Penerapan Algoritma Depth First Search (DFS) Dinamis Untuk Menentukan Apakah Sebuah String Diterima Oleh Bahasa Reguler yang Didefinisikan Nondeterministic Finite Automata (NFA)

Muhammad Ihsan[1], Ilden Abi Neri[2], Hafda Bayu Nanda[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Program Studi Teknik Informatika,Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if14120@students.if.itb.ac.id[1], if14145@students.if.itb.ac.id[2],
if14155@students.if.itb.ac.id[3]

Abstrak

Nondeterministic Finite Automata (NFA) adalah salah satu jenis dari *Finite State Automata* (FSA) yang berguna sebagai pengenal Bahasa Reguler. Karakteristik kunci dari NFA ini adalah membolehkan membaca satu atau lebih dari satu transisi untuk satu simbol masukan yang sama atau dengan kata lain untuk masukan simbol yang sama NFA bisa mentransisikannya ke beberapa state yang berbeda. Dengan menggunakan Algoritma *Depth First Search* (DFS) kita bisa menentukan apakah suatu string masukan didefinisikan oleh Bahasa Reguler yang digambarkan oleh suatu NFA, jika ya maka karakter terakhir dari string tersebut mentransisikan tepat ke *final state* dari NFA, jika tidak berarti karakter terakhir dari string tersebut tidak sampai ke *final state* dari NFA atau simbol atau karakter dari string masukan tersebut tidak didefinisikan sebagai kumpulan input simbol yang terdapat dalam bahasa reguler yang digambarkan oleh NFA. Karena sebuah NFA memungkinkan mempunyai lebih dari satu *final state* maka jika transisi telah sampai kesalah satu *final state* tapi karakter dari string masukan belum habis maka akan terjadi *backtracking* untuk kemudian mencari ke *final state* yang lain. Dalam kondisi lain *backtracing* terjadi jika string masukan sudah habis dibaca oleh NFA, tetapi transisi tidak mencapai *final state* sehingga akan terjadi *backtracking* untuk menemukan *final state* yang lainnya.

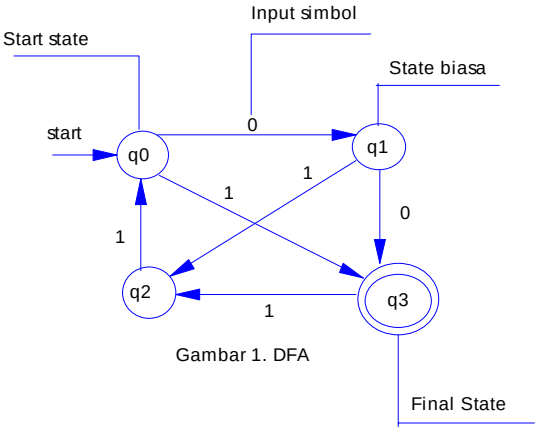
***Kata kunci**: makalah, karya ilmiah, strategi algoritmik, DFS, NFA, FSA, bahasa reguler*

1. Pendahuluan

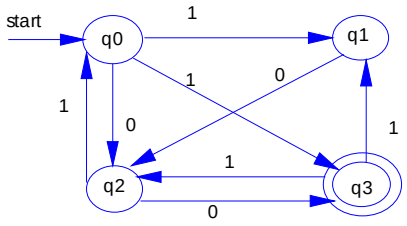
Menurut Noam Chomsky tata bahasa formal diklasifikasikan kedalam 4 kelas yang pertama kelas tata bahasa regular (tata bahasa jenis ke-3), kedua kelas tata bahasa bebas konteks (tata bahasa jenis ke-2), ketiga kelas tata bahasa peka konteks (tata bahasa jenis 1), seta yang keempat kelas tata bahasa tanpa pembatasan (tata bahasa jenis 0).

Kelas tata bahasa regular yang menjadi fokus kami dalam pembuatan makalah ini dapat dikenali dengan mesin pengenal bahasa regular atau dalam istilahnya disebut dengan *Finite State Automata* (FSA). FSA sendiri terbagi atas 2 macam yaitu Deterministic Finite Automata (DFA) dan Nondeterministic Finite Automata (NFA). Perbedaan yang mendasar dari DFA dan NFA ini adalah DFA untuk satu simbol masukan hanya bisa mentransisikan kesatu state sedangkan NFA untuk satu simbol masukan dapat mentransisikannya ke satu atau lebih state yang berbeda. Jadi, kita dapat menyimpulkan semua DFA merupakan bagian dari NFA.

NFA dan DFA dapat digambarkan dalam bentuk diagram transisi yang berupa graf berarah. Berikut contoh dari diagram NFA dan DFA :



Gambar 1. DFA



Gambar 2. NFA

Untuk memecahkan persoalan NFA dalam hal penentuan apakah sebuah string didefinisikan oleh bahasa regular yang digambarkan melalui NFA, kita dapat memanfaatkan algoritma *Depth First Search* (DFS) dimana setiap state dalam NFA dijadikan sebagai node dalam pohon yang dibentuk secara DFS.

NFA ini sendiri sangat berguna sekali sebagai dasar pembuatan *Compiler* berbagai bahasa pemograman yang kita kenal saat ini dan juga salah satu penerapan dari NFA yang sangat popular saat ini adalah Mesin Jaja (*Vending Machine*) yakni mesin yang dapat mengeluarkan makanan/minuman yang diinginkan pembeli setelah ia memasukkan sejumlah koin dan menekan tombol tertentu sesuai dengan jenis makanan/minuman yang dikehendakinya.

2. Ruang Lingkup

Makalah ini membahas tentang:

a. Konsep dari Nondeterministic Automata

b. Algoritma Depth First Search (DFS) untuk NFA

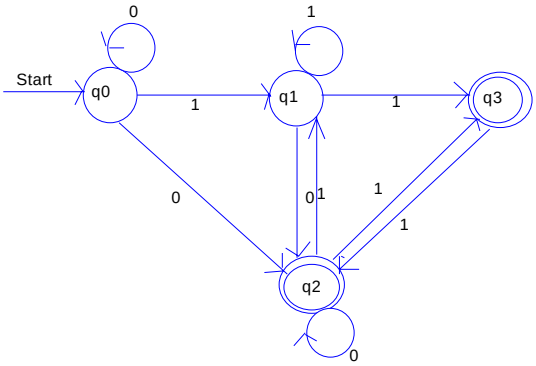
3. Pembahasan

3.1 Konsep dari Nondeterministic Automata (NFA)

Secara Formal, mesin NFA dinyatakan dalam 5-*tuple* (Q, E, f, q, FS), yang masing masing artinya adalah :

Q : himpuna berhingga status
E : himpunan berhingga symbol alphabet
f : fungsi transisi yang memetakan Q dengan parameter masukan Q dan E
q0 : merupakan status awal dari NFA dan salah satu bagian dari Q
FS : himpunan status akhir (*final state*) yang juga merupakan bagian dari Q

Sebagai contoh :
Misalkan didefinisikan sebuah NFA dengan gambar diagram transisi sebagai berikut :



Gambar 3 NFA

Pertanyaan:

a. Tentukan hasil f (q0, 1)?
b. Tentukan apakah string 1001 didefinisikan oleh bahasa regular yang digambarkan NFA diatas?

Jawab :

a. f(q0, 1) = q1
b. f(q0, 1001) = f(f(f(f (q0, 1), 0), 0), 1)
= f(f(f(q1, 0), 0), 1)
= f(f(q2, 0), 1)
= f(q2, 1)
= q3
karena q3 adalah *final state* maka string 1001 didefinisikan oleh Bahasa Regular yang digambarkan oleh NFA tersebut.

3.2 Algoritma DFS untuk NFA

Untuk persoalan secara umum algoritma DFS didefinisikan oleh runtutan langkah sebagai berikut :

1. Masukkan simpul akar ke dalam antrian *Q*. Jika simpul akar = simpul solusi, maka Stop.
2. Jika *Q* kosong, tidak ada solusi. Stop.
3. Ambil simpul *v* dari kepala (*head*) antrian.
4. Jika kedalaman simpul *v* sama dengan batas kedalaman maksimum, kembali ke langkah 2.
5. Bangkitkan semua anak dari simpul *v*. Jika *v* tidak mempunyai anak lagi, kembali ke langkah 2. Tempatkan semua anak dari *v* di awal antrian *Q*.
6. Jika anak dari simpul *v* adalah simpul tujuan, berarti solusi telah ditemukan, kalau tidak, kembali lagi ke langkah 2.

Khusus untuk penyelesaian masalah NFA simpul akar dalam pohon selalu merupakan q0 (*start state*), dan setiap node didalam pohon tidak *unique* artinya node yang sama bisa muncul beberapa kali didalam pohon yang dibangun dengan menggunakan algoritma DFS, untuk satu simbol masukan hanya boleh menghasilkan state maksimal 2 buah.

Aturan-aturan untuk pembentukan pohon secara DFS ini kami spesifikasikan sebagai berikut :

Jika untuk 1 simbol masukan bila dilakukan transisi mempunyai kemungkinan hasil transisi 2 atau lebih kemungkinan maka urutan prioritas hasil transisi adalah :

1. State hasil transisi yang belum pernah dikunjungi misal f(q3, 0) hasil nya bisa 2 kemungkinan yaitu q1 dan q2 tapi q1 pernah dikunjungi sebelumnya maka hasil transisi yang dipilih terlebih dahulu adalah q2

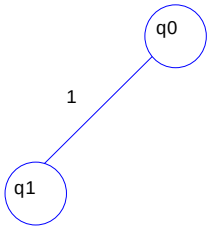
2. Jika kedua-duanya telah dikunjungi maka state yang dipilih adalah state yang paling dulu dikunjungi, misal f(q3, 0) hasil nya bisa 2 kemungkinan yaitu q1 dan q2 tapi q1 dan q2 pernah dikunjungi sebelumnya tapi q1 lebih dahulu dikunjungi maka hasil transisi yang dipilih terlebih dahulu adalah q1

3. Jika seluruh state hasil transisi belum pernah dikunjungi maka hasil yang dipilih adalah index statenya yang terkecil

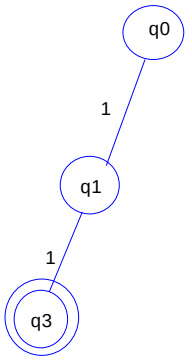
4. State yang tidak sama dengan parameter masukan misal f(q3, 0) hasil nya bisa 2 kemungkinan yaitu q3 dan q2 tapi q3 juga merupakan parameter masukan maka hasil transisi yang dipilih terlebih dahulu adalah q2.

Untuk penjelasan lebih baik NFA pada Gambar 3 kita manfaatkan kembali untuk menentukan apakah string 1100101 diterima oleh bahasa regular yang digambarkan oleh NFA pada gambar 3:

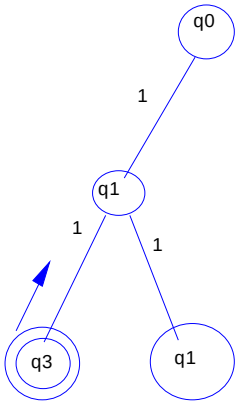
Langkah I :
NFA membaca Input symbol 1, state bertransisi dari q0 ke q1



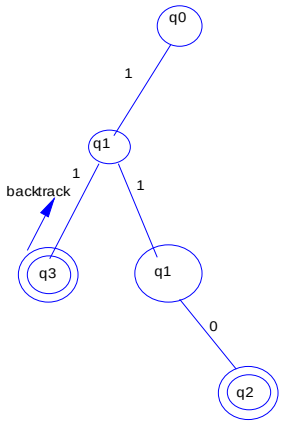
Langkah II :
NFA membaca Input symbol 1, state bertransisi dari pada 2 kemungkinan berdasarkan aturan maka hasil transisi adalah ke q3.



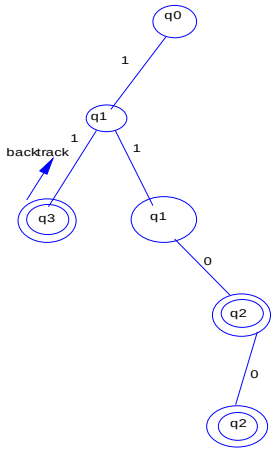
Langkah III :
NFA membaca Input symbol 0, karena state f(q3,0) tidak mempunyai hasil maka dari state q3 akan *backtrack* ke state q1, dari state q1 kemudian baca lagi input simbol 1 dan akan bertransisi tetap ke state q1



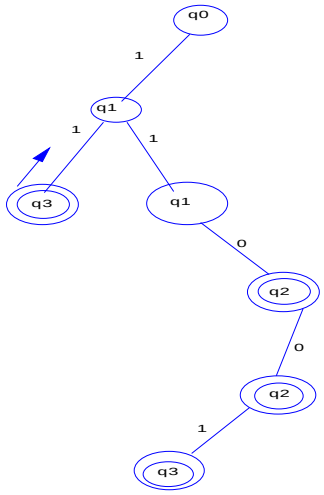
Langkah IV :
NFA membaca Input symbol 0, state bertransisi dari q1 ke q2



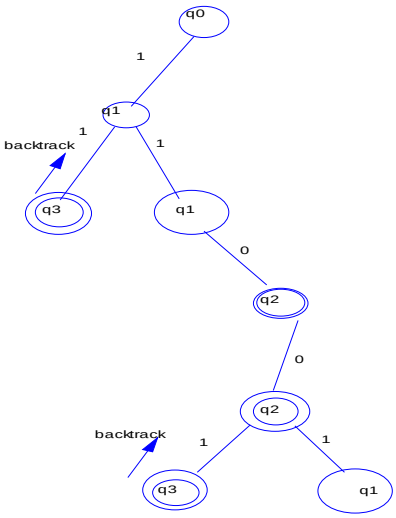
Langkah V :
NFA membaca Input symbol 0, state bertransisi dari q2 tetap ke q2



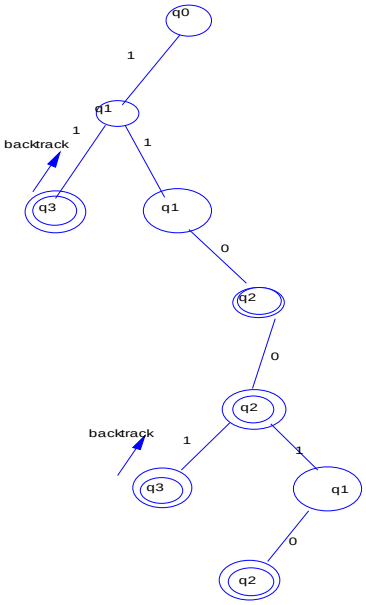
Langkah VI :
NFA membaca Input symbol 1, state bertransisi dari q2 ke q3



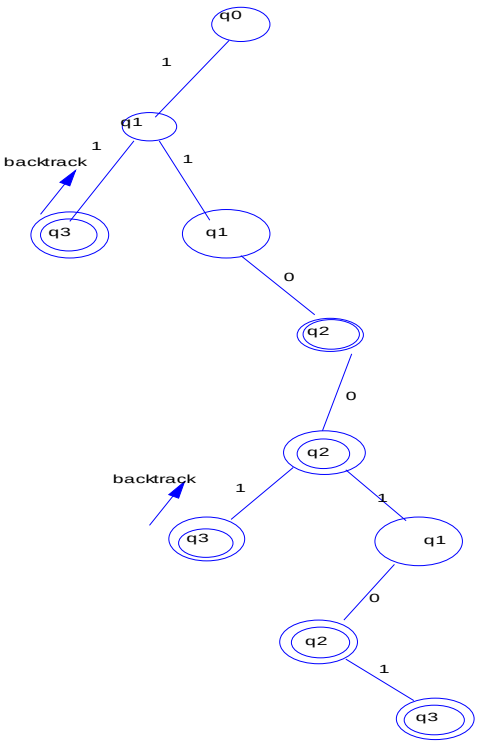
Langkah VI :
NFA membaca Input simbol 0, karena state f(q3,0) tidak mempunyai hasil maka dari state q3 akan *backtrack* ke state q2, dari state q2 kemudian baca lagi input simbol 1 dan akan bertransisi ke state q1



Langkah VII :
NFA membaca Input simbol 0, state bertransisi dari q1 tetap ke q2



Langkah VII :
NFA membaca Input simbol 1, state bertransisi dari q2 tetap ke q3. Karena semua input simbol telah dikunjungi dan state q3 juga merupakan state akhir (*final state*) maka string 1100101 diterima sebgai bahasa regular yang digambarkan oleh DFA gambar 3.



Berdasarkan keterangan diatas kita bias merumuskan pseucode dari Algoritma DFS untuk menyelesaikan apakan sebuah string diterima oleh sebuah NFA atau tidak :

```
Procedure DFStoNFA (input simbol : String, output Diterima : Boolean, input/ouput T : DFSTree  )

{ NFA sudah terdefinisi dalam bentuk fungsi fungsi transisi beserta hasilnya }
{ Diterima akan bernilai true jika String masukan diterima sebagai bahasa regular yang digambarkan oleh NFA dan akan bernilai false jika tidak diterima }

Deklarasi

        CurrentState : NodeTree

        {Merupakan NodeTree yang akan menjadi parameter masukan dan akan selalu berubah setiap penambahan state ke PohonDFS}

        TempHasil : array [0..1] of NodeTree
        Temp : NodeTree
        NBHasil : integer
        I : integer
        Tempsimbol = String
        cc : character

        Procedure BentukPohonDFS(input T : DFSTree)
        {Membentuk Pohon denga Algoritma DFS dengan akar merupaka state awal}

        ProcedureBacaKaraktersimbol(input/ouput S : String, ouput c : character)
        {Membaca satu karakter dari s dan kemudian karakter yang telah dibaca dikurangi dari s kemudian di outputkan lagi, dan juga menguotputkan karakter yang dibaca}

        Procedure TransisikanSimbol(input N : NodeTree, ouput I : integer, ouput Temp : array [1..2]of NodeTree)
        {Berguna untuk mentransisikan state kemudian mengoutputkan state hasil transisi dalam bentuk array of nodetree dan jumlah State yang dihasilkan oleh transisi tersebut }

        Procedure AddTree(input/output T : DFSTree , input N : NodeTree)
        { Menambahakan node baru ke Pohon DFS sekaligus merubah Current State dari PohonDFS
        }

        Procedure BackTrack(input/output T : DFSTree)
        {Mengubah posisi dari Current State balik ke node sebelumnya atau parent current node}

        function NBNode (input T : DFSTree) : integer
        {Menghasilkan jumlahnode yang terdapat didalam DFSTree}

        function PernahDikunjungi(N : NodeTree) : boolean
        {Jika N Pernah Dikunjungi di dalam NFA maka return true jika tidak false}

        function FinalState (N : NodeTree) : boolean
        {Jika N adalah Final State dalam NFA maka return true jika tidak return false}

{Ket : Queue pada DFS secara umum kami ganti dengan array yakni TempHasil}

Algoritma

BentukPohonDFS(T)
Tempsimbol = simbol
BacaKarakterSimbol(TempSimbol, cc)
TransisikanSimbol(CurrentState,    cc,    I, TempHasil)


  While (TempSimbol != StringKosong)

        If (I = 0) then
                If (NBNode(T) > 1) then
                        BackTrack(T)
        elseif (I = 1) then
                AddTree(T, TempHasil[0])

        else {I = 2}
                if
                   (PernahDikunjungi(TempHasi
                   l[0])) and
                   (!PernahDikunjungi(TempHas
                   il[1])) then


                AddTree(T,TempHasil[1])

                Elseif
                   (!PernahDikunjungi(TempHas
                   il[0])) and
                   (!PernahDikunjungi(TempHas
                   il[1])) then

                    if
                       (!ParameterMasukan
                      (TempHasil[0])and
                      !ParameterMasukan
                      (TempHasil[1])) then

                       Temp=
                       PertamaDikunjungi(Temp
                       Hasil)
                       AddTree(T,Temp)

                   ElseIf
                       (ParameterMasukan
                      (TempHasil[0])and
                      !ParameterMasukan
                      (TempHasil[1])) then
```

```
                    AddTree(T,TempHasil[1]
                    )

                ElseIf
                    (!ParameterMasukan
                   (TempHasil[0])and
                   ParameterMasukan
                   (TempHasil[1])) then

                    AddTree(T,TempHasil[0]
                    )

            Elseif

               (!PernahDikunjungi(TempHas
               il[0])) and
               (PernahDikunjungi(TempHasi
               l[1])) then


               AddTree(T,TempHasil[0])

            Elseif

               (PernahDikunjungi(TempHasi
               l[0])) and
               (PernahDikunjungi(TempHasi
               l[1])) then

                    Temp=
                    PertamaDikunjungi(Temp
                    Hasil)
                    AddTree(T, Temp)

          BacaKarakterSimbol(TempSimbol,
cc)
         TransisikanSimbol(CurrentState,
          cc, I, TempHasil)

   {Akhir dari While}
  if (FinalState(CurrentState)) then
      Diterima = true
  Elseif
      Diterima = false
```

4. Kesimpulan

Untuk menentukan apakah sebuah string dapat diterima oleh sebuah NFA kita dapat menggunakan algoritma DFS, namun makalah ini memiliki banyak keterbatasan berupa asumsi-asumsi dan spesifikasi yang kami rumuskan sendiri sehingga NFA yang akan di proses pun hanya merupakan NFA yang sederhana atau tidak terlalu kompleks.

Hal ini tentunya bukan masalah menurut kami, karena untuk bisa menentukan apakah sebuah string diterima atau tidak pada sebuah NFA yang kompleks dengan menggunakan DFS, membutuhkan penelitian lebih lanjut dan lebih mendalam lagi.

Dan mudah-mudahan apa yang telah kami kerjakan pada makalah ini bisa menjadi awal untuk penelitian lebih lanjut penentuan apakah sebuah string diterima oleh NFA atau tidak dengan memanfaatkan algoritma DFS.

Atas segala kekurangan kami mohon maaf, dan kalau ada masukan atau saran demi perbaikan makalah ini kedepan silahkan hubungi *e-mail* kami yang diatas.

5. Daftar Pustaka

[1]. Ir. Rinaldi Munir M.T, *Strategi Algoritmik*, Program Studi Teknik Informatika ITB, 2006.

[2]. Dra. Harlili M.Sc, Ir. Hanye S. Dulimarta Ph.D, Ir. Rinaldi Munir M.T., *IF351 Teori Bahasa Formal*, Departemen Teknik Informatika ITB, 2001