Teknik: Jurnal Ilmu Teknik dan Informatika
Volume 6 Nomor 2 Oktober 2026
e-ISSN: 2798-2513; p-ISSN: 2808-8751, Hal *158-171*
DOI: https://doi.org/10.51903/teknik.v6i1.1276
Available online at: https://journal.stiestekom.ac.id/index.php/TEKNIK

Rancang Bangun Penganalisis Sintaksis (*Syntax Analyzer*) Ekspresi Matematika Menggunakan Pendekatan *Context-Free Grammar*

Muhammad Tsabat Muhyiyuddin[1], Zulfahmi Indra[2], Muhammad Rois Lukman Damanik[3], Juhraini Helfiana Lexa[4]

1-4 Program Studi Pendidikan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Medan

Korespondensi penulis: izuddintsabat@gmail.com

Abstract. *The development of a mathematical expression syntax analyzer is an important part of computer science because it supports the development of compilers, computer algebra systems, and interactive learning media. However, the syntactic validation process of arithmetic expressions is still difficult for many students to understand. This research aims to design and develop a web-based syntax analyzer application to verify the structure of arithmetic expressions using the Context-Free Grammar (CFG) approach. The grammar is written in Backus-Naur Form (BNF) notation which includes basic arithmetic operators, parentheses, integers, and decimal numbers with operator precedence and associativity settings. The application development uses the Waterfall method which includes requirements analysis, design, implementation, testing, and evaluation. The syntactic analysis is implemented using the Recursive Descent Parsing algorithm in JavaScript and is equipped with real-time parse tree visualization and clear syntactic error information. The results of black-box testing show that the application is able to identify valid and invalid expressions according to grammatical rules. The resulting application has the potential to be an effective learning medium to help students understand formal languages and compiler constructions interactively.*
Keywords: *Context-Free Grammar (CFG); Recursive Descent Parsing; Analisis Sintaksis; Ekspresi Matematika,;Parse Tree.*

Abstrak. Pembangunan penganalisis sintaksis ekspresi matematika merupakan bagian penting dalam ilmu komputer karena mendukung pengembangan kompilator, sistem aljabar komputer, dan media pembelajaran interaktif. Namun, proses validasi sintaksis ekspresi aritmatika masih sulit dipahami oleh banyak mahasiswa. Penelitian ini bertujuan merancang dan mengembangkan aplikasi penganalisis sintaksis berbasis web untuk memverifikasi struktur ekspresi aritmatika menggunakan pendekatan Context-Free Grammar (CFG). Tata bahasa ditulis dalam notasi Backus-Naur Form (BNF) yang mencakup operator aritmatika dasar, tanda kurung, bilangan bulat, dan bilangan desimal dengan pengaturan presedensi serta asosiativitas operator. Pengembangan aplikasi menggunakan metode Waterfall yang meliputi analisis kebutuhan, perancangan, implementasi, pengujian, dan evaluasi. Analisis sintaksis diimplementasikan menggunakan algoritma Recursive Descent Parsing dalam JavaScript serta dilengkapi visualisasi pohon parse secara real-time dan informasi kesalahan sintaksis yang jelas. Hasil pengujian black-box menunjukkan bahwa aplikasi mampu mengidentifikasi ekspresi valid dan tidak valid sesuai aturan tata bahasa. Aplikasi yang dihasilkan berpotensi menjadi media pembelajaran yang efektif untuk membantu mahasiswa memahami bahasa formal dan konstruksi kompilator secara interaktif.
Kata kunci: CFG; Parsing Rekursi;, Analisis Sintaksis; Ekspresi Matematika; Pohon Parse.

1. LATAR BELAKANG

Perkembangan sains komputasi tidak dapat dilepaskan dari peran teori bahasa formal yang menjadi fondasi bagi pembangunan kompilator, interpreter, hingga sistem aljabar komputer modern (Ningsih & Nurfauziah, 2023). Setiap kali sebuah ekspresi matematika dimasukkan ke dalam program komputer, komputer harus terlebih dahulu memahami struktur ekspresi tersebut sebelum dapat mengevaluasi nilainya. Proses pemahaman struktur ini dikenal sebagai analisis sintaksis atau parsing yang merupakan salah satu tahap krusial dalam pemrosesan bahasa formal (Pratama et al., 2017.)

Diterima: 07 Juni 2026; Direvisi: 22 Juli 2026; Disetujui: 10 September 2026; Tersedia Online: 29 September 2026; Diterbitkan: 17 Oktober 2026.

Secara teknis, ekspresi matematika wajib diverifikasi oleh sebuah penganalisis sintaksis untuk memastikan bahwa rangkaian token-tokennya sesuai dengan aturan tata bahasa yang berlaku. Tanpa proses ini, komputer tidak akan mampu membedakan antara ekspresi yang sah secara sintaksis dengan ekspresi yang mengandung kesalahan struktural seperti kurung yang tidak seimbang atau operator yang berdampingan (Hosseinpour et al., 2018)

Salah satu pendekatan paling umum untuk mendeskripsikan struktur sintaksis ekspresi matematika adalah Context-Free Grammar (CFG). CFG memungkinkan pendefinisian aturan produksi yang bersifat rekursif sehingga dapat menangkap sifat hierarkis ekspresi matematika, termasuk presedensi dan asosiativitas (Pratama et al., 2017). Munir (2021) menyatakan bahwa CFG merupakan bagian dari hierarki Chomsky tipe 2 yang lebih ekspresif dibandingkan ekspresi reguler sehingga sangat sesuai untuk merepresentasikan bahasa pemrograman dan ekspresi matematis. Notasi yang lazim digunakan untuk menuliskan CFG adalah Backus-Naur Form (BNF) yang sederhana namun ekspresif (Krishnaswami & Yallop, 2019).

Meskipun materi CFG, parser, dan analisis sintaksis sudah diajarkan di berbagai mata kuliah, sebagian besar mahasiswa masih kesulitan memahami bagaimana aturan tata bahasa dapat diterjemahkan menjadi sebuah program nyata yang mampu memvalidasi ekspresi (Apriansyah et al., 2019). Bahan ajar yang tersedia umumnya berhenti pada level teoritis dan jarang menyentuh implementasi konkret yang dapat dijalankan secara interaktif oleh mahasiswa (Febiharsa et al., 2018)

Beberapa penelitian sebelumnya telah membahas penerapan CFG. Pratama et al. (2017) menerapkan LALR Parser berbasis CFG untuk analisis kalimat bahasa Indonesia. Sementara itu, pembahasan mengenai dasar-dasar tata bahasa formal dan teori automata yang menjadi landasan teknik kompilasi. Penelitian terdahulu mengenai Context-Free Grammar dan parsing telah menunjukkan efektivitas metode tersebut dalam melakukan analisis sintaksis terhadap berbagai bentuk bahasa formal. Akan tetapi, sebagian besar penelitian masih menitikberatkan pada proses validasi sintaksis dan belum menyediakan visualisasi pohon parse secara real-time yang dapat mempermudah pemahaman pengguna terhadap struktur hierarkis hasil parsing.

Dari tinjauan tersebut, teridentifikasi celah penelitian berupa belum tersedianya media pembelajaran interaktif berbasis web yang menyajikan pohon parse ekspresi matematika secara langsung, termasuk dukungan untuk operator perpangkatan dengan asosiativitas kanan. Penelitian ini bertujuan merancang dan membangun aplikasi penganalisis sintaksis berbasis web yang berfungsi sebagai (1) media pembelajaran interaktif, (2) implementasi nyata teori bahasa formal, dan (3) artefak yang terbuka untuk dikembangkan lebih lanjut.

2. KAJIAN TEORITIS

Sains Komputasi dan Teori Bahasa Formal

Sains komputasi adalah disiplin ilmu yang menggabungkan teori matematika dan ilmu komputer untuk menyelesaikan persoalan kompleks (Munir, 2021). Teori bahasa formal menyediakan kerangka kerja matematis untuk mendefinisikan struktur sintaksis suatu bahasa. (Nabhan et al., 2023.) menyatakan bahwa setiap bahasa formal dapat dikategorikan ke dalam hierarki Chomsky berdasarkan kekuatan aturan produksinya, dari bahasa reguler hingga bahasa rekursif enumerasi.

Sintaksis dan Analisis Sintaksis

Sintaksis adalah seperangkat aturan yang menentukan bagaimana simbol-simbol dasar atau token suatu bahasa dapat dikombinasikan menjadi kalimat yang sah (Pratama et al., 2017). analisis sintaksis adalah proses memeriksa apakah suatu rangkaian token mengikuti aturan tata bahasa yang telah didefinisikan, sekaligus membangun representasi struktural berupa pohon sintaksis. Sebuah penganalisis sintaksis bekerja setelah tahap analisis leksikal; jika input tidak valid, parser melaporkan syntax error beserta posisinya (Schröder & Cito, 2022).

Context-Free Grammar (CFG)

Context-Free Grammar (CFG) merupakan jenis tata bahasa formal tingkat 2 dalam hierarki Chomsky (Munir, 2021). CFG didefenisiskan sebagai tuple $G = (V, T, P, S)$ di mana V adalah himpunan simbol non-terminal, T adalah himpunan simbol terminal, P adalah himpunan aturan produksi, dan S adalah simbol awal. Setiap aturan produksi berbentuk $A \rightarrow \alpha$. Sifat 'context-free' berarti setiap non-terminal dapat diturunkan tanpa bergantung pada konteks di sekitarnya.

Backus-Naur Form (BNF)

Backus-Naur Form (BNF) adalah notasi standar untuk menuliskan aturan produksi CFG yang pertama kali digunakan untuk mendefinisikan bahasa pemrograman ALGOL 60. Untuk ekspresi aritmatika, hierarki produksi diatur sehingga penjumlahan dan pengurangan berada pada level terendah, perkalian dan pembagian pada level menengah, perpangkatan pada level lebih tinggi dengan asosiativitas kanan, dan tanda kurung pada level tertinggi (Munir, 2021). Extended BNF (EBNF) menambahkan notasi opsional dan perulangan untuk menyederhanakan grammar yang bersifat iteratif.

Pohon Parse (Parse Tree)

Pohon parse adalah representasi grafis dari proses derivasi sebuah string oleh CFG (Munir, 2021). Akar pohon merepresentasikan simbol awal grammar, sedangkan daun-daunnya

merupakan token masukan yang dihasilkan dari proses analisis leksikal. Simpul internal menunjukkan simbol non-terminal yang terbentuk melalui penerapan aturan produksi grammar. truktur ini menggambarkan hubungan antara simbol terminal dan non-terminal yang terbentuk melalui penerapan aturan produksi grammar, sehingga dapat digunakan untuk memvisualisasikan proses parsing dan membantu interpretasi struktur sintaks suatu ekspresi (Pratama et al., 2017). Visualisasi pohon parse sangat bermanfaat dalam konteks pembelajaran karena membuat struktur hierarkis ekspresi menjadi terlihat secara langsung (Febiharsa et al., 2018).

Recursive Descent Parsing

Recursive Descent Parser (RDP) merupakan teknik parsing top-down yang membangun struktur sintaksis berdasarkan aturan Context-Free Grammar (CFG). Pendekatan ini digunakan untuk melakukan analisis sintaks terhadap masukan dengan mengikuti aturan grammar yang telah didefinisikan sehingga struktur ekspresi dapat direpresentasikan secara sistematis (Scott & Johnstone, 2018). Karena grammar aritmatika mengandung rekursi kiri, grammar tersebut harus ditransformasi menjadi bentuk bebas rekursi kiri sebelum diimplementasikan dengan RDP. Transformasi ini menghasilkan produksi iteratif yang diwujudkan dengan perulangan while di dalam fungsi parser (Pratama et al., 2017).

Metode Pengembangan Waterfall

Model Waterfall adalah salah satu model SDLC yang paling banyak digunakan karena tahapannya yang sistematis dan berurutan (Sumbodo & Mustofa, 2025).Model Waterfall merupakan metode pengembangan perangkat lunak yang dilakukan secara berurutan melalui tahapan analisis kebutuhan, perancangan, implementasi, pengujian, dan pemeliharaan sehingga setiap tahap harus diselesaikan sebelum melanjutkan ke tahap berikutnya. Model ini dipilih pada penelitian dengan kebutuhan yang stabil dan terdefinisi jelas karena dapat meminimalkan risiko kesalahan desain di tahap awal (Ningsih & Nurfauziah, 2023).

Pengujian Black-Box

Black-box testing adalah teknik pengujian perangkat lunak yang berfokus pada perilaku fungsional sistem tanpa memandang struktur internalnya. Pengujian dilakukan dengan menyiapkan kasus uji berupa input tertentu dan membandingkan output aktual dengan output yang diharapkan (Wijaya & Wardijono, 2023). Black-box testing sangat sesuai untuk memvalidasi apakah parser memberikan respons yang benar terhadap ekspresi valid maupun tidak valid (Uminingsih et al., 2022).

Penelitian Terdahulu yang Relevan

Pratama et al., (2017) menerapkan algoritma LALR Parser berbasis CFG untuk menganalisis struktur kalimat bahasa Indonesia dan membuktikan bahwa CFG efektif untuk parsing bahasa formal. Mengembangkan aplikasi multimedia pembelajaran berbasis web menggunakan JavaScript, namun fokusnya pada konten pembelajaran bukan pada analisis sintaksis (Wardana et al., 2019). Mengembangkan media pembelajaran interaktif teori bahasa otomata berbasis web yang sudah mencakup simulasi automata, namun belum menyertakan implementasi parser dengan visualisasi pohon parse (Maulana et al., 2019).

Penelitian ini memposisikan diri sebagai pelengkap penelitian tersebut dengan tiga kebaruan: (1) parser dibangun berbasis web menggunakan JavaScript murni tanpa dependensi eksternal; (2) visualisasi pohon parse ditampilkan secara langsung saat ekspresi dimasukkan; dan (3) mendukung perpangkatan dengan asosiativitas kanan yang sering diabaikan pada implementasi sederhana sebelumnya.

3. METODE PENELITIAN

Jenis dan Pendekatan Penelitian

Penelitian ini merupakan penelitian rancang bangun (research and development) dengan pendekatan kualitatif-deskriptif (Ardiansyah et al., 2022).Tujuannya adalah menghasilkan artefak perangkat lunak berupa aplikasi penganalisis sintaksis ekspresi matematika berbasis web lengkap dengan dokumentasi rancangan tata bahasa, algoritma parsing, dan pengujian fungsionalnya.

Model Pengembangan Perangkat Lunak

Model pengembangan yang digunakan adalah Waterfall yang dipilih karena cakupan penelitian yang terdefinisi jelas dan kebutuhan yang . MenurutSommerville (2016), Waterfall terdiri atas lima tahap berurutan: analisis kebutuhan, perancangan sistem, implementasi, pengujian, dan evaluasi. Tahapan ini dijalankan secara linier dengan kemungkinan iterasi terbatas apabila ditemukan kesalahan pada tahap akhir.

Analisis Kebutuhan

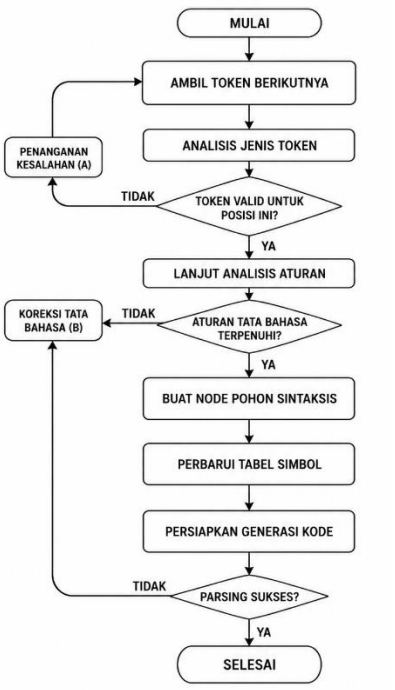
Pada tahap ini diidentifikasi kebutuhan fungsional dan non-fungsional sistem. Kebutuhan fungsional meliputi: (a) menerima input ekspresi matematika, (b) melakukan tokenisasi, (c) memvalidasi struktur sintaksis berdasarkan grammar, (d) menampilkan pohon parse dari ekspresi valid, dan (e) menampilkan pesan error informatif dengan posisi kesalahan. Kebutuhan non-fungsional meliputi: berbasis web, dapat dijalankan di browser modern tanpa instalasi, antarmuka responsif, serta proses parsing tampil secara real-time (Febiharsa et al., 2018).

Perancangan Tata Bahasa

Tata bahasa dirancang dalam notasi BNF mencakup lima operator aritmatika, tanda kurung, serta bilangan bulat dan desimal. Presedensi diatur melalui hierarki produksi dengan asosiativitas kanan untuk perpangkatan dan kiri untuk operator lainnya. Grammar dirancang bebas ambiguitas dan bebas rekursi kiri agar dapat diimplementasikan langsung dengan Recursive Descent Parsing (Apriansyah et al., 2019).

Perancangan Algoritma Parsing

Algoritma yang dipilih adalah Recursive Descent Parsing karena kesesuaiannya dengan grammar yang dirancang dan kemudahan pemahaman dari sisi pedagogis. Setiap non-terminal diterjemahkan menjadi satu fungsi parser yang dipanggil secara rekursif. Saat aturan berhasil diterapkan, parser membangun simpul pohon parse baru sebagai anak dari simpul pemanggilnya (Slivnik & Mernik, 2023). Alur kerja proses analisis sintaksis secara keseluruhan digambarkan pada flowchart berikut.



Gambar 1. Flowchart Proses Analisis Sintaksis

Implementasi

Implementasi dilakukan menggunakan JavaScript yang berjalan di sisi klien sehingga tidak membutuhkan server back-end. Antarmuka dibangun dengan HTML5 dan CSS3, sementara logika tokenizer, parser, dan visualisasi pohon parse ditulis dalam JavaScript murni (vanilla JS) tanpa dependensi eksternal. Visualisasi pohon parse dilakukan dengan penggambaran berbasis SVG yang menempatkan setiap simpul pada koordinat yang dihitung secara rekursif dari struktur pohon (Febiharsa et al., 2018).

Pengujian

Pengujian dilakukan dengan metode black-box testing yang berfokus pada perilaku fungsional sistem (Apriansyah et al., 2019). Kasus uji terdiri atas ekspresi valid dengan berbagai variasi presedensi, asosiativitas, dan kedalaman kurung; serta ekspresi tidak valid berupa operator ganda, kurung tidak seimbang, dan token tak dikenal. Untuk setiap kasus uji, sistem diharapkan menghasilkan pohon parse yang benar atau pesan error dengan posisi yang tepat (Maidl et al., 2016)

Evaluasi

Evaluasi dilakukan melalui dua pendekatan: (1) evaluasi fungsional dengan menghitung persentase kasus uji yang berhasil ditangani sesuai dan (2) evaluasi kualitatif dengan menelaah kesesuaian struktur pohon parse terhadap aturan presedensi dan asosiativitas yang berlaku secara matematis (Munir, 2021).

4. HASIL DAN PEMBAHASAN

Perancangan Tata Bahasa (Grammar BNF)

Tata bahasa ekspresi matematika yang digunakan dalam sistem ini dirancang menggunakan notasi Backus-Naur Form (BNF). Grammar didesain bebas ambiguitas dan bebas rekursi kiri agar sepenuhnya kompatibel dengan metode Recursive Descent Parsing. Hierarki produksi disusun sedemikian rupa sehingga presedensi dan asosiativitas operator terpenuhi secara alami. Tabel 1 menyajikan seluruh aturan produksi tata bahasa yang diimplementasikan.

Tabel 1. Aturan Produksi Tata Bahasa BNF Ekspresi Matematika

Non-Terminal	Produksi (BNF)
<expr>	<term> (('+'|'-') <term>)*
<term>	<power> (('*'|'/') <power>)*
<power>	<unary> ('^' <power>)?
<unary>	'-' <unary> | <primary>
<primary>	'(' <expr> ')' | NUMBER
NUMBER	[0-9]+ ('.' [0-9]+)?

Berdasarkan tata bahasa pada Tabel 1, ekspresi <expr> diturunkan menjadi rangkaian <term> yang dihubungkan oleh operator penjumlahan atau pengurangan. Operator dengan presedensi lebih tinggi ditempatkan pada lapisan produksi yang lebih dalam. Penggunaan rekursi kanan pada aturan <power> menghasilkan asosiativitas kanan untuk perpangkatan, sehingga ekspresi 2^3^2 diinterpretasikan sebagai 2^(3^2) = 512 sesuai konvensi matematis yang berlaku.

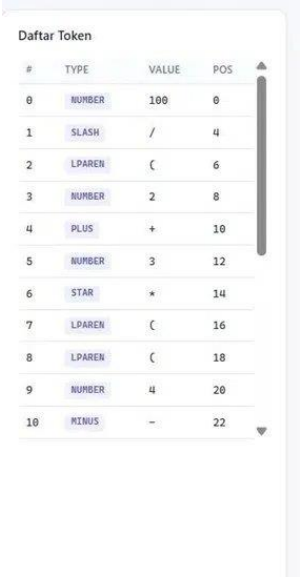
Implementasi Tokenizer

Langkah pertama dalam proses analisis sintaksis adalah tokenisasi, yakni penguraian string masukan menjadi satuan-satuan token yang bermakna. Tokenizer yang dikembangkan mengklasifikasikan setiap karakter masukan ke dalam tipe token yang sesuai berdasarkan pola leksikalnya. Tabel 2 merinci tipe-tipe token yang dikenali oleh sistem beserta simbol dan contoh kemunculannya pada ekspresi masukan.

Tabel 2. Tipe Token yang Dikenali oleh Tokenizer

Tipe Token	Simbol	Contoh	Keterangan
NUMBER	-	3, 100, 3.14	Bilangan bulat atau desimal
PLUS	+	+	Operator penjumlahan
MINUS	-	-	Operator pengurangan/unary minus
STAR	*	*	Operator perkalian
SLASH	/	/	Operator pembagian
CARET	^	^	Operator perpangkatan
LPAREN	(	(	Tanda kurung buka
RPAREN	)	)	Tanda kurung tutup

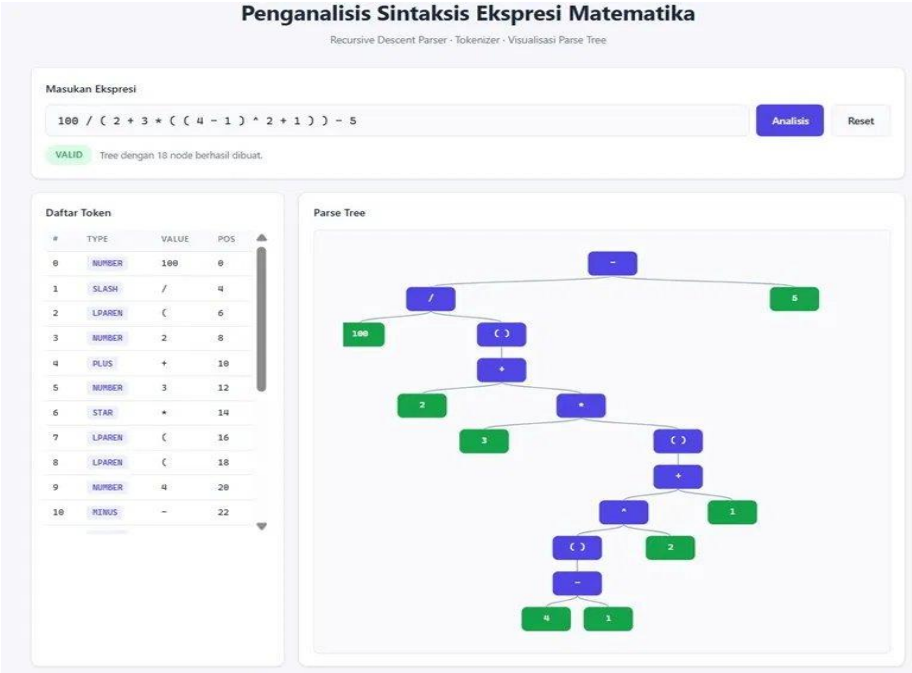
Proses tokenisasi terhadap ekspresi 100 / (2 + 3 * ((4 - 1) ^ 2 + 1)) - 5 menghasilkan 18 token yang ditampilkan secara lengkap pada panel Daftar Token di antarmuka aplikasi. Setiap token memiliki atribut TYPE, VALUE, dan POS yang masing-masing mencatat tipe, nilai leksikal, serta posisi karakter dalam string masukan. Informasi posisi ini dimanfaatkan oleh parser untuk menyusun pesan kesalahan yang informatif apabila ditemukan error sintaksis.



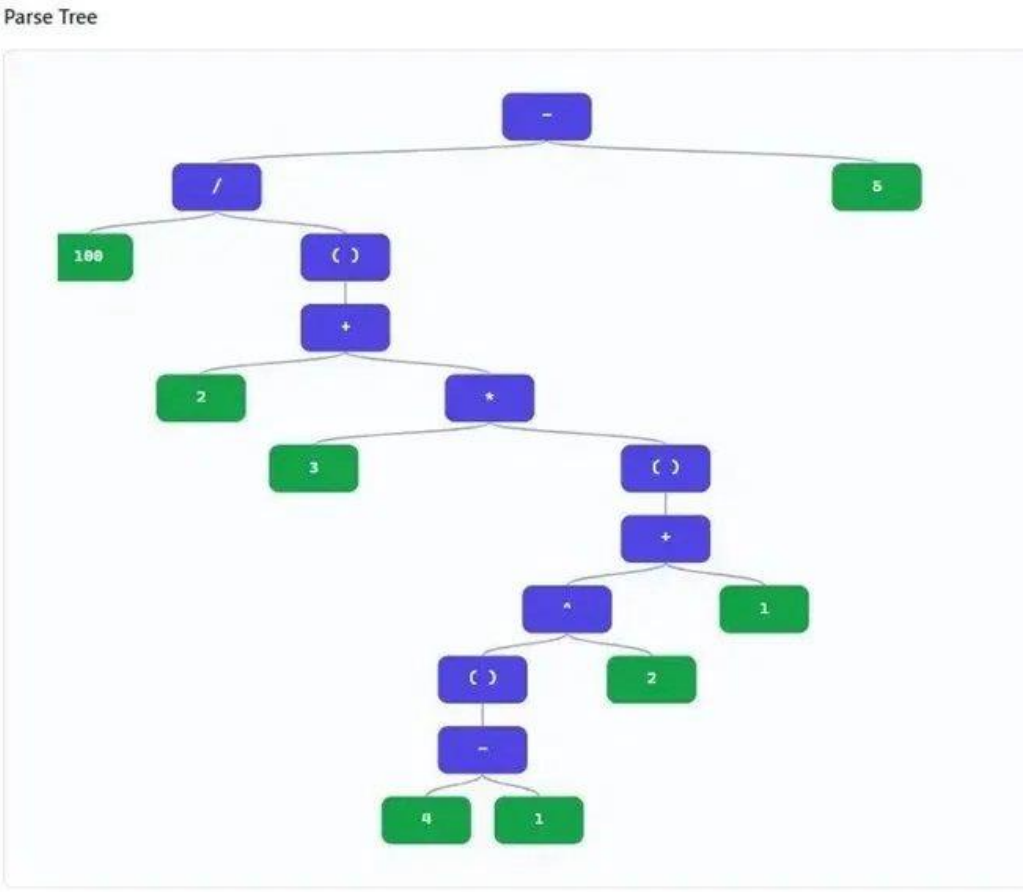
Gambar 2. Panel Daftar Token (Tokenizer Output) pada Antarmuka Aplikasi

Tampilan Antarmuka Aplikasi dan Visualisasi Parse Tree

Antarmuka aplikasi penganalisis sintaksis dirancang agar mudah digunakan dan responsif di berbagai perangkat. Pengguna memasukkan ekspresi matematika pada kolom input, kemudian menekan tombol "Analisis" untuk menjalankan proses tokenisasi dan parsing. Jika ekspresi yang dimasukkan valid secara sintaksis, sistem menampilkan badge "VALID" disertai informasi jumlah node yang terbentuk pada pohon parse. Sebaliknya, bila ekspresi mengandung kesalahan, sistem menampilkan badge "TIDAK VALID" diikuti pesan error yang menunjukkan posisi kesalahan secara tepat. Tampilan lengkap antarmuka aplikasi beserta visualisasi pohon parse dapat dilihat pada Gambar 3.



Gambar 3. Tampilan Lengkap Aplikasi Penganalisis Sintaksis



Gambar 4. Visualisasi Pohon Parse Ekspresi Matematika

Visualisasi pohon parse dirender secara grafis di bagian kanan antarmuka menggunakan teknologi SVG. Simpul operator ditampilkan dengan warna biru-ungu, sedangkan simpul yang merepresentasikan nilai angka (token NUMBER) ditampilkan dengan warna hijau. Tanda kurung yang berfungsi sebagai pengelompokan diwakili oleh simbol "( )" pada simpul tersendiri. Garis penghubung antarsimpul menggambarkan relasi induk-anak sesuai hierarki derivasi grammar. Sebagai contoh, ekspresi 100 / (2 + 3 * ((4 - 1) ^ 2 + 1)) - 5 menghasilkan pohon parse dengan 18 node; simpul akar adalah operator pengurangan (-) dengan cabang kiri berupa subpohon pembagian (/) dan cabang kanan berupa nilai 5.

Contoh Ekspresi Valid dan Tidak Valid

Aplikasi dilengkapi panel contoh ekspresi yang terbagi ke dalam dua kategori: ekspresi yang dinyatakan valid dan ekspresi yang dinyatakan tidak valid. Fitur ini dirancang untuk membantu mahasiswa memahami pola ekspresi yang diterima maupun ditolak oleh grammar. Kategori valid mencakup variasi operator, presedensi, kedalaman kurung, bilangan negatif unary, serta bilangan desimal. Adapun kategori tidak valid meliputi operator berurutan (3 + +

4), kurung tidak seimbang ((2 + 3), ekspresi tidak lengkap (5 *), token tak dikenal (3 . 14), dan dua angka tanpa operator (2 + 3 4).



Gambar 5. Panel Contoh Ekspresi Valid dan Tidak Valid pada Antarmuka

Hasil Pengujian Black-Box

Pengujian fungsional dilaksanakan menggunakan metode black-box testing terhadap 12 kasus uji yang terdiri atas 7 ekspresi valid dan 5 ekspresi tidak valid. Tabel 3 menyajikan hasil pengujian ekspresi valid, sedangkan Tabel 4 menyajikan hasil pengujian ekspresi tidak valid.

Tabel 3. Hasil Pengujian Black-Box - Ekspresi Valid

No.	Ekspresi Masukan	Output yang Diharapkan	Hasil Aktual
1	3 + 4	VALID - 2 node	VALID - 2 node (Sesuai)
2	2 * (3 + 4)	VALID - 5 node	VALID - 5 node (Sesuai)
3	-5 + 2.5 * 3	VALID - 5 node	VALID - 5 node (Sesuai)
4	2 ^ 3 ^ 2	VALID - 5 node (asosiativitas kanan)	VALID - 5 node (Sesuai)
5	(1 + 2) * (3 - 4) / 5	VALID - 7 node	VALID - 7 node (Sesuai)
6	-(3 + 4) ^ 2	VALID - 5 node	VALID - 5 node (Sesuai)
7	100/(2+3*((4-1)^2+1))-5	VALID - 18 node	VALID - 18 node (Sesuai)

Tabel 4. Hasil Pengujian Black-Box - Ekspresi Tidak Valid

No.	Ekspresi Masukan	Output yang Diharapkan	Hasil Aktual
1	3 + + 4	TIDAK VALID - operator berurutan	TIDAK VALID - posisi error (Sesuai)
2	(2 + 3	TIDAK VALID - kurung tidak seimbang	TIDAK VALID - posisi error (Sesuai)
3	5 *	TIDAK VALID - ekspresi tidak lengkap	TIDAK VALID - posisi error (Sesuai)
4	3 . 14	TIDAK VALID - token tak dikenal	TIDAK VALID - posisi error (Sesuai)

5	2 + 3 4	TIDAK VALID - dua angka berurutan	TIDAK VALID - posisi error (Sesuai)

Berdasarkan Tabel 3 dan Tabel 4, seluruh 12 kasus uji menghasilkan keluaran yang sesuai dengan ekspektasi, sehingga tingkat keberhasilan pengujian fungsional mencapai 100%. Sistem terbukti secara konsisten mampu membedakan ekspresi valid dari yang tidak valid, menghitung jumlah node pohon parse dengan tepat, serta menyajikan pesan kesalahan yang informatif disertai indikasi posisi error untuk setiap ekspresi yang tidak valid.

Perbandingan dengan Penelitian Terdahulu

Untuk menegaskan kontribusi penelitian ini, berikut diulas perbandingannya dengan empat penelitian terdahulu yang paling relevan. Pratama et al. (2017) membangun LALR Parser berbasis CFG untuk analisis kalimat bahasa Indonesia; sistem tersebut tidak berjalan berbasis web dan tidak menyertakan visualisasi pohon parse. Wardana et al. (2019) mengembangkan pemeriksa pola kalimat bahasa Indonesia berbasis algoritma left-corner parsing yang sudah berjalan di web, tetapi tidak dilengkapi visualisasi parse tree maupun dukungan operator perpangkatan. Maulana et al. (2019) menghadirkan simulator FSA interaktif berbasis web; meski responsif dan mudah digunakan, implementasinya terbatas pada automata dan tidak mencakup parsing ekspresi dengan pohon sintaksis yang dapat ditelusuri. Penelitian sebelumnya telah mengembangkan aplikasi transformasi Bentuk Normal Chomsky berbasis web untuk menyederhanakan tata bahasa bebas konteks. Meskipun sistem tersebut mampu melakukan transformasi aturan produksi dan mendukung perpangkatan secara parsial, visualisasi pohon parse secara real-time sebagai sarana interpretasi hasil analisis sintaksis belum diimplementasikan.

Dibandingkan dengan keempat penelitian tersebut, aplikasi yang dikembangkan dalam penelitian ini merupakan satu-satunya yang secara bersamaan memenuhi tiga kriteria kebaruan: berjalan sepenuhnya berbasis web tanpa instalasi tambahan, menyertakan visualisasi pohon parse secara real-time menggunakan SVG, serta menangani operator perpangkatan dengan asosiativitas kanan yang tepat sesuai konvensi matematis. Ketiga aspek ini secara kolektif menjadikan sistem yang dikembangkan lebih komprehensif sebagai media pembelajaran teori bahasa formal, khususnya pada topik analisis sintaksis dan konstruksi kompilator.

5. KESIMPULAN DAN SARAN

Penelitian ini berhasil merancang dan membangun aplikasi penganalisis sintaksis ekspresi matematika berbasis web dengan menggunakan pendekatan Context-Free Grammar (CFG) dan algoritma Recursive Descent Parsing. Tata bahasa yang dirancang dalam notasi BNF mencakup lima operator aritmatika dengan hierarki presedensi yang tepat, termasuk

perpangkatan dengan asosiativitas kanan, serta mendukung bilangan bulat, bilangan desimal, unary minus, dan pengelompokan menggunakan tanda kurung.

Hasil pengujian black-box menunjukkan bahwa sistem mampu menangani seluruh 12 kasus uji dengan tingkat keberhasilan 100%. Visualisasi pohon parse yang direnderer secara real-time menggunakan SVG berhasil merepresentasikan struktur hierarkis derivasi ekspresi sesuai aturan grammar. Pesan kesalahan informatif yang menyertakan posisi error juga terbukti berfungsi akurat pada semua kasus uji ekspresi tidak valid.

Untuk pengembangan selanjutnya, disarankan agar sistem diperluas dengan kemampuan evaluasi nilai ekspresi, dukungan variabel dan fungsi matematis, serta mode pembelajaran step-by-step yang menampilkan proses derivasi secara bertahap. Penelitian mendatang dapat pula mengeksplorasi penerapan Extended BNF (EBNF) dan parser generator otomatis untuk memperluas cakupan grammar yang dapat dianalisis.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada dosen pengampu mata kuliah Pengantar Sains Komputasi Program Studi Pendidikan Ilmu Komputer Universitas Negeri Medan atas bimbingan dan arahan yang diberikan selama penyusunan artikel ini.

DAFTAR REFERENSI

Apriansyah, A., Ichsan Saputra, T., & Fauzi Wijaya, Y. (2019). Desain Mesin Compiler untuk Penganalisa Leksikal, Sintaksis, Semantik, Kode Antara dan Error Handling Pada Bahasa Pemrograman Sederhana. In *Journal of Applied Informatics and Computing (JAIC)*, 3(1), 1-7. http://jurnal.polibatam.ac.id/index.php/JAIC

Ardiansyah, A., Gata, W., Bayhaqy, A., Rahayu, S., Hamdan, & Supriyanto, E. (2022). Penerapan konsep finite state automata pada proses pengajuan antivirus di lingkungan Provinsi DKI Jakarta. *Jurnal Ilmiah Ilmu Komputer*, *8*(2), 58–62. http://ejournal.fikom-unasman.ac.id

Febiharsa, D., Sudana, I. M., & Hudallah, N. (2018). Uji fungsionalitas (blackbox testing) sistem informasi lembaga sertifikasi profesi (SILSP) batik dengan AppPerfect Web Test dan uji pengguna. *Journal of Informatics Education*, *1*(2), 117–125.

Hosseinpour, S., Alavi Milani, M. M. R., & Pehlivan, H. (2018). A step-by-step solution methodology for mathematical expressions. *Symmetry*, *10*(7), Artikel 285. https://doi.org/10.3390/sym10070285

Krishnaswami, N. R., & Yallop, J. (2019). A typed, algebraic approach to parsing. *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'19)*, 379–393. https://doi.org/10.1145/3314221.3314625

Maidl, A. M., Mascarenhas, F., Medeiros, S., & Ierusalimschy, R. (2016). Error reporting in Parsing Expression Grammars. *Science of Computer Programming*, *132*, 129–140. https://doi.org/10.1016/j.scico.2016.08.004

Maulana, A. S., Azizah, H. N., & Kirana, K. C. (2019). Implementasi finite state automata (FSA) dengan simulasi vending machine pada aplikasi android. *Jurnal Edukasi Elektro*, *3*(2), 110–120. Retrieved http://journal.uny.ac.id/index.php/jee/

Munir, R. (2021). *Matematika diskrit* (Revisi ke-6). Informatika.

Nabhan, M. F., Lantana, D. A., Zidni, P., Arofi, A., Habibbullah, P., Mukti, A. F., & Khadavi, M. I. (2023). Simulator mesin deterministic finite automata (DFA) berdasarkan diagram transisi menggunakan Python. *Jurnal Sistem Komputer dan Kecerdasan Buatan*, *7*(1), 22–29.

Ningsih, W., & Nurfauziah, H. (2023). Perbandingan model waterfall dan metode prototype untuk pengembangan aplikasi pada sistem informasi. *Jurnal Ilmiah Metadata*, *5*(1), 83–95.

Pratama, M. R., Kusumadewi, S., & Hidayat, T. (2017). Penerapan algoritma LALR parser dan context-free grammar untuk struktur kalimat bahasa Indonesia. *Jurnal Teknologi Elektro*, *8*(1), 1–8.

Schröder, M., & Cito, J. (2022). Grammars for free: Toward grammar inference for ad hoc parsers. Dalam *Proceedings of the 2022 IEEE/ACM 44th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)* (hlm. 1–5). ACM. https://doi.org/10.1145/3510455.3512787

Scott, E., & Johnstone, A. (2018). GLL syntax analysers for EBNF grammars. Science of Computer Programming, 166, 120–145. https://doi.org/10.1016/j.scico.2018.06.002

Slivnik, B., & Mernik, M. (2023). On Parsing Programming Languages with Turing-Complete Parser. *Mathematics*, *11*(7). https://doi.org/10.3390/math11071594

Sommerville, Ian. (2016). *Software engineering*. Pearson.

Sumbodo, M., & Mustofa. (2025). Analisis metode system development life cycle (SDLC) dalam rancangan sistem informasi. *IKRAITH-INFORMATIKA*, *9*(3), 261–262. https://doi.org/10.37817/ikraith-informatika.v9i3

Uminingsih, Ichsanudin, M. N., Yusuf, M., & Suraya. (2022). Pengujian fungsional perangkat lunak sistem informasi perpustakaan dengan metode black box testing bagi pemula. *STORAGE: Jurnal Ilmiah Teknik dan Ilmu Komputer*, *1*(2), 1–8. https://doi.org/10.55123/storage.v1i2.235

Wardana, H. K., Swanita, I., & Yohanes, B. W. (2019). Sistem pemeriksa pola kalimat bahasa Indonesia berbasis algoritme left-corner parsing dengan stemming. *JNTETI (Jurnal Nasional Teknik Elektro dan Teknologi Informasi)*, *8*(3), 211–217. https://journal.ugm.ac.id/v3/JNTETI/article/view/2570

Wijaya, A. F., & Wardijono, B. A. (2023). Pengukuran kualitas aplikasi custody berdasarkan ISO 25010 menggunakan otomatisasi pengujian blackbox. *KLIK: Kajian Ilmiah Informatika dan Komputer*, *4*(2), 937–416. https://doi.org/10.30865/klik.v4i2.1215