SKRIPSI

PARSING KODE SUMBER ECMASCRIPT2015 KE DALAM *ABSTRACT SYNTAX TREE* DAN *CONTROL FLOW GRAPH*

Jason Yehezkiel

NPM: 2016730087

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2020

UNDERGRADUATE THESIS

PARSING ECMASCRIPT2015 SOURCE CODE INTO ABSTRACT SYNTAX TREE AND CONTROL FLOW GRAPH

Jason Yehezkiel

NPM: 2016730087

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2020

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

PARSING* KODE SUMBER ECMASCRIPT2015 KE DALAM *ABSTRACT SYNTAX TREE* DAN *CONTROL FLOW GRAPH

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 18 Juni 2020

Jason Yehezkiel
NPM: 2016730087

LEMBAR PENGESAHAN

PARSING KODE SUMBER ECMASCRIPT2015 KE DALAM *ABSTRACT SYNTAX TREE* DAN *CONTROL FLOW GRAPH*

Jason Yehezkiel

NPM: 2016730087

Bandung, 18 Juni 2020

Menyetujui,

Pembimbing

Elisati Hulu, M.T.

Ketua Tim Penguji

Anggota Tim Penguji

Husnul Hakim, M.T.

Chandra Wijaya, M.T.

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

PARSING KODE SUMBER ECMASCRIPT2015 KE DALAM *ABSTRACT SYNTAX TREE* DAN *CONTROL FLOW GRAPH*

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 18 Juni 2020

Jason Yehezkiel
NPM: 2016730087

ABSTRAK

Perangkat lunak dibangun dengan menulis kode sumber dalam satu bahasa pemrograman dan melakukan kompilasi kode sumber tersebut menjadi program yang dapat dieksekusi. Banyak bahasa pemrograman yang bermunculan untuk menjawab kebutuhan spesifik dalam pembuatan program. *Abstract Syntax Tree* dan *Control Flow Graph* dapat membantu memahami bahasa pemrograman yang ada.

Abstract Syntax Tree, atau AST, merepresentasikan struktur sintaks dari sebuah kode program. Di dalam AST, tata cara penulisan kode program dapat diabaikan dan mengurangi *language barrier* antarbahasa pemrograman. *Control Flow Graph*, atau CFG, merepresentasikan alur eksekusi kode program. Kedua struktur data tersebut dapat dibangun dengan mengimplementasikan dua proses dari sebuah *compiler*, yaitu analisis leksikal dan analisis sintaksis.

Dalam penelitian ini, dibangun sebuah perangkat lunak yang mampu membangun AST dan CFG dari kode sumber yang ditulis dalam bahasa pemrograman *Javascript* (ECMAScript2015). Perangkat lunak terdiri dari tiga komponen, yaitu komponen antarmuka, komponen *tokenizer* untuk melakukan analisis leksikal, dan komponen *parser* untuk melakukan analisis sintaksis; ketiganya dibangun dengan bahasa pemrograman *Javascript* dan *Typescript* menggunakan Node.js dan Vue.js. Hasil AST yang dibangun divisualisasikan dalam dua bentuk yaitu dalam notasi JSON dan dalam gambar visualisasi struktur data pohon, dan hasil CFG yang dibangun akan divisualisasikan dalam gambar struktur data graf berarah.

Pengujian dilakukan untuk memastikan perangkat lunak membangun AST dan CFG dari kode program *Javascript*. *Unit test* dilakukan menggunakan 69 kasus pengujian untuk memastikan bahwa *parser* membangun AST yang benar untuk setiap sintaks, 10 kasus pengujian untuk memastikan CFG yang dibangun adalah benar, dan 3 kasus pengujian untuk memastikan visualisasi AST sesuai dengan struktur data AST yang dihasilkan. Berdasarkan hasil pengujian, *parser* mampu membangun AST dan menampilkan visualisasi yang benar, tetapi *parser* belum bisa menghasilkan CFG yang sempurna dalam beberapa kasus pengujian. Dengan pengujian yang mengukur performa, didapatkan bahwa faktor banyaknya baris kode program dan kompleksitas kode program mempengaruhi waktu eksekusi program. Semakin banyak baris kode program dan semakin kompleks kode program, maka semakin lama waktu yang dibutuhkan untuk melakukan *parsing*.

Kata-kata kunci: *Parsing*, Kode Sumber, *Abstract Syntax Tree*, *Control Flow Graph*, Javascript, ECMAScipt2015

ABSTRACT

Softwares are developed by writing a source code in a programming language and compiling the source code into an executeable program. A lot of specific programming languages were then built to answer specific needs in building softwares. Abstract Syntax Tree and Control Flow Graph can help us understand these programming languages.

Abstract Syntax Tree, or AST for short, represents the syntax structure of a source code. With AST, specific grammar of a programming language can be ignored, and thus, reducing the language barrier between programming languages. Control Flow Graph, or CFG, represents the flow of program execution. These data structures can be built by implementing two processes of a compiler, which is lexical analysis and syntactic analysis.

In this research, we developed a software that builds AST and CFG from a source code written in Javascript (ECMAScript2015). The software consists of three components, which are the user-interface, the tokenizer to do lexical analysis, and the parser to do syntactic analysis. All of them are built in Javascript and Typescript, using Node.js and Vue.js framework. The result of AST will be visualized in two forms which are in JSON and with the tree data structure visualization, while the result of CFG will be visualized in the form of a directed graph.

Several tests were done to ensure the developed software can parse a valid AST and CFG from the Javascript source code. Unit testing was done using 69 test cases to ensure that the parser builds the correct AST for each syntax, 10 test cases to ensure that the parser build the correct CFG, and 3 test cases to ensure that the visualization of AST represents the AST object correctly. According to the result of the test, the parser can build the correct AST and show the correct visualization, but the parser could not produce the complete CFG in several test cases. An experiment to measure the performance of the software shows that total lines of code and code complexity affects program execution time. Increasing the number of lines of code and code complexity will increase the time needed to do parsing.

Keywords: Parsing, Source Code, Abstract Syntax Tree, Control Flow Graph, Javascript, ECMAScript

"TUHAN memberkati engkau dan melindungi engkau; TUHAN menyinari engkau dengan wajah-Nya dan memberi engkau kasih karunia; Tuhan menghadapkan wajah-Nya kepadamu dan memberi engkau damai sejahtera" Bilangan 6:24-26

Dipersembahkan kepada Tuhan Yang Maha Esa, keluarga, para dosen, teman-teman, serta diri sendiri.

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yesus Kristus untuk pimpinan, rahmat, dan kasih karunia-Nya, penulis dapat menyelesaikan penelitian dan penyusunan skripsi yang berjudul "*Parsing* Kode Sumber ECMAScript2015 ke dalam *Abstract Syntax Tree* dan *Control Flow Graph*". Skripsi ini dibuat dan diajukan untuk memenuhi salah satu syarat kelulusan Program Studi Teknik Informatika, Universitas Katolik Parahyangan, serta untuk memperoleh gelar Sarjana pada program studi tersebut. Selain itu, skripsi ini ditulis untuk memberikan pengetahuan kepada pembaca mengenai proses *parsing* yang dilakukan terhadap kode sumber, lebih tepatnya kode sumber Javascript. Penulis menyadari bahwa skripsi ini tidak dapat diselesaikan tanpa bantuan orang-orang terdekat. Oleh karena itu, penulis mengungkapkan rasa terima kasih kepada:

1. Bapak Raymond Chandra Putra, S.T., M.T. selaku dosen pembimbing yang telah membimbing dan mendukung penulis selama proses penyusunan skripsi ini.
2. Bapak Husnul Hakim, S.Kom., M.T. dan Bapak Chandra Wijaya, S.T, M.T. selaku penguji yang telah memberikan kritik dan saran yang membangun.
3. Keluarga yang selalu mendukung dan membantu penulis selama proses perkuliahan.
4. Ibu Mariskha Tri Adithia, S.Si., M.Sc., PD.Eng. selaku kepala program studi Teknik Informatika.
5. Ibu Dr.rer.nat. Cecilia Esti Nugraheni, S.T., M.T. selaku dosen wali selama proses perkuliahan.
6. Kakak, sahabat, dan teman hidup tercinta yang menjadi tempat istirahat dan mengisi tenaga dan menemani penulis dalam proses penyusunan skripsi serta memberikan dorongan dan motivasi dalam berbagai bentuk.
7. Manusia Setelah Ibadah untuk dukungan moral, ilmu hidup, hiburan, dan motivasi selama proses perkuliahan, serta memberikan kesempatan untuk menjadi lebih dekat dengan kakak, sahabat, dan teman hidup tercinta.
8. Ferdian Benyamin, Aldrich Suryawan, Steven Permana, Krisna Novianto, dan Yosua Abiezer yang memberikan tempat istirahat, hiburan, serta berbagi *meme*.
9. Tim Korzen untuk kesempatan dan pengalaman kerja, dan pelajaran yang sangat berharga.
10. Yehezkiel Rusli dan Cahyadi Hartanto sebagai teman seperjuangan mengerjakan skripsi sampai pagi.
11. Group kacang (Joshua, Cantika, Kevin) yang memotivasi dan membantu selama proses perkuliahan.
12. Komsel KomKop, Youth Reborn, Panji si Penakluk Ular, dan keluarga Bethel Teens dan Youth yang membantu penulis dalam perjalanan kehidupan.
13. Styadi Senjaya yang memperkenalkan penulis dengan hobi yang baru.

14. J. A. Muran, Gemi, dan Bul, serta belasan ikan lainnya yang nyawanya tidak terselamatkan, karena telah membawa hiburan dan kesegaran ke dalam rumah penulis.

15. Seluruh angkatan 2016 program studi Teknik Informatika sebagai teman-teman seperjuangan dalam menempuh perkuliahan.

16. Staff admin laboratorium komputasi FTIS yang menyediakan tempat mengerjakan tugas dan memberikan hiburan.

17. Teman-teman dari penulis yang namanya tidak bisa disebutkan satu demi satu.

Penulis menyadari bahwa penelitian ini jauh dari sempurna. Maka, penulis memohon maaf jika terdapat kekurangan pada penelitian ini dan menerima semua kritik dan saran untuk menyempurnakan penelitian ini. Semoga penelitian ini dapat bermanfaat bagi semua pembaca dan pihak yang berkepentingan.

Bandung, Juni 2020

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE PROGRAM

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Semua perangkat lunak berawal dari kode sumber yang di-*compile*. Kode sumber adalah kumpulan instruksi untuk mengatur operasi-operasi yang harus dilakukan komputer. Sebelum kode sumber di-*compile* menjadi sebuah perangkat lunak, kode sumber tersebut hanyalah sebuah *file* teks. Walaupun manusia bisa membaca teks kode ini, banyak informasi yang tidak bisa didapatkan secara langsung dengan membaca teks kode sumber tersebut [3].

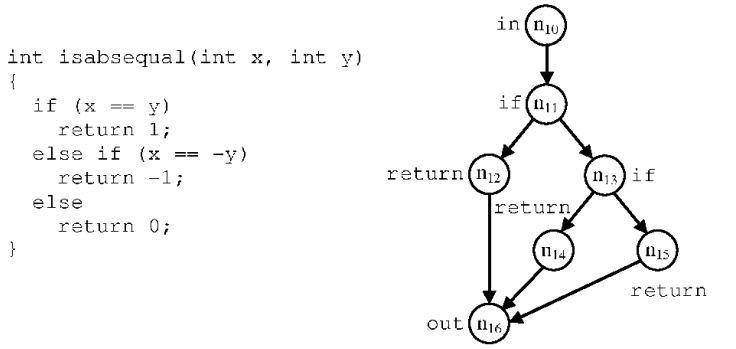
Kode sumber dari suatu program ditulis dalam suatu bahasa pemrograman. Bahasa pemrograman dirancang untuk mempermudah proses pembuatan program dengan cara membuat kode program lebih mirip dengan bahasa manusia dibandingkan bahasa mesin, sehingga komunikasi antara manusia dan komputer dapat terjalin karena bahasa tersebut dapat dimengerti oleh dua belah pihak.

Ada kemiripan antara bahasa pemrograman dan bahasa manusia, dan keduanya mempunyai masalah yang sama, yaitu tingkat kesulitan untuk dikuasai dan fasih dalam semua bahasa. Dibutuhkan penerjemah untuk seseorang bisa mengetahui maksud dari sebuah pernyataan dalam bahasa yang tidak ia ketahui. *Abstract Syntax Tree* dan *Control Flow Graph* bisa menjadi langkah pertama untuk menyelesaikan masalah ini.

Gambar 1.1: Contoh visualisasi hasil AST [1]

Abstract Syntax Tree (AST) adalah sebuah struktur data yang merepresentasikan struktur dari

kode program [2]. Berbeda dengan kode sumber program, AST mengabaikan gramatika, dialek, cara-cara penulisan, dan tanda baca, sehingga pembaca tidak dibuat bingung dengan perbedaan tata bahasa antara satu bahasa pemrograman dan bahasa pemrograman lainnya. Contoh sebuah AST dapat dilihat pada Gambar 1.1.



Gambar 1.2: Contoh visualisasi hasil CFG[4]

Control Flow Graph (CFG) merupakan struktur data graf berarah yang merepresentasikan alur eksekusi kode program [5]. Sesuai dengan definisinya, graf CFG mempunyai simpul yang merepresentasikan suatu kalimat dalam teks kode program dan sisi berarah yang merepresentasikan alur eksekusi kode program. Penelusuran kode atau struktur data CFG dimulai dari *root node* pertama dan dilanjutkan ke node selanjutnya, atau *next node*. Node bisa mempunyai beberapa *next node* seperti pada node percabangan. Contoh dari sebuah CFG dapat dilihat pada Gambar 1.2.

Kode sumber program yang sudah diubah ke dalam bentuk AST atau CFG kemudian bisa dianalisis untuk mendapatkan metrik-metrik yang dapat membantu proses pembangunan perangkat lunak, seperti kompleksitas kode program dan *code coverage* pada pengujian perangkat lunak [5]. *Code coverage* merupakan sebuah tolak ukur yang mengukur sejauh mana sebuah perangkat lunak sudah diuji. Pengembangan AST dan CFG lebih lanjut juga memungkinkan seorang *programmer* untuk mengubah kode sumber saat program dieksekusi melalui AST atau CFG, atau memungkinkan untuk mengeksekusi program dari salah satu node tertentu.

Perangkat lunak, AST, dan CFG yang akan dibangun hanya akan menggunakan satu bahasa pemrograman dan hanya bisa memproses bahasa pemrograman tersebut.

1.2 Rumusan Masalah

Rumusan masalah yang akan dibahas pada skripsi ini adalah:

- Apa bahasa pemrograman yang cocok digunakan untuk membangun *Abstract Syntax Tree* dan *Control Flow Graph*?
- Bagaimana cara mengubah dan memproses kode sumber menjadi *Abstract Syntax Tree* dan *Control Flow Graph*?
- Bagaimana cara mengimplementasikan *Abstract Syntax Tree* dan *Control Flow Graph* dalam sebuah perangkat lunak?
- Bagaimana AST dan CFG yang sudah dibuat dapat divisualisasikan?

[1]Sumber: https://tomassetti.me/antlr-mega-tutorial/, diakses pada 15 November 2019

1.3 Tujuan

Tujuan yang akan dicapai pada skripsi ini adalah:

- Menganalisis dan memilih bahasa pemrograman untuk membangun AST dan CFG dari kode sumber dalam bahasa pemrograman tersebut.
- Mempelajari cara membangun perangkat lunak untuk mengubah kode sumber menjadi AST dan CFG.
- Membangun perangkat lunak untuk mengubah kode sumber menjadi AST dan CFG menggunakan bahasa pemrograman yang sudah dipilih.
- Memvisualisasikan hasil AST dan CFG dari kode sumber.

1.4 Batasan Masalah

Rumusan masalah yang telah disebutkan memiliki lingkup yang cukup luas. Menyadari terbatasnya waktu serta kemampuan, penelitian ini akan difokuskan dengan memperhaitkan batasan-batasan berikut ini:

1. Kode sumber yang bisa diproses secara maksimal oleh perangkat lunak adalah kode sumber dalam bahasa pemrograman Javascript sesuai spesifikasi ECMAScript versi 6 tahun 2015 [10].
2. Terdapat dua sintaks yang belum bisa diproses oleh perangkat lunak yaitu *generator function* dan *yield statement*. Penggunaan kedua sintaks tersebut sama seperti *function declaration* dan *return statement*, tetapi *generator function* dan *yield statement* sangat jarang ditemukan.
3. Perangkat lunak juga hanya bisa mendeteksi pemanggilan fungsi secara rekursif secara terbatas, yaitu mendeteksi fungsi rekursif yang memanggil fungsi di dalam *call stack*. Perlu dilakukan analisis lebih jauh terhadap *syntax tree* untuk mendeteksi pemanggilan fungsi yang menggunakan dua atau lebih fungsi independen yang saling memanggil.
4. Perangkat lunak yang dibangun bisa mendeteksi kesalahan penulisan sintaks kode program, tetapi belum bisa memberitahu kesalahan penulisan seperti layaknya *error message* yang dikeluarkan pada proses kompilasi. Penelitian yang dilakukan difokuskan untuk membangun AST dan CFG, dan keluaran *error message* tidak dibutuhkan untuk membangun AST dan CFG.

1.5 Metode Penelitian

Berikut merupakan metodologi penelitan yang akan dilakukan pada penelitian ini:

1. Melakukan studi literatur mengenai struktur data *Abstract Syntax Tree* dan *Control Flow Graph*.
2. Melakukan studi literatur tentang pembangunan *Abstract Syntax Tree* dan *Control Flow Graph*.
3. Mempelajari kandidat bahasa pemrograman yang akan digunakan, berkaitan dengan pembangunan *Abstract Syntax Tree* dan *Control Flow Graph*.
4. Memilih bahasa pemrograman untuk membangun *Abstract Syntax Tree* dan *Control Flow Graph*.

5. Merancang implementasi *Abstract Syntax Tree* dan *Control Flow Graph* pada bahasa pemrograman yang sudah dipilih.
6. Membangun perangkat lunak pembangun *Abstract Syntax Tree* dan *Control Flow Graph*.
7. Merancang visualisasi hasil *Abstract Syntax Tree* dan *Control Flow Graph*.
8. Melakukan pengujian terhadap perangkat lunak.
9. Membuat dokumen skripsi.

1.6 Sistematika Pembahasan

1. Bab Pendahuluan
   Bab 1 berisi latar belakang mengenai *Abstract Syntax Tree* dan *Control Flow Graph*, rumusan masalah, tujuan, batasan masalah, metode penelitian, sistematika pembahasan.
2. Bab Landasan Teori
   Bab 2 berisi penjelasan tentang dasar-dasar teori yang digunakan dalam penelitian ini, yaitu teori graf, struktur data *Control Flow Graph*, struktur data *Abstract Syntax Tree*, bahasa pemrograman dan kode sumber, *compiler*, analisis leksikal, analisis sintaksis, *cyclomatic complexity*, Javascript, dan ECMAScript.
3. Bab Analisis
   Bab 3 berisi analisis *parsing*, struktur data *abstract syntax tree* dan *control flow graph*, dan cara membangunnya. Bab ini juga menganalisis perangkat lunak yang dibangun untuk melakukan *parsing*.
4. Bab Perancangan
   Bab 4 berisi perancangan perangkat lunak yang mampu membangun AST dan CFG, dalam bentuk perancangan antarmuka, diagram kelas dan deskripsi rinci dari kelas-kelas tersebut, dan diagram sekuens.
5. Bab Implementasi dan Pengujian
   Bab 5 berisi implementasi perangkat lunak yang mampu membangun AST dan CFG, lingkungan implementasi perangkat lunak, pengujian, dan hasil dari pengujian perangkat lunak tersebut.
6. Bab Kesimpulan dan Saran
   Bab 6 berisi kesimpulan yang didapat dari hasil penelitian dan saran yang diusulkan untuk penelitian lebih lanjut.