2 ALGORITMA DAN MESIN KOMPUTASI

2–1 PROSEDUR DAN ALGORITMA

Untuk menyelesaikan suatu masalah, kita dapat mengupayakannya dengan menyusun langkah-langkah berupa instruksi untuk dikerjakan.

Sebuah *PROSEDUR* yang efektif didefinisikan sebagai deretan instruksi yang banyaknya hingga, diskrit dan jelas, serta dapat dikerjakan secara mekanik. Kalau masalah kita merupakan masalah komputasi, maka pengertian "dapat dikerjakan secara mekanik" dapat diartikan bahwa kita dapat membuat suatu program komputer untuk mengerjakan instruksi tersebut.

Sebagai contoh yang mudah, berikut ini adalah prosedur yang dapat kita gunakan ketika kita ingin mengirim surat kepada teman, yakni:

1. Tulis surat, pada secarik kertas surat
2. Ambil sampul surat
3. Masukkan surat ke dalam sampul
4. Tutup sampul surat menggunakan perekat
5. Jika kita ingat alamat teman tersebut, maka tulis alamat pada sampul surat, jika tidak ingat, lebih dahulu pada buku alamat, baru kemudian kita tulis alamat pada sampul surat
6. Tempel perangko pada surat
7. Bawa surat ke kantor pos untuk diposkan.

Ketujuh langkah di atas merupakan prosedur yang efektif. Banyaknya instruksi adalah hingga, masing-masing bersifat diskrit dan jelas, serta dapat kita kerjakan (secara mekanik).

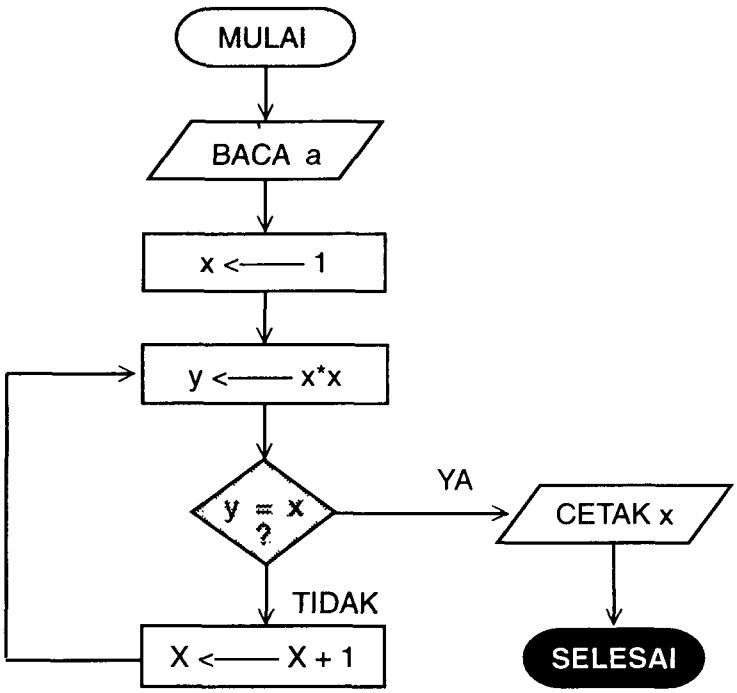
Berikut ini contoh prosedur untuk menyelesaikan masalah komputasi akar bulat positif dari suatu bilangan bulat (integer) positif a, yakni:

1. Baca a
2. Masukkan x sama dengan 1
3. Hitung y sebesar x*x
4. Jika y = a, maka cetak x sebagai akar dari a, selesai.
5. Tambah nilai x dengan 1
6. Pergi ke langkah 3

Selain menyajikan prosedur seperti cara di atas, yang sering disebut sebagai *pseudocode*, kita dapat pula menyajikan dalam bentuk flowchart atau diagram alur dari prosedur, seperti terlihat pada gambar 2-1.

Kita dapat pula membuat sebuah program bagi prosedur ini. Contohnya adalah program dalam bahasa BASIC berikut:

```
10 INPUT A
20 X=1
30 Y=X*X
40 IF Y=A THEN PRINT X:END
50 X=X+1
60 GO TO 30
```



Gambar 2-1

Kalau a kita masukkan nilai 4 misalnya, jelas akan tercetak nilai 2 sebagai akar dari 4. Namun, bila kita masukkan nilai 5 misalnya, prosedur akan berlanjut tak berhenti-henti. Memang, untuk a = 5, akar bulatnya tidak ada.

Suatu prosedur yang bersifat bahwa ia tidak dapat berhenti bila jawaban tidak ada, disebut *SEMI-ALGORITMA*.

Sedangkan prosedur yang mampu berhenti, baik bila jawaban ada ataupun tidak, disebut *ALGORITMA*.

Algoritma akan terhenti dan memberikan pesan bahwa jawaban tidak ada bila memang ternyata jawaban tidak ada.

Jadi prosedur pada contoh kita di atas adalah sebuah Semi-algoritma.

Sebuah semi-algoritma selalu dapat disempurnakan, dengan suatu cara atau dimodifikasi, menjadi sebuah algoritma.

Semi-algoritma contoh kita di atas dengan mudah kita sempurnakan menjadi sebuah algoritma, dengan cara menyelipkan satu instruksi

```
45 IF Y>A THEN PRINT "TAK ADA AKAR":END
```

pada program BASIC kita yang lalu.

Untuk maksud yang sama, silakan anda memodifikasi diagram alur yang bersangkutan.

Berikut ini sebuah Contoh lain dari Semi-algoritma.

Pandang bahwa kita mempunyai mesin yang hanya sanggup melakukan penjumlahan bilangan bulat positif.

Kita ingin mencari hasil pengurangan b-a, untuk 2 bilangan bulat positif b dan a. Di sini sama saja kita menghitung nilai x, sedemikian sehingga a + x = b.

Berikut ini prosedurnya.

1. Baca a, b
2. Masukkan x bernilai 0
3. Jika a + x = b, maka cetak x "adalah b-a", selesai.
4. Tambah nilai x dengan 1
5. Pergi ke langkah 3

Anda dapat melihat bahwa prosedur tidak akan berhenti, bila a > b.

2-2 MESIN TURING

Jauh sebelum lahirnya program komputer, *Alan Turing* pada tahun 1936 telah mengeluarkan gagasannya berupa model mesin abstrak sebagai alat mekanik untuk mengerjakan prosedur yang efektif. Model ini disebut *Mesin Turing*, dan biasa disingkat MT.

Selain mesin Turing ini, masih terdapat beberapa model lain untuk keperluan serupa, di antaranya adalah *masin Post*

Pada bab ini, kedua mesin tersebut kita bahas secara singkat, sebagai suatu pengantar untuk studi yang lebih lanjut.

Cara kerja MT adalah ekivalen dengan cara kerja komputer digital sekarang ini. Ia juga ekivalen dengan problema komputasi matematika.

Sebagai input dari MT adalah kata atau untai atas suatu alfabet T. Ia berhenti dengan keadaan *menerima* atau *menolak* untai. Kadang-kadang terjadi pula perulangan atau looping tak hingga.

Himpunan untai yang diterima oleh mesin Turing M, kita sebut ACCEPT(M), yang ditolak kita sebut REJECT(M), dan yang looping kita sebut LOOP(M).

Jelas bahwa ACCEPT(M) U REJECT(M) U LOOP(M) $=T^*$ dan ACCEPT(M), REJECT(M), LOOP(M) saling lepas.

Dua MT, M1 dan M2 disebut ekivalen jika ACCEPT(M1) = ACCEPT(M2), jadi jelas bahwa REJECT(M1) U LOOP(M1) = REJECT(M2) U LOOP(M2).

Koleksi MT, {M} disebut mempunyai power yang sama dengan koleksi MT, {N} bila untuk setiap mesin M di {M} ada mesin N di {N} yang ekivalen, dan sebaliknya.

Selanjutnya kita katakan {M} > {N}, atau {M} lebih powerful dari {N} jika untuk setiap mesin N di {N} ada mesin M di {M} yang ekivalen, tetapi sebaliknya tak berlaku. .

Sebagai contoh, kita kenal 3 mesin dengan power yang sama, yakni Mesin Turing – Mesin Post – Mesin Finite dengan dua pushdown store (pushdown automata dengan dua pushdown store atau stack).

1–3 PENYAJIAN MESIN TURING

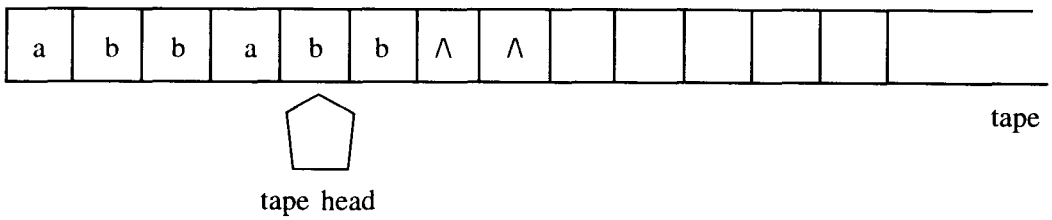
Ada bermacam-macam cara menyajikan MT. Sebelum itu, kita definisikan dahulu suatu MT secara formal.

Mesin Turing M atas alfabeet T, terdiri atas :

1. Tape atau pita yang terbentuk dari deretan sel. Tape mempunyai sel terkiri atau leftmost, tetapi mempunyai tak hingga sel ke kanan. Setiap sel hanya bisa berisi *satu* simbol pita.

   Simbol pita terdiri dari huruf dalam alfabet T, huruf pada alfabet hingga V (alfabet tambahan), serta simbol blank.

2. Tape head atau Kepala Pita, yang mengamati satu sel tape pada satu waktu. Head dapat bergerak. Pada setiap move atau gerak, head mencetak sebuah simbol pada sel yang diamati, menghapus apa yang telah tertulis sebelumnya pada sel itu, lalu move ke kiri atau ke kanan.

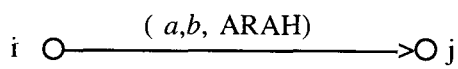


Gambar 2-2

3. Sebuah *program*, merupakan digraph hingga. Ia merupakan sebuah finite control. Simpul digraph merupakan Stata.

   Selalu terdapat Stata Awal yang disebut START, dan sebuah himpunan Stata Akhir (boleh hampa) yang disebut HALT.

   Setiap arkus digraph berbentuk :



Gambar 2-3

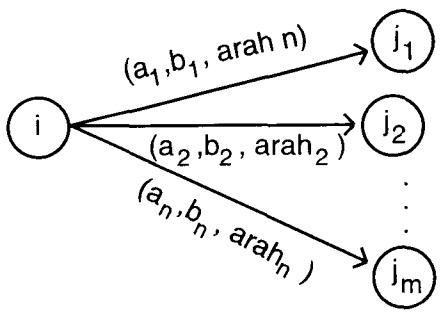
Di sini *a* serta *b* dalam T U V U {blank} (simboltape).

Sedangkan ARAH dalam {L,R}. Ini berarti: Jika selama komputasi kita ada di Stata i, head mengamati simbol *a* , maka kita menuju simpul j, sementara mencetak simbol *b*, kemudian bergerak satu sel ke kiri atau ke kanan, tergantung apakah ARAH = L(kiri) atau R(kanan).

Semua arkus yang keluar dari simpul i harus mempunyai label yang berbeda.

2-4 CARA KERJA MESIN TURING

Mula-mula untai *w* ditempatkan di bagian paling kiri dari tape, sisa di bagian kanan diisi simbol blank. Tape head menunjuk pada leftmost sel. Program bermula pada Stata START. Kalau tercapai Stata HALT, komputasi kita hentikan, untai *diterima* mesin Turing. Apabila sampai pada simpul i dengan situasi sebagai berikut :



Gambar 2–4

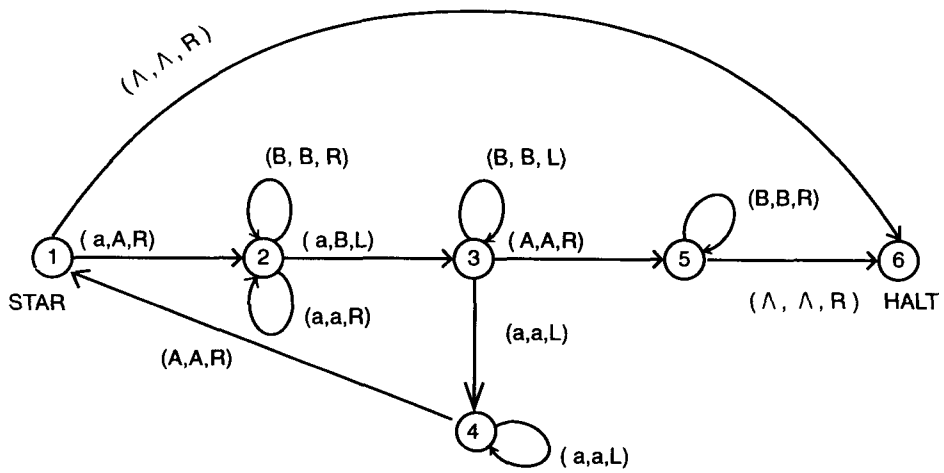
dengan simbol yang diamati bukan salah satu dari $a_1, a_2, ..., a_n$, maka tak ada jalan untuk melanjutkan proses.
Kita hentikan proses, dan untai *w* tersebut ditolak mesin.
Kasus lain penolakan w adalah pada saat tape head mengamati sel paling kiri, ia di-instruksiksan bergerak ke kiri.

Berikut ini contoh MT, M atas alfabet T ={a,b}, yang dirancang untuk dapat menerima semua untai berbentuk

a^nb^n, n >= 0

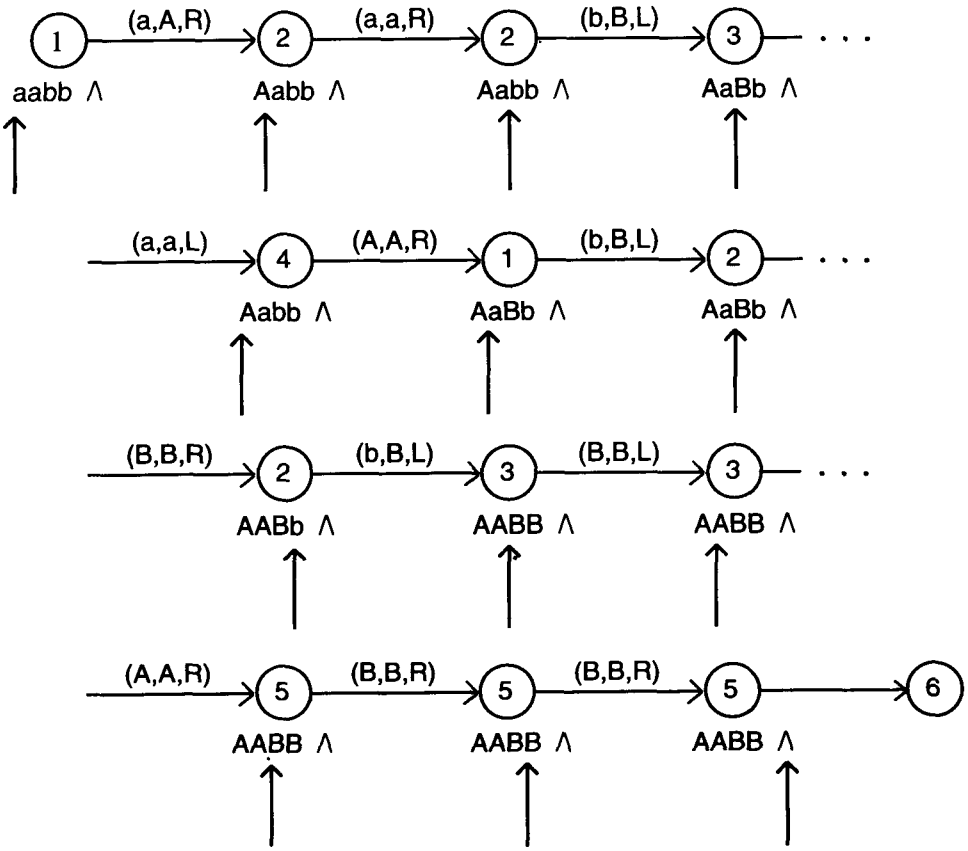
serta menolak untai yang lain.

Jadi ACCEPT(M) = $\{a^nb^n \mid n >= 0\}$
REJECT(M) = T^* – ACCEPT(M), LOOP(M) = hampa



Gambar 2-5

V = {A, B} adalah variabel tambahan.
Perhatikan bagaimana MT di atas menerima untai aabb.



Gambar 2-6

2-5 MESIN POST DAN DIAGRAM ALURNYA

Dalam studi bahasa formal, salah satu masalah yang harus dipecahkan adalah masalah pengenalan bahasa atau languages recognition. Selain menggunakan teknik grammar seperti yang telah kita bicarakan pada bab 1 yang lalu, masih ada lagi teknik pengenalan menggunakan mesin. Untuk itu, orang

membuat berbagai mesin logika, di antaranya kita kenal mesin Turing, mesin Post, automata pushdown, automata hingga, mesin Stata Hingga atau mesin sekuensial lengkap , dan sebagainya.

Masing-masing mesin dipergunakan untuk menguji bahasa jenis tertentu. Automata pushdown misalnya, dipakai untuk mengenali jenis bahasa context-free, sementara automata hingga dipergunakan mengenali bahasa regular. Kedua jenis automata tersebut akan kita bahas nanti dalam bab tersendiri.

Untuk menyajikan mesin tersebut, banyak cara dapat kita lakukan. Ada yang mempergunakan penyajian berupa Tabel, ada yang berupa graph berarah, dan ada pula berupa diagram alur atau flowchart.

Setelah dalam bagian terdahulu kita bahas tentang mesin Turing, dalam bagian ini akan disajikan diagram alur dari mesin Post yang kita gunakan untuk mengenali bahasa $L = \{a^n b^n \mid n >= 0\}$. Jelas anggota L adalah untai hampa, ab, aabb, aaabbb, aaaabbbb, ... dan seterusnya.

Perhatikan 3 fungsi terhadap T^*, sebagai berikut :

* head(x) adalah huruf paling kiri/leftmost dari x
* tail(x) adalah ekor dari x, diperoleh dengan menghapus head(x)
* a.x adalah menyambung huruf a di depan untai x

Sebagai Contoh :

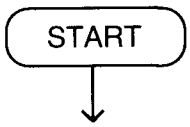
bila alfabet T = {a,b}, maka :
head(baab) = b
tail(baab) = aab
a.baab = abaab
head(Λ) = tail(Λ) = Λ, di sini Λ adalah untai hampa.
untuk setiap huruf h dalam T berlaku head(h) = h, tail(h) = Λ dan h.Λ = h

Fungsi yang berperan penting dalam diagram alur untuk mesin Post ini adalah head(x) dan fungsi tail(x).

Sekarang kita definisikan mesin Post M atas alfabet T = {a,b} adalah suatu diagram alur atau flowchart dengan satu variabel x yang dapat mempunyai harga suatu untai sebarang dengan V x, ditambah pula dengan simbol ## adalah simbol khusus tambahan.

Statement dalam diagram alur terdiri atas :

1. Statement START (hanya satu)

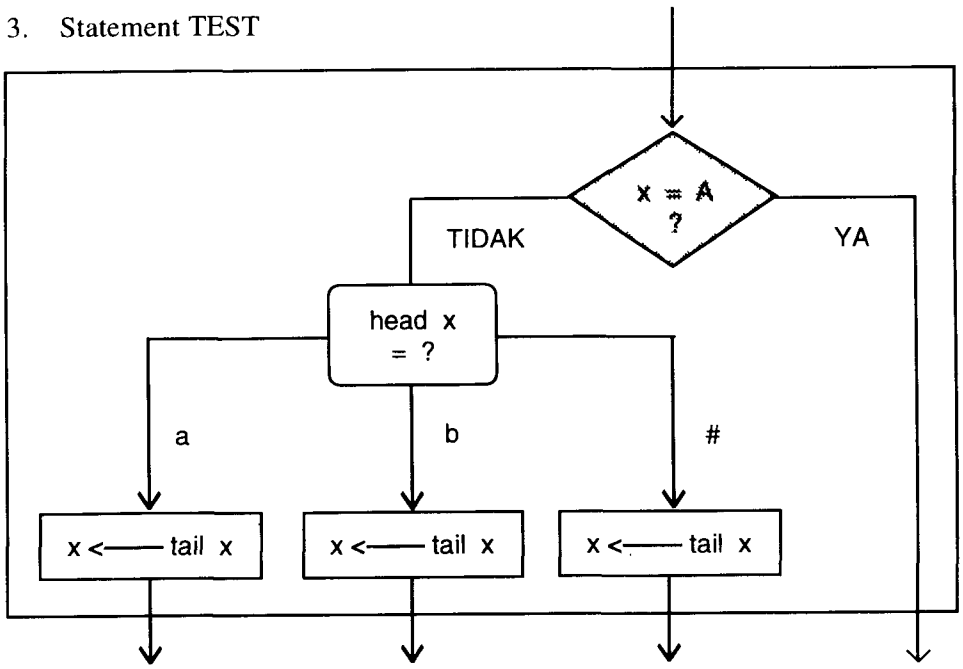


Gambar 2-7

2. Statement HALT

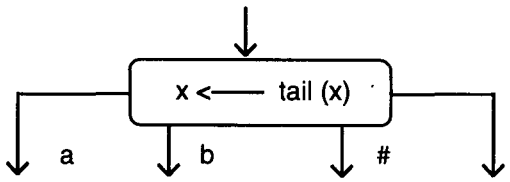
Gambar 2-8

3. Statement TEST



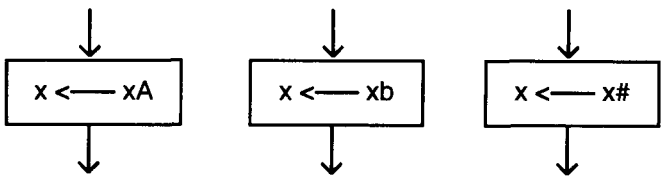
Gambar 2-9

Statement TEST di atas dapat digambar singkat sebagai :



Gambar 2-10

4. Statement ASSIGNMENT :



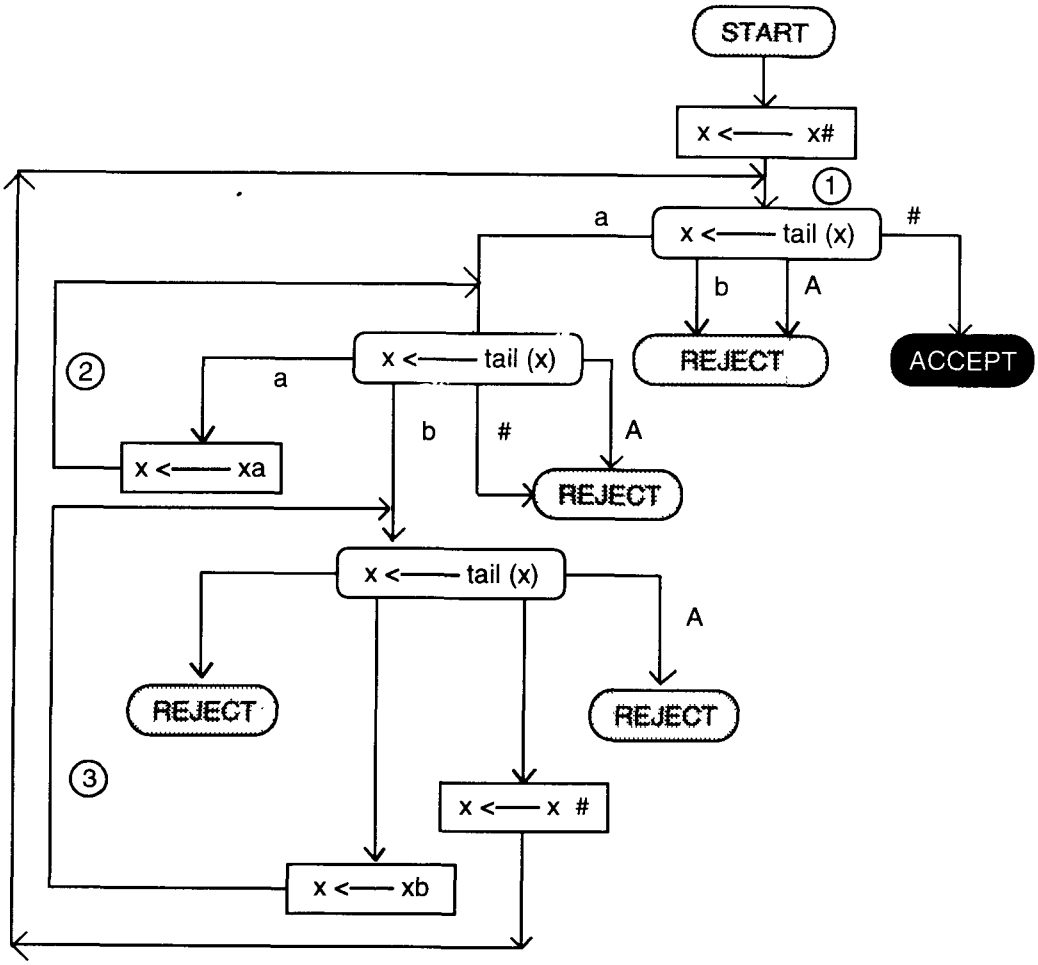
Gambar 2-11

Suatu untai *w* diterima/ditolak oleh mesin Post M, apabila untai *w* dimasukkan sebagai input, yakni x = *w*, penelusuran flowchart akan tiba di ACCEPT/REJECT.

Berikut ini diagram alur mesin Post yang dirancang untuk menerima bahasa $\{a^nb^n, n >= 0\}$, serta menolak
untai lain.

Misal $w = a^nb^n$, $n >= 0$. Kuncinya adalah bila kita sampai ke point (1), x berisi untai $a^ib^i\#$, $0 <= i <= n$. Sekarang bila kita jumpai a sebagai head(x) kita kelilingi loop (2), sebanyak i-1 kali, sampai diperoleh $b^i\#a^{i-1}$. Maka, dilakukan i-1 kali loop (3) sampai diperoleh $\#a^{i-1}b^{i-1}$. Selanjutnya, move # menjadi $a^{i-1}b^{i-1}\#$.
Proses diulang lagi dari point (1).

Silakan anda menggunakan diagram alur di atas untuk menerima untai aabb. Kemudian cobalah amati apabila anda masukkan input untai abb. Tentunya akan sampai ke REJECT.


START
x <— x#
1
a
x <— tail (x)
#
b
A
REJECT
ACCEPT
2
a
x <— tail (x)
b
#
A
x <— xa
REJECT
x <— tail (x)
A
REJECT
REJECT
3
x <— x #
x <— xb

Gambar 2-12

2-6 MESIN STATA HINGGA (FINITE STATE MACHINES)

Sebuah Mesin Stata Hingga, adalah suatu struktur abstrak yang didefinisikan terdiri atas :

(1) Himpunan hingga A berisi simbol input
(2) Himpunan hingga S berisi Stata (internal state)
(3) Himpunan hingga Z berisi simbol output
(4) Sebuah fungsi f : S x A $\rightarrow$ S, disebut fungsi next-state
(5) Sebuah fungsi g : 5 x A $\rightarrow$ Z, disebut fungsi output

Mesin Stata Hingga ini kadang-kadang disebut juga Mesin Sekuensial, atau juga Automata Hingga Beroutput.

Secara lengkap, Mesin M ditulis sebagai M(A,S,Z,f,g), yang kadang-kadang diberikan pula Stata Awal q_0 di dalam S, sehingga M dinyatakan sebagai $M(A,S,Z,q_0,f,g)$.

CONTOH :

Berikut ini contoh Mesin Stata Hingga dengan 2 simbol input, 3 internal Stata, dan 3 simbol output.

(1) $A = (a,b)$
(2) $S = (q_0,q_1,q_2\}$
(3) $Z = (x,y,z\}$
(4) Fungsi next-state, f : S x A $\rightarrow$ S, yang didefinisikan sebagai

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

(5) Fungsi output g : S x A $\rightarrow$ Z, didefinisikan sebagai :

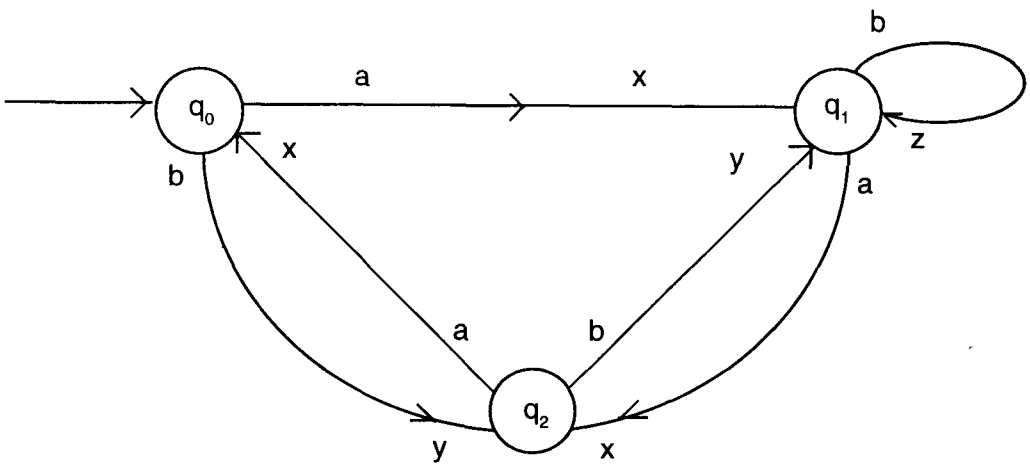
$g(q_0,a) = x$ $\quad$ $g(q_0,b) = y$
$g(q_1,a) = y$ $\quad$ $g(q_1,b) = z$
$g(q_2,a) = z$ $\quad$ $g(q_2,b) = y$

Untuk menyajikan Mesin Stata Hingga, kita mempunyai 2 cara, yaitu dengan Tabel (disebut Tabel Stata), dan dengan Graph Berarah (disebut diagram dari Mesin).

Tabel dari Mesin pada contoh di atas adalah :

	a	b
q_0	q_1,x	q_2,y
q_1	q_2,x	q_1,z
q_2	q_0,z	q_1,y

Diagram Mesin di atas adalah sebagai berikut :



Gambar 2-13

Perhatikan bahwa elemen Tabel baris q_i, kolom a_j. merupakan $f(x_i,a_j)$ dan $g(x_i,a_j)$.

Untuk menggambarkan diagram Mesin, Stata dinyatakan sebagai simpul, dan bila $f(q_i,a_j) = q_k$ dan $g(q_i,a_j) = z_m$,
maka akan terdapat sebuah arkus mengarah dari simpul q_i ke q_k yang berlabel pasangan $(a_j.z_m)$.
Di dalam hal terdapat Stata Awal q_o, kita menambahkan panah mengarah ke q_0 tersebut.

Pandang M suatu Mesin Stata Hingga. Diberikan untai simbol input :

$$U = a_1\ a_2\ \ldots\ a_n$$

Kita bayangkan bahwa simbol input tersebut terdapat dalam "pita input." Mesin membaca simbol tersebut satu persatu dan secara simultan mengubahnya melalui untai Stata :

$$V = s_0\ s_1 s_2\ \ldots\ s_n$$

di sini s_0 adalah Stata Awal, sambil mencetak untai simbol output

$$W = z_1\ z_2 \ldots\ z_n$$

pada sebuah pita output,

Dengan perkataan lain: Stata Awal s_0 dan untai input U menentukan untai V, dan W dengan hubungan :

$$s_i = f(s_{i-1}, a_i)$$

dan $z_i = g(s_{i-1}.a_i)$

dengan i = 1,2, ...,, n

CONTOH

q_o adalah Stata Awal dati Mesin M yang lalu, misalkan M mendapat untai input:

aaba

Kita hendak menentukan untai Stata dan output dari diagram Mesin, dimulai dengan Stata q_o :

$$q_0 \xrightarrow{a} q_1 \xrightarrow{x} q_2 \xrightarrow{a} q_1 \xrightarrow{x} q_2 \xrightarrow{b} q_1 \xrightarrow{y} q_2 \xrightarrow{a} q_1 \xrightarrow{x} q_2$$

yang menghasilkan untai Stata :

$q_0\ q_1\ q_2\ q_1\ q_2$

dan untai output :

xxyx

CONTOH :

Kita hendak menyatakan sebuah mesin yang mampu melakukan penjumlahan binar. Sebagai contoh kita menjumlahkan bilangan 1101011 dengan 0111011.

```
1101011
0111011
-------- +
10100110
```

Dapat dilihat bahwa input merupakan untai sepasang bilangan binar yang akan dijumlah (dari kanan ke kiri) : 11,11,00,11,01,01,ll,lO,b, di sini b menyatakan spasi kosong (blank) dan output merupakan untai:

0,1,1,0,0,1,0,1

Kita inginkan pula bahwa Mesin mempunyai suatu Stata berupa "stop" apabila Mesin menyelesaikan penjumlahan. Jadi dapat kita tentukan :

Simbol input A = (OO,lO,Ol,ll,b) Sinbol output Z =(0,1,b)
Himpunan Stata S terdiri atas 3 Stata yaitu :

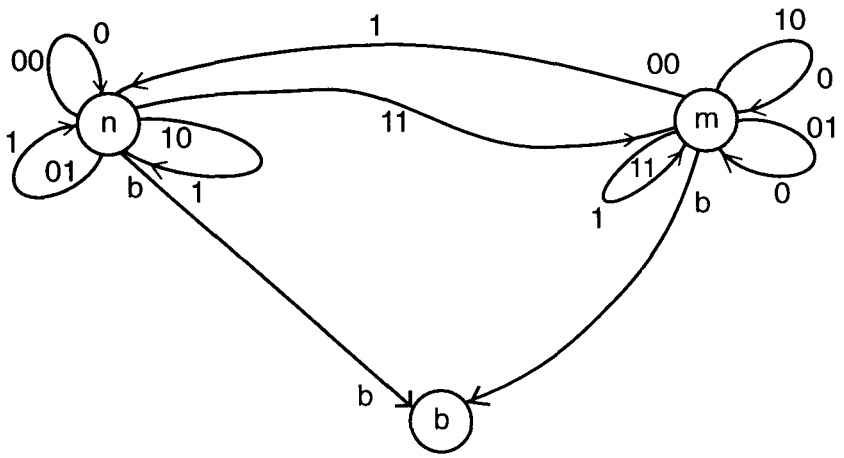
n = "tidak menyimpan," merupakan Stata Awal

m = "menyimpan"

s = "stop"

atau S = (n,m,s)

Diagram mesin ini adalah sebagai berikut:



Gambar 2-14

2–7 PENDEFINISIAN REKURSIF BAGI SEBUAH HIMPUNAN

Definisi rekursif merupakan sebuah metode untuk mendefinisikan anggota himpunan yang ada. Cara kerja dari definisi rekursif pada dasarnya adalah sebuah proses yang terdiri dari tiga langkah. Ketiga langkah tersebut adalah :

* Sebagai langkah Awal kita menentukan beberapa objek dasar dalam himpunan
* Kedua, kita berikan aturan guna penambahan objek dalam himpunan selain yang telah ada pada langkah pertama.
* Selanjutnya kita tetapkan bahwa tidak ada objek lain selain dari objek yang telah dibentuk pada kedua langkah di atas.

Di sini langkah ketiga kadang-kadang tidak kita tulis.

Sebagai contoh, kita akan mendefinisikan himpunan bilangan integer genap yang positif. Kita mempunyai tiga definisi.
Definisi pertama mengatakan bahwa himpunan GENAP adalah himpunan dari semua bilangan bulat positif yang habis dibagi dua. Kita dapat pula mendefisikan dalam definisi kedua berikut, yang mengatakan bahwa himpunan GENAP adalah himpunan dari semua bilangan 2n, dengan n = 1, 2, 3, ...

Pendefinisian rekursif dapat diberikan sebagai berikut.

Aturan 1 : Bilangan 2 adalah anggota himpunan GENAP.
Aturan 2 : Jika x anggota himpunan GENAP, begitu pula x + 2.
Aturan 3 : Elemen dari himpunan GENAP adalah bilangan yang dapat dihasilkan oleh dua aturan di atas.

Perlu dicatat, bahwa pendefinisian ketiga lebih sulit dalam aplikasinya.

Sebagai contoh, kita akan membuktikan bahwa 14 anggota himpunan GENAP. Dengan memakai definisi pertama, 14 dibagi 2 tidak menghasilkan sisa. Oleh sebab itu, 14 ada dalam himpunan GENAP. Dengan memakai definisi kedua, untuk membuktikan 14 anggota himpunan GENAP kita harus mempunyai bilangan 7 terlebih dahulu. Karena 14 = (2)(7), kita dapat katakan bahwa 14 anggota himpunan GENAP. Dengan memakai definisi rekursif pembuktian akan lebih panjang. Hal itu akan dijabarkan sebagai berikut.

Dari aturan pertama, 2 anggota himpunan GENAP. Kemudian dari aturan kedua didapat 2 + 2 = 4 yang merupakan anggota himpunan GENAP. Ulangi kembali aturan kedua, didapat 4 + 2 = 6 yang merupakan anggota himpunan GENAP. Kita kerjakan aturan kedua ini sebanyak empat kali lagi, sehingga diperoleh 12 + 2 = 14, yang memang ada dalam himpunan GENAP.

Cara kerja dari pendefinisian rekursif untuk contoh himpunan GENAP di atas hanyalah merupakan salah satu cara. Cara lain yang dapat dipakai adalah sebagai berikut.

Himpunan GENAP didefinisikan dengan tiga aturan.
Aturan 1 : 2 anggota himpunan GENAP.
Aturan 2 : Jika x dan y anggota himpunan GENAP, begitu pula x + y
Aturan 3 : Tidak ada bilangan dalam himpunan GENAP, kecuali jika dihasilkan oleh aturan pertama dan kedua.

Jadi, jika kita hendak membuktikan 14 ada dalam himpunan GENAP, penjabaran adalah sebagai berikut.

Oleh aturan pertama : 2 anggota himpunan GENAP.

Oleh aturan kedua : x = 2, y = 2, maka 4 anggota himpunan GENAP.
x = 2, y = 4, maka 6 anggota himpunan GENAP.
x = 2, y = 6, maka 8 anggota himpunan GENAP.
x = 6, y = 8, maka 14 anggota himpunan GENAP.

Cara kedua ini lebih baik dari cara pertama tadi, karena pembuktiannya lebih pendek. Bagaimanapun juga, cara kedua dari pendifinisian rekursif ini lebih sulit dipakai daripada kedua definisi non-rekursif yang terdahulu, walaupun ada beberapa keuntungannya.

Jadi dalam menetapkan pendefinisian rekursif, kita tergantung pada dua hal. Kedua hal tersebut adalah :

(i) Pendefinisian yang mudah dimengerti, dan

(ii) Jenis teorema yang dipakai untuk pembuktian.

Alasan mengapa definisi ini disebut rekursif adalah bahwa aturan dalam mendefinisikan himpunan adalah menggunakan himpunan itu sendiri, seperti yang telah dijelaskan dalam contoh yang lalu. Dalam bahasa komputer, jika ada prosedur memanggil prosedur itu sendiri kita sebut program itu rekursif.

CONTOH

Contoh 1

Aturan 1 : Λ anggota L_1
Aturan 2 : Jika Q sembarang untai dalam L_1 maka xQ juga ada dalam L_2.

Di sini akan ternyata bahwa L_1 adalah

$$L_1 = x^* = [\ \Lambda, x, xx, xxx, xxxx, ...]$$

Contoh 2

Aturan 1 : x anggota L_2
Aturan 2 : Jika Q sembarang untai dalam L_2 maka xxQ juga ada dalam L_2.

Di sini akan ternyata bahwa L_2 adalah

$$L_2 = [x^m \mid m \text{ ganjil}\} = [\ x,\ xxx,\ xxxxx,\ ...]$$

Contoh 3

Aturan 1 : 1, 2, 3, 4, 5, 6, 7, 8, dan 9 anggota L_3.
Aturan 2 : Jika Q sembarang untai dalam L_3, maka Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9 adalah untai dalam L_3.

$L_3 = [\ 1, 2, 3, 4\ ...\ ]$

Di sini L_3 adalah himpunan Integer positif.

Contoh 4

Bentuk yang bagaimanakah yang dapat dimengerti oleh komputer dalam pembentukan ekspresi aritmetika yang berlaku, dan yang dapat ditulis dalam satu baris?

Alfabet untuk bahasa ini adalah

I = { 0 1 2 3 4 5 6 7 8 9 + - * / ( ) }

Tentunya untai berikut ini kurang baik :

(3+5)+6)

2*(/8+9)

(3+(4-)8)

2)-(4

Ekspresi pertama mengandung tanda kurung (parantesis) yang tidak setimbang, sedangkan ekspresi kedua mengandung sub-untai yang tidak diperbolehkan. Ekspresi ketiga juga mengandung sub-untai yang tidak diperbolehkan. Ekspresi keempat mengandung parantesis tutup sebelum parantesis buka. Dalam mendefinisikan sebuah ekspresi aritmetika EA yang berlaku, sebaiknya kita memakai definisi rekursif.

Definisi dapat ditulis sebagai berikut :

Aturan 1 : Sebarang bilangan, baik positif, negatif, ataupun nol anggota EA
Aturan 2 : Jika x anggota EA, begitu pula (x) dan -(x)
Aturan 3 : Jika x dan y anggota EA, begitu pula
(i) x + y , asalkan simbol pertama dalam y tidak -
(ii) x - y , asalkan simbol pertama dalam y tidak -
(iii) x*y
(iv) x/y
(v) x**y (merupakan notasi eksponensial)

Definisi ini merupakan definisi yang "paling layak."

Jika kita mempunyai ekspresi

(2+4)*(7*(9-3)/4)/4*(2+8)-1

maka untuk menentukan apakah ekspresi ini valid atau tidak, kita tidak mengamati segenap untai untuk mencari subuntai yang tidak diperbolehkan, ataupun menghitung parantesisnya.

Di sini dapat kita lakukan sebagai berikut :

2 4 7 9 3 8 1	adalah EA (aturan 1)
• 2+4	adalah EA (aturan 3(i))
(2+4)	adalah EA (aturan 2)
9–3	adalah EA (aturan 3(ii))
(9–3)	adalah EA (aturan 2)
7*(9–3)	adalah EA (aturan 3(iii))
7*(9–3)/4	adalah EA (aturan 3(iv))
(7*(9–3)/4)	adalah EA (aturan 2)
(2+4)*(7*(9–3)/4)	adalah EA (aturan 3(iii))
(2+4)*(7*(9–3)/4)/4	adalah EA (aturan 3(iv))
2+8	adalah EA (aturan 3(i))
(2+8)	adalah EA (aturan 2)
(2+4)*(7*(9-3)/4)/4*(2+8)	adalah EA (aturan 3(iii))
(2+4)*(7*(9-3)/4)/4*(2+8)–1	adalah EA (aturan 3(ii))

Definisi di atas memberi pula kemungkinan penulisan 2+3+4 yang tidak memberi dua pengertian. Lain halnya dengan 8/4/2. Karena di sini 8/(4/2) = 4, sedangkan (8/4)/2 = 1, maka di sini terdapat dua pengertian. Begitu pula halnya 3+4*5 yang memberi dua pengertian. Dengan memakai aturan 2 dari definisi, kita dapat memakai parantesis secukupnya, untuk menghindari terjadinya kesalahan. Jadi masalah pengertian ganda dalam untai 8/4/2 adalah tergantung dari pengertian kita. Sehingga tidak ada keraguan kita bahwa untai adalah anggota EA.

Definisi di atas menentukan himpunan EA dalam suatu aturan, yang dapat berguna untuk pembuktian teorema tentang ekspresi aritmetika. Kita mempunyai tiga teorema berikut ini:

Teorema 1

Sebuah ekspresi aritmetika tidak dapat mengandung $.

Bukti :

$ tidak merupakan bagian dari bilangan apapun, jadi tidak dapat dimasukkan dalam EA oleh aturan 1. Jika untai x tidak mengandung $, begitu pula untai (x) dan -(x); sehingga tidak dapat dimasukkan dalam EA oleh aturan 2. Jika baik x ataupun y tidak mengandung $, begitu pula ekspresi yang didefinisikan oleh aturan 3. Sehingga $ tidak dapat masuk ke dalam EA.

Teorema 2

Tidak ada EA yang dimulai atau diakhiri oleh simbol /

Bukti :

Tidak ada bilangan yang dimulai atau diakhiri oleh simbol /. Jadi oleh aturan 1, hal ini tidak akan muncul dalam EA. Sembarang EA yang dibentuk oleh aturan 2 harus dimulai dan diakhiri oleh parantesis atau dimulai oleh sebuah tanda minus, sehingga / tidak dapat dimasukkan sebagai simbol Awal ataupun simbol Akhir dalam EA oleh aturan 2. Jika x tidak dimulai oleh sebuah /, serta y tidak diakhiri oleh sebuah /, maka EA yang dibentuk melalui aturan 3 tidak akan dimulai atau diakhiri oleh sebuah /. Oleh karena itu, aturan tersebut tidak akan pernah menghasilkan sebuah ekspresi yang dimulai atau diakhiri oleh sebuah /.

Terorema 3

Tidak ada EA yang dapat mengandung //

Bukti :

Dengan Kontradiksi. Misalkan ada beberapa EA yang mengandung sub untai //. Ambil yang paling pendek (mungkin ada beberapa; pilih satu saja), sebut untai itu sebagai w. Ini berarti w adalah EA yang valid dan mengandung //, tetapi tidak ada untai lain yang lebih pendek dalam EA yang mengandung subuntai // ini.

Kita ketahui bahwa w dibentuk melalui sederetan penggunaan aturan 1, 2, dan 3. Timbul pertanyaan: aturan terakhir mana yang dipakai dalam menghasilkan w? Jawabnya mudah, yaitu aturan 3 (iv). Akan dijelaskan: Misalkan bukan 3(iv), tetapi 3(iii); maka // harus terdapat di bagian x atau di bagian y saja. Akan tetapi x dan y diasumsikan adalah anggota EA> Ini berarti terdapat beberapa untai dalam EA yang lebih pendek dari w, yang mengandung //. Hal tersebut mengkontradisksikan asumsi bahwa w adalah kata terpendek. Oleh karena itu, aturan terakhir yang dipakai haruslah aturan 3(iv). Dalam hal ini adanya // dalam w dapat terjadi kalau bagian x berakhir dengan / atau bagian y dimulai oleh sebuah /. Tetapi karena x dan y adalah anggota EA, berdasar teorema (2) hal tersebut tidak dapat terjadi. Jadi, aturan 3(iv) tidak dapat memasukkan subuntai // dalam EA. Karena itu, tidak ada kemungkinan bagi syarat terakhir untuk dapat membentuk w. Karena itu w tidak terdapat dalam EA. Sehingga, tidak ada EA yang terpendek yang mengandung subuntia //. Dengan kata lain, tidak ada satupun anggota himpunan EA dapat mengandung subuntai //.

Contoh 5

Penggunaan lain dari pendefinisian rekursif adalah pada penentuan ekspresi yang valid dalam logika simbolik. Khususnya pada cabang dari logika simbolik yang disebut kalkulus kalimat atau kalkulus proposisi. Di sini kita membatasi hanya memakai negasi - dan implikasi $\rightarrow$ dengan beberapa variabel kalimat, walaupun konjungsi dan disjungsi dapat pula dipakai dalam sistem. Ekspresi yang valid dalam bahasa ini disebut WFF, kependekan dari *well formed formulas*.Di sini kita menggunakan alfabet

$$E = \{- \rightarrow ( ) \; a \; b \; c \; d \; ... \; z\}$$

Aturan dalam pembentukan WFF adalah :

Aturan 1 : Sebarang huruf latin tunggal adalah sebuah WFF a b c d ... z
Aturan 2 : Jika p adalah sebuah WFF, begitu pula (p) dan –p
Aturan 3 : Jika p dan q adalah WFF, begitu pula untuk p → q

Beberapa kali penggunaan dari aturan ini dapt membantu kita untuk memperlihatkan :

p → ((p → p) → q) adalah sebuah WFF.

Tanpa kesulitan kita dapat memperlihatkan bahwa :

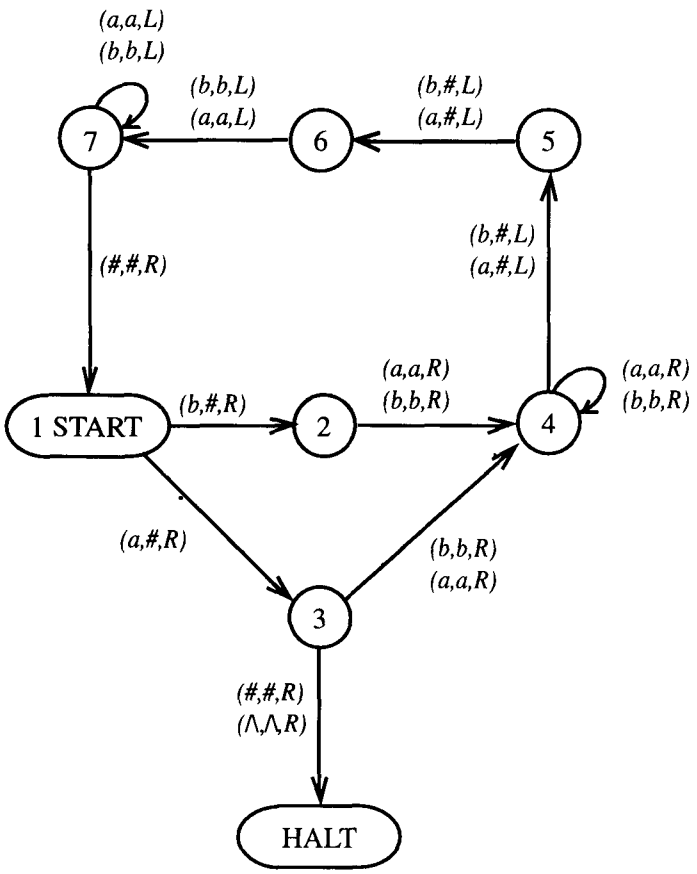
p →
→ p
(p →
p)
p) →
p(

semuanya bukan merupakan WFF.

SOAL LATIHAN

MESIN TURING

Untuk Soal 2-1 dan 2-2, pandang MT :



2-1 Telusuri pelaksanaan dari untai input berikut ini, pada mesin di atas

(i) aaa
(ii) aba
(iii) baaba
(iv) ababb

2-2 Bahasa yang diterima oleh MT di atas adalah semua untai yang terdiri atas sejumlah ganjil huruf dengan a adalah huruf tengahnya. Tunjukkan bahwa hal ini benar, dengan menerangkan algoritma yang dipergunakan oleh mesin, dan arti dari masing-masing Stata.

2-3 (i) Buatlah sebuah MT yang menerima Bahasa dari semua untai yang berisi subuntai *bbb*.

(ii) Buatlah sebuah MT yang menerima Bahasa dari semua untai yang tidak berisi subuntai *bbb*.

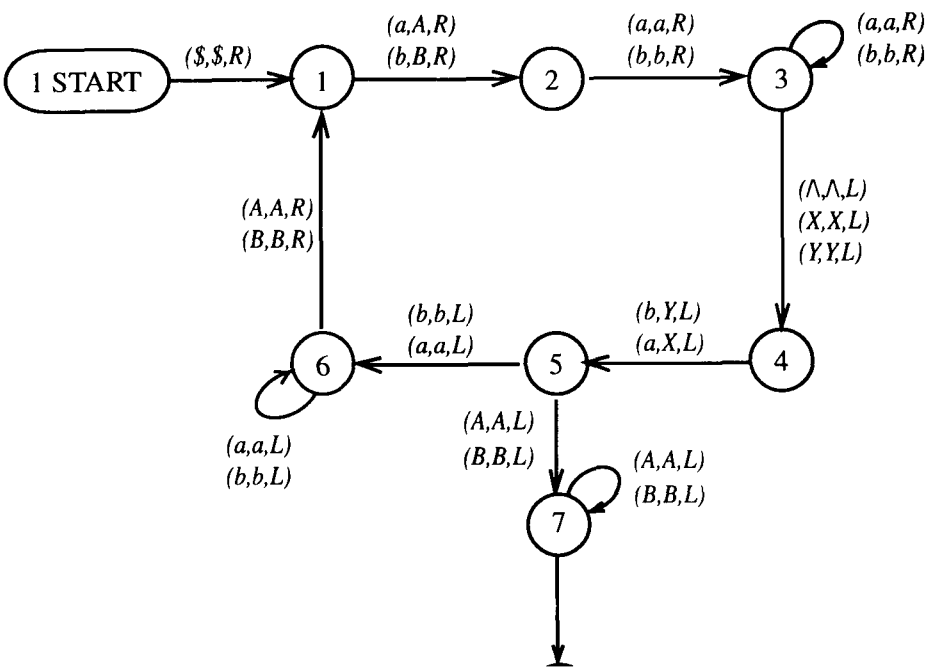
2-4 Buatlah sebuah MT yang menerima Bahasa ODDPALINDROME.

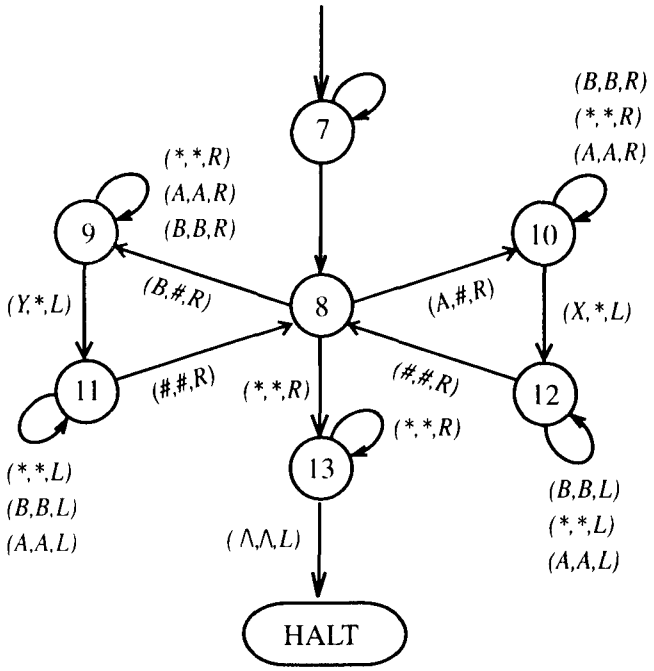
2-5 Buatlah sebuah MT yang menerima semua untai dengan jumlah a lebih banyak dari b.

2-6 (i) Buatlah sebuah MT yang menerima Bahasa $\{a^n b^{n+1}\}$.

(ii) Buatlah sebuah MT yang menerima Bahasa $\{a^n b^{2n}\}$.

Soal 7, 8, dan 9 mengacu kepada MT berikut ini





Kita asumsikan bahwa input untai ditaruh pada Tape dengan simbol # disisipkan di depannya, yakni pada sel pertama. Sebagai contoh, input ba akan dijalankan dengan Tape bermula dalam bentuk #ba.....

2-7 Telusuri pelaksanaan dari untai input berikut ini, pada mesin di atas

(i) aa
(ii) aaa
(iii) aaaa
(iv) aabaab
(v) abab

2-8 Bahasa dari MT ini adalah DOUBLEWORD, himpunan dari semua untai berbentuk ww dengan w adalah untai tidak nol dalam {a,b}.

DOUBLEWORD = {aa bb aaaa abab baba bbbb aaaaaa aabaab ...}

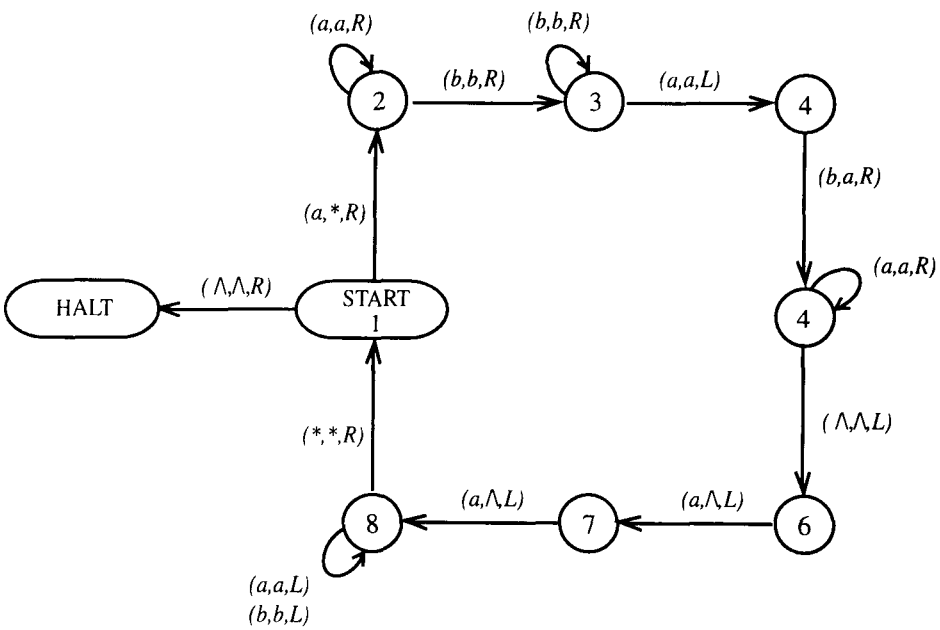
(i) Jelaskan peranan dari masing-masing Stata dan buktikan bahwa semua untai dalam DOUBLEWORD diterima oleh MT ini.

(ii) Tunjukkan bahwa semua untai bukan anggota DOUBLEWORD ditolak oleh mesin ini.

2-9 (i) Tunjukkan bahwa Stata 11 dan 12 dapat dikombinasi tanpa mengubah Bahasa.

(ii) Perubahan lain yang mana dapat dibuat?

2-10 Pandang MT untuk Bahasa $\{a^n b^n a^n\}$ berikut ini.



Telusuri untai berikut:

(i) aabbaa

(ii) aabbaaa

(iii) aabaa

(iv) aabbaabb

MESIN STATA HINGGA

2-11 Sebuah Mesin Stata Hingga mempunyai simbol Input a, b, c dan simbol Output x, y, z.

Diagram Mesin tersebut adalah sebagai berikut :

(a) Tentukan Tabel dari Mesin M tersebut.

(b) Bagaimana Outputnya, jika diberikan Input berupa untai aacabacca

2-12 M adalah Mesin Stata Hingga dengan Tabel sebagai berikut:

	a	b
q_0	q_0,y	q_1,z
q_1	q_2,x	q_2,y
q_2	q_2,y	q_1,z
q_3	q_2,z	q_0,x

(a) Tentukan Diagram Mesin tersebut

(b) Tentukan Output yang dihasilkan jika Input adalah untai aaabbaabab

PENDEFINISIAN REKURSIF

2-13 Selidikilah bahwa definisi rekursif ini merupakan juga definisi bagi himpunan bilangan GENAP positif.

Aturan 1 : 2 dan 4 adalah anggota GENAP

Aturan 2 : Jika x anggota GENAP maka demikian pula halnya x+4

Kemudian tunjukkan bahwa 20 adalah bilangan GENAP

2-14 Tunjukkan bahwa terdapat tak hingga banyaknya definisi rekursif untuk himpunan GENAP

2-15 Dengan menggunakan suatu definisi rekursif, tunjukkan bahwa setiap anggota GENAP selalu berakhir dengan digit 0, 2, 4, 6, atau 8

2-16 Selidikilah bahwa definisi rekursif ibi merupakan definisi bagi himpunan polinomial, yang kita sebut POLI

Aturan 1 : Semua bilangan real adalah POLI

Aturan 2 : Variabel x adalah POLI

Aturan 3: Jika p serta q adalah POLI maka demikian pula halnya p+q, (p), serta pq

Kemudian tunjukkan bahwa $x^2 + 7x - 5$ adalah POLI.

2-17 Berikan definisi rekursif untuk polinom dengan dua variabel x dan y

2-18 Perhatikan definisi dari Fungsi Aljabar atau ALEX (algebraic expression)

Aturan 1 : Setiap polinom adalah ALEX

Aturan 2 : Jika f(x) serta g(x) adalah ALEX, maka demikian pula halnya :

* (f(x))
* –(f(x))
* f(x) + g(x)
* f(x) – g(x)
* f(x)g(x)
* f(x)/g(x)
* $f(x)^{g(x)}$
* f(g)x))

Kemudian tunjukkan bahwa $(3x + 4)^{3x}$ adalah ALEX

2-19 Himpunan {ALINDROME adalah himpunan yang bila dibaca dari kiri ke kanan akan sama dengan bila dibaca dari kanan ke kiri. Contohnya KATAK, KOKOK, TAMAT, dan sebagainya

Berikut ini sebuah definisi rekursif terhadap PALINDROME GANJIL (kita singkat PALJIL) atas alfabet { a,b }:

Aturan 1 : a dan b adalah PALJIL

Aturan 2 : Jika x anggota PALJIL maka demikian pula halnya axa serta bxb

Tunjukkan bahwa aababababaa serta baababaab adalah anggota PALJIL.

2-20 Buatlah suatu definisi rekursif untuk PALINDROME GENAP, yakni palindrome yang panjang setiap untai anggotanya adalah genap.

2-21 Buatlah definisi rekursif untuk PALINDROME secara umum.

2-22 Buatlah definisi rekursif untuk himpunan bilangan GANJIL positif {1, 2, 3, 4, ...}.

2-23 Buatlah definisi rekursif untuk himpunan S*, jika S={aa, b}

2-24 Buatlah dua buah definisi rekursif untuk himpunan 2-PANGKAT, yakni {1, 2, 4, 8, ...}

2-25 Dengan menggunakan salah satu definisi anda di atas, buktikan bahwa hasilkali dua bilangan 2-pangkat adalah bilangan 2-pangkat pula.

2-26 Buatlah definisi rekursif untuk himpunan untai atas alfabet {a,b}, sebagai berikut:

1. Himpunan untai dengan panjang genap
2. Himpunan untai dengan panjang ganjil
3. Himpunan untai yang mengandung subuntai aa
4. Himpunan untai yang tidak mengandung subuntai bb

2-27 Perhatikan definisi rekursif untuk PERMUTASI-3, yakni

Aturan 1 : 123 adalah PERMUTASI-3

Aturan 2 : Jika xyz adalah PERMUTASI-3, maka demikian pula halnya yzx serta zyx

Tentukan keenam anggota PERMUTASI-3 tersebut.

2-28 Perhatikan definisi rekursif untuk PERMUTASI-4, yakni

Aturan 1 : 1234 adalah PERMUTASI-4

Aturan 2 : Jika xyzw adalah PERMUTASI-4, maka demikian pula halnya wzyx serta zwyx

Tentukan berapa banyak anggota PERMUTASI-4 tersebut, serta sebutkan mereka itu.