Penerapan Graf Transisi dalam Mendefinisikan Bahasa Formal

Abdurrahman Dihya R./13509060
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
a.dihya@s.itb.ac.id

Abstract*—Komputer selalu identik dengan istilah komputasi, yaitu perhitungan. Dua kata tersebut lahir dari kata dasar yang sama. Dalam komputasi ada yang disebut bahasa formal, yaitu bahasa yang digunakan untuk menyatakan cara kerja mesin abstrak yang dinyatakan sebagai model-model komputasi. Setiap mesin tersebut dapat dimodelkan secara matematis, mulai dari yang paling sederhana, sampai yang paling rumit sekalipun. Untuk memodelkan suatu bahasa formal dalam permasalahan komputasi dapat digunakan beberapa cara, yaitu dengan Finite Automata, Ekspresi Regular, dan graf transisi. Pada makalah ini akan dijelaskan mengenai cara memodelkan suatu bahasa formal dengan graf transisi dan pembuktiannya apabila dimodelkan dengan permodelan lainnya. Hal ini sangat penting dalam menentukan cara mesin menjawab masalah keputusan berdasarkan masukan string dengan memberikan keluaran 'ya' atau 'tidak'. Kita akan melihat, bagaimana masalah komputasi akan dapat dilihat sacara sederhana dengan representasi graf transisi.

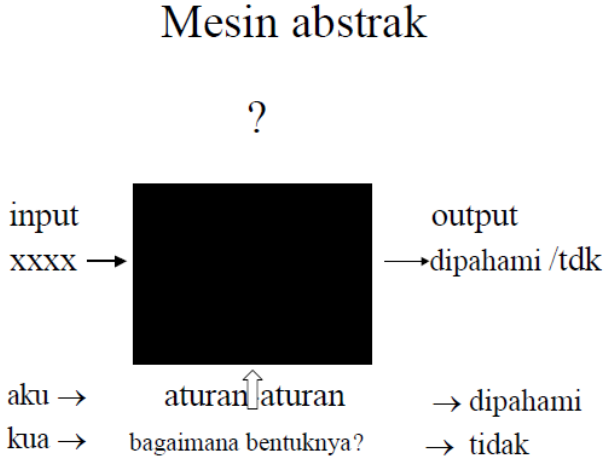
***Index Terms*—Bahasa formal, ekspresi regular, graf transisi, teorema Kleene.**

I. PENDAHULUAN

Teori Bahasa dan Automata

Bahasa formal adalah ketentuan khusus dalam kebahasaan yang karakteristiknya sudah ditentukan secara formal. Secara eksplisit, aturan yang mengatur bahasa formal dinyatakan dalam suatu terms yang akan menentukan output berupa string-string dari simbol. Oleh karena itu, bahasa formal lebih kepada pelambangan dalam simbol-simbol, bukan menyatakan bahasa manusia. Definisi formal juga merujuk kepada formalitas cara yang digunakan untuk menentukan output string dari simbol-simbol, bukan kepada formalnya bahasa itu sendiri. 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.

Kita seperti mempelajari sebuah mesin abstrak yang mampu mengenali masukan string, seperti ditunjukkan oleh gambar di bawah ini.



Gambar 1: Mesin abstrak

Kotak hitam di atas merupakan kunci untuk mengenal masukan string, seperti halnya bahasa formal. Dalam mempelajari bahasa formal ini dikenal apa yang dinamakan automata. Automata adalah mesin abstrak yang dapat mengenali (*recognize*), menerima (*accept*), atau membangkitkan (*generate*) sebuah kalimat dalam bahasa tertentu.

Suatu alphabet atau abjad adalah himpunan terbatas tak kosong dari symbol-simbol dilambangkan Σ. Barisan berhingga dari symbol-simbol dari suatu abjad kadang-kadang dinamakan sebuah kata (word) yang terbentuk dari abjad itu dilambangkan *w*, sedangkan suatu string yang tidak terdapat abjad didalamnya disebut empty string dilambangkan Λ. Kumpulan dari string-string yang dibentuk dari alphabet disebut Bahasa dilambangkan L. Terdapat suatu bahasa yang tidak terdiri dari untai – untai – *bahasa kosong* (empty language) dilambangkan ϕ. Terdapat operasi-operasi yang terjadi pada untai dan untai diantaranya : Concatenasi (Perangkaian) , Reversal (Pembalikan), Eksponensiasi.

Kemudian, kita juga harus mengenal bahasa regular, yaitu bahasa yang dapat dihasilkan dari bahasa paling sederhana dari suatu alfabet S dengan menggunakan operasi – operasi gabungan, konkatenasi, dan Kleene.

Mengenai operasi konkatenasi, berikut macam-macamnya.

Operasi konkatenasi pada string-string

String a dan b apabila dokonkatenasi akan menjadi string ab. Contohnya, jika ada a=1100 dan b=0010, maka hasil konkatenasi a dan b adalah ab=11000010.

Operasi konkatenasi berulang pada string

Operasi konkatenasi dapat dilakukan berulang kali dalam string yang sama. Konkatenasi beberapa abjad a dapat ditulis dengan pangkat untuk memudahkan. Milanya, a^4=aaaa dan sebagainya.

Operasi konkatenasi pada himpunan string

Operasi ini berarti, jika ada L_1 dan L_2 himpunan string maka L_1L_2 adalah himpunan string {xy|x L_1 dan y L_2}. Jika L_1={aa,bab} dan L_2 ={bb,a} maka L_1L_2 = {aabb,aaa,babbb,baba}. Jelaslah bahwa $L_1L_2 = L_2L_1$.

Operasi konkatenasi alfabet

Semua operasi string di atas juga berlaku pada alfabet.

Operasi Kleene-* pada Alfabet

Untuk alfabet S maka himpunan seluruh string yang mungkin terbentuk dari S adalah S* (Kleene-* dari S). Dan, setiap bahasa yang menggunakan alfabet S adalahsubset dari S*.

Operasi Kleene-* pada himpunan string

Bila L* adalah himpunan semua string yang terbentuk dari bebagai konkatenasi yang bisa dilakukan pada setiap elemen dari L. Ditulis sebagai berikut:

$$L^* = 8\ L_i$$

Sebagai contoh, jika kita nyatakan bahasa dari {a,b} yang mana setiap anggotanya merupakan hasil konkatenasi dari sejumlah substring bba dan diakhiri oleh satu simbol apa saja (a atau b). Bahasa tersebut dapat dituliskan sebagai ekspresi operasi himpunan {bba}*{a,b}. Ternyata bahasa tersebut adalah regular karena ekspresi operasi himpunan itu memenuhi definisi bahasa regular di atas, yaitu sebagai berikut.

- Konkatenasi {b}{b}{a} menjadi {bba}
- Kleen {bba} menjadi {bba}*
- Penggabungan {a}{b} menjadi {a,b}
- Konkatenasi hasil Kleen {bba} dengan hasil penggabungan {a,b} menjadi {bba}*{a,b}

Suatu Bahasa dapat kita definisikan dengan berbagai cara, dalam pembahasan kali ini menggunakan tiga cara pendefinisisan suatu bahasa yaitu: Dengan menggunakan Ekspresi Regular, menggunakan Finite Automata, dan menggunakan Graph transisi. Hal ini sesuai dengan teorema berikut ini, **" Jika suatu bahasa dapat didefinisikan oleh salah satu cara pendefinisisan maka akan juga dapat didefinisikan kedua cara yang lainnya "(Teorema Kleene 1956)**. Secara singkat bisa dikatakan bahwa ketiga metode pendefinisian diatas adalah Equivalent.

Finite Automata

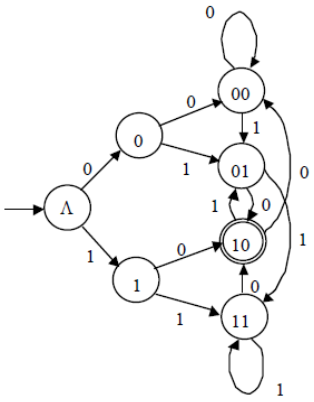
Finite Automata, atau dapat disebut juga Finite State Automata (FSA) adalah mesin yangdapat mengenali bahasa regular tanpa menggunaanstorage/memori. Kecuali, sejumlah status didefinisikan oada mesin untuk "mengingat" beberapa hal secara terbatas. Secara formal FA didefinisikan sebagai berikut.

"Suatu Finite Automata (FA) atau Mesin Status-Berhingga adalah 5-tuple $(Q,\sum,q_0,\delta)$ yang mana:

- *Q adalah himpunan berhingga dari status*
- *$\sum$, himpunan berhingga **alfabet** dari simbol masukan*
- *$q_0 \in Q$ adalah status **inisial** (status awal)*
- *$A \subseteq Q$ yaitu himpunan status **menerima** (accepting state, kadang-kadang disebut **final state**)*
- *δ adalah fungsi transisi yang memetakan $Q\ x\sum$ ke Q (ditulis $\delta\ Qx\sum \rightarrow Q$)"*

Dengan diagram status digambarkan dengan bulatan dan transisi dengan panah, yaitu sebagai berikut.

Status \ masukan	0	1
Λ	0	1
0	00	01
1	10	11
00	00	01
01	10	11
10	00	01
11	10	11



Gambar 2: Contoh Finite Automata

Bila kita lihat gambar di atas, bentuk diagram dari FA mendekati bentuk graf. Maka sangat mungkin bila kita dapat merepresentasikannya ke dalam bentuk graf.

Untuk dapat merepresentasikan ke dalam bentuk graf kta terlebih dahulu harus memahami apa itu graf, cara memakainya dan kegunaannya.

Definisi Graf

Secara matematis, graf didefinisikan sebagai berikut.

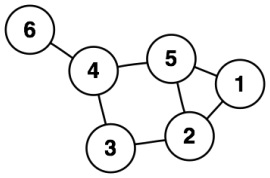
Graf G didefinisikan sebagai pasangan himpunan (V,E) yang dalam hal ini:

V = himpunan tak kosong dari simpul-simpul (verticles atau node) = $\{v_1,v_2,..,v_n\}$ dan

E = himpunan sisi (edges atau arcs) yang menghubungkan sepasang simpul = $\{e_1,e_2,...,e_n\}$

Atau dapat ditulis singkat notasi G = (V,E)

Gambar berikut ini adalah contoh graf dengan 6 simpul dan 7 sisi.



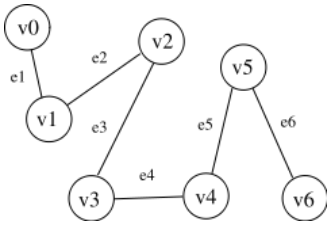
Gambar 3: Contoh graf

Dalam teori graf, formalisasi ini untuk memudahkan ketika nanti harus membahas terminologi selanjutnya

yang berhubungan dengan graph. Beberapa terminologi berhubungan dengan teori graf :

- *Degree* atau derajat dari suatu node, jumlah edge yang dimulai atau berakhir pada node tersebut. Node 5 berderajat 3. Node 1 berderajat 2.
- *Path* suatu jalur yang ada pada graph, misalnya antara 1 dan 6 ada path $b \longrightarrow c \longrightarrow g$
- *Cycle* siklus ? path yang kembali melalui titik asal 2 $f \longrightarrow c \longrightarrow d \longrightarrow e$ kembali ke 2.
- *Tree* merupakan salah satu jenis graf yang tidak mengandung cycle. Jika edge f dan a dalam digraf diatas dihilangkan, digraf tersebut menjadi sebuah tree. Jumlah edge dalam suatu tree adalah nV - 1. Dimana nV adalah jumlah vertex

Jalur yang menghubungkan antara node juga dapat diberi inisial(nama), seperti pada gambar di bawah ini. Sifat inilah yang nantinya akan dipakai untuk membuat graf transisi.



Gambar 4: Contoh disgraf

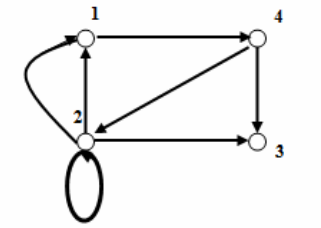
Berdasarkan orientasi arahnya, graf terbagi menjadi dua, yaitu graf tak berarah dan graf berarah.

Graf Tak Berarah

Graf tidak berarah adalah graf yang sisinya tidak mebmiliki orientasi arah.

Grat Berarah

Graf berarah adalah graf yang sisinya memiliki orientasi arah. Graf berarah ditunjukkan seperti gambar berikut.



Gambar 5: Contoh graf berarah

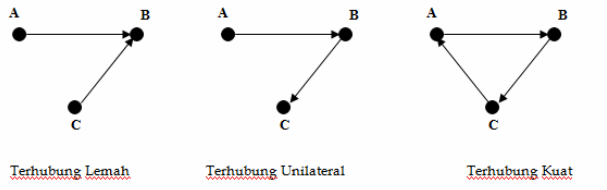
Beberapa pengertian dalam graf berarah :

- Derajat ke luar (out degree) suatu simpul adalah banyaknya ruas yang mulai / keluar dair simpul tersebut
- Derajat ke dalam (in degree) suatu simpul adalah banyaknya ruas yang berakhir / masuk ke simpul tersebut.
- Simpul berderajat ke dalam = 0 disebut sumber (source), sedangkan simpul berderajat ke luar = 0 disebut muara (sink).
- Pengertian Walk, Trail, Path (Jalur) dan Sirkuit (Cycle) berlaku pula pada graf berarah, dimana harus sesuai dengan arah ruas. Kalau tidak sesuai dengan arah ruas-nya, maka disebut sebagai semi walk, semi path atau semi trail

Pada graf berarah terdapat tiga pengertian keterhubungan, yaitu:

- Terhubung lemah, jika terdapat suatu semi path antara setiap 2 simpul dari D.
- Terhubung unilateral, jika antara setiap 2 simpul u danv dari D, terdapat jalur dari u ke v atau dari v ke u.
- Terhubung kuat, jika antara setiap 2 simpul u dan v dari D, terdapat jalur dari u ke v dan dari v ke u.

Contoh :



Gambar 6: Keterhubungan graf berarah

Graf Transisi

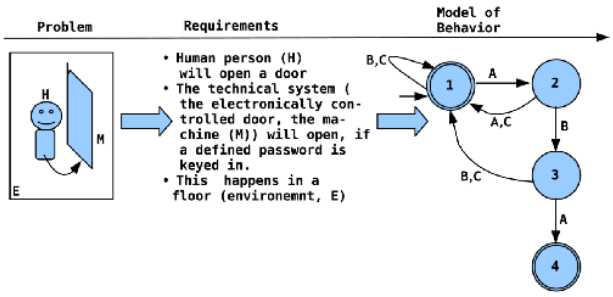
Graf transisi dapat dideskripsikan dengan 3 hal berikut ini.

- Pengaturan Finite state pada setiap keadaan, berupa keadaan mulai maupun keadaan menerima
- alfabet Σ of berupa inputan string
- Finite state dari transisi yang menunjukkan proses masuk ke dalam state yang baru bagi keadaan-keadaan yang berpasangan dan bagian dari string yang masuk, ataupun alfabet (setiap pasangan bisa memiliki 0, 1, atau lebih).

Komponen-komponen yang dimiliki oleh graf transisi adalah sebagai berikut.

- State awal
- State akhir
- State transisi
- Input string (Є)
- Tabel transisi

Contoh penggunaan graf transisi adalah seperti ditunjukkan pada gambar di bawah ini.



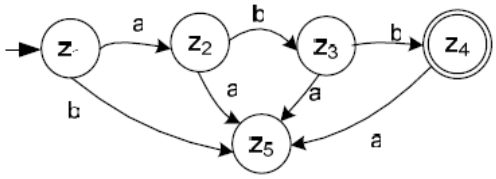
Gambar 7: Salah satu aplikasi graf transisi

Pada gambar di atas, dimodelkan sebuah sistem kendali pintu elektrik elektrikdengan menggunakan graf transisi. Banyak sekali permodelan yang dapat dilakukan dengan graf transisi, pada permodelan Final State Machine dalam perancangan rangkaian digital misalnya, atau permodelan, untuk memodelkan transisi sinyal dalam suatu rangkaian, dan masih banyak lagi. Salah satunya adalah memodelkan komputasi dari ekspresi regular yang akan kita bahas di sini.

II. PEMBAHASAN

Permodelan Menggunakan FA

Diberikan ekspresi regular abb, maka premodelan dengan Finite Automata (FA) adalah sebagai berikut.



Gambar 8: FA dari ekspresi regular abb

Dari gambar di atas, dapat kita lihat, sistem akan masuk ke state awal, yaitu z untuk membaca karakter (digit) pertama. Ada dua kondisi setelah itu, jika digit yang terbaca adalah 'a', maka state berpindah ke z_2, dan berpindah ke z_5 apabila yang dibaca adalah 'b'. Kemudian mesin mulai membaca karakter selanjutnya.

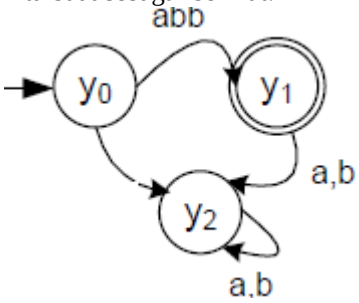
Dari z_2, state akan berpindah ke z_3 jika mendapatkan 'b', dan berpindah ke z_5 jika mendapatkan 'a'.

Dari z_3, state akan berpindah ke z_4 jika mendapatkan 'b', dan berpindah ke z_5 jika mendapatkan 'a'.

Dari z_4, state hanya akan berpindah ke z_5 jika mendapatkan 'a'. Dan tidak berpindah state jika yang dibaca adalah selainnya.

Permodelan menggunakan graf transisi

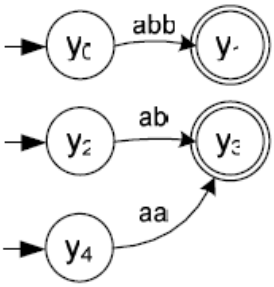
Dengan graf transisi, untuk memodelkan ekspresi regular abb, kita gunakan y_0 untuk state awal, dan dua state lagi yaitu y_2 dan y_1 sebagai state transisi dan state akhir. Maka kita buat sebagai berikut.



Gambar 9: Graf Transisi

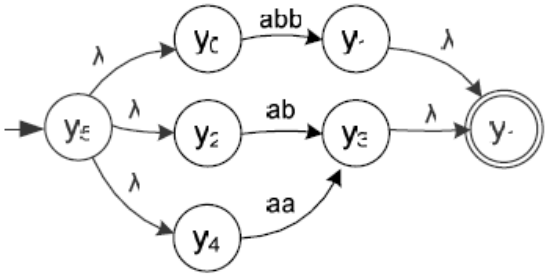
Pada gambar di atas, setiap membaca hasil konkatenasi dari ketiga masukan {abb} state langsung berpindah ke state y_1. Namun, jika masukan yang lain, ia langusng menuju ke state y_2. Dapat dilihat bahwa penggunaan graf transisi di sini meminimalisir state dan menyederhanakan permodelan.

Berbeda dengan FA, graf transisi dapat memiliki lebih dari satu state awal, dan lebih dari satu state akhir. Misal, untuk graf berikut ini.



Gambar10: lebih dari satu state awal

Dapat diubah menjadi:

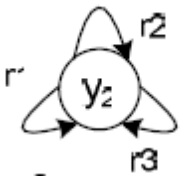


Gambar 11: satu state awal

Kedua graf transisi di atas saling ekuivalen.

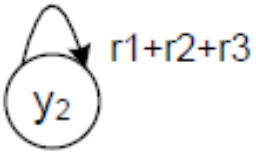
Untuk mengubah FA menjadi graf transisi, kita dapat memodifikasi state awal dan state akhir. Perbedaan antara FA dengan Graf Transisi adalah. State awal dari FA selalu satu. Sedangkan state awal di graf transisi mungkin lebih dari satu. Begitu pula state akhir. Selain itu, kita juga dapat mengubah masukan dari tiap state yang asalnya berupa string menjadi abjad tunggal, substring, ekspresi regular , dan string kosong.

Untuk ekspresi regular $r = r_1+r_2+r_3$, kita dapat memodelkan dengan FA seperti ini.



Gambar 12: r1+r2+r3

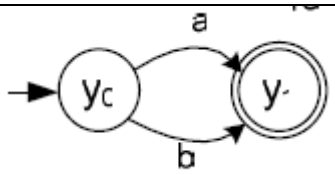
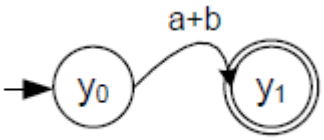
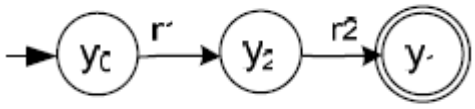
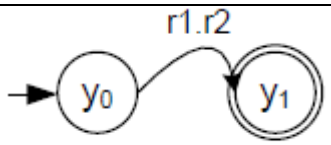
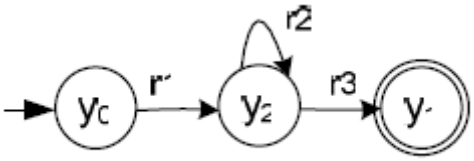
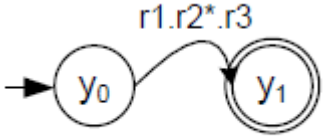
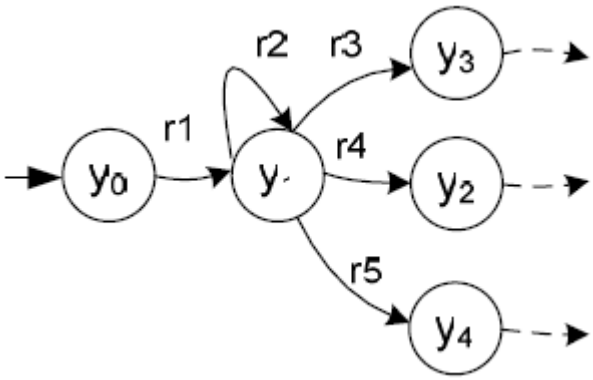
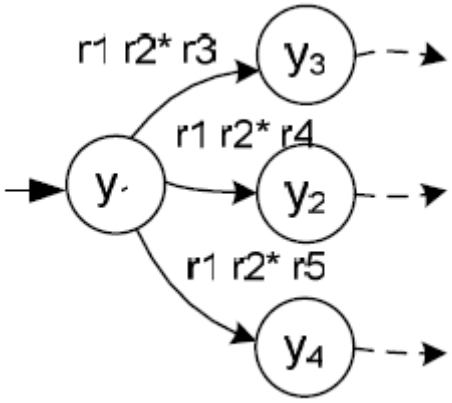
Sedangkan pada graf transisi, kita dapat membuatnya seperti ini.



Gambar 13: Bentuk Graf Transisinya

Berikut beberapa macam ekspresi regular dalam dua bentuk FA dan graf transisi.

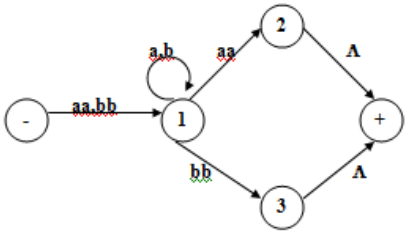
Tabel 1: FA dan Graf Transisi

Finite Automata	Graf Transisi
	
	
	
	

III. ANALISIS DAN KESIMPULAN

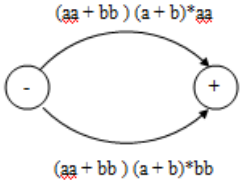
Analisis

Untuk mengetahui apakah suatu graf transisi benar-benar merepresentasikan ekspresi regular, kita dapat melakukan simplifikasi graf transisi, dengan mengubah parameter transisi menjadi ekspresi. Misalkan kita mempunya graf awal sebagai berikut.



Gambar 14: Graf awal

Kita dapati state (-) adalah state awal dan state (+) adalah state akhir. Jika kita perhatikan, perpindahan ke state 1 bernilai benar jika terbaca 'aa atau bb, maka ekspresi yang sesuai untuk menggambarkan transisi ini adalah (aa+bb). Kemudian, untuk satu masukan 'a' atau 'b', keadaan mesin tetap pada state 1, maka untuk ekspresi (aa+bb)(a+b) keadaan mesin akan tetap di state 1. Perpindahan dari state 1 ke state 2 dipenuhi jika terbaca aa. Artinya, state 2 terpenuhi jika ekspresi dari state awal ke state 2 adalah (aa+bb)(a+b)*aa. Sedangkan untuk mencapai state 3, ekspresinya adaah (aa+bb)(a+b)*bb. Untuk mencapai state akhir, dibutuhkan input string kosong. Maka graf hasil simplifikasi yang kita dapat adalah sebagai berikut.



Gambar 15: Graf hasil simplifikasi

Dengan tabel transisi, kita dapat melihat apakah graf transisi sebelum simplifikasi sama dengan graf yang menyatakan ekspresi regularnya. Graf awal kita namakan graf 1. Sedangkan graf yang kedua kita namakan graf 2. String kosong dilambangkan dengan '[ ]'.

Tabel 2: Tabel transisi

Input	State Graf 1	State Graf 2
a	-	-
a	1	-
b	1	-
a	1	-
b	1	-
a	1	-
b	1	-
b	3	-
[ ]	+	+
b	-	-
b	1	-
b	1	-
a	1	-
a	2	-
[ ]	+	+
a	-	-
b	-	-
b	1	-
b	1	-
a	1	-
b	1	-

State 1 dan 2 tidak ada di graf kedua. Cukup melihat transisi kedua graf ke state akhir dan kembali lagi ke state awal. Jika kita perhatikan, kedua graf ekuivalen karena transisi statenya sesuai.

Kesimpulan

Untuk mendefinisikan suatu bahasa formal, kita dapat menggunakan graf transisi sebagai elternatif dari Finite Automata, ekspresi regular, dan mesin permodelan yang lain. Antara setiap mesin permodelan tersebut saling ekuivalen. Sehingga bahasa yang diterima di satu mesin akan dipahami dengan cara yang sama di mesin yang lainnya.

Terdapat perbedaan mendasar antara Finite Automata dengan graf transisi, yaitu jenis inputnya dan jumlah state awal dan akhirnya. Input FA hanya bertipe string, sedangkan graf transisi dapat berupa substring, ekspresi regular, abjad tunggal, dan string kosong. Jumlah state awal dan akhir pada FA hanya satu, sedangkan pada graf transisi dapat diubah menjadi lebih dari satu.

IV. REFERENCES

[1] Munir, Rinaldi, "**Diktat Kuliah IF2091: Struktur Diskrit**", Penerbit ITB, 2008

[2] Rosen, Kenneth.H, *"Discrete Mathematics and Its Applications"*, 6th ed. New York: McGraw-Hill, 2007, hal. 589–743.

[3] "Formal Languages and Automata Theory", Samarjit Chakraborty. http://www.tik.ee.ethz.ch/tik/education/lectures/DES/Book/des_book_automata.pdf diakses tanggal 13 Desember 2010

[4] Tremblay, JP and Manohar, R. (1987). Discrete Mathematical Structure with Application to Computer Science, Mc Graw Hill.

[5] Manna, Z. (1074).Mathematical Theory of Computation, Mc Graw Hill.

[6] http://en.wikipedia.org/wiki/Introduction_to_Automata_Theory,_Languages,_and_Computation diakses tanggal 15 Desember 2010

[7] http://en.wikipedia.org/wiki/Graph_theory diakses tanggal 15 Desember 2010

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Desember 2010

ttd

Abdurrahman Dihya Ramadhan/13509060