PENGENALAN AUTOMATA

Objektif:

1. Mahasiswa Mampu Memahami Konsep Dasar Automata.
2. Mahasiswa Mampu Menggunakan Aplikasi JFLAP.

1.1 Dasar-dasar Automata

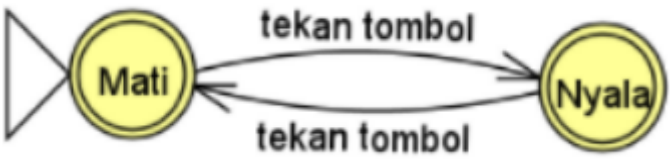
Teori automata adalah pembelajaran tentang alat komputasi atau mesin. Automata ditemukan oleh Alan Turing. Tujuan Turing mempelajari automata adalah untuk mendeskripsikan secara tepat tentang apa yang dapat dan tidak dapat dilakukan oleh mesin komputasi. Pada antara tahun 1940 – 1950an, terdapat sebuah mesin sederhana yang sekarang disebut *finite automata*. Mesin ini mirip fungsi otak manusia yang berfungsi untuk banyak tujuan. Pada akhir tahun 1950an, seorang Ahli Bahasa yang bernama Noam Chomsky mempelajari bahasa formal Grammar. Grammar ini memiliki hubungan dekat dengan automata yang merupakan basis utama komponen penting *software,* termasuk *compiler.*

Berikut beberapa alasan mengapa harus mempelajari automata:

- Aplikasi untuk mendesain dan memeriksa keadaan atau kondisi dari sebuah sirkuit *digital*.
- *Lexical analyzer*, yaitu komponen *compiler* yang memecah *input* teks menjadi bentuk unit *logical* seperti gambar, *identifier*, kata kunci, dan tanda baca.
- Aplikasi untuk memindai dokumen teks yang berukuran besar seperti koleksi dari halaman web
- Aplikasi untuk memverifikasi sistem yang mengandung keadaan atau *state* dalam jumlah terbatas.

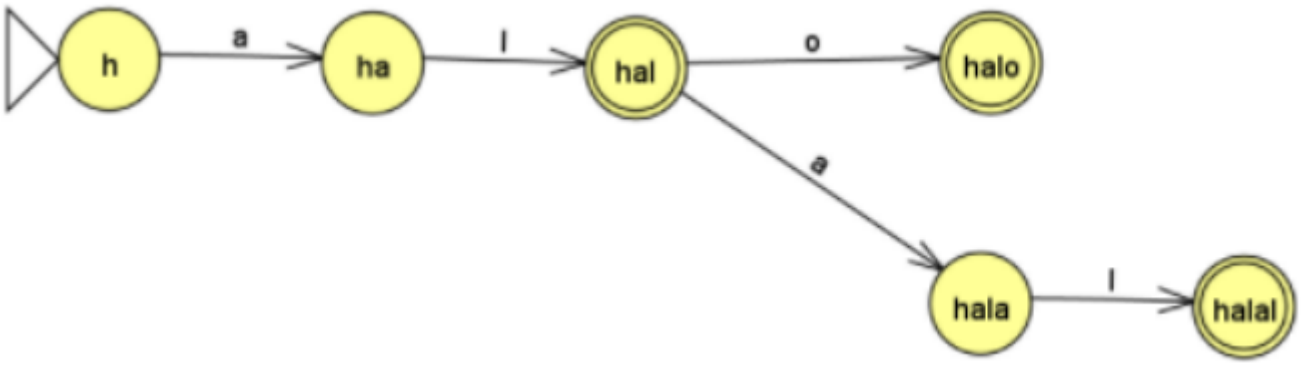
Contoh sederhana dari *finite automata* yaitu ketika menentukan suatu kondisi lampu jika ditekan tombolnya lampu tersebut bisa menyala atau mati dan suatu mesin untuk menerima *input* teks seperti kamus *online*.

Pada gambar dibawah terlihat terdapat dua *state* yaitu *state* mati dan *state* nyala, kedua *state* menandakan lampu mati atau menyala jika kita menekan salah satu tombol. Dari gambar diatas menjelaskan bahwa automata memiliki fungsi sebagai *state* yang menandakan suatu keadaan tertentu.



Gambar 1.1 Contoh Kondisi Lampu

Selain itu automata juga berfungsi untuk menerima *input string*. Bisa dilihat pada gambar dibawah terdapat automata yang menerima *input string* berupa huruf atau alfabet. Jika *input string* tersebut disusun maka akan menjadi sebuah kata tertentu. Salah satu fungsi automata yang sudah sering digunakan dengan menerima *input string* yaitu google *translate*.



Gambar 1.2 Kondisi Automata Menerima *Input String*

Automata sendiri memiliki notasi representasi struktural, yaitu:

- *Grammar*. Automata memiliki susunan grammar tertentu, seperti grammar pada Bahasa Inggris yang mengenal *noun*, *verb*, *adjective*, atau *grammar* pada Bahasa Indonesia yang mengenal subjek, predikat, objek, dan keterangan. *Grammar* pada automata adalah model untuk mendesain proses data dengan struktur rekursif. Contoh penggunaan *grammar* automata adalah "parser", yaitu komponen *compiler* yang menangani perulangan bersarang pada Bahasa pemrograman tertentu.
- *Regular Expression*. Ekspresi regular juga menotasikan struktur data, terutama *string*. Ekspresi regular pada umumnya terbentuk dari susunan huruf `"[A - Z] [a - b] * [ ] [A – Z][A - Z]"`.

1.1.1 Himpunan

Himpunan adalah kumpulan dari objek-objek. Dalam automata, objek yang dimaksud adalah suatu *state* atau keadaan tertentu yang terkumpul menjadi himpunan. Himpunan dapat dituliskan sebagai: `X = {a, b, c, d}`.

Beberapa operasi yang ada pada himpunan:

- Gabungan
  Himpunan gabungan adalah penggabungan dari dua atau lebih himpunan. Himpunan gabungan dilambangkan dengan notasi ∪.

  Contoh:
  `A = {1, 2, 3, 4, 5}  B = {6, 7, 8, 9, 10}`
  `A ∪ B = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}`

- Irisan
  Himpunan irisan adalah himpunan yang berisikan anggota yang sama antara dua atau lebih himpunan. Himpunan irisan dilambangkan dengan notasi ∩.

  Contoh:
  $A = \{1, 2, 3, 4, 5\}$ $B = \{1, 3, 5, 7, 9\}$
  $A \cap B = \{1, 3, 5, 7\}$

- Beda Simetris
  Himpunan beda simetris adalah himpunan yang berisikan anggota dari dua atau lebih himpunan yang tidak mengandung anggota yang sama dari himpunan tersebut. Himpunan beda simetris dilambangkan dengan notasi ⊕.

  Contoh :
  $A = \{1, 2, 3, 4, 5\}$ $B = \{2, 4, 6, 8, 10\}$
  $A \oplus B = \{1, 3, 5, 6, 8, 10\}$

- Komplemen
  Himpunan komplemen adalah himpunan yang anggotanya bukan merupakan anggota dari himpunan tersebut, tapi terdapat di himpunan semesta. Himpunan komplemen dilambangkan dengan notasi ` atau c

  Contoh:
  $U = \{1, 2, 3, 4, 5, . . . 10\}$
  $A = \{1, 2, 3, 4, 5\}$ $B = \{2, 4, 6, 8, 10\}$
  $A' = Ac = \{4, 5, 6, 7, 8, 9, 10\}$
  $B' = Bc = \{1, 3, 5, 7, 9\}$

1.1.2 Relasi dan Fungsi

Relasi adalah penggambaran interaksi atau koneksi antara elemen dari dua buah himpunan atau lebih. Dalam automata, relasi menggambarkan hubungan antara masing-masing *state* yang terhubung dengan suatu *input* atau tidak.

Berikut adalah sifat-sifat relasi, yaitu:

- Refleksif
  Suatu relasi R pada himpunan A dinamakan bersifat refleksi jika $(a, a) \in R$ untuk setiap $a \in A$.

  Contoh:
  $A = \{1, 2, 3, 4, 5\}$ terdapat relasi R yaitu $\geq$ (lebih dari atau sama dengan) yang didefinisikan pada himpunan A, maka:

$R = \{(1, 1), (2, 1), (2, 2), (3, 2), (3, 3), (4, 3), (4, 4), (5, 4), (5, 5)\}$

Terdapat relasi $(1, 1), (2, 2), (3, 3), (4, 4), (5, 5)$ yang merupakan $\in$ R. Maka relasi ini bersifat refleksif.

- Simetris
  Suatu relasi pada himpunan A bersifat simetris jika $(a, b) \in R$, untuk setiap a dan $b \in A$, dan $(b, a) \in R$. Suatu relasi bersifat tidak simetris jika $(a, b) \in R$, namun $(b, a) \notin R$.

  Contoh:
  Terdapat relasi $R = \{(1, 2), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4), (3, 1), (3, 2), (3, 4), (4, 1), (4, 2), (4, 3)\}$

  Terdapat relasi $(1, 2), (2, 1), (1, 3), (3, 1), (1, 4), (4, 1), (2, 3), (3, 2), (2, 4), (4, 2), (3, 4), (4, 3)$. Maka relasi ini bersifat simetris.

- Antisimetris
  Suatu relasi pada himpunan A bersifat antisimetris apabila setiap $a, b \in A, (a, b) \in R$, dan $(b, a) \in R$, jika $a = b$.

  Contoh:
  $A = \{1, 2, 3, 4\}$ terdapat relasi R yaitu $\geq$ (lebih dari atau sama dengan) yang didefinisikan pada himpunan A, maka:

  $R = \{(1, 1), (2, 2), (2, 1), (3, 3), (3, 2), (3, 1), (4, 4), (4, 3), (4, 2), (4, 1)\}$

  Apabila $a \geq b$, dan $b \geq a$, berarti $a = b$. Maka relasi ini bersifat antisimetris.

- Transitif

  Suatu relasi pada himpunan A bersifat transitif apabila $(a, b) \in R$ dan $(b, c) \in R$, maka $(a, c) \in R$ untuk a , $b, c \in A$.

  Contoh:
  $A = \{2, 4, 6, 8\}$ terdapat relasi R yaitu $<$ (kurang dari) yang didefinisikan pada himpunan A, maka:

  $R = \{(2, 4), (2, 6), (2, 8), (4, 6), (4, 8), (6, 8)\}$

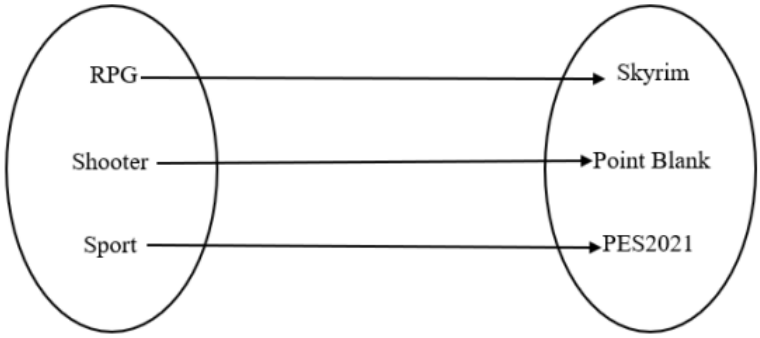
  Terdapat relasi $(2, 4), (4, 6),$ dan $(2, 6)$, relasi $(2, 6), (6, 8),$ dan $(2, 8)$. Maka relasi ini bersifat transitif.

Fungsi adalah relasi dari masing-masing anggota himpunan domain yang hanya mempunyai satu bayangan di himpunan kodomain.

Macam-macam jenis fungsi:

- Fungsi Satu-satu
  Fungsi satu-satu adalah fungsi yang masing-masing anggota hanya memiliki satu bayangan pada himpunan domain dan kodomainnya. Fungsi ini hanya berlaku jika jumlah anggota domain dan kodomainnya sama.

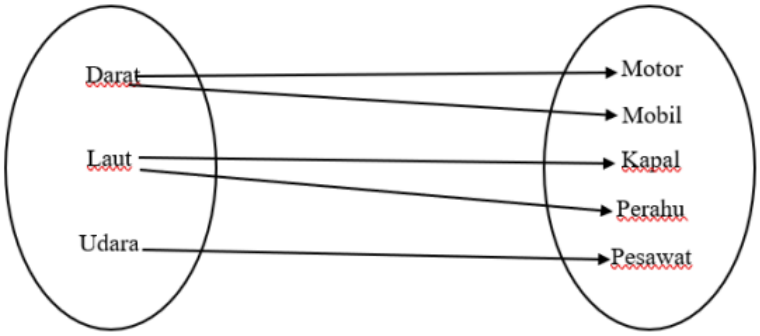
  Contoh:
  `X = {RPG, Shooter, Sport}`
  `Y = {Skyrim, Point Blank, PES2021}`
  Fungsi `f: X → Y`



Gambar 1.3 Fungsi satu-satu

- Fungsi Pada
  Fungsi pada adalah fungsi yang setiap anggota himpunan kodomain merupakan bayangan dari minimal satu dari himpunan domain.

  Contoh:
  `X = {darat, laut, udara}`
  `Y = {motor, mobil, kapal, perahu, pesawat}`
  Fungsi `f: X → Y`



Gambar 1.4 Fungsi pada

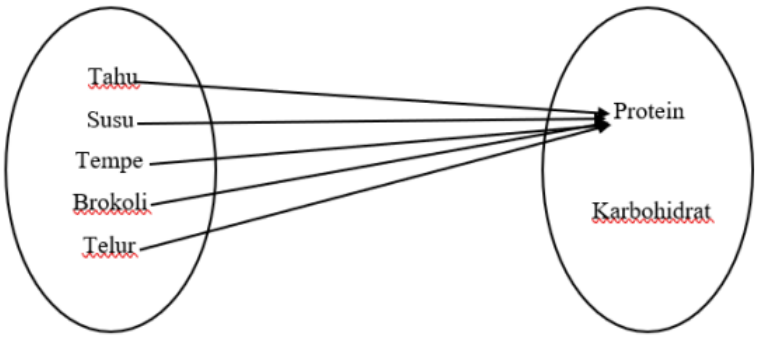
- Fungsi Konstan
  Fungsi konstan adalah fungsi yang satu anggota dan himpunan kodomain menjadi bayangan dari semua anggota domain.

Contoh:

X = {tahu, tempe, telur, susu, brokoli}

Y = {protein, karbohidrat}

Fungsi f: X → Y



Gambar 1.5 Fungsi konstan

1.1.3 Graph dan Tree

Graph dan Tree merupakan dasar dari automata, karena automata mirip dengan graph berarah, namun memiliki *input*. Suatu graph terdiri dari *vertex*, sejumlah *edge*, dan fungsi yang menunjukkan hubungan antara *vertex* dan *edge*. Ketiga komponen ini dapat ditulis G = {V, E, y} .

Contoh:

Terdapat graph dengan *vertex* = {P, Q, R, S} dan *edge* = {a1, a2, a3, a4, a5} dan y didefinisikan dengan:

y (a1) = {P, Q}

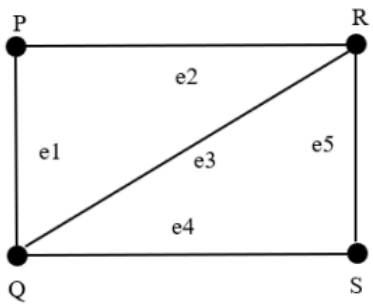
y (a2) = {P, R}

y (a3) = {Q, R}

y (a4) = {Q, S}

y (a5) = {R, S}

Maka graph tersebut:

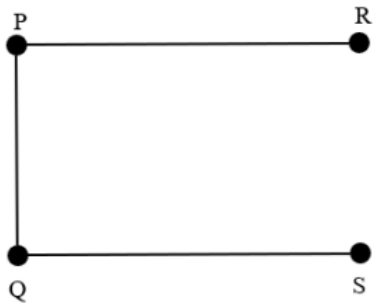


Gambar 1.6 Graph

Fungsi yang menunjukkan hubungan antara *vertex* dan *edge* mirip dengan fungsi transisi pada *finite automata*.

Pohon adalah sebuah *graph* yang tidak memiliki *cycle* atau putaran.

Contoh:



Gambar 1.7 Pohon

1.1.4 Alfabet

Alfabet adalah sebuah karakter atau simbol. Dalam *finite* automata, alfabet adalah himpunan karakter atau simbol. Himpunan alfabet dalam automata disimbolkan dengan ∑.

Contoh:

∑ = {0, 1, 2, ... 100}

∑ = {a, b, c, d, ... z}

∑ = {himpunan karakter ASCII}

1.1.5 *String*

String adalah kumpulan dari beberapa alfabet. Suatu mesin automata dapat menolak atau menerima *input string* yang diberikan, selain itu mesin automata juga dapat menerima *input string* kosong yang disebut Ɛ-*Non Deterministic Finite Automata*.

Contoh:

1. ∑ = {0, 1, 2}, maka *string* yang dibentuk yaitu:
   00, 101, 0101012, 100101, ...
   String memiliki panjang yaitu banyaknya alfabet yang membentuk *string* dan dinotasikan dengan |x|.
2. |001120| = 6
   Sebuah *string* dapat juga sebagai operasi perpangkatan dari alfabet.
3. ∑ = {0, 1}, maka:
   $\sum^1$ = {0, 1}
   $\sum^1$ = {00, 01, 10, 11}
   $\sum^1$ = {000, 001, 010, 011, 100, 101, 110, 111}
   $\sum^0$ = {Ɛ}

Suatu *string* dapat digabungkan dengan *string* lain. Himpunan seluruh alfabet dalam ∑ dapat dinotasikan dengan ∑*. * dalam notasi tersebut disebut *kleene star*. Himpunan seluruh alfabet tanpa Ɛ *string* dinotasikan dengan $\sum^+$.

∑* = Alfabet bisa muncul seluruhnya dan bisa tidak muncul dalam suatu *string*
$\sum^+$ = Alfabet muncul minimal satu kali dalam suatu *string*

Konkatenasi *string* adalah penggabungan dari *string*. Jika terdapat *string* p dan *string* q, maka pq merupakan hasil konkatenasi *string* tersebut.

Contoh:

1. `p = beli q = buku`
   Maka:
   `pq = belibuku`
   `qp = bukubeli`
2. ∑ = `{0, 1}` , maka:
   $\sum^*$ = `{Ɛ, 00, 11, 100, 001, . . . .}`
   $\sum^+$ = `{00, 11, 100, 101, 10101, 111, . . . .}`
3. Tuliskan *string* yang dihasilkan dari:
   $(0, 1)^*11(0)^+$
   Maka, *string* yang dihasilkan = `{110, 01100, 1110, 01110, . . . .}`
4. Tuliskan *string* yang dihasilkan dari:
   `{a, ab}*`
   Maka, *string* yang dihasilkan = `{Ɛ, a, ab, aab, aaab, aaaab, . . . .}`

1.1.6 Bahasa

Bahasa pada automata adalah ruang lingkup atau ketentuan dari suatu *string*.
Contoh:

1. ∑ = `{0, 1}`
   Sebuah mesin automata hanya bisa menerima *input string* yang memenuhi bahasa L = {diawali *input* 0 dan diakhiri *input* 1}. Maka:
   String yang diterima adalah `{01, 00001111, 0 .... 1}` jika dinotasikan maka $L = 0(01)^*1$.

2. ∑ = `{0, 1}`
   Sebuah mesin automata hanya bisa menerima *input string* yang memenuhi bahasa L = {*input* 0 berjumlah genap dan *input* 1 berjumlah ganjil}. Maka:
   String yang diterima adalah `{001, 00111, 00001, 0000111, .....}` jika dinotasikan maka $L = (00)^+1(11)^*$.

3. ∑ = {x, y}

   Terdapat bahasa L = {$x^n y^n$: n > 0}. Periksa apakah *string* xxyy, xy, xxxyy termasuk di bahasa tersebut?

   xxyy -> jumlah x = 2 , jumlah y = 2 . Memenuhi Bahasa tersebut karena jumlah x dan y sama

   xy -> jumlah x dan jumlah y = 1 . Memenuhi Bahasa tersebut karena jumlah x dan y sama

   xxxyy -> jumlah x = 3 , jumlah y = 2 . Tidak memenuhi Bahasa tersebut karena x dan y tidak sama

4. ∑ = {x, y, z}

   Terdapat bahasa L = {xy, z} . Tentukan L3 dan L0 !

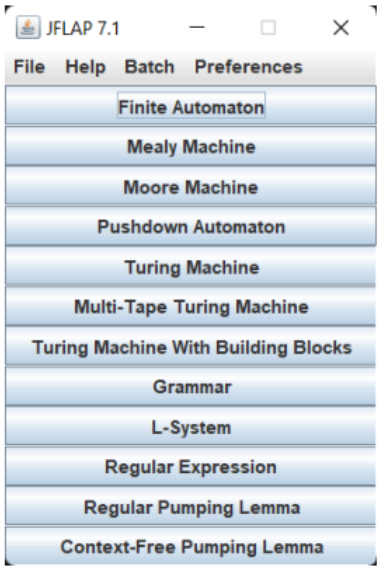
   Maka, L3 adalah 3 kata dari Bahasa tersebut

   L3 = {xyxyxy, xyxyz, xyzxy, xyzz, zxyxy, zxyz, zzxy, zzz}

   L0 = kata kosong λ

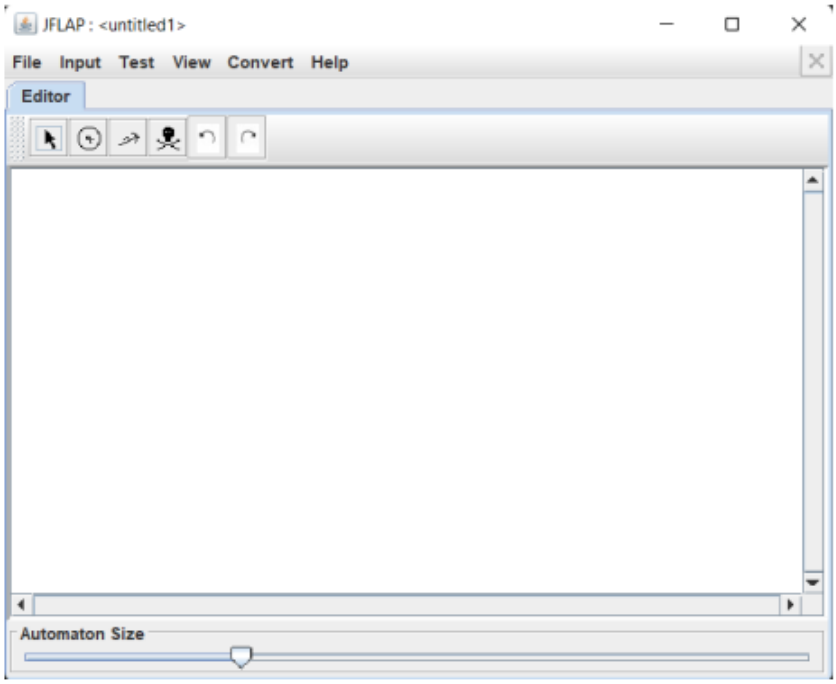
1.2 Pengenalan dan Instalasi JFLAP

JFLAP adalah aplikasi yang digunakan untuk merancang Bahasa formal termasuk NFA(Non-Deterministic Finite Automata), DFA (Deterministic Finite Automata), Pushdown Automata, Mealy Machine, Moore Machine, Turing Machine, dan beberapa tipe grammar. JFLAP menggunakan bahasa pemrograman java dan memiliki ekstensi .jar.



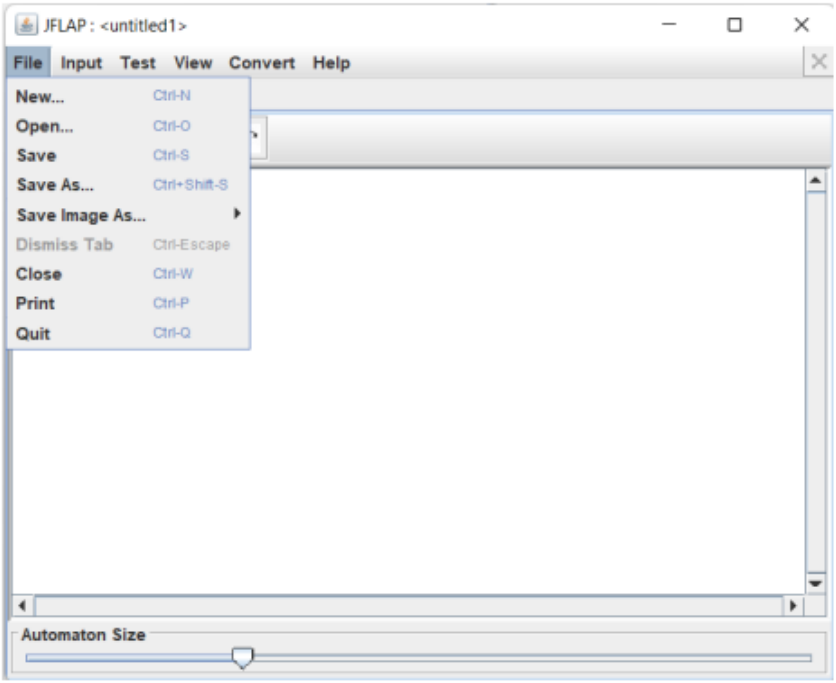
Gambar 1.8 JFLAP

Jika memilih menu *Finite Automaton* maka akan muncul tampilan awal pada JFLAP seperti berikut ini.



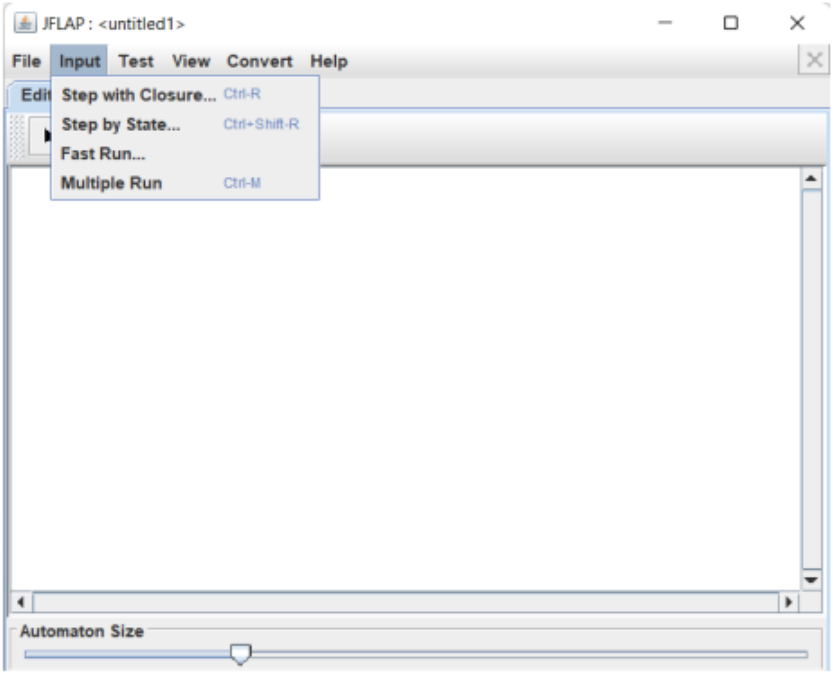
Gambar 1.9 Tampilan Awal JFLAP

Pada tampilan awal terdapat menu awal yaitu *File, Input, Test, View, Convert* dan *Help*. Pada menu *File* terdapat beberapa sub menu lagi seperti yang ada pada gambar berikut.



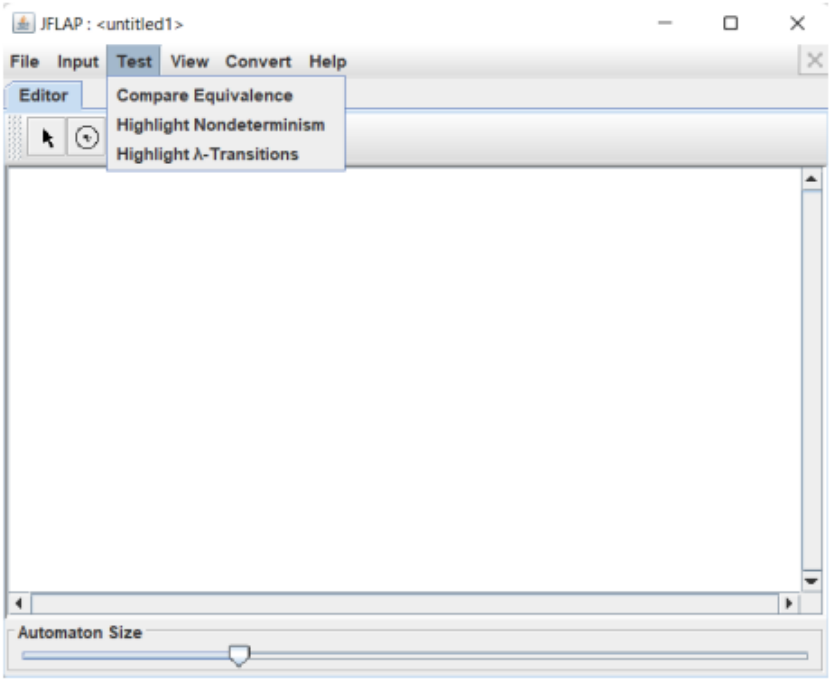
Gambar 1.10 Menu *File* JFLAP

Pada menu *Input* juga terdapat beberapa sub menu yang berfungsi untuk menguji suatu *string* dari automaton tersebut dapat diterima atau tidak, selain itu juga dapat diketahui langkah-langkah pengecekan pada bagian sub menu ini.



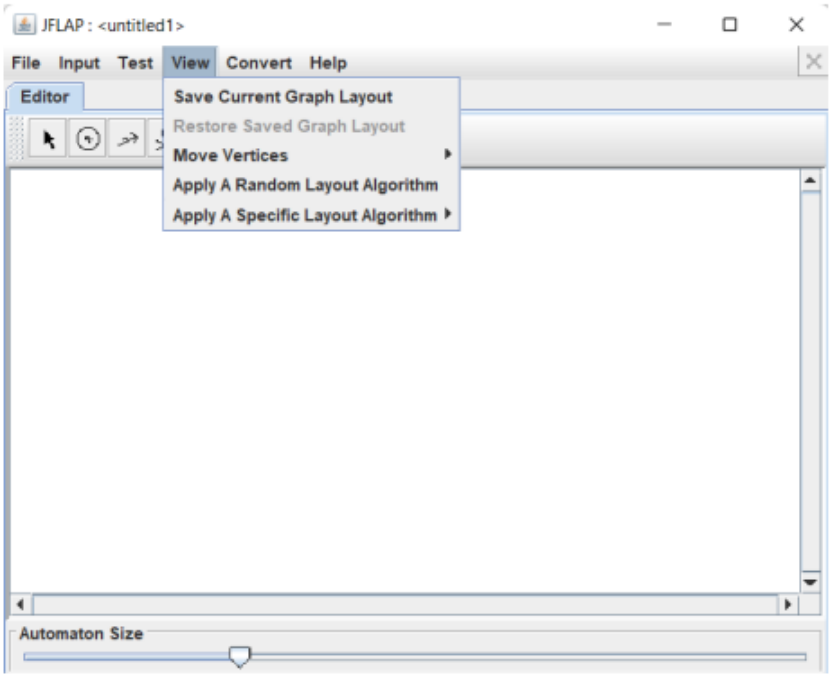
Gambar 1.11 Menu *Input*

Pada menu *Test* terdapat pilihan untuk menampilkan transisi yang merupakan *non-deterministic* atau transisi yang mengandung Ɛ *string*.



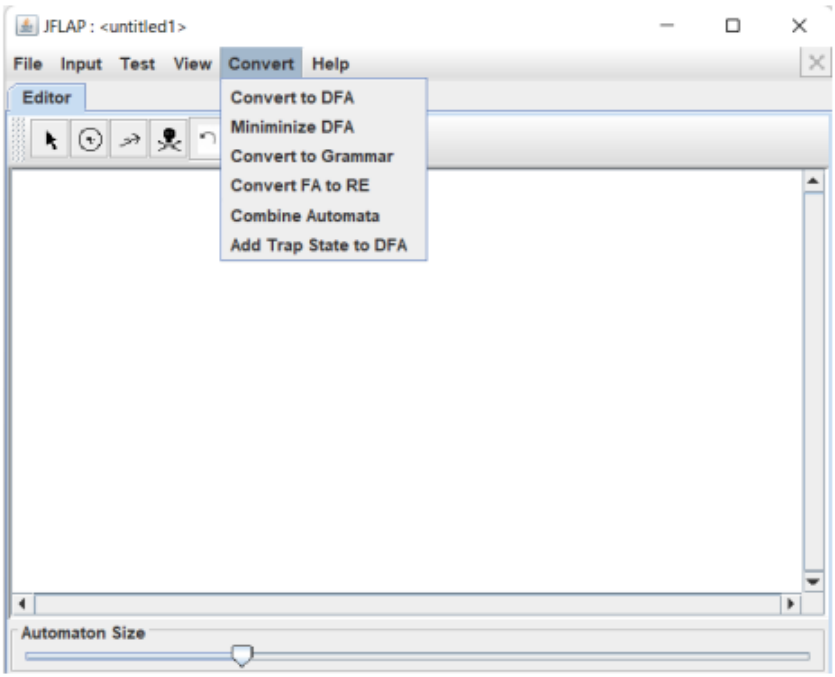
Gambar 1.12 Menu *Test* JFLAP

Pada tampilan menu *View* terdapat beberapa pilihan yang ada pada sub menunya. Menu ini berfungsi untuk melihat suatu algoritma tertentu pada automaton.



Gambar 1.13 Menu *View* JFLAP

Pada menu *Convert* digunakan untuk konversi *Nondeterministic Finite Automata* (NFA) menjadi *Deterministic Finite Automata* (DFA), untuk meminimasi DFA yang memiliki banyak *state*, konversi grammar ke ekspresi *regular* atau sebaliknya dan untuk menggabungkan dua atau lebih mesin automaton pada JFLAP.



Gambar 1.14 Menu *Convert* JFLAP

Selain terdapat beberapa menu yang di setiap menunya ada sub menu, pada JFLAP juga terdapat beberapa *icon* yang dapat digunakan untuk membuat atau menggambar automaton.

Tabel 1.1 Fungsi *Icon* JFLAP

Icon	Fungsi
	Pointer penunjuk untuk memindahkan state Mengubah bentuk garis transisi Mengubah nama state
	Menambahkan state baru
	Menambahkan fungsi transisi
	Menghapus state Menghapus fungsi transisi
	Undo
	Redo

RANGKUMAN

- Teori automata adalah pembelajaran tentang alat komputasi atau mesin. Automata ditemukan oleh Alan Turing
- JFLAP adalah aplikasi yang digunakan untuk merancang Bahasa formal termasuk NFA (non deterministic finite automata), DFA (Deterministic Finite Automata), Pushdown Automata, Mealy Machine, Moore Machine, Turing Machine, dan beberapa tipe grammar.
- Himpunan adalah kumpulan dari objek-objek.
- Relasi adalah penggambaran interaksi atau koneksi antara elemen dari 2 buah himpunan atau lebih.
- Fungsi adalah relasi dari masing-masing anggota himpunan domain yang hanya mempunyai satu bayangan di himpunan kodomain.
- Graph dan pohon merupakan dasar dari automata, karena automata mirip dengan graph berarah, namun memiliki input.
- Fungsi yang menunjukkan hubungan antara vertex dan edge mirip dengan fungsi transisi pada mesin finite automata
- Alphabet adalah sebuah karakter atau simbol. Dalam finite automata, alfabet adalah himpunan karakter atau simbol. Himpunan alfabet dalam automata disimbolkan dengan ∑.
- String adalah kumpulan dari beberapa alfabet
- Bahasa (language) pada automata adalah seperti ruang lingkup atau ketentuan dari string.

Referensi

[1] Dirgantara, Harya Bima. 2016. *Teori Bahasa dan Automata: Merancang Automata Dengan JFLAP*. Yogyakarta: Penerbit Andi.