Penerapan Struktur Data Pohon Dalam *Context-Free Grammar*

Ditra Rizqa Amadia - 13521019[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521019@std.stei.itb.ac.id

***Abstrak*—Mesin dapat berfungsi sesuai instruksi yang diberikan oleh manusia melalui program. Program yang dibuat oleh manusia dalam bentuk bahasa pemrograman belum dapat dibaca langsung oleh mesin. Program *Compiler* dibuat untuk mengubah program dalam bahasa pemrograman menjadi bahasa mesin yang dapat dibaca lalu dieksekusi oleh mesin. Salah satu tahapan pada proses penerjemahan oleh *compiler* yaitu *parsing*. Tahapan *parsing* bertujuan untuk memeriksa sintaksis program untuk mencegah terjadinya gangguan pada program. Aturan tata bahasa *(grammar)* yang digunakan yaitu *Context-Free Grammar* (CFG). Untuk merumuskan CFG tersebut, penulis menggunakan struktur data pohon.**

***Keywords*—Struktur Data Pohon, *Context-Free Grammar* (CFG), *Parser, Compiler*.**

I. PENDAHULUAN

Alat komputasi, atau komputer merupakan hal yang wajib dimiliki pada zaman modern ini. Komputer adalah salah satu bagian dari perkembangan teknologi untuk membantu manusia dalam menjalani kehidupan sehari-hari. Beberapa contoh teknologi yang memanfaatkan komputer yaitu gawai, komputer pribadi (PC), televisi, dan *microwave.* Dengan adanya komputer, penyebaran informasi, aktivitas bisnis, aktivitas rumah tangga, dan kegiatan manusia lainnya menjadi lebih mudah dan cepat.

Komputer memerlukan sebuah program yang berfungsi untuk mengatur keberjalanan mesin tersebut. Program adalah suatu set instruksi yang terstruktur dan beralur. Program atau yang biasa disebut aplikasi memberikan instruksi kepada mesin untuk melakukan suatu pekerjaan. Program terdiri dari untaian-untaian kode berurut yang biasa disebut algoritma. Algoritma ini akan dieksekusi oleh mesin dan akhirnya menentukan perilaku perangkat tersebut.

Program tentunya dibuat oleh manusia, dan orang yang membuat suatu program disebut *programmer*. Namun bahasa manusia dengan bahasa mesin tentunya berbeda. Oleh karena itu, manusia membuat program dengan bantuan bahasa pemrograman. Bahasa pemrograman inilah yang disusun dengan algoritma dan logika manusia lalu diterjemahkan menjadi bahasa mesin. Terdapat banyak bahasa pemrograman yang memiliki fungsi khususnya masing-masing. Namun tujuan utama dari bahasa pemrograman yaitu untuk memudahkan manusia membuat suatu program.

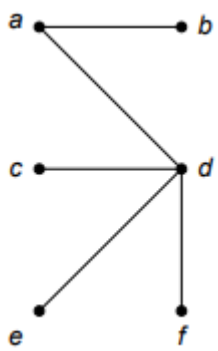
Terdapat suatu program untuk menerjemahkan bahasa pemrograman manusia menjadi bahasa yang dapat dimengerti dan dieksekusi oleh mesin. Program tersebut diberi nama *compiler*. Setiap bahasa pemrograman memiliki *compiler*-nya masing-masing. Salah satu tahapan *compiler* dalam menerjemahkan program yaitu memeriksa sintaksis atau tata bahasa pada bahasa pemrograman. Tahapan ini disebut *parsing* dan program yang melakukan tahapan tersebut disebut *parser*. Kata *parsing* berasal dari bahasa Inggris yang artinya mengurai.

Tata bahasa pemrograman adalah kalimat atau urutan-urutan kalimat pada kode yang terdaftar (sintaksis) sehingga dapat dimengerti dan dieksekusi oleh mesin. Tata bahasa pemrograman secara kesuluruhan disebut *grammar*. Salah satu bentuk *grammar* yang dipakai untuk bahasa pemrograman yaitu *Context-Free Grammer* (CFG). Apabila tata bahasa tidak sesuai dengan aturan tata bahasa yang ada pada bahasa pemrograman, program akan mengalami gangguan dan pada akhirnya tidak dapat dieksekusi oleh mesin. Oleh karena itu, *parser* berperan penting dalam memeriksa sintaksis dan tata bahasa suatu program.

II. LANDASAN TEORI

A. Pohon

Pohon adalah graf tak berarah terhubung yang tidak mengandung sirkuit [1].

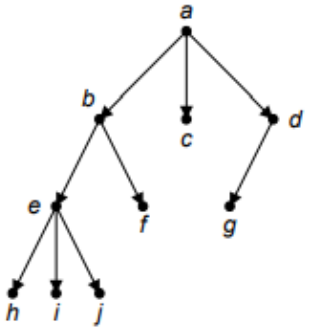


Gambar 2.1 Pohon, sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf

Misalkan G = (V, E) adalah graf tak berarah sederhana dan jumlah simpulnya n. Maka, semua pernyataan di bawah ini adalah ekivalen:

- G adalah pohon
- Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal
- G terhubung dan memiliki m = n – 1 buah sisi
- G tidak mengandung sirkuit dan memiliki m = n – 1 buah sisi
- G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit
- G terhubung dan semua sisinya adalah jembatan

Pohon berakar (*rooted tree*) adalah pohon yang satu buah simpulnya diperlukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah [2].



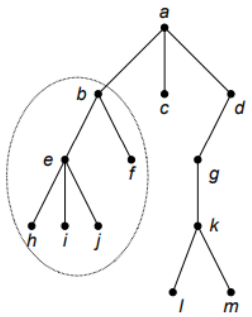
Gambar 2.2 Pohon berakar, sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf

Anak (*child*) adalah anak simpul dan orangtua (*parent*) adalah orangtua dari simpul. Pada 2.2 simpul b adalah orangtua dari simpul e dan f. Simpul e dan f adalah anak dari simpul b, namun simpul b bukan orangtua dari simpul g.

Lintasan (*path*) adalah rute dari suatu simpul ke simpul lain melewati satu atau beberapa sisi. Pada 2.2 lintasan a ke j adalah a, b, e, j. Panjang lintasan dari a ke j yaitu 3.

Saudara kandung (*sibling*) adalah relasi simpul dengan simpul orangtua yang sama.Pada gambar pohon berakar simpul f adalah saudara kandung simpul e, tetapi simpul g bukan saudara kandung simpul e karena simpul orangtua mereka berbeda.

Upapohon (*subtree*) adalah bagian pohon yang terdapat pada pohon utama. Pohon dapat dibentuk dari beberapa upapohon secara rekursif.



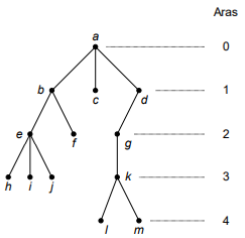
Gambar 2.3 Upapohon sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf

Derajat (*degree*) adalah banyaknya upapohon (atau jumlah anak) pada simpul tersebut. Pada 2.2 derajat a adalah 3, derajat b adalah 2, derajat d adalah 1, dan derajat c adalah 0. Derajat yang dimaksud pada pohon adalah derajat keluar. Derajat maksimum dari semua simpul merupakan derajat pohon itu sendiri. Pada gambar pohon berakar derajat maksimum pohon yaitu berderaja 3.

Daun (*leaf*) adalah simpul yang derajatnya 0 (atau tidak mempunyai anak). Pada 2.2 simpul h, i, j, f, c, dan g adalah daun.

Simpul dalam (*internal nodes*) adalah simpul yang mempunyai anak. Pada 2.2 simpul a, b, d, dan e adalah simpul dalam.

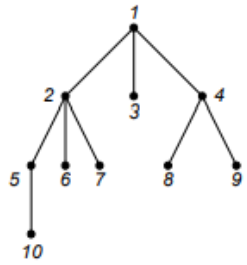
Aras (*level*) adalah tingkat suatu simpul pada pohon relatif terhadap akarnya.



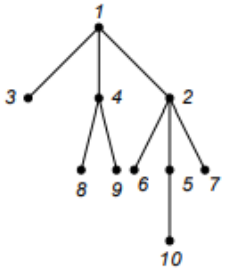
Gambar 2.4 Aras pohon, sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf

Tinggi (*height*) atau kedalaman (*depth*) adalah aras maksimum dari suatu pohon. Pada 2.4 kedalamannya adalah 4.

Pohon terurut (*ordered tree*) adalah pohon berakar yang urutan anak-anaknya penting.



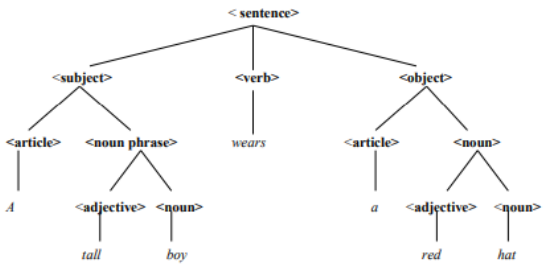
Gambar 2.5 Pohon terurut a, sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf



Gambar 2.6 Pohon terurut b, sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf

Pohon pada 2.5 dan pohon pada 2.6 adalah dua pohon terurut berbeda.

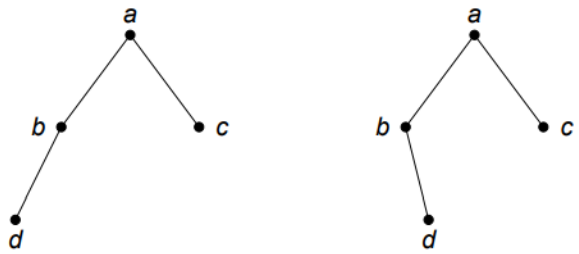
Pohon n-ary adalah pohon berakar yang setiap simpul cabangnya mempunyai paling banyak n buah anak.



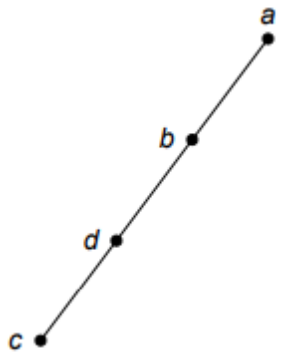
Gambar 2.7 Pohon n-ary, sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf

Pohon n-ary dikatakan teratur atau penuh jika setiap simpul cabangnya mempunyai tepat n anak.

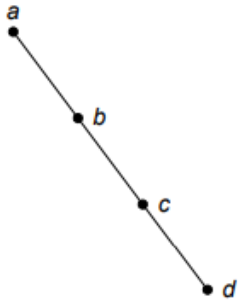
Pohon biner (*binary tree*) adalah pohon n-ary dengan n = 2. Pohon ini adalah pohon yang paling penting karena banyak aplikasinya. Setiap simpul pada pohon biner mempunyai paling banyak 2 buah anak. Simpul anak kiri dan anak kanan dibedakan. Pohon biner adalah pohon terurut karena terdapat perbedaan anak.



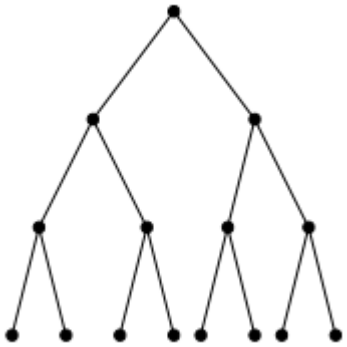
Gambar 2.7 Dua pohon biner berbeda, sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf



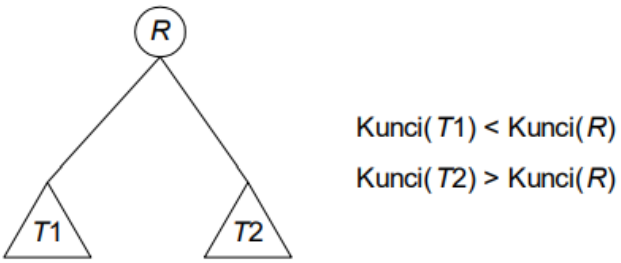
Gambar 2.8 Pohon condong kiri, sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf



Gambar 2.9 Pohon condong kanan, sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf

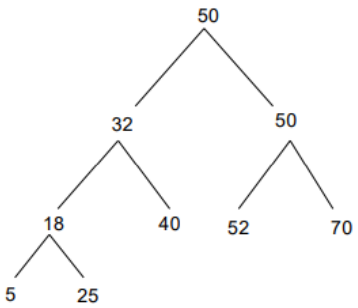


Gambar 2.10 Pohon biner penuh, sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf



Gambar 2.11 Pohon pencarian biner, sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf

Pohon pencarian biner adalah pohon biner dengan sifat nilai simpul anak kiri harus lebih kecil dari nilai simpul orangtuanya dan nilai simpul anak kanan harus lebih besar dari simpul orangtuanya. Terdapat sebuah data 50, 32, 18, 40, 60, 52, 5, 25 , dan 70. Maka representasi pohonnya adalah 2.12.



Gamabr 2.12 Pohon pencarian biner, sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf

Seperti pada namanya, pohon pencarian biner memudahkan program untuk mencari suatu nilai pada suatu simpul.

B. Context-Free Grammar

Dalam teori bahasa formal, *Context-Free Grammar* (CFG) adalah suatu notasi formal untuk mengekspresikan aturan tata bahasa (sintaksis) yang berulang sehingga membentuk suatu *grammar* [2]. Terdapat 4 komponen penting dalam *grammar* suatu bahasa:

- T = Simbol terminal
- V = Simbol non-terminal
- S = Simbol *Start*
- P = Simbol aturan produksi

Simbol terminal adalah simbol yang merepresentasikan kata pada suatu bahasa. Simbol non-terminal adalah simbol yang merepresentasikan suatu sintaks atau susunan kata non-terminal. Simbol *start* adalah simbol *grammar*-nya itu sendiri. Simbol aturan produksi adalah simbol yang merepresentasikan sifat pengulangan dari tata bahasa. Bentuk CFG dapat direpresentasikan menjadi

$$G = (V, T, P, S). \quad (1)$$

Aturan produksi dalam CFG dapat ditulis ke dalam bentuk

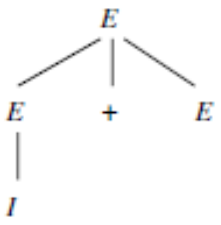
$$E \rightarrow I \quad (2)$$

di mana E adalah pemproduksi dan I adalah hasil produksi. Pemproduksi merupakan sebuah variabel, dapat berupa V, S, ataupun P. Sedangkan hasil produksi dapat berupa T, V, S, maupun P. Beberapa CFG yang benar termasuk:

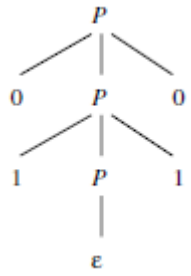
$$I \rightarrow a \quad (3)$$
$$I \rightarrow Ia \quad (4)$$
$$E \rightarrow I \quad (5)$$

C. Pohon Penurunan

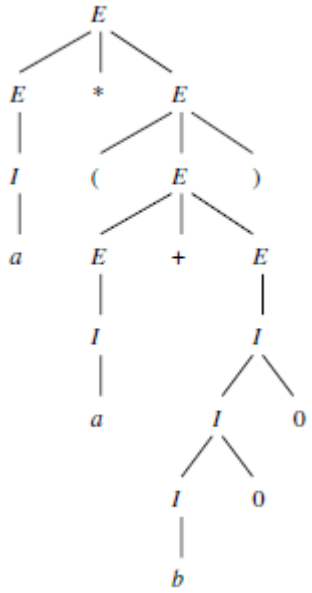
Pohon Penurunan (*parse tree*) adalah representasi pohon untuk turunan grammar yang terbukti berguna [3]. Pohon ini memvisualisasikan secara jelas bagaimana setiap simbol yang terdiri dari beberapa simbol terminal disatukan menjadi bagian-bagian tersendiri untuk membentuk suatu tata bahasa.



Gambar 2.13 Pohon penurunan dari I + E[3]



Gambar 2.14 Pohon penurunan dari 0110[3]



Gambar 2.15 Pohon penurunan dari a*(a+b00)[3]

Pohon penurunan adalah representasi pohon untuk turunan grammar yang terbukti berguna [3]. Pohon ini memvisualisasikan secara jelas bagaimana setiap simbol yang terdiri dari beberapa simbol terminal disatukan menjadi bagian-bagian tersendiri untuk membentuk suatu tata bahasa. Pohon penurunan berguna untuk menggambarkan simbol-simbol variabel menjadi terminal. Semua simbol variabel diturunkan menjadi terminal secara rekursif sampai tidak ada lagi simbol yang berupa variabel.

III. METODOLOGI

A. Membuat CFG

Terdapat beberapa tahapan dalam membuat CFG yaitu:

Pertama yaitu identifikasi terminal yang akan digunakan. Terminal ini adalah basis dari sintaks, umumnya hanya terdiri dari 1 kata atau simbol. Terminal dapat berupa token per kata dari suatu program. Beberapa contoh simbol terminal pada bahasa javaScript yaitu import, let, var, if, dan else. Penulisan terminal menggunakan huruf non-kapital

Kedua, identifikasi variabel yang akan digunakan. Variabel adalah sintaksis yang dikategorikan. Variabel ini umumnya berbentuk frasa. Variabel adalah susunan dari beberapa variabel lainnya atau terminal. Contoh simbol variabel pada bahasa

javaScript yaitu IF_STATEMENT dan IMPORT_STATEMENT. Penulisan variabel menggunakan huruf kapital

Ketiga yaitu definisikan simbol *start*. Simbol *start* adalah variabel yang berperan sebagai akar dalam pohon penurunan suatu tata bahasa. Simbol start adalah sintaks dari untaian-untaian variabel sehingga membentuk suatu program utuh. Variabel yang akan penulis gunakan untuk simbol *start* adalah huruf S.

Keempat, tulis aturan produksi. Produksi adalah sintaksis atau untaian-kata-kata dapat berupa variabel maupun terminal. Cara penulisan produksi yaitu dengan menuliskan suatu variabel di ruas kiri, diikuti tanda panah ke kanan, dan di ruas kanan dapat berupa untaian variabel atau terminal. Contoh suatu produksi dalam bahasa javaScript yaitu produksi *if statement*. Produksi ini direpresentasikan dengan

IF_STATEMENT → if lb LOGIC_STATEMENT rb lcb CODE_BLOCK rcb.

Bentuk asli dari produksi tersebut dapat dilihat pada gambar 3.1.

```
1   if (k != 3) {
2       console.log(do_something)
3   }
```

Gambar 3.1 Sintaks *if statement* dalam javaScript

Pada produksi, token if direpresentasikan oleh terminal if, kurung buka dan kurung tutup direpresentasikan oleh terminal lb dan rb, dan kurung kurawal buka dan kurung kurawal tutup direpresentasikan dengan lcb dan rcb. LOGIC_STATEMENT adalah variabel yang memiliki sintaksnya tersendiri, pada 3.1 k != 3 adalah LOGIC_STATEMENT. Begitu pula dengan CODE BLOCK, yaitu variabel dengan sintaksnya tersendiri. Pada 3.1 console.log(do_something) adalah CODE_BLOCK. Variabel LOGIC_STATEMENT dapat diurai kembali menjadi terminalnya yaitu

LOGIC_STATEMENT → var neq number .

Begitu pula dengan CODE_BLOCK yang dapat diurai menjadi terminalnya menjadi

CODE_BLOCK → var dot var lb var rb.

Dari penurunan tersebut, produksi IF_STATEMENT dapat ditulis menjadi

IF_STATEMENT → if lb var neq number rb lcb var dot var lb var rb rcb.

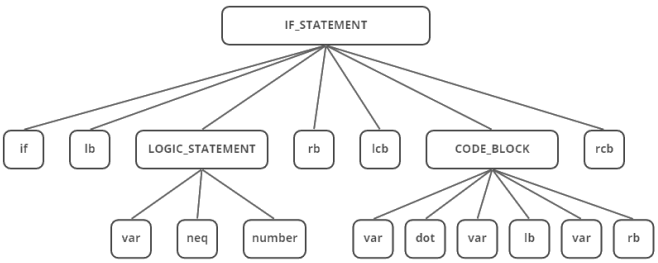
IF_STATEMENT ini nantinya adalah bagian dari sintaks S yang merupakan sintaks dari keseluruhan program.

Membuat suatu CFG membutuhkan pemahaman mendalam mengenai bahasa yang bersangkutan dan pengalaman dalam tata bahasa dan sintaks bahasa pemrograman tersebut.

B. Menvisualisasikan CFG dalam bentuk pohon penurunan

Pohon penurunan dapat dimanfaatkan untuk memvisualisasikan CFG yang telah dibuat. Pohon pennurunan ini akan menunjukan relasi hierarki antara masing-masing kata dan frasa dalam suatu kalimat. Selain itu, pohon penurunan dapat menunjukkan peran dan fungsi dari masing-masing kalimat. Pohon penurunan akan menurunkan semua variabel hingga ke terminalnya.

Dalam kasus IF_STATEMENT, pohon penurunan dapat direpresentasikan menjadi 3.2.



Gambar 3.2 Pohon penurunan *if statement* dalam javaScript

IV. IMPLEMENTASI

CFG dan pohon penurunan dapat diimplementasikan sebagai *parser* dalam program *compiler*.

Kasus pertama yaitu penulis membuat sebuah program 3.3.

```
1   let n = 2;
2   if (n > 0) {
3     sayHello();
4   }
5
```

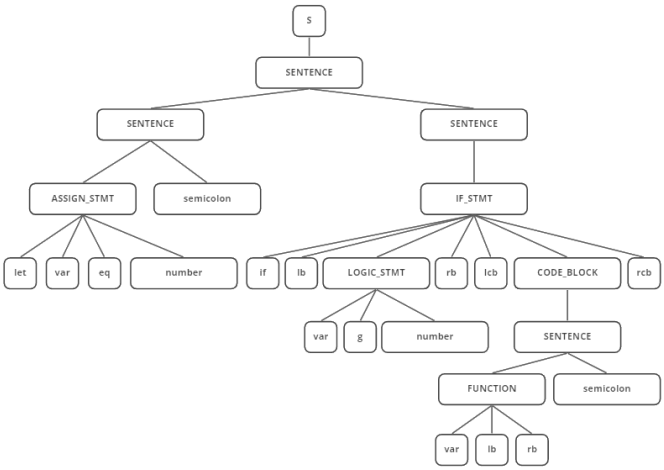
Gambar 3.3 Program kasus 1

CFG yang dibuat untuk melakukan *parsing* pada program tersebut yaitu 3.4.

```
1   S -> SENTENCE
2   SENTENCE -> ASSIGN_STMT semicolon
3   SENTENCE -> IF_STMT
4   SENTENCE -> FUNCTION semicolon
5   SENTENCE -> SENTENCE SENTENCE
6   ASSIGN_STMT -> let var eq number
7   LOGIC_STMT -> var g number
8   IF_STMT -> if lb LOGIC_STMT rb lcb CODE_BLOCK rcb
9   FUNCTION -> var lb rb
10  CODE_BLOCK -> SENTENCE
```

Gambar 3.4 CFG

CFG dari program 3.3 dapat direpresentasikan oleh pohon penelusuran pada gambar 3.5



Gambar 3.5 Pohon penurunan kasus 1

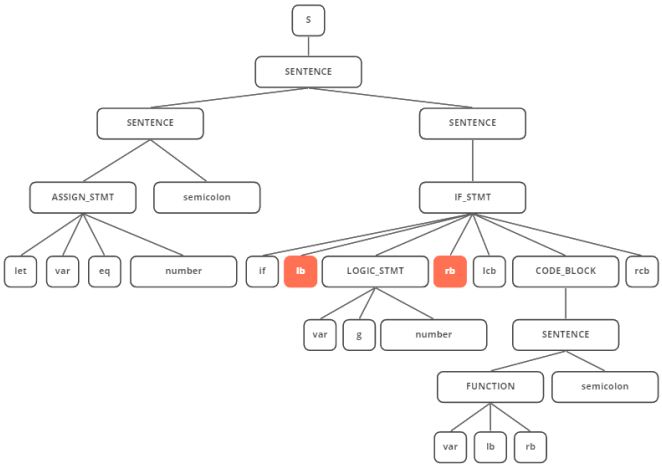
Tata bahasa program sesuai dengan CFG yang ada sehingga tidak menghasilkan gangguan. Dapat diperhatikan bahwa semua daun pada pohonnya berupa terminal dan akarnya yaitu variabel *start.*

Pada kasus kedua, penulis membuat program 3.6.

```
1  let n = 2;
2  if n > 0 {
3      sayHello();
4  }
```

Gambar 3.6 Program kasus 2

Dengan menggunakan CFG 3.4, program akan menghasilkan gangguan berupa *syntax error* karena sintaksis tidak sesuai dengan aturan yang ada. Visualisasinya dapat dilihat pada 3.7.



Gambar 3.7 Pohon penurunan kasus 2

Program mengalami gangguan pada IF_STMT. Kotak yang dimerahkan tidak terdapat pada program sehingga variabel IF_STMT tidak dapat terbentuk. Akhirnya variabel S tidak dapat terbentuk juga dan program mengeluarkan *syntax error*.

V. KESIMPULAN

Berdasarkan hasil implementasi di atas, teori pohon dapat diaplikasikan ke dalam proses *compiling* khusunya dalam tahap *parsing*. Program *parser* menggunakan pohon penurunan untuk memvalidasi tata bahasa suatu program yang bersangkutan. Adanya aturan dalam bahasa pemrograman bertujuan agar program dapat dimengerti dan dieksekusi oleh mesin.

VI. PENUTUP

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa yang telah mengaruniakan kelancaran dan kemudahan untuk penulis dalam penyusunan makalah ini. Penulis ucapkan terima kasih untuk kedua orangtua penulis atas dukungannya selama proses pendidikan pennulis. Tidak lupa juga penulis ucapkan terima kasih kepada Dr. Ir. Rinaldi Munir, M.T. sebagai dosen pengampu mata kuliah matematika diskrit (IF2120) untuk kelas K3 yang telah menjadi pembimbing penulis dalam menjalani mata kuliah ini hingga memberi kesempatan kepada penulis untuk menulis makalah ini.

REFERENSI

[1] Munir, Rinaldi (2003). Pohon (Bag. 1) Bahan Kuliah IF2120 Matematika Diskrit, URL: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf>
Diakses: 11 Desember 2022

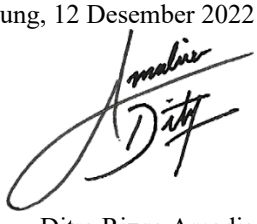
[2] Munir, Rinaldi (2003). Pohon (Bag. 2) Bahan Kuliah IF2120 Matematika Diskrit, URL: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon-2021-Bag2.pdf>
Diakses: 11 Desember 2022

[3] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation,* 3rd ed, Delhi: Pearson Education, 2022, pp. 171-275

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, 12 Desember 2022



Ditra Rizqa Amadia
13521019