Laporan Tugas II
IF2220 Teori Bahasa Formal dan Automata

Jonathan Christopher
(13515001, K-01)

Nicholas Thie
(13515079, K-01)

November 22, 2016

1 Deskripsi Persoalan

Dibutuhkan sebuah program untuk mengenali dan menghitung ekspresi aritmatika menggunakan tata bahasa bebas konteks (*context-free grammar*). Bila diberikan sebuah ekspresi aritmatika, maka program harus bisa mengenali apakah ekspresi tersebut valid atau tidak (*syntax error*). Jika ekspresi yang diberikan valid, maka program tersebut harus menghitung nilai dari ekspresi tersebut dengan mengubah terlebih dahulu setiap simbol terminal (angka) menjadi nilai numerik yang bersesuaian. Contoh ekspresi aritmatika yang valid adalah $(-457.01 + 1280) * (35.7 - 11.0233)/(-6.1450)$ (setelah dieksekusi akan menampilkan hasil perhitungan ekspresi tersebut yaitu -3304.91). Contoh ekspresi tidak valid adalah $3 * + - 12/(57)$ (setelah dieksekusi akan ditampilkan pesan *syntax error*).

1.1 Batasan Masalah

- Terminal terdiri dari karakter-karakter $\{+, -, *, /, (,), 0..9, .\}$.
- Operator terdiri dari $\{+, -, *, /, (,)\}$.
- Ekspresi di dalam kurung dievaluasi terlebih dahulu, kemudian operator perkalian dan pembagian, baru sesudahnya operator penambahan dan pengurangan.
- Operan terdiri dari bilangan bulat dan bilangan desimal positif atau negatif.
- Program merupakan implementasi dari tata bahasa dan *pushdown automata* yang dibuat terlebih dahulu (disain sendiri) menggunakan teori yang telah dipelajari.
- Implementasi program menggunakan bahasa pemrograman prosedural C atau Pascal.

2 Jawab Persoalan

Untuk validasi sintaks dan perhitungan ekspresi aritmatika, digunakan CFG (*context-free grammar*). Ekspresi aritmatika terlebih dahulu dimodelkan dalam context-free grammar tersebut. Karakter-karakter yang dierima dalam ekspresi, yaitu $\{+, -, *, /, (,), 0..9, .\}$, menjadi simbol terminal dalam CFG yang dibuat. Selain simbol terminal, CFG tersebut juga mengandung sejumlah variabel yang merepresentasikan fungsi dari potongan-potongan ekspresi tertentu. Salah satu variabel melambangkan keseluruhan ekspresi aritmatika itu sendiri dan menjadi *start symbol*. Eksekusi untuk validasi dan perhitungan akan selalu dimulai dengan variabel tersebut, lalu berlanjut ke sejumlah aturan produksi yang mendefinisikan tata bahasa ekspresi secara rekursif.

Implementasi *parser* yang dibuat menggunakan metode rekursif, sehingga program yang dihasilkan strukturnya tidak jauh berbeda dengan CFG yang digunakan. Sebuah variabel dalam CFG diubah menjadi fungsi dalam program, yang dapat memanggil fungsi-fungsi lainnya yang melambangkan variabel lainnya sesuai aturan produksi CFG. Karakter terminal dalam CFG berarti program harus membaca karakter tersebut dari *string* masukan. Akan tetapi, terdapat beberapa kriteria yang harus dipenuhi oleh CFG yang digunakan agar implementasi dengan metode rekursif ini dapat berjalan. Pertama, CFG tidak boleh ambigu, atau memiliki beberapa kemungkinan alur *parsing*. Kedua, tidak boleh ada aturan produksi dalam CFG dimana suatu variabel mengandung dirinya sendiri sebelum variabel lain atau simbol terminal.

Sebuah CFG pada dasarnya hanya dapat digunakan untuk mengecek apakah suatu *string* memenuhi aturan CFG tersebut atau tidak (dalam hal ini, validitas sintaks sebuah ekspresi aritmatika). Untuk dapat menghitung juga nilai hasil evaluasi suatu ekspresi aritmatika, dibutuhkan modifikasi pada program yang mengimplementasikannya. Selain dapat mem-*parse* simbol terminal dan variabel, program yang dibuat juga dapat mengartikan nilai numerik simbol terminal yang berupa digit serta mengaitkannya dengan urutan dan makna yang benar tergantung pada

simbol terminal operator yang ditemukan. Hasil evaluasi dikembalikan dalam *return value* tiap fungsi yang merepresentasikan variabel dalam CFG. Selain itu, untuk mempertahankan urutan evaluasi yang asosiatif-kiri (untuk operator yang prioritasnya sama), terpaksa digunakan iterasi untuk bagian tersebut, bukan rekursi.

Selain implementasi CFG dan evaluasi ekspresi, terdapat beberapa fitur tambahan yang dimuat dalam program ini. Terdapat empat tipe hasil evaluasi ekspresi yang dikenali, yaitu bilangan bulat, bilangan desimal, *syntax error*, serta *division error* (terjadi jika ada pembagian dengan nol). Bilangan bulat disimpan dalam tipe *integer* 64-bit, sedangkan bilangan desimal disimpan dalam tipe *double-precision floating point*. Program sedapat mungkin melakukan evaluasi dalam tipe bilangan bulat dan menggunakan tipe *floating-point* jika ada hasil yang memang tidak bulat. Hal ini dilakukan untuk sedapat mungkin mencegah ketidaktelitian perhitungan yang diakibatkan keterbatasan representasi *floating-point*. Selain itu, jika terdapat *syntax error*, program juga akan mencatat posisi karakter pertama yang mengakibatkan *syntax error* untuk mempermudah pencarian kesalahan.

3 *Context-Free Grammar*

Berikut adalah tata bahasa bebas konteks (*context-free grammar*) yang digunakan untuk memodelkan ekspresi aritmatika yang akan divalidasi dan dievaluasi:

$$G = (\{E, T, F, W, N, F, I\}, \{+, -, *, /, (,), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .\}, P, E)$$

dengan aturan produksi P yang didefinisikan sebagai berikut:

$$\begin{aligned}
E &\rightarrow E + T \mid E - T \mid T \\
T &\rightarrow T * F \mid T/F \mid F \\
F &\rightarrow -F \mid +F \mid W \\
W &\rightarrow (E) \mid N \\
N &\rightarrow I.F \mid I \\
F &\rightarrow D \mid DF \\
I &\rightarrow D \mid DI
\end{aligned}$$

Setiap variabel dalam CFG tersebut merepresentasikan sebuah komponen ekspresi aritmatika:

Variabel	Nama fungsi dalam program	Deskripsi
E	expression	keseluruhan ekspresi aritmatika
T	term	ekspresi angka, hasil perkalian atau hasil pembagian
F	factor	ekspresi yang dapat menjadi bagian dari operasi biner
W	factorWithoutUnary	angka atau ekspresi aritmatika yang dibatasi kurung
N	number	bilangan bulat atau desimal tidak negatif
I	integer	bilangan bulat tidak negatif
F	fractional	urutan digit yang terletak di belakang koma
D	digit	sebuah karakter digit (0..9)

4 *Source Code*

5 Contoh Interaksi dengan Program

Berikut adalah contoh interaksi pengguna dengan program. Input pengguna digarisbawahi.

```
       ...

Evaluator - Tugas II IF2220 Teori Bahasa Formal dan Automata
============================================================
```

Referensi

[1] Hopcroft, John E.; Motwani, Rajeev; Ullman, Jeffrey D. (2013). *Introduction to Automata Theory, Languages, and Computation (3rd ed.)*. Pearson.