Aplikasi Algoritma Runut Balik dan Pencocokan String dalam *Code Translation* dari Suatu Bahasa Pemrograman ke Bahasa Pemrograman Lain pada Level Analisis Leksikal

Herbert Bontor Meilando Sinaga[1],
Made Harta Dwijaksara[2], Yohanes[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if14125@students.if.itb.ac.id[1], if14137@students.if.itb.ac.id[2],
if14158@students.if.itb.ac.id[3]

Abstrak

Algoritma runut balik dan pencocokan string merupakan algoritma yang sangat sering dimanfaatkan dalam berbagai program khususnya yang erat kaitannya dengan proses *searching*. Algoritma runut balik yang dimanfaatkan untuk searching didalam pohon merupakan optimalisasi dari metode DFS. Sedangkan algoritma pecocokan string juga merupakan algoritma yang merepresentasikan proses searching namun dilakukan pada rangkaian karakter (*string*). Dalam translasi suatu kode program salah satu caranya dilakukan dengan menggantikan semua token-token dalam bahasa yang bersangkutan dengan token yang bersesuain pada bahasa tujuan. Proses ini memanfaatkan algoritma pencocokan string untuk menemukan token-token yang dimaksudkan. Hanya saja proses ini akan sangat memakan waktu lama dalam proses searching token, karena seperti kita ketahui token dalam suatu bahasa tidaklah sedikit, maka dari itu diperlukan suatu teknik untuk mempersingkat proses pencarian tersebut salah satu caranya yaitu dengan mempersempit ruang pencarian token. Jadi yang dicari hanya token-token yang dipergunakan dalam bahasa yang akan ditranslasikan.

***Kata kunci**: algoritma runut balik,algoritma pencocokan string, DFS, token*

1. Pendahuluan

Translasi kode program dari suatu bahasa ke bahasa yang lain memang belum begitu terasa manfaatnya. Namun bagi seseorang yang sedang menekuni berbagai ragam bahasa pemrograman hal ini mungkin menjadi perlu. Karena ketika mereka ingin berpindah dari satu bahasa pemrograman ke bahasa pemrogram lain maka mereka tidak lagi perlu mengatikan secara manual kode program yang telah mereka buat sebelumnya. Sehingga nantinya akan dapat menghemat waktu dan tenaga.

Namun karena sepengetahuan penulis belum ada suatu teknik yang membahas hal ini secara khusus maka makalah ini dibuat untuk membahas teknik itu. Pertama dengan menjabarkan metode translasinya dan berikutnya memaparkan kendala yang dihadapi ketika menggunakan teknik ini.

Cara yang digunakan pada pembahasan ini merupakan pengetahuan penulis setelah mengikuti beberapa mata kuliah diataranya adalah Strategi Algoritmik dan Teori Bahasa Formal sehingga nantinya pembahasan mungkin akan sedikit terbatas. Namun dengan itu penulis merasa sudah cukup dapat membahas teknik ini walaupun tidak dengan sempurna.

2. Ruang Lingkup

Pembahasan pada makalah ini hanya menyangkut suatu metode translasi kode program dan terutamanya dikhususkan untuk bahasa dengan standar bahasa yang jelas dan memiliki standar baku internasional. Contohnya bahasa C dan bahasa Pascal. Dan juga karena nantinya dalam translasi hanya akan menggunakan bantuan DFA (*deuterministic finite automata*) sebagai sarana pemodelan suatu bahasa maka translasi hanya akan bisa sampai level analisis leksikal.

3. Susunan Bahasa dalam Suatu Kode Program

Suatu code program biasanya terdiri dari suatu token-token yang berasal dari bahasa bawaan bahasa pemrograman tersebut dan kumpulan karakter yang bukan milik bahasa pemrograman yang kita kenal sebagai *idetifier*. Token-token itu akan merepresentasikan suatu fungsi setelah kode tersebut dikompilasi dan akan dikerjakan program setelah program dijalankan (*run*). Sedangkan identifier berperan sebagai parameter untuk berjalannya suatu program. Jadi, keberadaan dua komponen ini menjadi syarat harus suatu program yang benar.

4. Tahapan Translasi

Seperti yang telah dijelaskan sebelumnya bahwa translasi dilakukan dengan mengidetifikasi token-token suatu bahasa yang akan ditranslasikan. Ada dua tahapan utama dalam traslasi ini, yaitu :

Tahap I : Traslasi token.

a) Pada awal *translate* DFA yang memodelkan bahasa pemrograman asal dan yang dituju kedalam struktur pohon sehingga dapat diterapkan algoritma runut balik didalamnya. Sebenarnya proses ini bisa dilakukan hanya sekali saja pada awal program dibuat, karena bagaimanapun juga DFA yang memodelkan suatu bahasa pemrograman akan tetap sama sehingga bisa dipakai terus.

b) Buat daftar tabel token dari bahasa yang dari bahasa yang akan dituju beserta nama dari tokennya, sama halnya dengan poin a bahwa proses ini tidak perlu dilkukan berulang karena tabel ini akan bisa dipakai selamanya, kecuali bahasa pemrogrammannya berubah.

c) Lakukan traversal pada kode program yang akan ditranslasikan. Jika ditemukan token bahasa pemrograman maka dimasukan kedalam DFA bahasa yang akan dituju untuk diketahui apakah token tersebut juga dikenali pada bahasa yang dituju.

Tahap II : Penyesuain token

a) Diawali dengan membentuk tabel penyesuaian token dari bahasa yang ditranslate ke bahasa yang dituju. Proses ini juga hanya perlu dilakukan sekali saja.

b) Traslate semua token yang memiliki yang berpasangan

.Contoh, jika kita ingin metranslasikan kode yang ditulis dalam bahasa Pascal berikut ke dalam bahasa C

Kode program asal dalam bahasa Pascal:

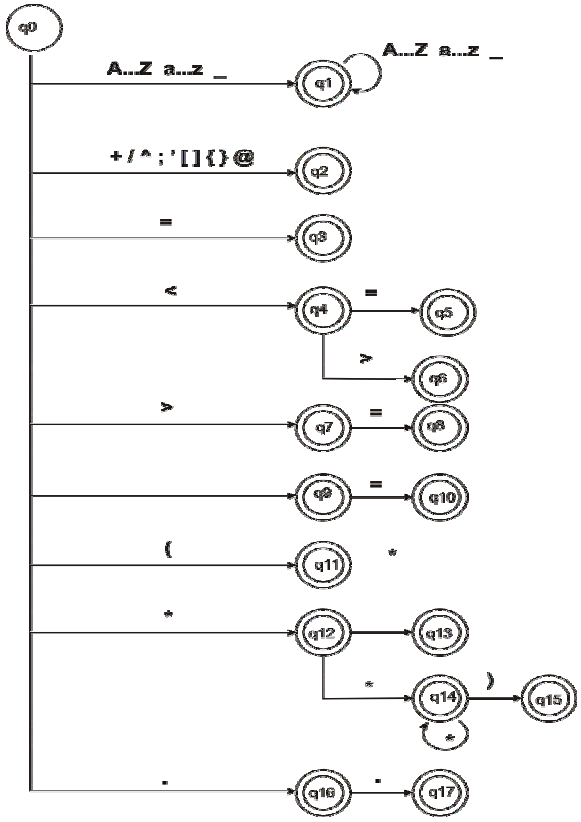
```
Contoh 1 : Perulangan
i := 0;
a := 0;
while(i<5)
begin
        a := a + 1;
        i := i + 1;
end;
```

Proses Translasi :

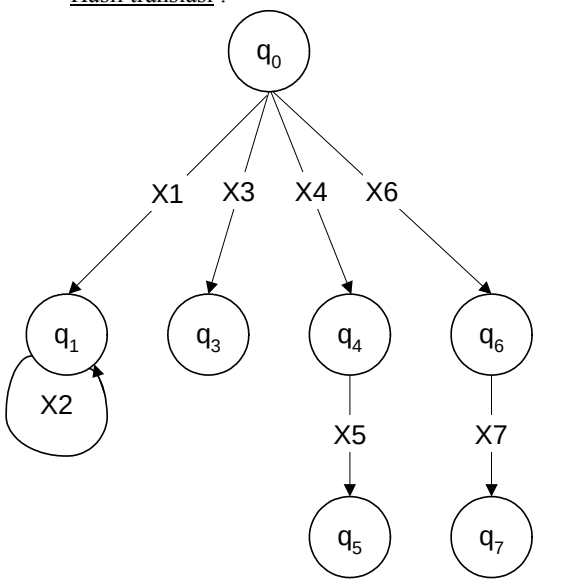
4.1 Tahap I

4.1.1 Translate DFA kedalam struktur pohon. DFA yang memodelkan bahasa Pascal itu sendiri adalah sebagai berikut :

{traslasi DFA ke struktur pohon pada contoh berikut dilakukan hanya sebagian disesuaikan dengan contoh potongan program yang akan ditranslate}



Hasil translasi :



X1 = { A…Z a…z _ } => token indeftifier
X1 = { A…Z a…z 1…9 _ } => token indeftifier
X3 = { + / ^ ; ' [ ] { } ( ) @ } => token x { x ϵ X3}
X4 = { : } => token titik dua
X5 = { := } => token asigment
X6 = { < > } => token x {x ϵ X6 }
X7 = { <> } => token tidak sama dengan

Nilai X1…X7 diatas maksudnya adalah input (karakter penyusun token) yang akan dimasukan merupakan kedalam pohon guna mengidentifikasi token yang dimaksudkan.

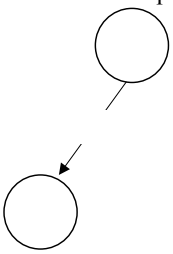
4.1.2 Selanjutnya buatkan daftar tabel penyesuain token untuk menemukan kesamaan fungsi token contoh:

Nama	Dalam Bahasa C
assignment	=
+	+
;	;
identifier	A…Z a…z _ 1…9

{*juga hanya dibuat untuk token yang akan digunakan pada contoh*}

4.1.3 Langkah berikutnya lakukan traversal, kemudian cari token apa yang ditemukan pada pohon yang telah terbentuk dengan metode runut balik (*backtracking*).

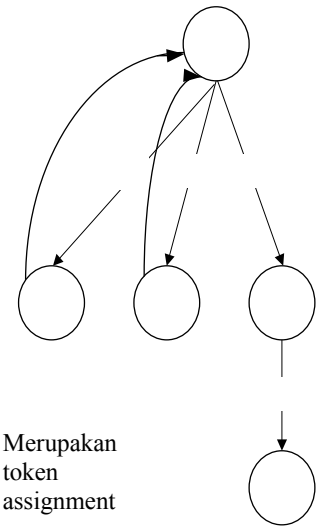
- Pertama yang akan dibaca adalah 'i', masukan kedalam pohon



Ditemukan bahwa i adalah token identifier, yaitu dari nilai X1

Karena diketahui token idetifier maka langsung disalin.

- Karakter selanjutnya yang dimasukan adalah ':='



Merupakan token assignment

Lihat kedalam tabel penyesuain bahwa token assignment pada bahasa C adalah '=', kemudian token ini disalin. Perlu diketahui bahwa algoritma backtracking yang diterapkan disini tidak seperti yang biasanya. Karena jika pada algoritma backtracking yang umum proses pencarian dilakukan sambil melakukan pembentukan node pohon, namun disini karena node pohon telah terbentuk sebelumnya jadi tidak perlu lagi membentuk node ketika melakukan pencarian. Yang perlu dilakukan membandingkan tiap karakter dan jika tidak ketemu lakukan backtracking.

Proses itu dilakukan terus sampai akhir program. Sehingga pada akhir nanti akan didapat hasil translasi pertama awal adalah sebagai berikut :

```
i = 0;
a = 0;
while(i<5)
begin
        a = a + 1;
        i = i + 1;
end;
```

4.2 Tahap II

4.2.1 Buat tabel penyesuain token dari bahasa Pascal ke bahasa C, contoh yaitu sebagai berikut :

Pascal	C
begin	{
end;	}
end.	}
procedure	void
function …..type	type (return type)
mod	%
div	/

{*juga hanya dibuat untuk token yang akan digunakan pada contoh*}

4.2.2 Kemudian lakukan penyesuain dengan mengganti semua identifier yang memiliki pasangan karena identifier itu merupakan token dalam bahasanya sehingga perlu dicarikan token yang identifier yang bersesuai dalam bahasas tujuan yang juga merupakan token.

Proses penggantian didahului dengan melakukan pencarian token pada kode program hasil translasi. Pencarian ini dilakukan dengan memanfaatkan algoritma pencocokan string. Hal ini karena mempertimbangkan bahwa yang dicari adalah token dimana token itu sendiri adalah berupa string dan proses pencarian dilakukan pada sekumpulan string.

Jika diterapkan sebagai berikut :
Untuk token <u>begin</u>

```
i = 0;
begin
 begin
  begin
   begin
    begin
     begin
      begin
       begin
a = 0;
begin
 begin
  begin
   begin
    begin
     begin
      begin
       begin
while(i<5)
begin
 begin
  begin
   begin
    begin
     begin
      begin
       begin
        begin
         begin
          begin
           begin
            begin
```

begin ➔ ketemu (ganti dng. pasangannya)

```
            ‘{ ‘
begin
            a = a + 1;
            i = i + 1;
end;
```

Proses itu dilakukan terus sampai semua indetifier yang berupa token dalam bahasa pascal tergantikan dengan pasangannya dalam bahasa C. Hasil dari proses diatas adalah kode program sebagai berikut.

```
i = 0;
a = 0;
while(i<5)
{
            a = a + 1;
            i = i + 1;
}
```

Akhirnya kita dapat kode akhir dair hasil translasi yang telah dalam bentuk kode program dalam bahasa C.

5. Analisis

Teknik translasi diatas mungkin hanya akan cocok pada bahasa-bahasa pemrograman tertentu. Karena seperti yang telah dibahas sebelumnya, bahasa pemrograman tersebut harus memiliki standar yang berlaku umum dan diterima oleh semua *compiler*. Kenapa harus demikian? hal ini untuk mempermudah dan memperjelas dalam pembuatan DFA, tabel token dan tabel penyesuaian identifier nantinya.

Hal lain yang mungkin sangat mencolok disini adalah bahwa teknik ini mungkin tidak akan menghasilkan suatu kode program hasil translasi yang benar-benar siap untuk di*compile* karena pada penyelesaiannya teknik ini hanya menggunkan sarana DFA untuk memodelkan suatu bahasa pemrograman padahal DFA itu sendiri hanya mempunyai efek kebenaran sampai level analisis leksikal saja. Artinya hanya benar untuk menentukan bahwa suatu token diterima oleh suatu bahasa pemrograman. Dilain sisi untuk menciptakan suatu bahasa yang benar-benar akurat diperlukan suatu analisis sampai level sintaks. Untuk itu diperlukan suatu alat bantu lagi yang dikenal dengan nama PDA yaitu *Pushdown Automata*. Oleh karena keterbatasan kemampuan penulis dalam artian disini ketika penulis membuat tulisan ini penulis belum sempat mempelajari materi tersebut, sehingga belum bisa memasukannya sebagai bahan solusi permasalahan. Namun intinya disini bukanlah bagaimana suatu kode itu ditranslasikan dengan benar. Melainkan bagaimana suatu kode dapat ditranslasikan dari suatu bahasa pemrograman ke dalam bahasa pemrograman yang lain dengan algoritma yang tepat.

6. Kesimpulan

Setelah mengulas salah satu teknik translasi kode program diatas dapat ditarik kesimpulan bahwa :

6.1 Algoritma runut balik dan pencocokan string merupakan salah satu algoritma yang multi guna karena dapat diterapkan dalam berbagai situasi dan kondisi. Serta pemanfaatannya tidak selalu sama untuk kasus yang berbeda.

6.2 Kode hasil translasi dari teknik yang disajikan diatas tidak selalu bisa langsung di*compile* karena translasi baru hanya sampai level analisis leksikal dan perlu berangkat lebih jauh ke analisis sintaks untuk memastikan keakuratannya. Tapi hal tersebut tidak juga menutup kemungkinan bahwa kode hasil translasi ada yang sudah bisa di*compile*.

Daftar Pustaka

[1] John E. Hopcroft and Jeffrey D. Ullman, 1979, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing Company.

[2] Rinaldi Munir, 2006, *Diktat Kuliah IF2251 Strategi Algoritmik*, Penerbit ITB.

[3] Yohannes, Made, Herbert, Joel and Hadi, 2006, *Tugas Kuliah Otamata dan Teori Bahasa Formal.*