Penerapan Program Dinamis dalam Algoritma Cocke-Younger-Kasami dan Earley untuk Pemrosesan Bahasa Natural

Muhammad Rizki Duwinanto - 13515006
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung. Jalan Ganesha 10 Bandung 40132, Indonesia
13515006@std.stei.itb.ac.id

***Abstract*—Pemrosesan bahasa natural sangat diperlukan bagi komputer dalam memproses masukan dan keluaran bagi manusia. Dalam ilmu komputer, terdapat upailmu yaitu teori bahasa formal dan automata yang mempelajari ilmu tentang bahasa. Dalam Teori Bahasa akan dikenal Bahasa ynag merupakan himpunan grammar, dan grammar atau tata bahasa yang merupakan aturan membentuk Bahasa. Grammr memiliki empat klasifikasi menurut chomsky, dan yang dipakai untuk analisis algoritma ini adalah grammar bebas berkonteks dengan bentuk normal chomsky. Tentu saja untuk menguraikan masukan string dalam bahasa sehingga dapat cocok dengan grammar bebas berkonteks dalam bentuk normal Chomsky diperlukan algoritma yang tentu juga harus memproses string sehingga bisa diketahui juga kalkulasi yang tepat dalam bahasa tersebut. Algoritma tersebut memakai strategi program dinamis sehingga bisa mengkalkulasi apakah string tersebut masuk kedalam himpunan grammar yang ada. Algoritma program dinamis juga memiliki karakteristik yaitu memiliki tahap dan berbeda dengan greedy dalam mengambil keputusan. Algoritma program dinamis yang dipakai bernama Cocke-Younger-Kasami dan Earley yang menguraikan string dalam bahasa. Algoritma tersebut memiliki approach yang berbeda dengan Cocke-Younger-Kasami memiliki approach mundur sedangkan Earley memiliki approach maju. Algoritma ini memiliki kekurangan dan kompleksitas waktu polinomial.**

Keywords : Bahasa, Penguraian, Tahap, Dinamis, Grammar

I. PENDAHULUAN

Pada semester tiga lalu, saya mempelajari Teori Bahasa Formal dan Automata. Tentunya Teori Bahasa Formal dan Automata merupakan mata kuliah yang favorit dan menyenangkan. Karena saya jadi mengerti bagaimana proses berjalannya penguraian (*parsing)* suatu bahasa. Yang dipelajari di Teori Bahasa Formal dan Automata juga sangat menarik untuk diimplementasikan di bidang Intelegensi Buatan.

Namun, Teori Bahasa membicarakan Bahasa formal, untuk kepentingan perancangan kompilator (*compiler*) dan pemrosesan naskah (*text processing*). Bahasa formal adalah kumpulan kalimat yang dibangkitkan oleh sebuah tata bahasa (*grammar*) yang sama atau dua atau lebih tata bahasa berbeda. Berbeda dengan dengan bahasa formal adalah bahasa natural.

Bahasa natural adalah bahasa yang dipakai di masyarakat. Bahasa Indonesia dan Inggris merupakan contoh bahasa natural. Tentunya grammar yang dibuat dipakai untuk meresmikan bahasa tersebut. Bahasa natural tentu perlu diproses sehingga masukan dan keluaran mesin dan menyamai kemampuan manusia, sehingga mempermudah kehidupan. Sehingga pemrosesan ini dapat menjadi dasar dari Intelegensi Buatan, dan akan dijelaskan di makalah ini bagaimana cara memproses dengan algoritma yang ada.

Dalam menguraikan Bahasa dikenal Algoritma Cocke Younger Kasami dan Earley, biasanya digunakan untuk memastikan apakah suatu string termasuk dalam Context-Free-Grammar dalam bentuk Chomsky Normal Form, yaitu suatu bentuk representasi tata bahasa dengan menggunakan himpunan aturan produksi. Algoritma diatas menggunakan Program Dinamis dengan dua cara yang berbeda pula untuk kedua Algoritma. Walaupun begitu performa algoritma, tetaplah time consuming dan memiliki kekurangan karena ambiguitas yang dihasilkan oleh Manusia sebagai makhluk yang tidak sempurna.

Sehingga dalam Makalah Strategi Algoritma tentang Algoritma Earley dan CKY dalam Pemrosesasan Bahasa Natural atau *natural language processing*, akan disertakan dengan eksperimen program dan penjelasan tentang aplikasi program dinamis dalam algoritma tersebut dan dijelaskan juga pengaplikasian Algoritma tersebut, seperti Intelejensi Buatan. Makalah ini diharapkan dapat menjelaskan pengaruh dan manfaat Strategi Algoritma Program Dinamis untuk Pemrosesan Bahasa Natural yang biasa dipakai untuk Pencarian seperti Google, Layanan Penerjemah seperti Google Translate, ataupun Layanan Intelegensi Buatan seperti Siri dan Google Voice.

Gambar 1. Contoh Penggunaan Pemrosesan Bahasa Natural [4]

II. DASAR TEORI PROGRAM DINAMIS

A. *Pengertian dan Konsep Dasar*

Algoritma Program Dinamis atau Dynamic Programming adalah suatu metode pemecahan masalah dengan cara menguraikan solusi yang kompleks menjadi beberapa tingkatan atau tahap. Setiap tahap dapat dipandang dari rangkaian keputusan yang saling berkaitan satu sama lain.

Tujuan utama dari model ini adalah untuk mempermmudah penyelesaian persoalan optimasi yang mempunyai karakteristik tertentu. Inti dari program dinamis ini adalah membagi satu persoalan atas beberapa bagian persoalan yang dalam program dinamis disbeut tahap (*stage*), kemudian memecahkan tiap tahap dengan mengoptimalkan keputusan atas tiap tahap sampai seluruh persoalan telah terpecahkan .

Istilah "Program Dinamis" muncul karena menggunakan proses memoisasi dan perhitungan solusi menggunakan tabel yang ukurannya dapat berubah tiap tahap. Kata "Program" tidak berarti berhubungan dengan kode program, namun adalah perencanaan pada penyelesaian masalah.

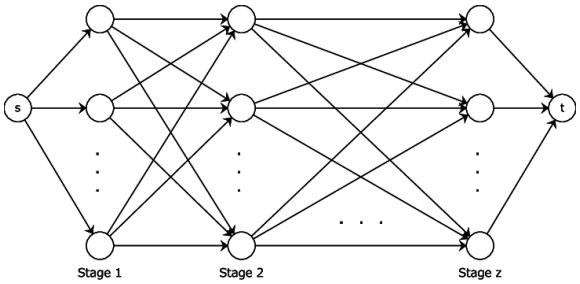
Prinsip dari program dinamis tentu sangat kongruen dengan algoritma *greedy* dimana perbedaanya adalah algoritma *greedy* hanya mempedulikan keputusan yang dibuat dan bersifat optimum lokal, sedangkan program dinamis terdapat lebih dari satu rangkain keputusan yang dipertimbangkan sehingga keputusan yang dibuat pada setiap tahapan akan menghasilkan solusi yang optimum global.

Program dinamis juga memiliki metode yang hampir sama dengan *bruteforce.* Akan tetapi algoritma Program dinamis tidak senaif *bruteforce.* Program dinamis pada dasarnya akan menyimpan semua hal (dapat berupa rumus/langkah yang menghasilkan nilai) yang telah dikomputasi sebelumnya, sehingga setiap kali melakukan komputasi akan mengecek terlebih apakah rumus/langkah yang menghasilkan nilai tersebut sudah pernah disimpan sebelumnya (sudah dikomputasi sebelumnya). Apabila hal tersebut sudah pernah dikomputasi, waktu untuk mengkomputasi hal yang sama untuk seterusnya setelah komputasi hal tersebut akan membutuhkan waktu yang konstan, sehingga algoritma ini dapat dilakukan dalam waktu polinomial. Sedangkan bruteforce akan mencoba semuanya tanpa komputasi yang menunjukan keputsan sehingga bisa memakan waktu lama bahkan hingga eksponensial.

B. *Karakteristik Persoalan Program Dinamis*

Persoalan yang hendak diselesaikan dengan program dinamis memiliki karasteristik sebagai berikut.

1. Persoalan dapat dibagi menjadi beberapa tahap, yang pada setiap tahapnya hanya diambil satu keputusan.
2. Masing-masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Secara umum, status merupakan bermacam kumngkinan masukan yang ada pada tahap teresbut. Jumlah satus bias berhingga (*finite*) atau tak berhinggan (*infinite*). Terlihat pada gambar di bawah perbedaan antara tahap dan status yang diberikan pada graf tahap-majemuk (*multistage graph*). Tiap simpul di dalam graf tersebut menyatakan status, sedangakan V1, V2 dan seterusnya menyatakan tahap.



Gambar 2. Graf Majemuk-Tahap [4]

https://www.researchgate.net/profile/Bartosz_Musznicki/publication/265412778/figure/fig1/AS:392039154896896@1470480826291/Figure-3-A-general-structure-of-a-multi-stage-graph.png)

3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya. Artinya, setiap hasil keputusan yang diambil pada setiap tahap dimemoisasi untuk digunakan pada tahap selanjutnya seperti konsep dasar program dinamis yang telah dipaparkan pada bagian sebelumnya.
4. Ongkos (*cost*) pada suatu tahap meningkat secar teratur (*steadily*) dengan bertambahnya jumlah tahapan. Hal ini dapat dimengerti dengan mudah karena setiap upapersoalan diintegrasikan kembali, ongkos global persoalan akan bertambah seiring berkurangnya upapersoalan akan bertambah seiring berkurangnya upapersoalan yang tersisa.
5. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan dan ongkos pada tahap tersebut. Hal ini cukup jelas dengan penjelasan yang sama dengan poin yang sama dengan poin 4.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan di tahap sebelumnya. Hal ini disebabkan oleh karena setiap upapersoalan (tahap) memiliki hasil optimal yang berbeda dengan upapersoalan lainnya.
7. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap k memberikan keputusan terbaik untuk setiap status pada tahap k+1. Karakteristik ini pun merupakan salah satu cara agar program dinamis dapat melakukan memoisasi terhadap nilai/hasil dari suatu rumus/rekurens/komputasi yang sudah pernah dilakukan oleh program dinamis agar dapat digunakan kembali pada komputasi selanjutnya yang memiliki parameter yang sama pada parameter nilai komputasi yang sudah dicatat oleh algoritma.
8. Prinsip optimalitas berlaku pada persoalan tersebut. Hal ini harus diutamakan karena tujjuan dari program dinamis salah satunya adalah agar permasalahan dapat diselesaikan dengan waktu yang tidak banyak.

Terdapat dua ancangan (*approach*) untuk menyelesaikan masalah dengan program dinamis, yaitu maju (*up-down*) dan mundur (*bottom-up*). Sebagai contoh x_1, x_2,..., menyatakan peubah keputusan yang harus dibuat masing-masing untuk tahap 1,2,...n. Jika diselesaikan dengan program dinamis maju makan akan bergerak mulai dari tahap ke-1, tahap ke-2, hingga tahap ke-n. Sedangkan jika diselesaikan dengan program dinamis mundur akan dimulai tahap ke-n, tahap ke-n-1 hingga tahap ke-1.

Secara Umum, dapat disimpulkan terdapat empat langkah untuk melakukan metode algoritma program dinamis, yaitu :

1. Mengklasifikan struktur solusi optimal apakah maju atau mundur.
2. Definisikan secara rekursif nilai dari solusi optimal (membuat rumus/komputasi yang mempunyai basis dan fungsi rekursif untuk mendapatkan nilai optimal global dan lokal).
3. Hitung nilai solusi optimal secara maju atau mundur (lalu mencatat hasil perhitungannya).
4. Konstruksi solusi optimal (mengkonstruksikan solusi optimal lokal menjadi solusi optimal global.

III. Dasar Teori Bahasa Formal Dan Automata

A. Konsep Dasar

Teori bahasa membicarakan bahasa formal (*formal language*), terutama untuk kepentingan perancangan kompilator (*compiler*) dan pemroses naskah (*text processor*). Bahasa formal adalah kumpulan kalimat. Semua kalimat dalam sebuah bahasa bisa dibangkitkan oleh dua atau lebih tata bahasa berbeda. Dikatakan bahasa formal karena grammar dicipatkan mendahului pembangkitan setiap kalimatnya. Bahasa natural atau manusia bersifat sebaliknya, yaitu grammar diciptakan untuk meresmikan kata-kata yang hidup di masyarakat.

Automata adalah mesin abstrak yang dapat mengenali (*recognize*), menerima (*accept*), atau membangkitkan (*generate*), sebuah kalimat dalam bahasa tertentu.

Bahasa memiliki simbol atau entitas abstrak (seperti halnya pengertian titik dalam geometri). Sebuah huruf atau sebuah angka adalah contoh simbol. String adalah deretan terbatas (*finite*) simbol-simbol. Sebagai contoh, jika a, b, dan c adalah tiga buah simbol maka abcb adalah sebuah string yang dibangun dari ketiga simbol tersebut. Jika w adalah sebuah string, maka | w | adalah panjang string. String kosong direpesentasikan dengan ε.

B. Tata Bahasa dan Bahasa

Tentu bahasa memiliki tata bahasa (*grammar*) yang direpresentasikan dengan anggota alfabet dinamakan simbol terminal. Kalimat adalah deretan hingga simbol-simbol terminal. Bahasa adalah himpunan kalimat-kalimat. Anggota bahasa bisa tak hingga kalimat. Tata Bahasa memiliki simbol terminal {a,b,c,-,+,x,(,),**if, then, else**} dan non-terminal {A, B, C, S (Awal), *expr*}. Huruf kecil akhir alphabet melambangkan string yang tersusun atas simbol-simbol terminal misalnya : x, y, z. Huruf yunani melambangkan string yang tersusun atas simbol-simbol non-terminal atau campuran keduanya, misal α, β, dan γ. $\alpha \rightarrow \beta$ menunjukan produksi yaitu derivasi dapat dilakukan yang berarti proses pembentukan sebuah kalimat atau sentensial. Sentensial adalah string yang tersusun atas simbol-simbol terminal atau simbol non-terminal atau campuran keduanya. Kalimat adalah string yang tersusun atas simbol terminal. Pengertian terminal artinya berakhir jika sentensial yang dihasilkan adalah sebuah kalimat. Pengertian non-terminal berasal belum berakhir artinya Derivasi belum berakhir jika sentensial mengandung simbol nonterminal.

Grammar G didefinisikan sebagai pasangan 4 tuple : V_T, V_N, S, dan Q, dan dituliskan G(: V_T, V_N, S, Q) dimana :

V_T : Himpunan simbol-simbol terminal (atau himpunan token-token atau alphabet)

V_N : Himpunan simbol-simbol non terminal

S : Simbol awal

Q : Himpunan Produksi

Berdasarkan komposisi bentuk ruas kiri dan ruas kanan produksinya ($\alpha \rightarrow \beta$), Noam Chomsky Mengklasifikan 4 tipe Grammar :

1. Grammar tipe ke-0 : Unrestricted Grammar (UG)

   Ciri : $\alpha, \beta \in (VT \mid V N)^*, |\alpha| > 0$
2. Grammar tipe ke-1 : Context Sensitive Grammar (CSG)

   Ciri : $\alpha, \beta \in (VT \mid V N)^*, 0 < |\alpha| \leq |\beta|$
3. Grammar tipe ke-2 : Context Free Grammar (CFG)

   Ciri : $\alpha \in V_N , \beta \in (V_T \mid V_N)^*$
4. Grammar tipe ke-3 : Regular Grammar (RG)

   Ciri : $\alpha \in V_N , \beta \in \{V_T , V_T V_N\}$ atau $\alpha \in V_N , \beta \in \{V_T , V_N V_T\}$

Untuk setiap kelas bahasa Chomsky, terdapat sebuah mesin pengenal bahasa, yaitu :

Kelas Bahasa	Mesin Pengenal Bahasa
Unrestricted Grammar	Mesin Turing
Context Sensitive Grammar	Linear Bounded Automata
Context Free Grammar	Pushdown Automata
Regular Grammar	Finite Automaton

Tabel 1. Mesin Pengenal Bahasa

Yang utama dibahas di makalah ini adalah Context Free Grammar dalam bentuk Chomsky Normal Form. Bentuk Normal Chomsky adalah grammar bebas konteks dengan setiap produksinya berbentuk : $A \rightarrow BC$ atau $A \rightarrow a$. Transformasi CFG ke CNF adalah transformasi sebagai berikut :

1. Eliminasi semua produksi hampa, produksi hampa dikaitkan dengan nullable sehingga harus mengeliminasi ε. Caranya adalah buang produksi hampa kemudian tambahkan produksi lain yang merupakan produksi lama tetapi simbol nullable-nya yang diruas kanan produksi dicoret.
2. Eliminasi semua produksi unitas yang berbentuk A → B. Sehingga jika ada produksi tersebut atau derivasi dan ada produksi non-unitas akan menghapus produksi unitas dan membuat produksi gabungan. Tidak dilakukan jika dilakukan dengan derivasi tertutup.
3. Penerapan batasan bentuk ruas kanan produksi sehingga semua bentuk produksi masuk kedalam bentuk A → a dan A → $B_1B_2...B_n$, n lebih dari 2.
4. Penerapan batasan panjang ruas kanan produksi sehingga panjang untai ruas kanannya tidak lebih dari sama dengan 2.

IV. Pemrosesan Bahasa Natural Dengan Program Dinamis

A. Algoritma Cocke-Younger-Kasami

Algoritma Cocke-Younger-Kasami atau biasa disingkat CYK adalah Algoritma untuk Penguraiana (*Parsing*) dengan Context Free Grammar dan menggunakan program dinamis mundur. Algoritma ini ditemukan oleh John Cocke, Daniel Younger, dan Tadao Kasami.

Algoritma ini membutuhkan grammar bebas berkonteks untuk diubah menjadi bentuk normal Chomsky. Masukan dari algoritma ini adalah Grammar **A** dalam Chomsky Normal Form dan Kalimat X. Ide dari Algoritma adalah secara rekursif membuat penguraian secara mundur dan keluran C[i,j] yang mengandung produksi untuk mengenerasikan X[i]...X[j]. Berikut adalah Pseudocode Algoritma Cocke-Younger-Kasami.

```
function CYK (A:Grammar, S:String): table
 for each i = 1 to n
  for each production A → a
   if a == X[i]
    add (A → a, 0) to C[i,i]
 for each L = 2 to n
  for each i = 1 to n-L+1
   for each j = 1 to L-1
    for each production A → A1, A2
     if A1 in C[i, i+j] and A2 in C[i+j+1,
i+L]
      add (A → A1, A2, i+j) to C[i, i+l]
```

Algoritma ini memakai program dinamis maju karena mengisi tabel dengan hasil yang sudah dikalkulasi dimana dapat diselesaikan dalam waktu polynomial, dan tidak mengulang seperti bruteforce.

Kita dapat mendapatkan pohon penguraian dengan traversal dalam table/ dibuat yang juga mengandung aturan produksi yang dipakai dan titik dimana upapohon menjadi dua.

Untuk mengetahui apakah String tersebut masuk kedalam bahasa dapat meilhat C[0,n] apakah diisi oleh simbol non-terminal atau bukan. Jika iya, maka String tersebut dapat diterima dan memenuhi kaidah tata bahasa. Sebagai contoh diberikan CFG dengan CNF seperti dibawah, CFG ini merepresentasikan tata bahasa bahasa natural yaitu bahasa inggris dimana String awal direpresentasikan dengan S, NP adalah Subyek, N adalah Kata Benda, VP adalah objek Predikat, dan V adalah Verb, dan Det adalah determinan.

S -> NP VP
NP -> DET N
NP -> NP PP
PP -> P NP
VP -> V NP
VP -> VP PP
DET -> the
NP -> I
N -> man
N -> telescope
P -> with
V -> saw
N -> cat

Dan string yang akan dianalisis adalah “i saw the man with the telescope”.

Hal yang pertama kali dilakukan adalah membuat table dengan panjang string atau jumlah karakter yang ada sedemikian rupa sehingga string dapat dianalisis kata per-kata sehingga dapat identifikasi apakah jenis kata yang ada. Dapat dihubungkan bahwa ini adalah tahap pertama program dinamis yang ada.

NP	V	Det	N	P	Det	N
i	saw	the	man	with	the	telescope

Kemudian untuk tahap kedua, algoritma akan mengisi baris kedua dengan mengisi C[i, i+1] dengan apakah A1 ada di C[i, i+j] and A2 ada di C[i+j+1, i+L] sehingga menjadi seperti ini.

		NP			NP	
NP	V	Det	N	P	Det	N
i	saw	the	man	with	the	telescope

Kemudian untuk tahap selanjutnya akan dilanjutkan dengan cara yang sama sehingga jika sampai ke baris terakhit maka tabel akan berubah menjadi berikut.

S						
	VP					
		NP				
S						
	VP					
		NP			NP	
NP	V	Det	N	P	Det	N
i	saw	the	man	with	the	telescope

Tabel terakhir menunjukan hasil penguraian Jika sel tabel C[0, n] terisi S atau Simbol non-terminal maka String dapat diterima dalam Tata Bahasa diatas. Jika tidak atau sel tabel C[0, n] kosong maka string tidak diterima. Sehingga komputasi program dinamis ini menunjukan goal state yaitu C[0,n].

Algoritma ini memiliki kompleksitas Algoritma dengan kasus terburuk yaitu adalah $O(n^3 |G|)$ yaitu n adalah jumlah string yang ada dan G adalah ukuran dari CNF Grammar G. Ini membuat algoritma CYK merupakan Algoritma yang paling efisien dalam mengurai Context Free Languange bahkan dalam bahasa natural.

Namun Algoritma ini memiliki Kekurangan yaitu Algoritma ini harus mengkonversi terlebih dahulu ke bentuk normal chomsky sehingga jika transformasi bentuk normal chomsky itu bisa membuat jumlah grammar menjadi besar sehingga memperlambat komputasi. Kekurangan lain adalah bentuk program dinamis yang mundur bisa mengakibatkan constituen yang ilegal dan tidak bisa dipakai di hasil string berikutnya.

B. Algoritma Earley

Algoritma Earley adalah Algoritma Parsing Context Free Grammar dan menggunakan Program dinamis maju. Algoritma ini ditemukan oleh Jay Earley.

Algoritma ini membutuhkan Context Free Grammar tanpa harus diubah ke bentuk normal Chomsky. Dalam masukannya membutuhkan Grammar dan String. Algoritma ini menggunakan notasi dot Earley dimana diberikan produksi X → αβ dimana X → α ●β notasi tersebut merepresentasikan string α sudah di urai dan β selanjutnya.

Algoritma ini memakai algoritma *left to right* di tabel untuk mengisi tabel dengan n+1 status dimana n adalah jumlah kata yang diproses. Simbol ● berfungsi untuk menyimpan kata yang sudah diproses. Dalam setiap posisi kata, tabel berisi himpunan kata yang sudah diproses dengan pohon penguraian parsial dan akan di proses.

Tiap status berisi tuple yang berisi produksi, posisi sekarang dan posisi i dalam input untuk matching. Status dari input posisi k disebut s(k). Parser dimulai dari 0 kemudian mulai maju dan memulai tiga operasi yaitu *predictor, scanner, dan completer*..

Predictor berfungsi untuk merepresentasikan status yang bisa terjadi, dan memasukannya ke chart, jika tidak sesuai maka tidak dimasukan. Scanner berfungsi untuk melihat status bagian string yang sudah diprediksi dengan melihat kata sekarang di masukan. Sedangkan Completer adalah untuk menjelajah constituen sehingga status harus ditemukan agar bisa berlanjut.

Algoritma ini sebenernya memprediksi status yang akan dihadapi dengan program dinamis maju kemudian membaca kata selanjutnya untuk diproses.

Algoritma ini menerima string jika (X → y ● 0) berakhir di S(n) dimana (X → y) adalah aturan tahap pertama dan n adalah panjang string, selain itu masukan ditolak. Sehingga Goal state yang dicapai program dinamis itu adalah S(n).

```
function init (S : String)
 l : array[length(s)]
 for each k = 0 to length(s)
       l[k] = {}

function earley (A:Grammar, S:String): table
 init(S)
 add((y → ●l, 0), l[0]);
 for each k = 0 to length(s)
  for each state in l[k]
   if not Finished(state)
    if NEXT(state) then
     predictor(state, k, A)
    else
     scanner(state, k, S)
    else
     complete(state, k)

procedure predictor((A → α•Bβ, j), int: k, A:
grammar)
 for each (B → γ) in grammar-rules(B,
grammar)
  add(((B → •γ, k), l[k]))

procedure scanner((A → α•Bβ, j), int : k, S :
string)
 if a in parts-of-speech(S[k])
  add(((A → αa•β, j), l[k+1]))

procedure completer((B → γ•, x), k)
 for each ((A → αa•β, j),  in l[x] do
  add((A → αB•β, j), S[k])
```

Berikut adalah pseudocode Algoritma Earley.

Sebagai contoh diberikan CFG dengan CNF seperti dibawah, CFG ini merepresentasikan tata bahasa bahasa natural yaitu bahasa inggris dimana String awal direpresentasikan dengan S, NP adalah Subyek, N adalah Kata Benda, VP adalah objek Predikat, dan V adalah Verb, dan Det adalah determinan.

S -> NP VP

NP -> DET N
NP -> NP PP
PP -> P NP
VP -> V NP
VP -> VP PP
DET -> the
NP -> I
N -> man
N -> telescope
P -> with
V -> saw
N -> cat

Dan string yang akan dianalisis adalah “i saw the man”.

Maka jika diurai maka dibuatlah tabel seed pertama yang berisi kata dengan Tabel[0] yang berisi state dan termasuk kedalam program dinamis maju tahap pertama yaitu adalah

y → ● S	[0,0]	Dummy start state
S → ● NP VP	[0,0]	Predictor
NP → ● Det N	[0,0]	Predictor
NP → ● NP PP	[0,0]	Predictor
PP → ● P NP	[0,0]	Predictor
VP → ● V	[0,0]	Predictor
VP → ● V NP	[0,0]	Predictor

Kemudian Tabel[1] yang juga tahap kedua dari porgram dinamis maju dilanjutkan menjadi seperti ini :

NP → i ●	[0,1]	Scanner
S → NP ● VP	[0,1]	Completer
NP → NP ● VP	[0,1]	Completer
VP → V ●	[0,1]	Completer
VP → V ● NP	[0,1]	Completer
NP → ● DET N	[1,1]	Predictor

Kemudian Tabel[2] yang juga tahap ketiga dari porgram dinamis maju dilanjutkan menjadi seperti ini :

V → saw ●	[1,2]	Scanner
VP → V● NP	[1,2]	Completer
S → VP● NP	[1,2]	Completer
VP → VP● NP	[1,2]	Completer
NP → Det● N	[2,2]	Predictor
NP → P ●NP	[2,2]	Predictor

Kemudian Tabel [3] yang mengurai kata “The” sehingga tabel menjadi :

Det → The●	[2,3]	Scanner
NP → Det● N	[2,3]	Completer

Kemudian Tabel [4] yang mengurai kata “The” sehingga tabel menjadi :

N → man●	[3,4]	Scanner
NP → DET● N	[3,4]	Completer
NP → NP● PP	[2,4]	Completer
VP → V● NP	[1,4]	Completer
S → NP● VP	[0,4]	Completer

Dikarenakan S → NP VP adalah [0,4] maka string dapat diterima dan merupakan bagian dari grammar.

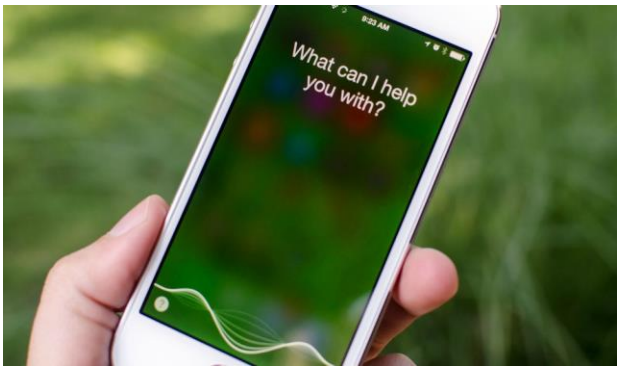
Algoritma ini memiliki kompleksitas waktu yaitu $O(n^3)$, n adalah jumlah huruf dalam string. Hal ini diakibatkan Algoritma ini memproses setiap iterasi adalah $O(n^2)$ dikali oleh elemen Completer yang memproses sebanyak O(n).

Algoritma ini mengoreksi apa yang Algoritma CYK masih kekurangan yaitu tidak harus mengonversi CYK. Walaupun begitu. Algoritma ini memang sedikit lebih rumit karena juga menginput aturan produksi grammar disetiap kata. Namun, algoritma ini lebih cepat dibandingkan Algoritma CYK.

C. Kesimpulan dan Saran

Kedua Algoritma dapat memang menguraikan string untuk diproses dengan Grammar yang ada dan mengecek kebenaran program. Tapi tentu dapat disimpulkan bahwa Algoritma Earley memang lebih mangkus dibandingkan Algoritma CYK walaupun sedikit rumit, dan juga Earley lebih bagus dikarenakan menggunakan Program Dinamis Maju dimana Informasi lebih baik, dan tidak perlu mengubahnya ke bentuk normal chomsky terlebih dahulu.

Kegunaan yang dapat digunakan dari kedua algoritma dengan penguraian bahasa dengan grammar tersebut adalah untuk pemrosesan bahasa natural lanjut sebagai learning agent bagi Intelegensi Buatan sehingga dapat menerima masukan dari manusia lebih akurat dan natural sesuai hukum alam dan tanpa bahasa mesin. Kegunaan lain adalah Algoritma ini dapat membuat query untuk search engine menjadi lebih akurat. Aplikasi yang sudah jelas ada adalah mesin pencari google, Asisten Suara Siri, dan Google Translate.



Gambar 3. Kegunaan Pemrosesan Bahasa[5]

UCAPAN TERIMA KASIH

Puji Syukur kita panjatkan kehadirat Allah SWT karenaNya Makalah ini dapat diselesaikan dengan baik. Terima Kasih saya Ucapkan kepada Pak Rinaldi Munir, Ibu Masayu Leylia Khodra, Ibu Nur Ulfa Mauldevi yang telah mengajar Strategi Algoritma dengan sabar, dan kepada teman teman saya di Teknik Informatika yang sudah memberikan ide terhadap makalah ini. Terima Kasih juga kepada Orang Tua saya yang sudah mendukung saya untuk berkuliah di ITB. Semoga saya mendapatkan nilai yang memuaskan untuk mata-kuliah ini dan ilmu yang dipakai dapat berguna untuk masa depan.

REFERENSI

[1] https://www.cs.princeton.edu/courses/archive/spr05/cos423/lectures/06dynamic-programming.pdf, diakses pada 17 Mei 2017.

[2] Munir, Rinaldi. Diktat Kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika. Institut Teknologi Bandung. 2009.

[3] Hopcroft, John E, Ullman, Jeffrey. Intro to Automata Theory, Languages And Computation.Addison Wesley. 2007

[4] https://www.researchgate.net/profile/Bartosz_Musznicki/publication/265412778/figure/fig1/AS:392039154896896@1470480826291/Figure-3-A-general-structure-of-a-multi-stage-graph.png diakses pada 16 Mei 2017

[5] http://cdn.bgr.com/2015/12/siri-iphone.jpg?quality=98&strip=all diakses pada 16 Mei 2017

[6] https://thenextweb.com/wp-content/blogs.dir/1/files/2011/06/Kannada-to-English-Translation.jpg diakses pada 16 Mei 2017

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, 18 Mei 2017

Muhammad Rizki Duwinanto (13515006)