PERTEMUAN 4 :

FINITE STATE AUTOMATA (FSA) / AUTOMATA BERHINGGA

A. TUJUAN PEMBELAJARAN

Pada bab ini akan dibahas mengenai bentuk suatu mesin otomata berhingga, definisi nya, serta sifatnya. Setelah mengikuti perkuliahan ini mahasiswa diharapkan mampu :

1) Menjelaskan Konsep FSA dalam suatu komputasi
2) Menjabarkan Definisi dari FSA
3) Menganalisa perbedaan antara satu string dengan string lainnya dalam suatu FSA

B. URAIAN MATERI

1. Konsep FSA Dalam Suatu Komputasi

Sebelum mengenal mesin otomata dengan definisi lebih formal, berikut ini contoh yang lebih alami yang menunjukan adanya suatu konsep otomata dalam suatu kegiatan. Bagaimana masalah untuk mendesain sebuah “komputer” menjadi menjual minuman secara otomatis yang.

Ketika ada seseorang pembeli yang ingin membeli minuman dingin pada mesin penjual minuman otomatis ini, maka pembeli ini harus memasukan uang dalam bentuk kertas dengan sejumlah tertentu agar dapat memproses pembelian pada mesin. asumsikan bahwa harga minuman adalah 5.000 dan mesin dapat hanya menerima uang kertas dalam pecahan 1.000, 2.000, dan 5.000. Asumsikan juga bahwa tidak akan memberikan uang kembali setelah menerima uang kecuali kondisi uang rusak sehingga tidak bisa membaca nilai uang.

Pembeli harus menentukan minuman yang ingin dibeli dan akan menyiapkan uang untuk selanjutnya diproses untuk memproses pembelian minuman tersebut. Dalam kondisi apapun, mesin harus mampu menentukan berhasil atau tidaknya proses pembelian untuk mengeluarkan minuman yang diorder. Untuk menentukan hal

tersebut, maka mesin harus berada pada satu state[1] dari 6 (enam) state, pada suatu waktu tertentu akan ada proses berikut :

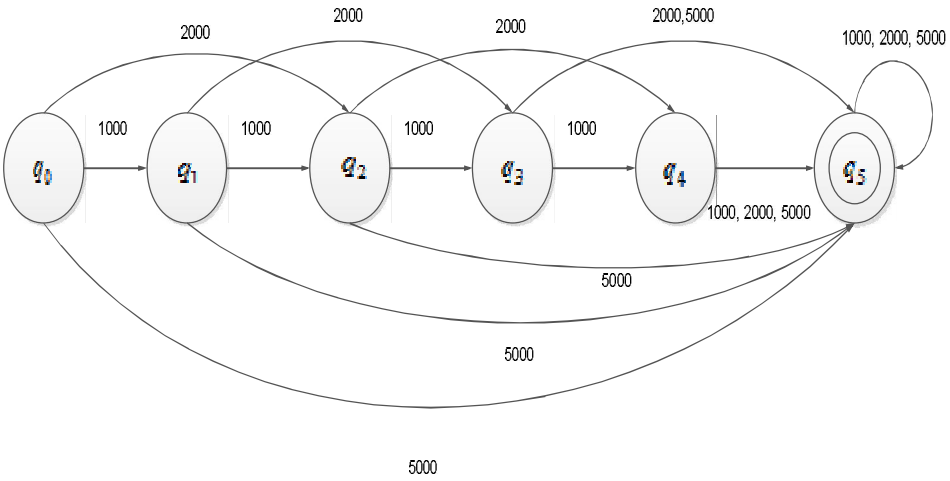
- Mesin pada kondisi state , jika belum menerima uang kertas.
- Mesin pada kondisi state , jika sudah tepat menerima uang kertas 1000.
- Mesin pada kondisi state , jika sudah tepat menerima uang kertas 2000
- Mesin pada kondisi state , jika sudah tepat menerima uang kertas 3000
- Mesin pada kondisi state , jika sudah tepat menerima uang kertas 4000
- Mesin pada kondisi state , jika sudah tepat menerima uang kertas 5000 atau lebih

Inisialisasi[2], mesin pada state . Asumsi nya, untuk contoh penerimaan uang dari pembeli minuman menyediakan uang kertas senilai 2000,1000,1000,dan 2000. Berikut proses yang akan terjadi :

1. Setelah menerima uang kertas 2000 pertama, mesin akan berganti dari state ke state .
2. Setelah menerima uang kertas 1000 pertama, mesin akan berganti dari state ke state .
3. Setelah menerima uang kertas 1000 kedua, mesin akan berganti dari state ke state .
4. Setelah menerima uang kertas 2000 kedua, mesin akan berganti dari state ke state . Pada posisi ini, mesin akan mengeluarkan minuman. (Ingat, tidak ada pengembalian uang).

Gambar berikut ini menerjemahkan behaviour dari semua kemungkinan apa yang akan terjadi pada setiap nilai uang kertas yang diterima. State digambarkan dengan dua lingkaran. Sesaat mesin mencapai kondisi state ini maka mesin akan mengeluarkan minuman.

[1] State atau dapat dikatakan sebagai posisi
[2] Kondisi awal saat mesin belum menerima uang kertas



Gambar 3.1 Mesin abstrak
untuk menjual minuman harga 5000

Hasil observasi bahwa mesin (atau komputer) hanya akan mengingat state pada posisi yang mana saat waktu tertentu. Oleh karena itu, hanya akan memerlukan penggunaan kapasitas memori yang sangat kecil. Dan mampu membedakan antara satu dari enam kemungkinan.

Sedikit lebih rumit adalah kasus di mana jawabannya tergantung pada masukan simbol diterima dan bukan pada setiap simbol sebelum itu. Misalnya, jika = {a, b}, perangkat mungkin mengumumkan setiap kali menerima simbol, dan hanya dalam hal ini, bahwa string saat ini diterima. Dalam kasus ini, bahasa ini menerima adalah himpunan semua string yang diakhiri dengan. Ini adalah contoh dari jenis bahasa Penerima disebut Finite State Automata (FSA), atau mesin berhingga. Setiap langkah, mesin berhingga terletak di salah satu dari state berhingga (itu adalah mesin state karena menetapkan himpunan finite) respon tergantung hanya pada keadaan saat ini dan simbol saat ini. "Respon"

dalam keadaan tertentu dan menerima masukan simbol tertentu adalah hanya untuk memasukkan state tertentu, mungkin sama itu sudah di FSA "mengumumkan" penerimaan atau penolakan dalam arti bahwa kondisi saat baik menerima state atau state tidak menerima.

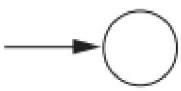
Dalam dua contoh sederhana dimana respon adalah selalu sama, hanya satu state yang diperlukan. Untuk bahasa string yang berakhir dengan dua state adalah sufficient, sebuah state yang menerima untuk string yang berakhir dengan dan tidak menerima state untuk semua yang lain. Sebelum otomaton finite telah menerima masukan apapun, itu adalah dalam keadaan awal, yang adalah sebuah state yang menerima tepat jika null string diterima. Setelah mengetahui bagaimana ada banyak state, yang salah satunya adalah keadaan awal, dan state yang menerima menyatakan,-hanya informasi yang butuhkan untuk menggambarkan operasi Mesin adalah fungsi transisi, yang spesifik untuk setiap kombinasi state dan masukan simbol state ke mesin FSA. Fungsi transisi dapat digambarkan oleh tabel nilai atau (cara menggunakan paling sering) diagram transisi. Dalam diagram, state diwakili oleh lingkaran, transisi yang diwakili oleh panah dengan masukan simbol sebagai label,

Gambar 3.2 Diagram transisi suatu state ke state lain

Dan state penerima yang digambarkan dengan gambar dua lingkaran yang menjadi satu

Gambar 3.3 State penerima

State awal akan memiliki panah yang menunjuk state tersebut, dan panah tersebut tidak berasal dari state lain.



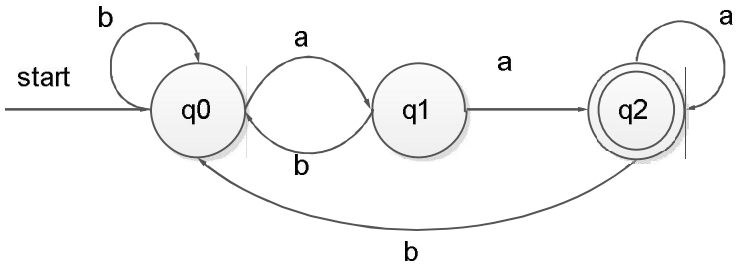
Gambar 3.4 State awal

FSA dapat melanjutkan melalui input, seperti robot, dengan hanya mengingat pada setiap langkah apa state yang dalam dan mengubah State menanggapi masukan simbol sesuai dengan fungsi transisi. Dengan diagram, boleh menjejaki komputasi untuk input string tertentu dengan hanya mengikuti panah yang sesuai untuk simbol-simbol string.

2. Definisi dari FSA

Mesin otomata berhingga atau FSA, merupakan model matematika yang dapat menerima input dan bahkan mengeluarkan output, memiliki state yang berhingga banyaknya dan dapat berpindah dari satu state ke state lainnya berdasar input dan fungsi transisi, tidak memiliki tempat penyimpanan/memory, hanya bisa mengingat state terkini. Mekanisme kerja dapat diaplikasikan pada : elevator, text editor, analisa leksikal, pencek parity dan lain sebagainya.

Pada pembahasan pertama, dapat kita amati bagaimana sebuah mesin penjual minuman otomatis untuk harga minuman 5000 dan dengan kemungkinan menerima uang kertas yang dibatasi 1000, 2000, dan 5000, dapat menerima setiap input dari sensor uang yang sudah di kenali dan tidak akan diproses sebelum sensor sudah merespons. Hal ini merupakan gambaran bagaimana FSA dapat berperan untuk mempermudah rancangan desain suatu mesin otomatis dengan kompleksitas proses input berdasarkan bahasa formal lainnya.



Gambar 3.5 sebuah mesin otomata
yang menerima string yang berakhiran aa

Sebagai contoh yang mudah dan sederhana, perhatikan pada gambar 3.5 berikut. FSA tersebut akan menerima suatu bahasa berikut :

L1 = {x $\in$ {a, b}* | x berakhir dengan aa} artinya bahasa L1 hanya akan menerima setiap input untai string dinyatakan dengan x yang elemen tersusun dari simbol alfabet a dan b saja.

FSA beroperasi dengan tiga (3) state yakni q0, q1 dan q2, sesuai dengan jumlah berturut-turut dari yang diterima berikutnya untuk menghasilkan string sesuai bahasa L1. Dengan state awal nya adalah q0. Pada state q0, masukan string b tidak akan menghasilkan transisi ke state lainnya karena hanya akan tetap pada state q0, sedangkan untuk masukan string a menyebabkan FSA memungkinkan untuk berpindah state dari q0 ke state q1.

Saat state berada di q1, masukan string b akan membatalkan FSA untuk berpindah maju ke state berikutnya karena hanya membawa transisi state kembali ke state q0. Jika diberikan masukan string a maka akan berpindah transisi ke state menerima yakni q2, sehingga memungkinkan diterima oleh mesin setelah mencapai state penerima yakni di state q2. Jika string a dimasukkan maka state akan tetap pada q2 sedangkan jika diberikan masukkan string b maka state akan dikirimkan kembali ke keadaan awal q0.

Maka dapat kita definisikan suatu FSA adalah 5-tupel (Q, S, Σ, F, δ,)

Q adalah himpunan state;

Σ adalah alfabet masukan state;

S adalah state awal yakni q0 dan q0 Q adalah keadaan awal;

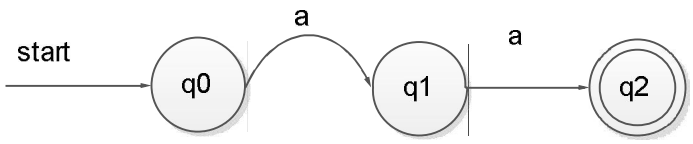
F Q adalah serangkaian State penerima;

δ: Q ×Σ → Q adalah fungsi transisi.

3. Perbedaan Antara String satu dengan String lainnya

Pada otomata berhingga gambar 3.5, dapat kita deskripsikan.

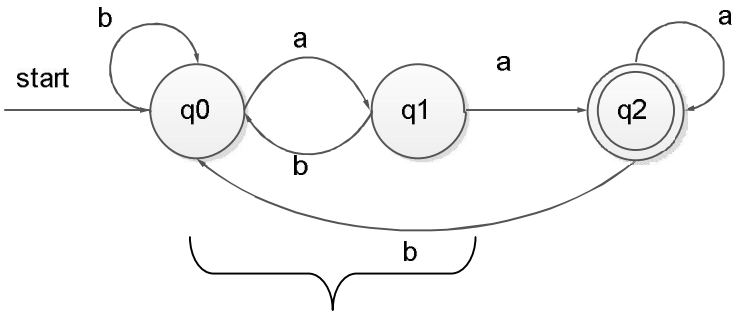
a. Menerima setiap bahasa L1 string yang merupakan elemen {a, b} dan berakhir dengan aa dengan kemungkinan dapat dilihat pada gambar berikut :



b. Memiliki tiga state, sesuai dengan yang dibutuhkan untuk string di bahasa L1. Di mana setiap FA dengan tiga state dan informasi untuk transisi mesin dengan input string {a,b}.

c. Dalam kasus mesin M tersebut, tidak ada bedanya Apakah untai string saat ini adalah aba atau aabbabbabaaaba, kedua input tersebut tidak berakhir dengan aa.

Namun bila untai string saat ini aba atau ab akan ada perbedaan yakni karena simbol masukan masih mungkin akan datang berikutnya selain dari a. Perhatikan gambaran berikut :



Input untai string aba akan melibatkan state q0 dan q1 bahkan ada kemungkinan untuk string selanjutnya ada “a”

Jika untai string nya adalah aba, maka pada input selanjutnya hanya mungkin a saja yakni dari state q1 hanya memiliki transisi berpindah ke state q2 dengan input string a.

Sedangkan untuk input untai string ab, akan melibatkan state q0 dengan q1 dengan posisi state berada pada q0. Maka untuk input string yang memungkinkan adalah banyak karena q0 memiliki perpindahan state q1 dengan input string a ataupun tak berpindah dengan string b. Artinya banyak kemungkinan input string yang mungkin dikombinasikan. Mesin Otomata berhingga (FSA) harus dapat membedakan aba dan ab, sehingga dalam kasus masukan simbol berikutnya akan mampu membedakan dua string. Materi mengenai string mampu dibedakan akan dibahas lebih lanjut pada bab 4.

Definisi membedakan String yang diterima oleh Bahasa L:

Jika L adalah bahasa dengan abjad x dan y adalah string di Σ maka x dan y adalah mampu dibedakan pada L. Jika ada string z Σ sedemikian rupa sehingga baik xz L dan yz ∉ L, atau xz ∉ L dan yz L. String z memiliki properti untuk membedakan antara x dan y sehubungan dengan bahasa L. Formulasi setara adalah untuk mengatakan bahwa x dan y dapar dibedakan atau distinguishable dalam bahasa L

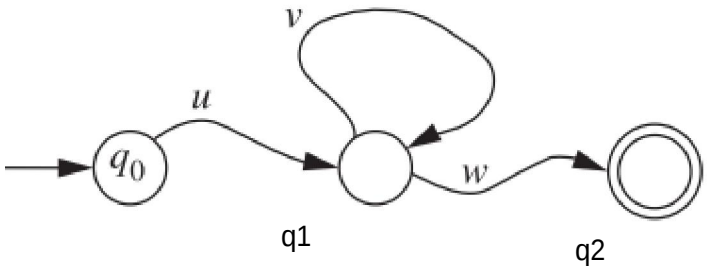
jika L/x ≠ L/y, untuk L / x = {z Σ | xz L}

Dua string x dan y adalah tidak dapat dibedakan atau indistinguishable pada bahasa L, jika L / x = L/y, yang berarti bahwa untuk setiap z Σ , xz L jika dan hanya jika yz L.

Lemma Pemompaan

Mesin terbatas menerima bahasa beroperasi dalam cara yang sangat sederhana. Maka bahasa yang dapat diterima dengan cara ini adalah bahasa "sederhana", tetapi belum jelas apa artinya. Dalam bagian ini, kita akan melihat satu properti yang harus dimiliki untuk setiap bahasa yang diterima oleh FSA. Jika diketahui bahwa M = (Q, Σ, q0, δ, F) adalah FSA menerima setiap yang ada pada F dan bahwa Q memiliki unsur-

unsur state. Jika x adalah string dalam L dengan |x| = n−1, jadi bahwa x memiliki n awalan yang berbeda, itu masih dibayangkan bahwa M adalah dalam keadaan berbeda setelah memproses setiap string. Jika |x|≥ n, namun, kemudian pada saat M telah membaca simbol-simbol x, memasuki beberapa dua kali state; harus ada dua awalan kita yang berbeda dan sedemikian rupa sehingga artinya bahwa jika x L dan w adalah menerima untai string x = uvw, perhatikan situasi ini digambarkan oleh gambar 3.6. Dalam membaca simbol x = uvw, M bergerak dari keadaan awal ke state yang menerima dengan mengikuti alur yang berisi sebuah pengulangan atau loop, sesuai dengan simbol-simbol v.



Gambar 3.6 Contoh Diagram Lemma Pemompaan

Kemungkinan lebih dari satu lingkaran tersebut, dan lebih dari satu cara melompat untai string x ke tiga potong u, v, dan w; tapi setidaknya salah satu loop harus selesai pada saat M telah membaca simbol n pertama dari x. Dengan kata lain, untuk setidaknya salah satu dari pilihan kita, v, dan w tersebut bahwa x = uvw dan v merupakan loop, |uv| ≤ n. Alasan ini harus diingat bahwa akan ada banyak string lain selain x yang juga diterima oleh M dan oleh karena itu akan menyebabkan M untuk mengikuti alur yang sama tetapi melintasi loop berbeda beberapa kali dalam string bahasa L. String yang diperoleh dari x dengan menghilangkan substring v adalah di L, karena M tidak harus melintasi sama sekali. Untuk setiap i ≥ 2, string uvw dalam bahasa L, M dapat mengambil loop sekian kali sebelum melanjutkan ke state penerima. Pernyataan ini membuktikan teori memompa Lemma untuk Bahasa L. istilah "Pompa" merujuk kepada ide memompa string x dengan memasukkan

tambahan salinan string v (tapi ingat bahwa juga mendapatkan salah satu string baru dengan meninggalkan keluar v).

Teori Lemma pompaan

Misalkan L adalah bahasa atas abjad ∑ . Jika L diterima oleh terbatas

robot M = (Q, ∑ , q0, δ, F) dan jika n adalah jumlah State m, maka setiap x L diterima |x|≥ n, ada tiga string u, v, dan w tersebut bahwa x = u v w dan berikut tiga kondisi ini benar:

1. |uv|≤ n.

2. |v| > 0 (yaitu, v ≠ hampa).

3. untuk setiap i ≥ 0, string uv(ke i)w juga milik bahasa L.

Kemudian dalam bagian ini akan ditemukan cara untuk menerapkan hasil ini untuk bahasa L yang diterima oleh FSA. Tapi aplikasi yang paling umum adalah menunjukkan bahwa bahasa yang tidak dapat diterima oleh FSA, dengan menunjukkan bahwa ia tidak memiliki properti yang dijelaskan dalam lemma pemompaan. Bukti menggunakan lemma pemompaan dari bahasa L itu tidak diterima oleh otomata berhingga adalah bukti adanya kontradiksi. Asumsinya, untuk kontradiksi, yang dapat diterima L oleh M, FSA dengan n state, dan dicoba untuk memilih string dalam L dengan panjang setidaknya n. Misalkan x = b , atau misalkan x = b, atau sesuatu sebanding, tergantung u,v dan w pada bahasa L. Dengan Lemma pemompaan akan diketahui properti apa yang memenuhi setiap string dalam L, setidaknya selama n panjangnya. Hal ini sangat mungkin untuk beberapa pilihan x, fakta bahwa x memiliki sifat-sifat ini tidak menghasilkan pertentangan. Jika tidak mendapatkan suatu kontradiksi, belum membuktikan apa-apa. Sebagai contoh, jika mencoba untuk menunjukkan bahwa bahasa atas {a, b} tidak dapat diterima oleh FSA. Setelah menemukan sebuah string x yang terlihat memungkinkan, lemma pemompaan akan ada beberapa cara untuk melompati x menjadi lebih pendek string u, v, dan w memenuhi tiga kondisi. Tanpa mengetahui apa string ini pendek saja, hanya jika memenuhi kondisi lemma. Jika v, u dan w menghasilkan suatu kontradiksi, harus

menunjukkan bahwa harus didapatkan suatu kontradiksi, yang memenuhi kondisi benar dari teori lemma pemompaan.

Contoh pada compiler untuk bahasa pemrograman tingkat tinggi harus mampu menerima bahasa tertentu. Ini termasuk kedua bahasa yang seimbang dan karena (m) n adalah sebuah string yang seimbang, dan n (ma) adalah ekspresi hukum, jika dan hanya jika m = n. Contoh lain adalah kumpulan L hukum program C. tidak perlu tahu banyak tentang sintaks C untuk menunjukkan bahwa bahasa ini tidak dapat diterima oleh FSA - hanya bahwa string main() {{{}}...}

dengan kemunculan m "{" dan kejadian n "}", adalah program C hukum yang tepat jika m = n. Seperti biasa, mulai bukti dengan mengasumsikan bahwa L diterima oleh beberapa FSA dengan n State dan membiarkan x menjadi string main() {n}n. Jika x = uvw, dan tiga string ini memenuhi persyaratan 1 – 3, maka cara termudah untuk mendapatkan suatu kontradiksi untuk menggunakan i = 0 dalam kondisi 3. String v tidak boleh berisi tanda kurung yang tepat karena kondisi 1; Jika string lebih pendek uw hilang salah satu simbol-simbol di "main()", maka itu tidak memiliki header yang diperlukan untuk program C, dan jika itu hilang salah satu tanda kurung siku kiri, kemudian dua angka tidak cocok.

Untuk menunjukkan kegunaan lemma memompa, salah satu cara untuk menjawab pertanyaan tentang bahasa adalah untuk memeriksa otomaton terbatas yang menerima. Secara khusus, untuk bahasa yang dapat diterima oleh FSA, ada beberapa masalah keputusan[3], bisa menjawab dengan beberapa cara yang memiliki algoritma keputusan yang mengambil keuntungan dari lemma pemompaan.

Masalah keputusan yang melibatkan bahasa yang diterima oleh Automata terbatas

Pertanyaan yang paling mendasar tentang bahasa L adalah string yang menjadi bagian dari bahasa L tersebut. Masalahnya keanggotaan untuk bahasa L diterima oleh M suatu FSA akan meminta, untuk setiap string x atas masukan huruf m, Apakah x L(M). Masalah sebagai spesifik untuk M, adalah contoh masalah string x tertentu; juga mungkin mengajukan pertanyaan untuk setiap M dan setiap x, dan mempertimbangkan yang menjadi pasangan memerintahkan (M, x). Dalam kedua

[3] pertanyaan dengan jawaban ya atau tidak

kasus, cara kerja otomaton terbatas membuatnya mudah untuk menemukan algoritma keputusan untuk masalah. Mengetahui string x dan spesifikasi untuk M = (Q, Σ, q0, δ, F) memungkinkan untuk menghitung state δ(q0,x) dan memeriksa apakah termasuk di dalamnya adalah elemen a.

Dua masalah di bawah ini adalah contoh lain pertanyaan yang mungkin bertanya tentang L(M):

1. Mengingat FSA M = (Q, ∑ , q0, δ,), adalah L(M) tidak kosong?

2. Mengingat FSA M = (Q, ∑ , q0, δ,), adalah L(M) tak terbatas?

Salah satu cara untuk bahasa L(M) menjadi kosong adalah untuk kosong, tapi ini bukan satu-satunya cara. Pertanyaan sebenarnya adalah Apakah M telah menerima setiap state, tetapi apakah ia memiliki apapun yang dapat dicapai dari q0. Algoritma yang sama yang gunakan untuk menemukan seperangkat kota dapat dijangkau dari kota S, algoritma lain yang kurang efisien tapi lebih mudah untuk menggambarkan adalah sebagai berikut :

Keputusan untuk algoritma pertama

Biarkan n menjadi jumlah State M. mencoba string dalam urutan panjang, untuk melihat apakah ada yang diterima oleh M; L(M) = jika dan hanya jika ada serangkaian panjang kurang dari n yang diterima, dimana n adalah jumlah State m.

Alasan algoritma ini adalah bahwa menurut lemma memompa, untuk setiap string x L(M) dengan |x|≥ n, ada string lebih pendek di L(M), yang diperoleh dengan menghilangkan bagian tengah v. Oleh karena itu, mustahil untuk string singkat diterima oleh M memiliki panjang n atau lebih.

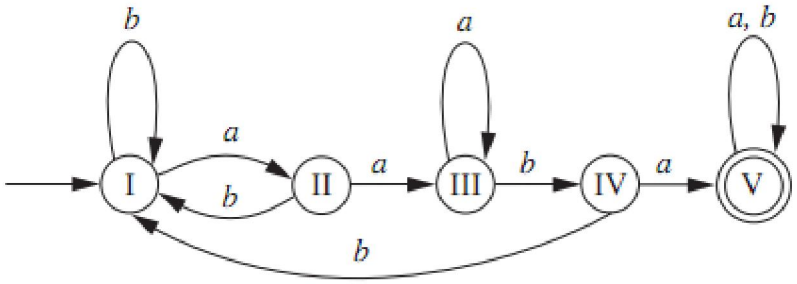
Jika n adalah jumlah State m, dan x adalah sebuah string di L(M) dengan |x|≥ n, kemudian ada banyak lagi string di L(M). Di sisi lain, jika x L(M) dan |x|≥ 2n, maka x = uvw untuk beberapa string u, v dan w menerima kondisi di lemma memompa, dan u w = uw adalah sebuah string yang lebih pendek di L(M) yang panjang adalah masih n atau lebih, karena |v|≤ n. Dengan kata lain, jika ada string dalam L dengan panjang setidaknya n, terpendek string tersebut harus memiliki

panjang kurang dari 2n. Berikut bahwa algoritma berikut ini, yang tidak diragukan lagi tidak efisien, adalah algoritma keputusan untuk masalah 2.

Algoritma yang akan menyelesaikan masalah ke 2

Mencoba string dalam urutan panjang, dimulai dengan string panjang n, untuk melihat apakah ada yang diterima oleh M. L(M) adalah jika dan hanya jika string x ditemukan dengan $n \leq |x| < 2n$.

C. SOAL LATIHAN 3 DAN TUGAS



1) Berdasarkan gambar berikut, Deskripsikan dengan sederhana pendapat Anda masing – masing mengenai Bahasa yang mungkin dapat diterima.
2) Coba gambarkan Mesin otomata dengan kondisi mampu menerima hanya bahasa antara {a,b} berikut :
    a. FA yang mampu menerima string aabb
    b. FA yang mampu menerima string abaa
    c. FA yang mampu menerima string bbaa

D. DAFTAR PUSTAKA

Martin, John C. 2010. Fourth Edition. Introduction to Language and The Theory of Computation. United State America : McGraw-Hill

Maheshwari, Anil & Smid Michiel, September 25, 2014. Introduction to Theory of Computation. Ottawa Canada : School of Computer Science Carleton University