_B) = \{B \rightarrow Bb \mid Db,\ D \rightarrow aDb \mid ab\}$

   $Q_4(L_4) = \{S \rightarrow A \mid B,\ A \rightarrow aA \mid aC,\ C \rightarrow aCb \mid ab,\ B \rightarrow Bb \mid Db,\ D \rightarrow aDb \mid ab\}$

5. Tentukan sebuah gramar bebas konteks untuk bahasa :

   L_5 = bilangan bulat non negatif genap. Jika bilangan tersebut terdiri dari dua digit atau lebih maka nol tidak boleh muncul sebagai digit pertama.

   Jawab :

   Langkah kunci : Digit terakhir bilangan harus genap. Digit pertama tidak boleh nol. Buat tiga himpunan terpisah : bilangan genap tanpa nol (G), bilangan genap dengan nol (N), serta bilangan ganjil (J).

   $Q_5(L_5) = \{S \rightarrow N \mid GA \mid JA,\ A \rightarrow N \mid NA \mid JA,\ G \rightarrow 2 \mid 4 \mid 6 \mid 8,\ N \rightarrow 0 \mid 2 \mid 4 \mid 6 \mid 8,$
   $J \rightarrow 1 \mid 3 \mid 5 \mid 7 \mid 9\}$

Mesin Pengenal Bahasa

Untuk setiap kelas bahasa Chomsky, terdapat sebuah mesin pengenal bahasa. Masing-masing mesin tersebut adalah :

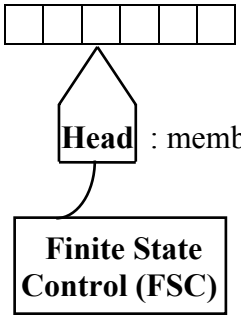
Kelas Bahasa	Mesin Pengenal Bahasa
Unrestricted Grammar (UG)	Mesin Turing (*Turing Machine*), TM
Context Sensitive Grammar (CSG)	*Linear Bounded Automaton*, LBA
Context Free Gammar (CFG)	Automata Pushdown (*Pushdown Automata*), PDA
Regular Grammar, RG	Automata Hingga (*Finite Automata*), FA

Catatan :

1. Pengenal bahasa adalah salah satu kemampuan mesin turing.
2. LBA adalah variasi dari Mesin Turing Nondeterministik.
3. Yang akan dibahasa dalam kuliah Teori Bahasa dan Automata adalah : TM (sekilas), FA, dan PDA.

III. MESIN TURING

Ilustrasi TM sebagai sebuah 'mesin':



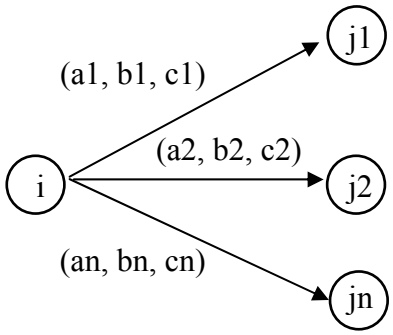
Pita TM. Terbatas di kiri. Setiap sel berisi sebuah karakter dari kalimat yang akan dikenali. Di kanan kalimat terdapat tak hingga simbol hampa.

Head : membaca dan menulisi sel pita TM, bisa bergerak ke kiri atau ke akan

FSC : otak dari TM, diimplementasikan dari algoritma pengenalan kalimat.

Ilustrasi TM sebagai sebuah graf berarah :

1. Sebagaimana graf, TM terdiri dari beberapa *node* dan beberapa *edge*. Dari satu node mungkin terdapat satu atau lebih edge yang menuju node lainnya atau dirinya sendiri.
2. Sebuah node menyatakan sebuah *stata* (*state*). Dua stata penting adalah stata awal S (*start*) dan stata penerima H (*halt*). Sesaat sebelum proses pengenalan sebuah kalimat, TM berada pada stata S. Jika kalimat tersebut dikenali maka, setelah selesai membaca kalimat tersebut, TM akan akan berhenti pada stata H.
3. Sebuah edge mempunyai 'bobot' yang dinotasikan sebagai triple : (a, b, d). a adalah karakter acuan bagi karakter dalam sel pita TM yang sedang dibaca head. Jika yang dibaca head adalah karakter a maka a akan di-*overwrite* dengan karakter b dan head akan berpindah satu sel ke arah d (kanan atau kiri).
4. Kondisi *crash* akan terjadi jika ditemui keadaan sebagai berikut :



TM sedang berada pada stata i. Jika TM sedang membaca simbol ax ≠ a1 ≠ a2 ≠ … ≠ an maka TM tidak mungkin beranjak dari stata i. Jadi pada kasus ini penelusuran (*tracing*) TM berakhir pada stata i.

Contoh :

Rancanglah sebuah mesin turing pengenal bahasa $L = \{a^n b^n \mid n \geq 0\}$.

Jawab :

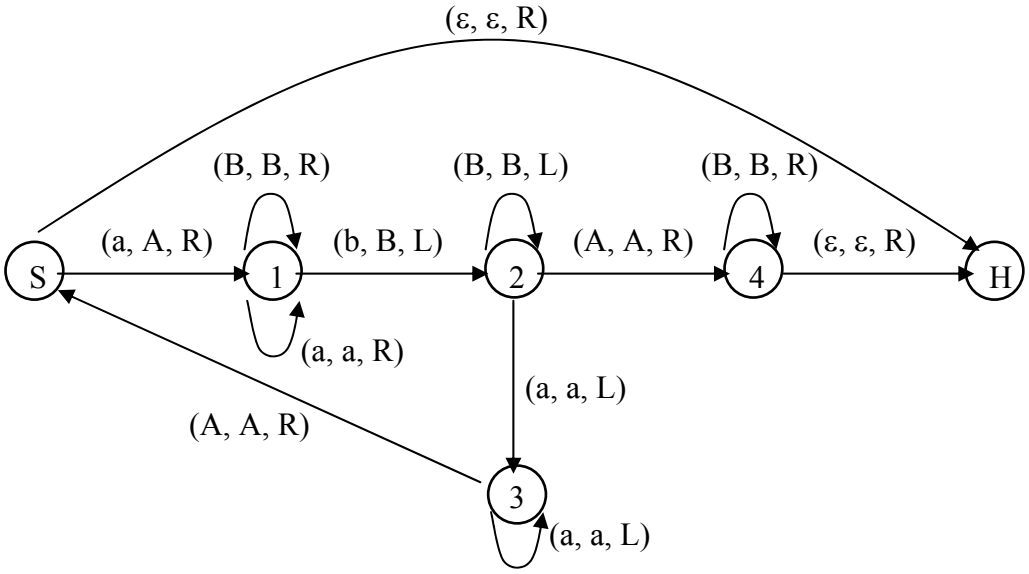
L tersebut terdiri dari 2 kelompok kalimat yaitu ε dan non-ε. Kelompok non-ε adalah : ab, aabb, aaabbb, dan seterusnya. Untuk dapat menerima kalimat ε TM harus mempunyai edge dari S ke H dengan bobot (ε ,ε , R). TM menerima kalimat-kalimat : ab, aabb, aaabbb, dan seterusnya, dengan algoritma sebagai berikut :

1. Mulai dari S, head membaca simbol a.
2. Head membaca simbol a. Tandai simbol a yang sudah dibaca tersebut, head bergerak ke kanan mencari simbol b pasangannya.
3. Head membaca simbol b. Tandai simbol b yang sudah dibaca tersebut, head bergerak ke kiri mencari simbol a baru yang belum dibaca/ditandai.
4. Ulangi langkah 2 dan 3.
5. Head sampai ke H hanya jika semua simbol a dan simbol b dalam kalimat $a^n b^n$ selesai dibaca.

Algoritma di atas lebih diperinci lagi sebagai berikut :

1. Mulai dari S, head membaca simbol a.
2. *Overwrite* a tersebut dengan suatu simbol (misalkan A) untuk menandakan bahwa a tersebut sudah dibaca. Selanjutnya head harus bergerak ke *kanan* untuk mencari sebuah b sebagai pasangan a yang sudah dibaca tersebut.
   i) Jika yang ditemukan adalah simbol a maka a tersebut harus dilewati (*tidak boleh dioverwrite*), dengan kata lain a *dioverwrite dengan* a *juga* dan head bergerak ke *kanan*.
   ii) Jika TM pernah membaca simbol b ada kemungkinan ditemukan simbol B. Simbol B tersebut harus dilewati (*tidak boleh dioverwrite*), artinya B *diover-write dengan* B *juga* dan head bergerak ke *kanan*.
3. Head membaca simbol b, maka b tersebut harus dioverwrite dengan simbol lain (misalnya B) untuk menandakan bahwa b tersebut (sebagai pasangan dari a) telah dibaca, dan head bergerak ke *kiri* untuk mencari simbol A.
   i) Jika ditemukan B maka B tersebut harus dilewati (*tidak boleh dioverwrite*), dengan kata lain B *dioverwrite dengan* B *juga* dan head bergerak ke *kiri*.
   ii) Jika ditemukan a maka a tersebut harus dilewati (*tidak boleh dioverwrite*), dengan kata lain a *dioverwrite dengan* a *juga* dan head bergerak ke *kiri*.
4. Head membaca simbol A, maka A tersebut harus dilewati (*tidak boleh dioverwrite*), dengan kata lain A *dioverwrite dengan* A *juga* dan head bergerak ke *kanan*.
5. Head membaca simbol a, ulangi langkah 2 dan 3.
6. (Setelah langkah 3) head membaca simbol A, maka A tersebut harus dilewati (*tidak boleh dioverwrite*), dengan kata lain A *dioverwrite dengan* A *juga* dan head bergerak ke *kanan*.
7. Head membaca simbol B, maka B tersebut harus dilewati (*tidak boleh dioverwrite*), dengan kata lain B *dioverwrite dengan* A *juga* dan head bergerak ke *kanan*.
8. Head membaca simbol ε, maka ε dioverwrite dengan ε dan head bergerak ke *kanan* menuju stata H.

Skema graf Mesin Turing di atas adalah :



Contoh :
Lakukan tracing dengan mesin turing di atas untuk kalimat-kalimat : aabb, aab.
Jawab :

i) (S,aabb) $\Rightarrow$ (1,Aabb) $\Rightarrow$ (1,Aabb) $\Rightarrow$ (2,AaBb) $\Rightarrow$ (3,AaBb) $\Rightarrow$ (S,AaBb)
$\Rightarrow$ (1,AABb) $\Rightarrow$ (1,AABb) $\Rightarrow$ (2,AABB) $\Rightarrow$ (2,AABB) $\Rightarrow$ (4,AABB)
$\Rightarrow$ (4,AABB) $\Rightarrow$ (4,AABBε) $\Rightarrow$ (H,AABBεε)

ii) (S,aab) $\Rightarrow$ (1,Aab) $\Rightarrow$ (1,Aab) $\Rightarrow$ (2,AaB) $\Rightarrow$ (3,AaB) $\Rightarrow$ (S,AaB) $\Rightarrow$ (1,AAB)
$\Rightarrow$ (1,AAbε) $\Rightarrow$ crash, karena dari node 1 tidak ada edge dengan bobot komponen pertamanya hampa (ε)

IV. AUTOMATA HINGGA (AH)

- AH didefinisikan sebagai pasangan 5 tupel : (K, V_T, M, S, Z).
  K : himpunan hingga stata,
  V_T : himpunan hingga simbol input (alfabet)
  M : fungsi transisi, menggambarkan transisi stata AH akibat pembacaan simbol input.
  Fungsi transisi ini biasanya diberikan dalam bentuk tabel.
  $S \in K$: stata awal
  $Z \subset K$: himpunan stata penerima
- Ada dua jenis automata hingga : deterministik (AHD, DFA = *deterministic finite automata*) dan non deterministik (AHN, NFA = *non deterministik finite automata*).
  - AHD : transisi stata AH akibat pembacaan sebuah simbol bersifat tertentu.
    $M(AHD) : K \times V_T \rightarrow K$
  - AHN : transisi stata AH akibat pembacaan sebuah simbol bersifat tak tentu.
    $M(AHN) : K \times V_T \rightarrow 2^K$

IV. 1. Automata Hingga Deterministik (AHD)

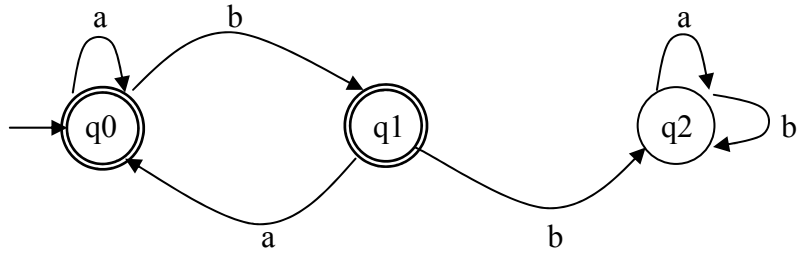
Berikut ini sebuah contoh AHD F(K, V_T, M, S, Z), dimana :

K = {q0, q1, q2}
V_T = {a, b}
S = q0
Z = {q0, q1}

M diberikan dalam tabel berikut :

	a	b
q0	q0	q1
q1	q0	q2
q2	q2	q2

Ilustrasi graf untuk AHD F adalah sebagai berikut :
Lambang stata awal adalah node dengan anak panah.
Lambang stata awal adalah node ganda.



Contoh kalimat yang diterima AHD : a, b, aa, ab, ba, aba, bab, abab, baba
Contoh kalimat yang tidak diterima AHD : bb, abb, abba
AHD ini menerima semua kalimat yang tersusun dari simbol a dan b yang tidak mengandung substring bb.

Contoh :
Telusurilah, apakah kalimat-kalimat berikut diterima AHD :
abababaa, aaaabab, aaabbaba
Jawab :

i) M(q0,abababaa) $\Rightarrow$ M(q0,bababaa) $\Rightarrow$ M(q1,ababaa) $\Rightarrow$ M(q0,babaa)
$\Rightarrow$ M(q1,abaa) $\Rightarrow$ M(q0,baa) $\Rightarrow$ M(q1,aa) $\Rightarrow$ M(q0,a) $\Rightarrow$ q0
Tracing berakhir di q0 (stata penerima) $\Rightarrow$ kalimat abababaa diterima

ii) M(q0, aaaabab) $\mapsto$ M(q0,aaabab) $\mapsto$ M(q0,aabab) $\mapsto$ M(q0,abab)
$\Rightarrow$ M(q0,bab) $\Rightarrow$ M(q1,ab) $\Rightarrow$ M(q0,b) $\mapsto$ q1
Tracing berakhir di q1 (stata penerima) $\Rightarrow$ kalimat aaaababa diterima

iii) M(q0, aaabbaba) $\Rightarrow$ M(q0, aabbaba) $\Rightarrow$ M(q0, abbaba) $\Rightarrow$ M(q0, bbaba)
$\Rightarrow$ M(q1,bbaba) $\Rightarrow$ M(q2,baba) $\Rightarrow$ M(q2,aba) $\Rightarrow$ M(q2,ba) $\Rightarrow$ M(q2,a) $\mapsto$ q2
Tracing berakhir di q2 (bukan stata penerima) $\Rightarrow$ kalimat aaabbaba ditolak

Kesimpulan : sebuah kalimat diterima oleh AHD jika tracingnya berakhir di salah satu stata penerima.

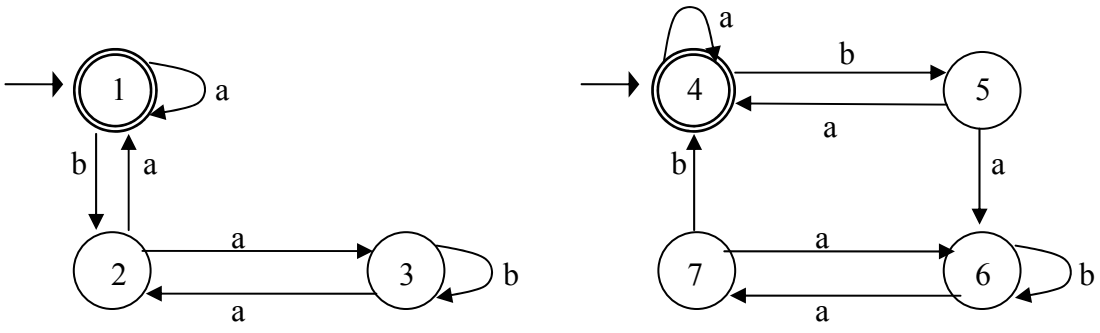
IV.2. Equivalensi 2 AHD

Dua buah AHD dikatakan equivalen jika keduanya dapat menerima bahasa yang sama. Misalkan kedua AHD tersebut adalah A dan A'. Misalkan pula bahasa yang diterima adalah bahasa L yang dibangun oleh alfabet V_T = {a1, a2, a3, ..., an}. Berikut ini algoritma untuk menguji equivalensi dua buah AHD.

1. Berikan nama kepada semua stata masing-masing AHD dengan nama berbeda. Misalkan nama-nama tersebut adalah : S, A1, A2, ... untuk AHD A, dan : S', A1', A2', ... untuk AHD A'.
2. Buat tabel (n+1) kolom, yaitu kolom-kolom : (v, v'), (v_{a1}, v_{a1}'), ..., (v_{an}, v_{an}'), yaitu pasangan terurut (stata AHD A, stata AHD A').
3. Isikan (S, S') pada baris pertama kolom (v, v'), dimana S dan S' masing-masing adalah *stata awal* masing-masing AHD.
4. Jika terdapat *edge* dari S ke A1 dengan label a1 dan jika terdapat *edge* dari S' ke A1' juga dengan label a1, isikan pasangan terurut (A1, A1') sebagai pada baris pertama kolom (v_{a1}, v_{a1}'). Lakukan hal yang sama untuk kolom-kolom berikutnya.
5. Perhatikan nilai-nilai pasangan terurut pada baris pertama. Jika terdapat nilai pasangan terurut pada kolom (v_{a1}, v_{a1}') s/d (v_{an}, v_{an}') *yang tidak sama* dengan nilai pasangan terurut (v, v'), tempatkan nilai tersebut pada kolom (v, v') *baris-baris* berikutnya. Lakukan hal yang sama seperti yang dilakukan pada langkah (4). Lanjutkan dengan langkah (5).
6. Jika selama proses di atas dihasilkan sebuah nilai pada kolom (v, v'), dengan komponen v merupakan *stata penerima* sedangkan komponen v' bukan, atau sebaliknya, maka *kedua AHD tersebut tidak ekuivalen*. Proses dihentikan.
7. Jika kondisi (6) tidak dipenuhi dan jika tidak ada lagi pasangan terurut baru yang harus ditempatkan pada kolom (v, v') maka proses dihentikan dan *kedua AHD tersebut ekuivalen*.

Contoh :

Periksalah ekuivalensi kedua AHD berikut :



AHD A AHD A'

Jawab :

Dengan menggunakan menggunakan algoritma di atas maka dapat dibentuk tabel berikut :

(v, v')	(v_a, v_a')	(v_b, v_b')	Keterangan :
(1, 4)	(1, 4)	(2, 5)	(2, 5) adalah pasangan terurut baru
(2,5)	(3, 6)	(1, 4)	(3, 6) adalah pasangan terurut baru
(3, 6)	(2, 7)	(3, 6)	(2, 7) adalah pasangan terurut baru
(2, 7)	(3, 6)	(1, 4)	tidak adal lagi pasangan terurut baru

IV. 3. Mesin Stata Hingga (MSH)

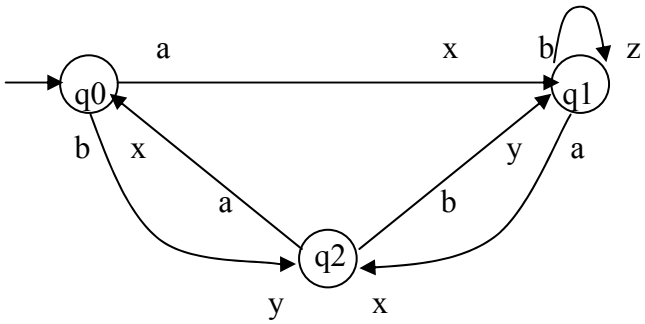
- MSH atau FSM (*Finite State Machine*) adalah sebuah varians automata hingga. MSH sering juga disebut sebagai automata hingga beroutput atau mesin sekuensial.
- MSH didefinisikan sebagai pasangan 6 tupel F(K, V_T, S, Z, f, g) dimana :
  K : himpunan hingga stata,
  V_T : himpunan hingga simbol input (alfabet)
  $S \in K$: stata awal
  Z : himpunan hingga simbol output
  $f : K \times V_T \rightarrow K$ disebut fungsi next state
  $g : K \times V_T \rightarrow Z$ disebut fungsi output

Contoh :
Berikut ini adalah contoh MSH dengan 2 simbol input, 3 stata, dan 3 simbol output :

K = {q0, q1, q2}	fungsi f :		fungsi g :	
S = q0	f(q0,a) = q1	f(q0,b) = q2	f(q0,a) = x	f(q0,b) = y
V_T = {a, b}	f(q1,a) = q2	f(q1,b) = q1	f(q1,a) = x	f(q1,b) = z
Z = {x, y, z}	f(q2,a) = q0	f(q2,b) = q1	f(q2,a) = z	f(q2,b) = y

MSH dapat disajikan dalam bentuk tabel atau graf. Untuk MSH contoh di atas tabel dan grafnya masing-masing adalah :

	a	b
q0	q1, x	q2, y
q1	q2, x	q1, z
q2	q0, z	q1, y



Jika MSH di atas mendapat untai masukan "aaba" maka akan dihasilkan :
- untai keluaran : xxyx
- untai stata : q0 q1 q2 q1 q2

IV. 4. MSH penjumlah biner

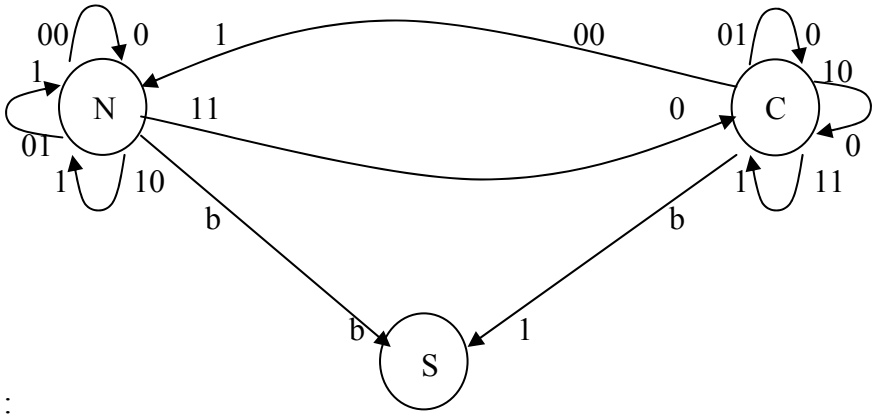
MSH dapat disajikan sebagai penjumlah biner. Sifat penjumlahan biner bergantung pada statusnya : carry atau not carry.
Pada status not carry berlaku : 0 + 0 = 0, 1 + 0 = 0 + 1 = 1, 1 + 1 = 0
Pada status carry berlaku : 0 + 0 = 1, 1 + 0 = 0 + 1 = 0, 1 + 1 = 1
Pada status not carry blank (b) menjadi b, sedangkan pada status carry menjadi 1.

Nilai setiap tupel untuk MSH ini adalah :

K = N (not carry), C (carry), dan S (stop)
S = N
V_T = {00, 01, 10, 11, b}
Z = {0, 1, b}

Tabel MSH					
	00	01	10	11	b
N	N, 0	N, 1	N, 1	C, 0	S, b
C	N, 1	C, 0	C, 0	C, 1	S, 1

Graf MSH penjumlah biner :



Contoh :
Hitunglah : 1101011 + 0111011
Jawab :
Input = pasangan digit kedua bilangan, mulai dari LSB (*least significant bit*)
= 11, 11, 00, 11, 01, 11, 11, b
Output = 0, 1, 1, 0, 0, 1, 1, 1 (jawab : dibaca dari kanan)
Stata = N, C, C, N, C, C, C, C, S
Periksa : 1 1 0 1 0 1 1
0 1 1 1 0 1 1 +
1 1 1 0 0 1 1 0

IV. 5. Ekspresi Regular

- Bahasa regular dapat dinyatakan sebagai ekspresi regular dengan menggunakan 3 operator : concate, alternate, dan closure.
- Dua buah ekspresi regular adalah ekuivalen jika keduanya menyatakan bahasa yang sama

Contoh :
$L_1 = \{a^n ba^m \mid n \geq 1, m \geq 1\} \Leftrightarrow er_1 = a^+ b a^+$
$L_2 = \{a^n ba^m \mid n \geq 0, m \geq 0\} \Leftrightarrow er_2 = a^* b a^*$
Perhatikan bahwa kita tidak bisa membuat ekspresi regular dari bahasa
$L_3 = \{a^n ba^n \mid n \geq 1\}$ atau $L_4 = \{a^n ba^n \mid n \geq 0\}$, karena keduanya tidak dihasilkan dari grammar regular.

Kesamaan 2 ekspresi regular :
$(a\ b)^*\ a = a\ (b\ a)^*$
Bukti :
$(a\ b)^*\ a = (\varepsilon \mid (ab) \mid (abab) \mid \ldots)\ a = (\varepsilon\ a \mid (aba) \mid (ababa) \mid \ldots) = (a \mid (aba) \mid (ababa) \mid \ldots)$
$= a\ (\varepsilon \mid (ba) \mid (baba) \mid \ldots) = a\ (b\ a)^*$

Latihan 2. Buktikan kesamaan ekspresi regular berikut :
1. $(a^* \mid b)^* = (a \mid b)^*$
2. $(a \mid b^*)^* = (a \mid b)^*$
3. $(a^*\ b)^*\ a^* = a^*\ (b\ a^*)^*$
4. $(a\ a^*)(\varepsilon \mid a) = a^*$
5. $a(b\ a \mid a)^*\ b = a\ a^*\ b(a\ a^*\ b)^*$

IV. 6. Automata Hingga Nondeterministik (AHN)

Berikut ini sebuah contoh AHN F(K, V_T, M, S, Z), dimana :

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

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

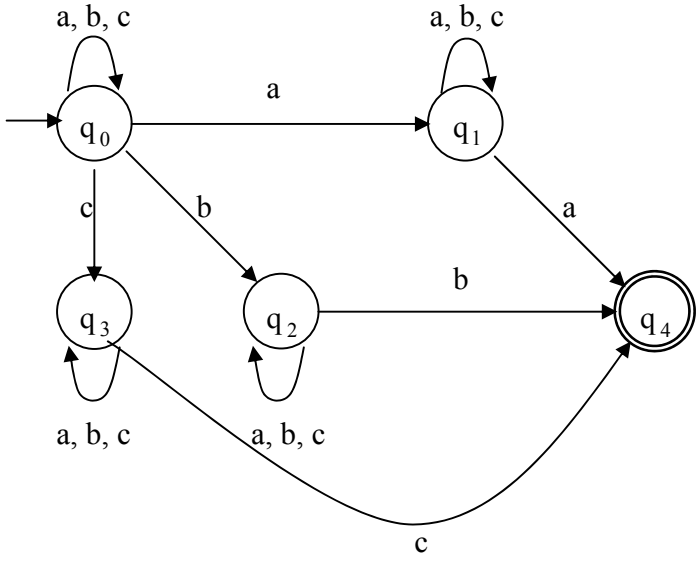
$S = q_0$

$Z = \{q_4\}$

M diberikan dalam tabel berikut :

	a	b	c
q_0	$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_3\}$
q_1	$\{q_1, q_4\}$	$\{q_1\}$	$\{q_1\}$
q_2	$\{q_2\}$	$\{q_2, q_4\}$	$\{q_2\}$
q_3	$\{q_3\}$	$\{q_3\}$	$\{q_3, q_4\}$
q_4	$\varnothing$	$\varnothing$	$\varnothing$

Ilustrasi graf untuk AHN F adalah sebagai berikut :



Contoh kalimat yang diterima AHN di atas : aa, bb, cc, aaa, abb, bcc, cbb
Contoh kalimat yang tidak diterima AHN di atas : a, b, c, ab, ba, ac, bc

Fungsi transisi M sebuah AHN dapat diperluas sebagai berikut :

1. $M(q, \varepsilon) = \{q\}$ untuk setiap $q \in K$
2. $M(q, t\,T) = \cup\, M(p_i, T)$ dimana $t \in V_T$, T adalah V_T^*, dan $M(q, t) = \{p_i\}$
3. $M(\{q_1, q_2, \ldots, q_n\}, x) = \cup\, M(q_i, x)$, untuk $x \in V_T^*$

Sebuah kalimat di terima AHN jika :

- salah satu tracing-nya berakhir di stata penerima, atau
- himpunan stata setelah membaca string tersebut mengandung stata penerima

Contoh :
Telusurilah, apakah kalimat-kalimat berikut diterima AHN : ab, abc, aabc, aabb

Jawab :

i) $M(q_0,ab) \Rightarrow M(q_0,b) \cup M(q_1,b) \Rightarrow \{q_0, q_2\} \cup \{q_1\} = \{q_0, q_1, q_2\}$
Himpunan stata tidak mengandung stata penerima ⇒ kalimat ab *tidak diterima*

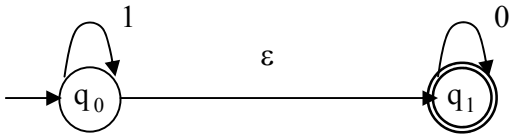
ii) $M(q_0,abc) \Rightarrow M(q_0,bc) \cup M(q_1,bc) \Rightarrow \{M(q_0,c) \cup M(q_2,c)\} \cup M(q_1, c)$
$\Rightarrow \{\{q_0, q_3\} \cup \{q_2\}\} \cup \{q_1\} = \{q_0, q_1, q_2, q_3\}$
Himpunan stata tidak mengandung stata penerima ⇒ kalimat abc *tidak diterima*

iii) $M(q_0,aabc) \Rightarrow M(q_0,abc) \cup M(q_1,abc) \Rightarrow \{M(q_0,bc) \cup M(q_1,bc)\} \cup M(q_1,bc)$
$\Rightarrow \{\{M(q_0, c) \cup M(q_2,c)\} \cup M(q_1, c)\} \cup M(q_1, c)$
$\Rightarrow \{\{\{q_0, q_3\} \cup \{q_2\}\} \cup \{q_1\}\} \cup \{q_1\} = \{q_0, q_1, q_2, q_3\}$
Himpunan stata tidak mengandung stata penerima ⇒ kalimat aabc *tidak diterima*

iv) $M(q_0,aabb) \Rightarrow M(q_0,abb) \cup M(q_1,abb) \Rightarrow \{M(q_0,bb) \cup M(q_1,bb)\} \cup M(q_1,bb)$
$\Rightarrow \{\{M(q_0, b) \cup M(q_2,b)\} \cup M(q_1, b)\} \cup M(q_1, b)$
$\Rightarrow \{\{\{q_0, q_2\} \cup \{q_2, q_4\}\} \cup \{q_1\}\} \cup \{q_1\} = \{q_0, q_1, q_2, q_4\}$
Himpunan stata tidak mengandung stata penerima ⇒ kalimat aabb *diterima*

AHN Dengan Transisi Hampa

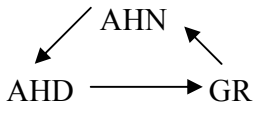
Perhatikan AHN berikut.



AHN di atas mengandung ruas dengan bobot ε. AHN demikian dinamakan AHN dengan transisi ε, atau singkatnya AHN-ε. AHN-ε di atas menerima bahasa $L = \{1^i 0^j \mid i, j \geq 0\}$

IV. 7. Ekuivalensi AHN, AHD, dan GR

AHD bisa dibentuk dari AHN.
GR bisa dibentuk dari AHD.
AHN bisa dibentuk dari GR.



Pembentukan AHD dari AHN

Diberikan sebuah AHN F = (K, V_T, M, S, Z). Akan dibentuk sebuah AHD F' = (K', V_T', M', S', Z') dari AHN F tersebut. Algoritma pembentukannya adalah sbb. :

1. Tetapkan : S' = S dan $V_T' = V_T$
2. Copykan tabel AHN F sebagai tabel AHD F'. Mula-mula K' = K dan M' = M
3. Setiap stata q yang merupakan *nilai* (atau *peta*) dari fungsi M dan $q \notin K$, ditetapkan sebagai elemen baru dari K'. Tempatkan q tersebut pada kolom Stata M', lakukan pemetaan berdasarkan fungsi M.
4. Ulangi langkah (3) sampai tidak diperoleh stata baru.
5. Elemen Z' adalah semua stata yang mengandung stata elemen Z.

Contoh :

Berikut ini diberikan sebuah AHN F = (K, V_T, M, S, Z) dengan :

K = {A, B, C}, V_T = {a, b}, S = A, Z = {C}, dan M didefinisikan sebagai berikut :

Stata K AHN F	Input	
	a	b
A	[A,B]	C
B	A	B
C	B	[A,B]

Tentukan AHD hasil transformasinya.

Jawab :

Berdasarkan algoritma di atas, maka :

1. S' = S = A, V_T' = V_T = {a, b}.
2. Hasil copy tabel AHN F menghasilkan tabel AHD F' berikut :

Stata K' AHD F'	Input	
	a	b
A	[A,B]	C
B	A	B
C	B	[A,B]

3. Pada tabel AHD F' di atas terdapat stata baru yaitu [A,B]. Pemetaan [A,B] adalah : M([A,B],a) = M(A,a) ∪ M(B,a) = [A,B] ∪ A = [A,B], dan
   M([A,B],b) = M(A,b) ∪ M(B,b) = C ∪ B = [B,C], sehingga diperoleh tabel berikut :

Stata K' dari AHD F'	Input	
	a	b
A	[A,B]	C
B	A	B
C	B	[A,B]
[A,B]	[A,B]	[B,C]

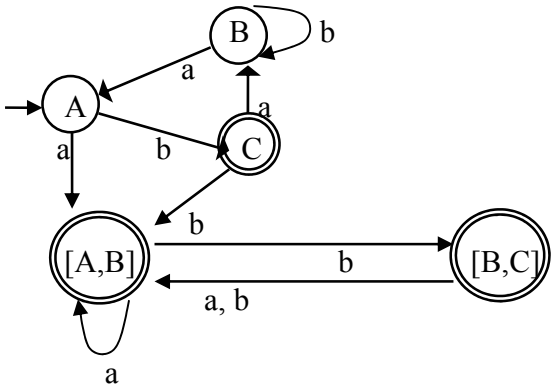
4. Langkah (3) di atas menghasilkan stata baru yaitu [B,C]. Setelah pemetaan terhadap [B,C] diperoleh tabel berikut :

Stata K' dari AHD F'	Input	
	a	b
A	[A,B]	C
B	A	B
C	B	[A,B]
[A,B]	[A,B]	[B,C]
[B,C]	[A,B]	[A,B]

5. Setelah langkah (4) di atas tidak terdapat lagi stata baru.

Dengan demikian AHD F' yang dihasilkan adalah : AHD F' = (K', V_T', M', S', Z'), dimana : K' = {A, B, C, [A,B], [B,C]}, V_T' = {a, b}, S' = A, Z' = {C, [B,C]}. Fungsi transisi M' serta graf dari AHD F' adalah sebagai berikut :

Stata K' dari AHD F'	Input	
	a	b
A	[A,B]	C
B	A	B
C	B	[A,B]
[A,B]	[A,B]	[B,C]
[B,C]	[A,B]	[A,B]



Pembentukan GR dari AHD

Diketahui sebuah AHD F = (K, V_T, M, S, Z). Akan dibentuk GR G = (V_T',V_N, S', Q).

Algoritma pembentukan GR dari AHD adalah sebagai berikut :

1. Tetapkan $V_T' = V_T$, S' = S, $V_N = S$
2. Jika $A_p, A_q \in K$ dan $a \in V_T$, maka :

   $M(A_p, a) = A_q$ ekuivalen dengan produksi : $\begin{cases} A_p \rightarrow aA_q, & \text{jika } A_q \notin Z \\ A_p \rightarrow a, & \text{jika } A_q \in Z \end{cases}$

Contoh

Diketahui sebuah AHD F dengan Z = {S} dan fungsi transisi M sebagai berikut :

Stata K AHD F	Input 0	Input 1
S	B	A
A	C	S
B	S	C
C	A	B

Dengan algoritma di atas maka diperoleh Q(GR) sbb. :

$M(S,0) = B \Leftrightarrow S \rightarrow 0B$ $\quad M(S,1) = A \Leftrightarrow S \rightarrow 1A$

$M(A,0) = C \Leftrightarrow A \rightarrow 0C$ $\quad M(A,1) = S \Leftrightarrow A \rightarrow 1$

$M(B,0) = S \Leftrightarrow B \rightarrow 0$ $\quad M(B,1) = C \Leftrightarrow B \rightarrow 1C$

$M(C,0) = A \Leftrightarrow C \rightarrow 0A$ $\quad M(C,1) = B \Leftrightarrow C \rightarrow 1B$

GR yang dihasilkan adalah G(V_T',V_N, S', Q), dengan $V_T' = \{0,1\}$, $V_N = \{S, A, B, C\}$, S' = S, dan Q = {$S \rightarrow 0B$, $S \rightarrow 1A$, $A \rightarrow 0C$, $B \rightarrow 1C$, $C \rightarrow 0A$, $C \rightarrow 1B$, $A \rightarrow 1$, $B \rightarrow 0$}

Pembentukan AHN dari GR

Diketahui GR G = (V_T,V_N, S, Q). Akan dibentuk AHN F = (K,V_T', M, S', Z).

Algoritma pembentukan AHN dari GR :

1. Tetapkan $V_T' = V_T$, S' = S, $K = V_N$
2. Produksi $A_p \rightarrow a\,A_q$ ekuivalen dengan $M(A_p, a) = A_q$

   Produksi $A_p \rightarrow a$ ekuivalen dengan $M(A_p, a) = X$, dimana $X \notin V_N$
3. $K = K \cup \{X\}$
4. $Z = \{X\}$

Contoh

Diketahui GR G = (V_T, V_N, S, Q) dengan : V_T = {a, b}, V_N = {S, A, B}, S = S, dan Q = {S → aS, S → bA, A → aA, A → aB, B → b}

Terapkan algoritma di atas untuk memperoleh AHN F sebagai berikut :

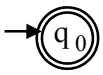
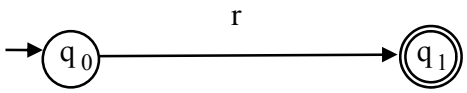
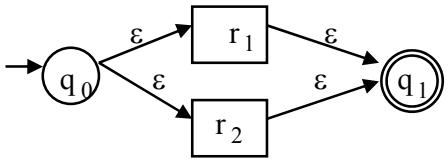
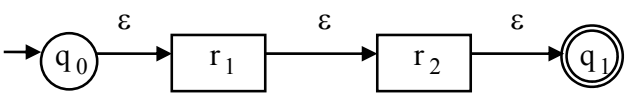
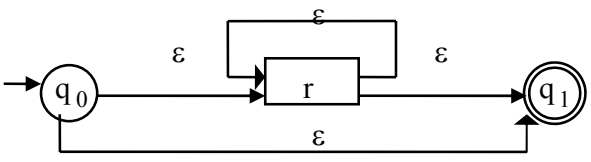
1. V_T' = V_T = {a, b}, S' = S, K = V_N = {S, A, B}
2. S → aS ⇔ M(S,a) = S, S → bA ⇔ M(S,b) = A,
   A → aA ⇔ M(A,a) = A, A → aB ⇔ M(A,a) = B,
   B → b ⇔ M(B,b) = X

AHN yang diperoleh : F(K,V_T', M, S', Z), dengan
K = {S, A, B, X}, V_T' = {a, b}, S' = S, Z = {X},

Tabel M :

Stata K / AHN F	Input	
	a	b
S	S	A
A	[A,B]	ϕ
B	ϕ	X
X	ϕ	ϕ

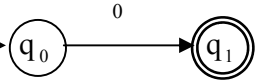
IV.8. Ekuivalensi Ahn-ε Dengan ER (Ekspresi Regular)

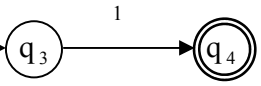
Jenis ER	Simbol ER	AHN
Simbol hampa	ε	
ER hampa	ϕ atau {}	
ER umum	r	
Alternation	$r_1 \mid r_2$	
Concatenation	$r_1 r_2$	
Kleene Clossure	r *	

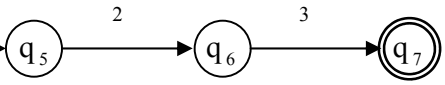
Contoh :

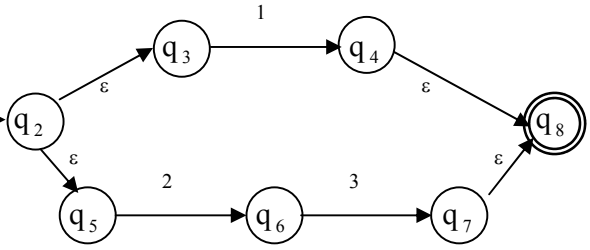
Tentukan AHN untuk ekspresi regular r = 0(1 | 23)*

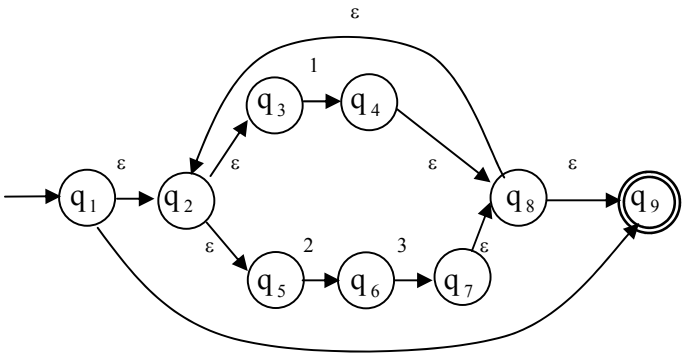
Jawab :

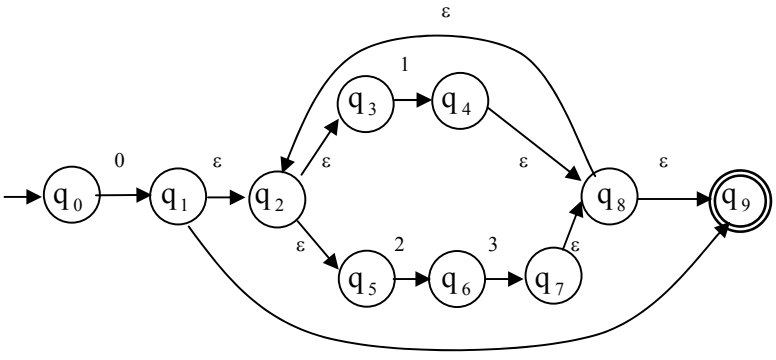
$r_1 = 0 \Leftrightarrow$ 

$r_2 = 1 \Leftrightarrow$ 

$r_3 = 23 \Leftrightarrow$ 

$r_4 = r_2 \mid r_3 = 1 \mid 23 \Leftrightarrow$ 

$r_5 = r_4^* = (1 \mid 23)^* \Leftrightarrow$ 

$r = r_5 = 0(1 \mid 23)^* \Leftrightarrow$ 

V. GRAMMAR CONTEXT-FREE DAN PARSING

Bentuk umum produksi CFG adalah : $\alpha \rightarrow \beta, \quad \alpha \in V_N, \quad \beta \in (V_N \mid V_T)^*$

Analisis sintaks adalah penelusuran sebuah kalimat (atau sentensial) sampai pada simbol awal grammar. Analisis sintaks dapat dilakukan melalui derivasi atau parsing. Penelusuran melalui parsing menghasilkan *pohon sintaks*.

Contoh 1 :

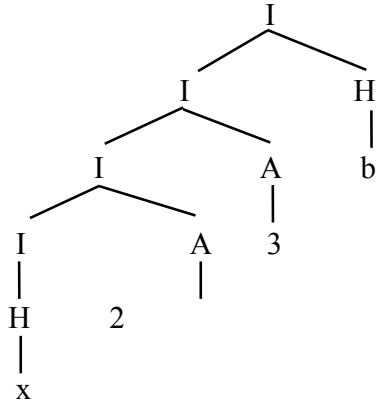
Diketahui grammar $G_1 = \{I \rightarrow H \mid I\,H \mid IA,\ H \rightarrow a \mid b \mid c \mid ... \mid z,\ A \rightarrow 0 \mid 1 \mid 2 \mid ... \mid 9\}$

dengan I adalah simbol awal. Berikut ini kedua cara analisa sintaks untuk kalimat x23b.

cara 1 (derivasi)

$I \Rightarrow IH$
$\Rightarrow IAH$
$\Rightarrow IAAH$
$\Rightarrow HAAH$
$\Rightarrow xAAH$
$\Rightarrow x2AH$
$\Rightarrow x23H$
$\Rightarrow x23b$

cara 2 (parsing)

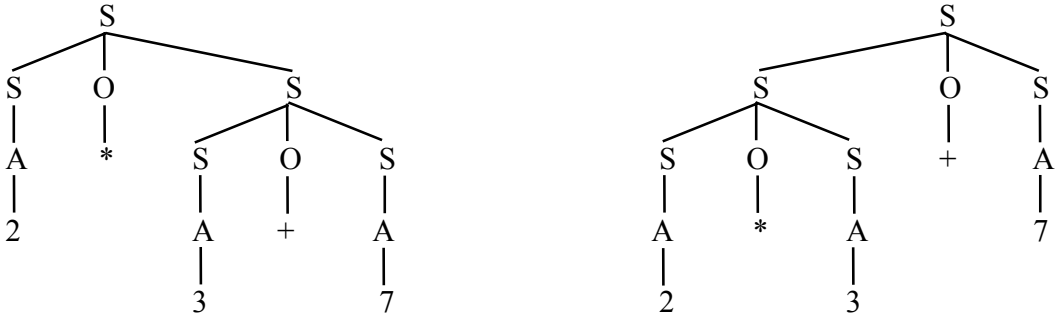


Sebuah kalimat dapat saja mempunyai lebih dari satu pohon.

Contoh 2 :

Diketahui grammar $G_2 = \{S \rightarrow SOS \mid A,\ O \rightarrow * \mid +,\ \ A \rightarrow 0 \mid 1 \mid 2 \mid ... \mid 9\}$

Kalimat : 2*3+7 mempunyai dua pohon sintaks berikut :



Sebuah kalimat yang mempunyai lebih dari satu pohon sintaks disebut *kalimat ambigu* (*ambiguous*). Grammar yang menghasilkan paling sedikit sebuah kalimat ambigu disebut *grammar ambigu*.

5.1. Metoda Parsing

Ada 2 metoda parsing : *top-down* dan *bottom-up*.

Parsing *top-down* : Diberikan kalimat x sebagai input. Parsing dimulai dari simbol awal S sampai kalimat x nyata (atau tidak nyata jika kalimat x memang tidak bisa diturunkan dari S) dari pembacaan semua *leaf* dari pohon parsing jika dibaca dari kiri ke kanan.

Parsing *bottom-up* : Diberikan kalimat x sebagai input. Parsing dimulai dari kalimat x yang nyata dari pembacaan semua *leaf* pohon parsing dari kiri ke kanan sampai tiba di simbol awal S (atau tidak sampai di S jika kalimat x memang tidak bisa diturunkan dari S)

Parsing Top-down

Ada 2 kelas metoda parsing *top-down*, yaitu kelas metoda *dengan backup* dan kelas metoda *tanpa backup*. Contoh metoda kelas *dengan backup* adalah metoda *Brute-Force*, sedangkan contoh metoda kelas *tanpa backup* adalah metoda *recursive descent*.

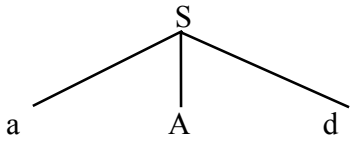
Metoda *Brute-Force*

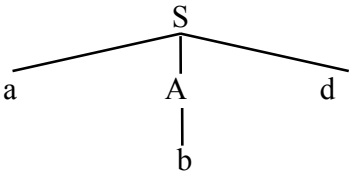
Kelas metoda *dengan backup*, termasuk metoda *Brute-Force*, adalah kelas metoda parsing yang menggunakan produksi alternatif, jika ada, ketika hasil penggunaan sebuah produksi tidak sesuai dengan simbol input. Penggunaan produksi sesuai dengan nomor urut produksi.

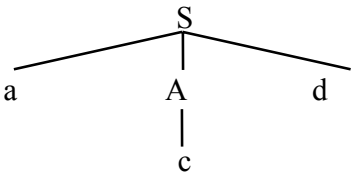
Contoh 3 :

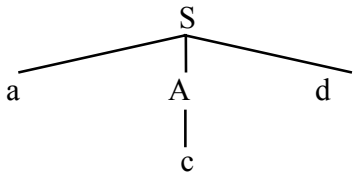
Diberikan grammar G = {S → aAd | aB, A → b | c, B → ccd | ddc}. Gunakan metoda Brute-Force untuk melakukan analisis sintaks terhadap kalimat x = accd.

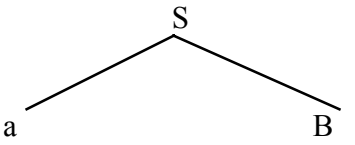
<table>
<tr>
<td>S

Hasil :
Input : Sisa : accd
Penjelasan : Gunakan produksi S pertama. Masukkan simbol terkiri kalimat sebagai input.</td>
<td> 

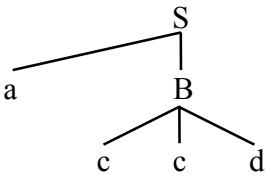
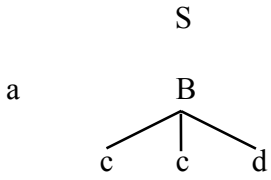
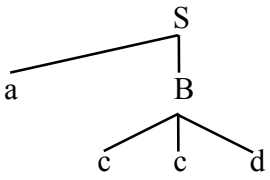
Hasil : a
Input : a Sisa : ccd
Penjelasan : Hasil = Input. Gunakan produksi A pertama.</td>
<td> 

Hasil : ab
Input : ac Sisa : cd
Penjelasan : Hasil ≠ Input. Backup : Gunakan produksi A alternatif pertama.</td>
</tr>
<tr>
<td> 

Hasil : ac
Input : ac Sisa : cd
Penjelasan : Hasil = Input. Karakter berikutnya adalah simbol terminal, Hasil dibandingkan dengan Input.</td>
<td> 

Hasil : acd
Input : acc Sisa : c
Penjelasan : Hasil ≠ Input. Tidak ada lagi produksi A alternatif, backup : gunakan produksi S alternatif pertama.</td>
<td> 

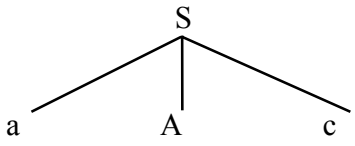
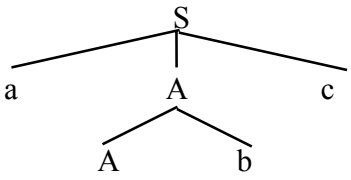
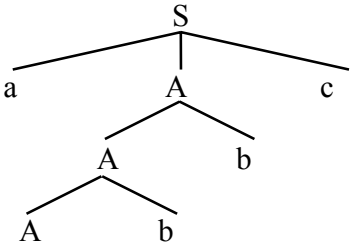
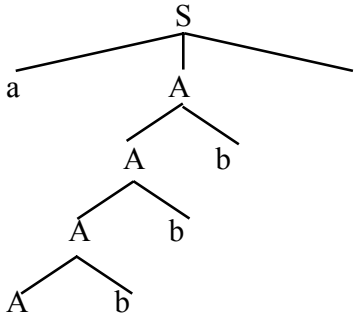
Hasil : a
Input : a Sisa : ccd
Penjelasan : Hasil = Input. Gunakan produksi B pertama.</td>
</tr>
</table>

 Hasil : ac Input : ac Sisa : cd Penjelasan : Hasil = Input. Karakter berikutnya adalah simbol terminal, Hasil dibandingkan dengan Input.	 Hasil : acc Input : acc Sisa : d Penjelasan : Hasil = Input. Karakter berikutnya adalah simbol terminal, Hasil dibandingkan dengan Input.	 Hasil : accd Input : accd Sisa : Penjelasan : Hasil = Input. SELESAI, SUKSES

Metoda *Brute-Force* tidak dapat menggunakan grammar rekursi kiri, yaitu grammar yang mengandung produksi rekursi kiri (*left recursion*) : $A \rightarrow A\propto$. Produksi rekursi kiri akan menyebabkan parsing mengalami looping tak hingga.

Contoh 4 :

Diberikan grammar G = $\{S \rightarrow aAc,\ A \rightarrow Ab \mid \varepsilon\}$. Gunakan metoda Brute-Force untuk melakukan analisis sintaks terhadap kalimat x = ac.

S Hasil : Input : Sisa : ac Penjelasan : Masukkan simbol terkiri kalimat sebagai input. Gunakan produksi S pertama.	 Hasil : a Input : a Sisa : c Penjelasan : Hasil = Input. Gunakan produksi A pertama.	 Hasil : a Input : a Sisa : c Penjelasan : Hasil = Input. Gunakan produksi A pertama.
 Hasil : a Input : a Sisa : c Penjelasan : Hasil = Input. Gunakan produksi A pertama.	 Hasil : a Input : a Sisa : c Penjelasan : Hasil = Input. Gunakan produksi A pertama.	dan seterusnya..... (looping)

Agar tidak menghasilkan looping tak hingga, grammar rekursi kiri harus ditransformasi. Untuk contoh di atas transformasi berarti merubah produksi $A \rightarrow Ab$ menjadi $A \rightarrow bA$.

Metoda *Recursive-Descent*

- Kelas metoda *tanpa backup*, termasuk metoda *recursive descent*, adalah kelas metoda parsing yang tidak menggunakan produksi alternatif ketika hasil akibat penggunaan sebuah produksi tidak sesuai dengan simbol input. Jika produksi A mempunyai dua buah ruas kanan atau lebih maka produksi yang dipilih untuk digunakan adalah *produksi dengan simbol pertama ruas kanannya sama dengan input yang sedang dibaca*. Jika tidak ada produksi yang demikian maka dikatakan bahwa parsing tidak dapat dilakukan.
- Ketentuan produksi yang digunakan metoda *recursive descent* adalah : *Jika terdapat dua atau lebih produksi dengan ruas kiri yang sama maka karakter pertama dari semua ruas kanan produksi tersebut tidak boleh sama*. Ketentuan ini tidak melarang adanya produksi yang bersifat rekursi kiri.

Contoh 5 :

Diketahui grammar G = {S → aB | A, A → a, B → b | d}. Gunakan metoda *recursive descent* untuk melakukan analisis sintaks terhadap kalimat x = ac.

S Hasil : Input : Sisa : ab Penjelasan : Masukkan simbol terkiri kalimat sebagai input. Gunakan produksi S dengan simbol pertama ruas kanan = a	S a B Hasil : a Input : a Sisa : c Penjelasan : Hasil = Input. Gunakan produksi B dengan simbol pertama ruas kanan = c. Karena produksi demikian maka parsing gagal dilakukan.	SELESAI, PARSING GAGAL

Parsing Bottom-Up

Salah satu contoh menarik dari parsing *bottom-up* adalah parsing pada *grammar preseden sederhana* (GPS). Sebelum sampai ke parsing tersebut, akan dikemukakan beberapa pengertian dasar serta relasi yang ada pada GPS.

Pengertian Dasar

- Jika α dan x keduanya diderivasi dari simbol awal grammar tertentu, maka α disebut *sentensial* jika $\alpha \in (V_T \mid V_N)^*$, dan x disebut *kalimat* jika $x \in (V_T)^*$
- Misalkan $\alpha = Q_1 \beta Q_2$ adalah sentensial dan $A \in V_N$:
  - β adalah *frase* dari sentensial α jika : $S \Rightarrow \ldots \Rightarrow Q_1 A Q_2$ dan $A \Rightarrow \ldots \Rightarrow \beta$
  - β adalah *simple frase* dari sentensial α jika : $S \Rightarrow \ldots \Rightarrow Q_1 A Q_2$ dan $A \Rightarrow \beta$
  - Simple frase terkiri dinamakan *handel*
  - frase, simple frase, dan handel adalah string dengan panjang 0 atau lebih..

Contoh 6 :

(1) $I \Rightarrow I H$ Hb adalah sentensial dan b adalah simple frase
$\Rightarrow H H$ (dibandingkan dengan $Q_1 \beta Q_2$ maka $Q_1 = H$, $\beta = b$, dan $Q_2 = \varepsilon$)
$\Rightarrow H b$ Perhatikan : simple frase (b) adalah yang terakhir diturunkan

(2) $I \Rightarrow I\ H$ Hb adalah sentensial dan H adalah simple frase

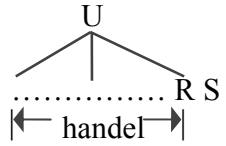
$\Rightarrow I\ b$ (dibandingkan dengan $Q_1\ \beta\ Q_2$ maka $Q_1 = \varepsilon$, $\beta = H$, dan $Q_2 = b$)

$\Rightarrow H\ b$ Perhatikan : simple frase (H) adalah yang terakhir diturunkan

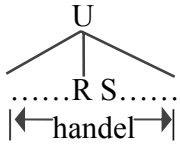
Sentensial Hb mempunyai dua simple frase (b dan H), sedangkan handelnya adalah H.

Relasi Preseden dan Grammar Preseden Sederhana

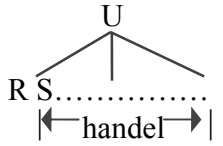
- Relasi preseden adalah relasi antara 2 simbol grammar (baik V_T maupun V_N) dimana paling tidak salah satu simbol tersebut adalah komponen handel.
- Misalkan S dan R adalah 2 simbol. Ada 3 relasi preseden yang : $\leftarrow$, $\leftrightarrow$, dan $\rightarrow$



Relasi : $R \rightarrow S$



Relasi : $R \leftrightarrow S$



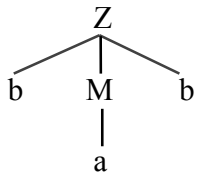
Relasi : $R \leftarrow S$

Perhatikan : komponen handel selalu 'menunjuk' yang simbol lainnya.

Contoh 7 :

Diketahui grammar dengan G = {Z $\rightarrow$ bMb, M $\rightarrow$ (L | a, L $\rightarrow$ Ma)}. Dari 3 sentensial : bab, b(Lb, b(Ma)b, tentukan handel dan relasi yang ada.

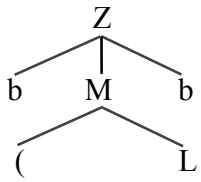
bab



Handel : a

Relasi : $b \leftarrow a$, $a \rightarrow b$

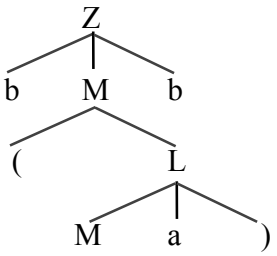
b(Lb



Handel : (L

Relasi : $b \leftarrow ($, $(\leftrightarrow L$, $L \rightarrow b$

b(Ma)b



Handel : Ma)

Relasi : ($\leftarrow$M, M $\leftrightarrow$ a, a $\leftrightarrow$),) $\rightarrow$ b

- Secara umum : jika A $\rightarrow$ aBc adalah sebuah produksi maka :
  - aBc adalah handel dari sentensial yang mengandung string "aBc"
  - relasi preseden antara a, B, dan c adalah : a $\leftrightarrow$ B, B $\leftrightarrow$ c
- Dengan memperhatikan ruas kanan produksi yang ada serta berbagai sentensial yang dapat diderivasi dari Z maka semua relasi preseden tercantum dalam tabel berikut :

	Z	b	M	L	a	(	)
Z							
b			$\leftrightarrow$		$\leftarrow$	$\leftarrow$	
M		$\leftrightarrow$			$\leftrightarrow$		
L		$\rightarrow$			$\rightarrow$		
a		$\rightarrow$			$\rightarrow$	$\leftarrow$	$\leftrightarrow$
(			$\leftarrow$	$\leftrightarrow$	$\leftarrow$	$\leftarrow$	
)		$\rightarrow$					

Grammar G disebut *grammar preseden sederhana* jika :
1. paling banyak terdapat satu relasi antara setiap dua simbolnya
2. tidak terdapat dua produksi produksi dengan ruas kanan yang sama

Parsing Grammar Preseden Sederhana

Prosedur parsing :
1. Buat tabel 3 kolom dengan label : sentensial dan relasi, handel, dan ruas kiri produksi.
2. Tuliskan kalimat (atau sentensial) yang diselidiki pada baris pertama kolom pertama.
3. Dengan menggunakan tabel relasi preseden cantumkan relasi preseden antara setiap dua simbol yang bertetangga.
4. Tentukan handel dari sentensial tersebut. Handel adalah string yang dibatasi "$\leftarrow$" terakhir dan "$\rightarrow$ " pertama jika dilakukan penelusuran dari kiri atau yang saling mempunyai relasi "$\leftrightarrow$". Handel tersebut pastilah merupakan ruas kanan produksi, karena itu tentukan ruas kiri dari handel tersebut.
5. Ganti handel dengan ruas kiri produksinya. GOTO 3.
6. Kalimat yang diselidiki adalah benar dapat diderivasi dari simbol awal jika kolom "ruas kiri produksi" menghasilkan simbol awal.

Contoh 8 :

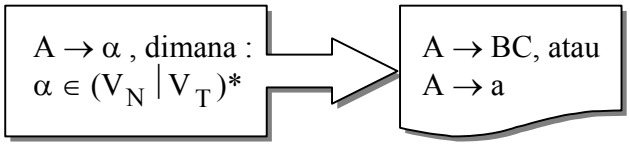
Lakukan parsing atas kalimat x = b(aa)b berdasarkan grammar G di atas.

sentensial dan relasi	handel	ruas kiri produksi
$b \leftarrow (\leftarrow a \rightarrow a \leftrightarrow) \rightarrow b$	a	M
$b \leftarrow (\leftarrow M \leftrightarrow a \leftrightarrow) \rightarrow b$	Ma)	L
$b \leftarrow (\leftrightarrow L \rightarrow b$	(L	M
$b \leftrightarrow M \leftrightarrow b$	bMb	Z

Prosedur parsing sampai di simbol awal (Z). Maka kalimat "b(aa)b" memang dapat diderivasi dari simbol awal Z dengan menggunakan grammar G.

5.2. Bentuk Normal Chomsky

- 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 trnasformasi berikut :



- 4 langkah konversi CFG – CNF adalah sebagai berikut :
  1. Eliminir semua produksi hampa
  2. Eliminir semua produksi unitas
  3. Terapkan prinsip batasan *bentuk* ruas kanan produksi
  4. Terapkan prinsip batasan *panjang* ruas kanan produksi

- Penjelasan Tentang 4 Langkah Konversi

  1. Eliminasi produksi hampa

  Produksi hampa dikaitkan dengan pengertian *nullable*

  Suatu simbol $A \in V_N$ dikatakan *nullable* jika :

  (a) terkait dengan produksi berbentuk : $A \rightarrow \varepsilon$, atau

  (b) terkait dengan derivasi berbentuk : $A \Rightarrow \ldots \Rightarrow \varepsilon$

  Eliminasi yang dilakukan terhadap simbol *nullable* adalah :

  (a) Buang produksi hampa

  (b) Tambahkan produksi lain yang merupakan produksi lama tetapi simbol *nullable*-nya yang di ruas kanan produksi dicoret.

  Contoh 9 :

  Lakukan eliminasi produksi hampa terhadap himpunan produksi berikut :

  $Q = \{ S \rightarrow a \mid Xb \mid aYa,\ X \rightarrow Y \mid \varepsilon,\ Y \rightarrow b \mid X\}$

  Solusi :

  - Simbol *nullable* adalah X (karena $X \rightarrow \varepsilon$) dan Y (karena $Y \Rightarrow X \Rightarrow \varepsilon$)
  - Dua langkah eliminasi simbol *nullable* adalah :
    - langkah (a) menghilangkan produksi $X \rightarrow \varepsilon$
    - langkah (b) menambahkan produksi $S \rightarrow b$ (pencoretan simbol *nullable* X pada produksi $S \rightarrow Xb$) dan produksi $S \rightarrow aa$ (pencoretan simbol *nullable* Y pada produksi $S \rightarrow aYa$)
  - Himpunan produksi setelah dilakukan eliminasi produksi hampa adalah :
    $Q = \{S \rightarrow a \mid Xb \mid aYa \mid b \mid aa,\ X \rightarrow Y,\ Y \rightarrow b \mid X\}$

  2. Eliminasi produksi unitas

  Produksi unitas berbentuk $A \rightarrow B$, dimana $A,B \in V_N$

  - Jika ada produksi berbentuk : $A \rightarrow B$, atau derivasi $A \Rightarrow X_1 \Rightarrow X_2 \Rightarrow \ldots \Rightarrow B$, dan jika ada produksi non-unitas dari B berbentuk : $B \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$, maka eliminasi yang dilakukan akan menghapus produksi $A \rightarrow B$ dan menghasilkan produksi : $A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$.

- Tidak dilakukan eliminasi terhadap derivasi tertutup karena tidak akan menghasilkan produksi baru. Bentuk derivasi tertutup adalah : $A \Rightarrow X_1 \Rightarrow X_2 \Rightarrow ... \Rightarrow A$

Contoh 10 :
Lakukan eliminasi produksi unitas terhadap himpunan produksi berikut :
$Q = \{S \rightarrow A \mid bb,\ A \rightarrow B \mid b,\ B \rightarrow S \mid a\}$
Solusi :
Untuk memudahkan, pisahkan produksi unitas dan non-unitas :
Produksi unitas : $S \rightarrow A$, $A \rightarrow B$, $B \rightarrow S$
Produksi non unitas : $S \rightarrow bb$, $A \rightarrow b$, $B \rightarrow a$
Proses eliminasi yang dilakukan adalah :
$S \rightarrow A$ dan $A \rightarrow b$ menghapus $S \rightarrow A$ dan menghasilkan $S \rightarrow b$
$S \Rightarrow A \Rightarrow B$ dan $B \rightarrow a$ menghasilkan $S \rightarrow a$
$A \rightarrow B$ dan $B \rightarrow a$ menghapus $A \rightarrow B$ dan menghasilkan $A \rightarrow a$
$A \Rightarrow B \Rightarrow S$ dan $S \rightarrow bb$ menghasilkan $A \rightarrow bb$
$B \rightarrow S$ dan $S \rightarrow bb$ menghapus $B \rightarrow S$ dan menghasilkan $B \rightarrow bb$
$B \Rightarrow S \Rightarrow A$ dan $A \rightarrow b$ menghasilkan $B \rightarrow b$
Perhatikan bahwa derivasi $S \Rightarrow A \Rightarrow B \Rightarrow S$ (derivasi tertutup) dan produksi $S \rightarrow bb$ akan menghasilkan produksi $S \rightarrow bb$ yang jelas bukan merupakan produksi baru. Karena itu terhadap derivasi ini tidak dilakukan eliminasi.

3. Penerapan batasan bentuk ruas kanan produksi
Penerapan batasan bentuk ruas kanan produksi adalah mengubah semua bentuk produksi ke dalam 2 bentuk berikut : $A \rightarrow a$ dan $A \rightarrow B_1 B_2 ... B_n$, $n \geq 2$.

Contoh 11:
Terapkan batasan *bentuk* ruas kanan produksi terhadap himpunan produksi berikut :
$Q = \{S \rightarrow Aa, A \rightarrow bAa\}$
Solusi :
- produksi $S \rightarrow Aa$ diubah menjadi : $S \rightarrow AX_a$, $X_a \rightarrow a$
- produksi $A \rightarrow bAa$ diubah menjdi : $A \rightarrow X_b A X_a$, $X_a \rightarrow a$, $X_b \rightarrow b$

sehingga himpunan produksi menjadi :
$Q = \{S \rightarrow AX_a, A \rightarrow X_b A X_a, X_a \rightarrow a, X_b \rightarrow b\}$

4. Penerapan batasan panjang ruas kanan produksi
Penerapan batasan panjang ruas kanan produksi adalah mengubah semua bentuk produksi sehingga panjang untai ruas kanannya ≤ 2.
Contoh 12 :
Terapkan batasan *panjang* ruas kanan produksi terhadap himpunan produksi berikut :
$Q = \{S \rightarrow ABCD \mid ABC,\ B \rightarrow X_b B X_a\}$
Solusi :
- produksi $S \rightarrow ABCD$ diubah menjadi : $S \rightarrow AT_1$, $T_1 \rightarrow BT_2$, $T_2 \rightarrow CD$
- produksi $S \rightarrow ABC$ diubah menjadi : $S \rightarrow AT_3$, $T_3 \rightarrow BC$
- produksi $B \rightarrow X_b B X_a$ diubah menjadi : $B \rightarrow X_b T_4$, $T_4 \rightarrow B X_a$

sehingga himpunan produksi menjadi :
$Q = \{S \rightarrow AT_1 , T_1 \rightarrow BT_2 , T_2 \rightarrow CD, S \rightarrow AT_3 , T_3 \rightarrow BC, B \rightarrow X_b T_4 ,$
$T_4 \rightarrow B X_a \}$

- Contoh 13 : *Penyelesaian Konversi CFG ke CNF*
  Diberikan $Q = \{S \rightarrow AACD , A \rightarrow aAb \mid \varepsilon , C \rightarrow aC \mid a , D \rightarrow aDa \mid bDb \mid \varepsilon \}$
  Transformasikan himpunan produksi tersebut ke dalam bentuk CNF-nya.
  1. *Eliminasi Produksi Hampa*
     Dari bentuk Q di atas, maka simbol *nullable* adalah A dan D. Dua langkah eliminasi yang dilakukan adalah :
     - penghilangan produksi hampa $A \rightarrow \varepsilon$ dan $D \rightarrow \varepsilon$
     - pembentukan produksi hampa dari produksi yang mengandung simbol *nullable* :
       - dari $S \rightarrow AACD$ dibentuk : $S \rightarrow ACD \mid AAC \mid AC \mid CD \mid C$
       - dari $A \rightarrow aAb$ dibentuk : $A \rightarrow ab$
       - dari $D \rightarrow aDa \mid bDb$ dibentuk : $D \rightarrow aa \mid bb$

     Dengan demikian Q berubah menjadi :
     $Q = \{S \rightarrow AACD \mid ACD \mid AAC \mid AC \mid CD \mid C ,$
     $A \rightarrow aAb \mid ab , C \rightarrow ac \mid a , D \rightarrow aDa \mid bDb \mid aa \mid bb \}$

  2. *Eliminasi Produksi Unitas*
     Q hasil langkah pertama di atas mengandung satu produksi unitas : $S \rightarrow C$. Proses eliminasi yang dilakukan adalah :
     $S \rightarrow C$ dan $C \rightarrow ac \mid a$ menghapus $S \rightarrow C$ dan menghasilkan $S \rightarrow ac \mid a$
     Dengan demikian Q berubah menjadi :
     $Q = \{S \rightarrow AACD \mid ACD \mid AAC \mid AC \mid CD \mid ac \mid a ,$
     $A \rightarrow aAb \mid ab , C \rightarrow ac \mid a , D \rightarrow aDa \mid bDb \mid aa \mid bb \}$

  3. *Penerapan Batasan Bentuk Ruas Kanan*
     Setelah langkah 2, ternyata Q masih mengandung produksi-prosuksi yang tidak ber-bentuk $A \rightarrow a$ dan $A \rightarrow B_1 B_2 ... B_n (n \geq 2)$. Produksi-produksi tersebut adalah :
     $S \rightarrow aC$, $A \rightarrow aAb \mid ab$, $C \rightarrow aC$, $D \rightarrow aDa \mid bDb \mid aa \mid bb$. Bentuk-bentuk produksi ini diubah sebagai berikut :
     $S \rightarrow aC$ menjadi $S \rightarrow X_a C$ dan $X_a \rightarrow a$
     $A \rightarrow aAb \mid ab$ menjadi $A \rightarrow X_a A X_b \mid X_a X_b$ dan $X_a \rightarrow a, X_b \rightarrow b$
     $C \rightarrow aC$ menjadi $C \rightarrow X_a C$ dan $X_a \rightarrow a$
     $D \rightarrow aDa \mid bDb \mid aa \mid bb$ menjadi $D \rightarrow X_a D X_a \mid X_b D X_b \mid X_a X_a \mid X_b X_b$
     dan $X_a \rightarrow a, X_b \rightarrow b$
     Bentuk Grammar sampai langkah 3 ini adalah :
     $Q = \{ S \rightarrow AACD \mid ACD \mid AAC \mid AC \mid CD \mid X_a C \mid a , A \rightarrow X_a A X_b \mid X_a X_b ,$
     $C \rightarrow X_a C \mid a , D \rightarrow X_a D X_a \mid X_b D X_b \mid X_a X_a \mid X_b X_b ,$

$X_a \rightarrow a$, $X_b \rightarrow b\}$

4. *Penerapan Batasan Panjang Ruas Kanan*
   Bentuk Q terakhir masih mengandung produksi-produksi dengan panjang untai ruas kanan ≥ 2. Produksi-produksi tersebut adalah : $S \rightarrow AACD \mid ACD \mid AAC$, $A \rightarrow X_a AX_b$, $D \rightarrow X_a DX_a \mid X_b DX_b$. Bentuk-bentuk produksi ini diubah sebagai berikut :
   $S \rightarrow AACD$ menjadi : $S \rightarrow A\,T_1$, $T_1 \rightarrow A\,T_2$, $T_2 \rightarrow CD$
   $S \rightarrow ACD$ menjadi : $S \rightarrow A\,T_2$, $T_2 \rightarrow CD$
   $S \rightarrow AAC$ menjadi : $S \rightarrow A\,T_3$, $T_3 \rightarrow AC$
   $A \rightarrow X_a AX_b$ menjadi : $A \rightarrow X_a\,T_4$, $T_4 \rightarrow AX_b$
   $D \rightarrow X_a DX_a$ menjadi : $D \rightarrow X_a\,T_5$, $T_5 \rightarrow DX_a$
   $D \rightarrow X_b DX_b$ menjadi : $D \rightarrow X_b\,T_6$, $T_6 \rightarrow DX_b$
   Bentuk grammar sampai langkah 4 ini adalah bentuk CNF, yaitu :
   $Q = \{S \rightarrow A\,T_1 \mid A\,T_2 \mid A\,T_3 \mid AC \mid CD \mid X_a C \mid a,$
   $T_1 \rightarrow A\,T_2$, $T_2 \rightarrow CD$, $T_3 \rightarrow AC$,
   $A \rightarrow X_a\,T_4 \mid X_a X_b$, $T_4 \rightarrow A\,X_b$,
   $C \rightarrow X_a C \mid a,$
   $D \rightarrow X_a\,T_5 \mid X_b T_6 \mid X_a\,X_a \mid X_b\,X_b$, $T_5 \rightarrow DX_a$, $T_6 \rightarrow DX_b$,
   $X_a \rightarrow a$, $X_b \rightarrow b\}$

5.3. Automata Pushdown (APD)

- Definisi : PDA adalah pasangan 7 tuple M = (Q, Σ, Γ, q_0, Z_0, δ, A), dimana :
  Q : himpunan hingga stata, Σ : alfabet input, Γ : alfabet *stack*, $q_0 \in Q$: stata awal,
  $Z_0 \in \Gamma$: simbol awal *stack*, $A \subseteq Q$: himpunan stata penerima,
  fungsi transisi $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$ (himpunan bagian dari $Q \times \Gamma^*$)
- Untuk stata $q \in Q$, simbol input $a \in \Sigma$, dan simbol *stack* $X \in \Gamma$, $\delta(q, a, X) = (p, \alpha)$ berarti : PDA bertransisi ke stata p dan mengganti X pada stack dengan string α.
- Konfigurasi PDA pada suatu saat dinyatakan sebagai triple (q, x, α), dimana :
  $q \in Q$: stata pada saat tersebut, $x \in \Sigma^*$: bagian string input yang belum dibaca, dan $\alpha \in \Gamma^*$: string yang menyatakan isi *stack* dengan karakter terkiri menyatakan *top of stack*.
- Misalkan (p, ay, Xβ) adalah sebuah konfigurasi, dimana : $a \in \Sigma$, $y \in \Sigma^*$, $X \in \Gamma$, dan $\beta \in \Gamma^*$. Misalkan pula $\delta(p, a, X) = (q, \gamma)$ untuk $q \in Q$ dan $\gamma \in \Gamma^*$. Dapat kita tuliskan bahwa : $(p, ay, X\beta) \Rightarrow (q, y, \gamma\beta)$.

Contoh 14 (PDA Deterministik):

PDA M = (Q, Σ, Γ, q_0, Z_0, δ, A) pengenal palindrome $L = \{xcx^T \mid x \in (a|b)^*\}$, dimana x^T adalah cermin(x), mempunyai tuple : $Q = \{q_0, q_1, q_2\}$, $A = \{q_2\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{a, b, Z_0\}$, dan fungsi transisi δ terdefinisi melalui tabel berikut :

No.	Stata	Input	*TopStack*	Hasil
1	q_0	a	Z_0	(q_0, aZ_0)
2	q_0	b	Z_0	(q_0, bZ_0)
3	q_0	a	a	(q_0, aa)
4	q_0	b	a	(q_0, ba)
5	q_0	a	b	(q_0, ab)
6	q_0	b	b	(q_0, bb)

No.	Stata	Input	*TopStack*	Hasil
7	q_0	c	Z_0	(q_1, Z_0)
8	q_0	c	a	(q_1, a)
9	q_0	c	b	(q_1, b)
10	q_1	a	a	(q_1, ε)
11	q_1	b	b	(q_1, ε)
12	q_1	ε	Z_0	(q_2, ε)

Sebagai contoh, perhatikan bahwa fungsi transisi No. 1 dapat dinyatakan sebagai : $\delta(q_0, a, Z_0) = (q_0, aZ_0)$. Pada tabel transisi tersebut terlihat bahwa pada stata q_0 PDA akan melakukan PUSH jika mendapat input a atau b dan melakukan transisi stata ke stata q_1 jika mendapat input c. Pada stata q_1 PDA akan melakukan POP.

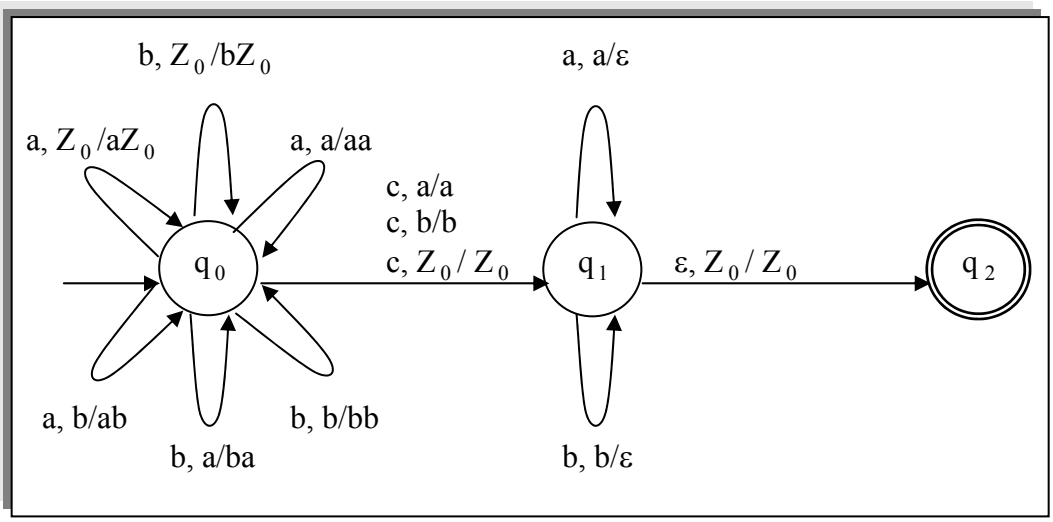
Berikut ini pengenalan dua string oleh PDA di atas :

1. abcba : $(q_0, abcba, Z_0) \Rightarrow (q_0, bcba, aZ_0)$ (1)
   $\Rightarrow (q_0, cba, baZ_0)$ (4)
   $\Rightarrow (q_1, ba, baZ_0)$ (9)
   $\Rightarrow (q_1, a, aZ_0)$ (11)
   $\Rightarrow (q_1, \varepsilon, Z_0)$ (10)
   $\Rightarrow (q_2, \varepsilon, Z_0)$ (12) (*diterima*)
2. acb : $(q_0, acb, Z_0) \Rightarrow (q_0, cb, aZ_0)$ (1)
   $\Rightarrow (q_1, b, aZ_0)$ (8), (*crash* → *ditolak*)
3. ab : $(q_0, ab, Z_0) \Rightarrow (q_0, b, aZ_0)$ (1)
   $\Rightarrow (q_0, \varepsilon, baZ_0)$ (4) (*crash* → *ditolak*)

Penerimaan dan penolakan tiga string di atas dapat dijelaskan sebagai berikut :

1. string abcba diterima karena *tracing* sampai di stata penerima (q_2) dan string "abcba" selesai dibaca (string yang belum dibaca = ε)
2. string acb ditolak karena konfigurasi akhir (q_1, b, a Z_0) sedangkan fungsi transisi $\delta(q_1, b, a)$ tidak terdefinsi
3. string ab ditolak karena konfigurasi akhir (q_0, ε, baZ_0) sedangkan fungsi transisi $\delta(q_0, \varepsilon, b)$ tidak terdefinsi

Ilustrasi graf fungsi transisi PDA di atas ditunjukkan melalui gambar berikut :



- Notasi (p, ay, Xβ) ⇒ (q, y, γβ) dapat diperluas menjadi : (p, x, α) ⇒* (q, y, β), yang berarti konfigurasi (q, y, β) dicapai melalui sejumlah (0 atau lebih) transisi.
- Ada dua cara penerimaan sebuah kalimat oleh PDA, yang masing-masing terlihat dari konfigurasi akhir, sebagaimana penjelasan berikut :
  Jika M = (Q, Σ, Γ, q_0, Z_0, δ, A) adalah PDA dan x ∈Σ*, maka x *diterima dengan stata akhir* (*accepted by final state*) oleh PDA M jika : (q_0, x, Z_0) ⇒* (q, ε, α) untuk α ∈ Γ * dan q ∈ A. x *diterima dengan stack hampa* (*accepted by empty stack*) oleh PDA M jika : (q_0, x, Z_0) ⇒* (q, ε, ε) untuk q ∈ Q.

Contoh 15 (PDA Non-Deterministik):

PDA M = (Q, Σ, Γ, q_0, Z_0, δ, A) pengenal palindrome L = $\{xx^T \mid x \in (a|b)^*\}$ mempunyai komponen tuple berikut : Q = $\{q_0, q_1, q_2\}$, A = $\{q_2\}$, Σ = {a, b}, Γ = {a, b, Z_0}, dan fungsi transisi δ terdefinisi melalui tabel berikut :

No.	St.	In.	*TS*	Hasil
1	q_0	a	Z_0	$(q_0, aZ_0),(q_1, Z_0)$
2	q_0	b	Z_0	$(q_0, bZ_0),(q_1, Z_0)$
3	q_0	a	a	$(q_0, aa),(q_1, a)$
4	q_0	b	a	$(q_0, ba),(q_1, a)$
5	q_0	a	b	$(q_0, ab),(q_1, b)$
6	q_0	b	b	$(q_0, bb),(q_1, b)$

No.	St.	In.	*TS*	Hasil
7	q_0	ε	Z_0	(q_1, Z_0)
8	q_0	ε	a	(q_1, a)
9	q_0	ε	b	(q_1, b)
10	q_1	a	a	(q_1, ε)
11	q_1	b	b	(q_1, ε)
12	q_1	ε	Z_0	(q_2, ε)

Pada tabel transisi tersebut terlihat bahwa pada stata q_0 PDA akan melakukan PUSH jika mendapat input a atau b dan melakukan transisi stata ke stata q_1 jika mendapat input ε. Pada stata q_1 PDA akan melakukan POP. Contoh 14 dan 15 menunjukkan bahwa PDA dapat dinyatakan sebagai mesin PUSH-POP.
Berikut ini pengenalan string "baab" oleh PDA di atas :

1. $(q_0, baab, Z_0)$ $\Rightarrow (q_0, aab, bZ_0)$ (2 kiri)
   $\Rightarrow (q_0, ab, abZ_0)$ (5 kiri)
   $\Rightarrow (q_1, ab, abZ_0)$ (3 kanan)
   $\Rightarrow (q_1, b, bZ_0)$ (11)
   $\Rightarrow (q_1, \varepsilon, Z_0)$ (10)
   $\Rightarrow (q_2, \varepsilon, Z_0)$ (12) (*diterima*)

2. $(q_0, baab, Z_0)$ $\Rightarrow (q_1, baab, Z_0)$ (2 kanan) (*crash* $\rightarrow$ *ditolak*)

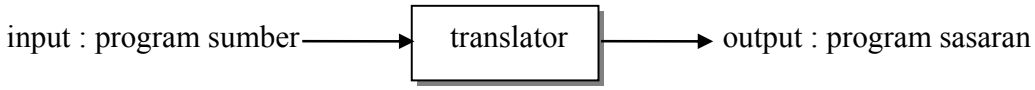
3. $(q_0, baab, Z_0)$ $\Rightarrow (q_0, aab, bZ_0)$ (2 kiri)
   $\Rightarrow (q_0, ab, abZ_0)$ (5 kiri)
   $\Rightarrow (q_0, b, aabZ_0)$ (3 kiri)
   $\Rightarrow (q_1, b, aabZ_0)$ (4 kanan) (*crash* $\rightarrow$ *ditolak*)

4. $(q_0, baab, Z_0)$ $\Rightarrow (q_0, aab, bZ_0)$ (2 kiri)
   $\Rightarrow (q_0, ab, abZ_0)$ (5 kiri)
   $\Rightarrow (q_0, b, aabZ_0)$ (3 kiri)
   $\Rightarrow (q_0, \varepsilon, baabZ_0)$ (4 kiri)
   $\Rightarrow (q_1, \varepsilon, baabZ_0)$ (9) (*crash* $\rightarrow$ *ditolak*)

VI. PENGANTAR KOMPILASI

6.1. Translator

Translator (penerjemah) adalah sebuah program yang menerjemahkan sebuah program sumber (*source program*) menjadi program sasaran (*target program*).

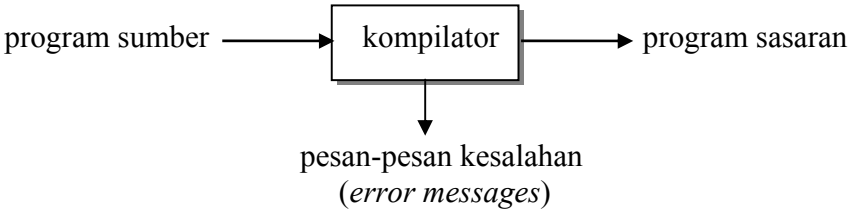


Jenis-jenis translator berdasarkan bahasa pemrograman yang bersesuaian dengan input dan outputnya adalah :

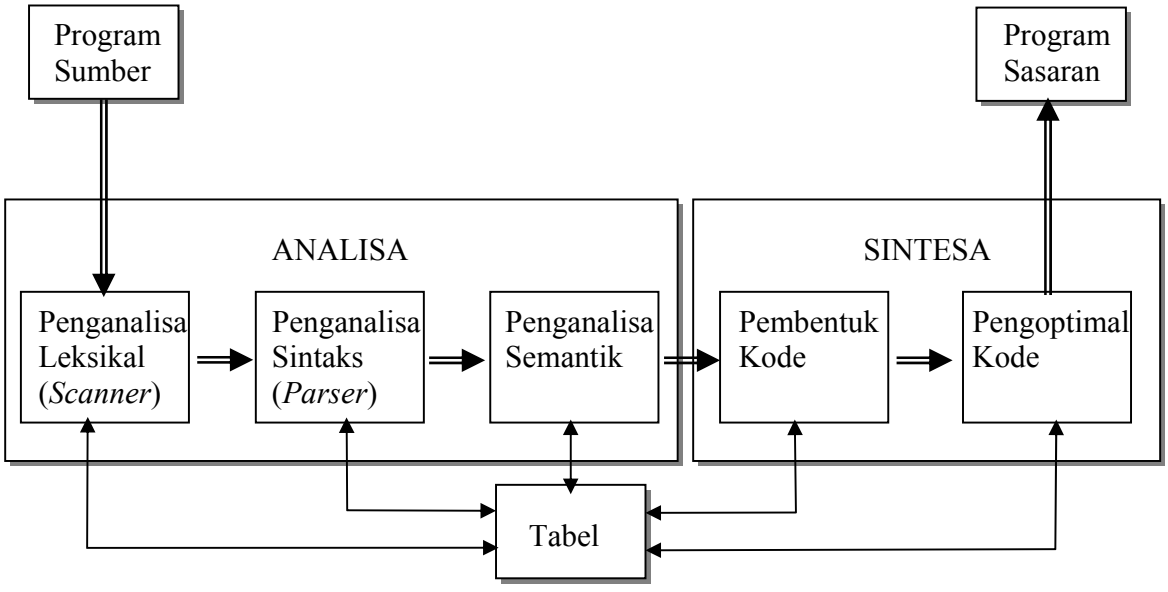
Jenis Translator	**Bahasa Pemrograman**	
	Input	**Output**
Assembler	Bahasa Rakitan	Bahasa mesin
Compiler (Kompilator)	Bahasa tingkat tinggi	Bahasa tingkat rendah

6.2. Kompilator dan komponennya

Black box sebuah kompilator dapat digambarkan melalui diagram berikut :



sedangkan diagram rincinya adalah sebagai berikut :



Proses kompilasi dikelompokkan ke dalam dua kelompok besar :

1. *analisa* : program sumber dipecah-pecah dan dibentuk menjadi bentuk antara (*intermediate representation*)
2. *sintesa* : membangun program sasaran yang diinginkan dari bentuk antara

Program sumber merupakan rangkaian karakter. Berikut ini hal-hal yang dilakukan oleh setiap fase pada proses kompilasi terhadap program sumber tersebut :

1. Penganalisa leksikal :
   membaca program sumber, karakter demi karakter. Sederetan (satu atau lebih) karakter dikelompokkan menjadi satu kesatuan mengacu kepada *pola kesatuan kelompok karakter* (*token*) yang ditentukan dalam *bahasa sumber*. Kelompok karakter yang membentuk sebuah token dinamakan *lexeme* untuk token tersebut. Setiap token yang dihasilkan disimpan di dalam *tabel simbol*. Sederetan karakter yang tidak mengikuti pola token akan dilaporkan sebagai *token tak dikenal* (*unidentified token*).
   Contoh : Misalnya pola token untuk *identifier* I adalah : I = *huruf*(*huruf*|*angka*)*. Lexeme *ab2c* dikenali sebagai token sementara lexeme *2abc* atau *abC* tidak dikenal.
2. Penganalisa sintaks :
   memeriksa kesesuaian *pola deretan token* dengan aturan sintaks yang ditentukan dalam *bahasa sumber*. Deretan token yang tidak sesuai aturan sintaks akan dilaporkan sebagai *kesalahan sintaks* (*sintax error*). Secara logika deretan token yang bersesuaian dengan sintaks tertentu akan dinyatakan sebagai pohon parsing (*parse tree*).
   Contoh : Misalnya sintaks untuk ekspresi *if-then* E adalah : E $\rightarrow$ *if* L *then*, L $\rightarrow$ IOA, I = *huruf*(*huruf*|*angka*)*, O $\rightarrow$ <|=|>|<=|>=, A $\rightarrow$ 0|1|...|9. Ekspresi *if a2 < 9 then* adalah ekspresi sesuai sintaks; sementara ekspresi *if a2 < 9 do* atau *if then a2B < 9* tidak sesuai. Perhatikan bahwa contoh ekspresi terakhir juga mengandung token yang tidak dikenal.
3. Penganalisa semantik :
   memeriksa token dan ekspresi dengan acuan batasan-batasan yang ditetapkan. Batasan-batasan tersebut misalnya :
   a. panjang maksimum token *identifier* adalah 8 karakter,
   b. panjang maksimum ekspresi tunggal adalah 80 karakter,
   c. nilai bilangan bulat adalah -32768 s/d 32767,
   d. operasi aritmatika harus melibatkan operan-operan yang bertipe sama.
4. Pembangkit kode (atau pembangkit kode antara):
   membangkitkan kode antara (*intermediate code*) berdasarkan pohon parsing. Pohon parse selanjutnya diterjemahkan oleh suatu penerjemah, misalnya oleh *penerjemah berdasarkan sintak* (*syntax-directed translator*). Hasil penerjemahan ini biasanya merupakan *perintah tiga alamat* (*three-address code*) yang merupakan representasi program untuk suatu *mesin abstrak*. Perintah tiga alamat bisa berbentuk *quadruples* (*op, arg1, arg2, result*), *tripels* (*op, arg1, arg2*). Ekspresi dengan satu argumen dinyatakan dengan menetapkan *arg2* dengan - (strip, *dash*).
5. Pengoptimal kode :

melakukan optimasi (penghematan *space* dan *waktu komputasi*), jika mungkin, terhadap kode antara.