LAPORAN TUGAS BESAR IF1210 TEORI BAHASA FORMAL DAN AUTOMATA

Daniel Egiant Sitanggang (13521056)
Go Dillon Audris (13521062)
Salomo Reinhart Gregory Manalu (13521063)

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung 2022

DAFTAR ISI

BAB I TEORI DASAR

1. CFG

CFG atau Context Free Grammar adalah tata bahasa formal di mana setiap aturan produksi adalah dalam bentuk A → B di mana A adalah yang memproduksi, dan B adalah hasil produksi. Batasannya hanyalah ruas kiri adalah sebuah simbol variabel. Dan pada ruas kanan bisa berupa terminal, symbol, variable ataupun ε, Contoh aturan produksi yang termasuk CFG adalah seperti berikut ini:

X -> bY | Za

Y -> aY | b

Z -> bZ | ε

CFG adalah tata bahasa yang mempunyai tujuan sama seperti halnya tata bahasa regular yaitu merupakan suatu cara untuk menunjukkan bagaimana menghasilkan suatu untai-untai dalam sebuah bahasa. CFG perlu disederhanakan dengan tujuan untuk melakukan pembatasan sehingga tidak menghasilkan pohon penurunan yang memiliki kerumitan yang tak perlu atau aturan produksi tak berarti

2. CNF

Bentuk normal Chomsky / Chomsky Normal Form (CNF) merupakan salah satu bentuk normal yang sangat berguna untuk Context Free Grammar (CFG). Bentuk normal Chomsky dapat dibuat dari sebuah tata bahasa bebas konteks yang telah mengalami penyederhanaan yaitu penghilangan produksi useless, unit, dan ε. Dengan kata lain, suatu tata bahasa bebas konteks dapat dibuat menjadi bentuk normal Chomsky dengan syarat tata bahasa bebas konteks tersebut:

- Tidak memiliki produksi useless
- Tidak memiliki produksi unit
- Tidak memiliki produksi ε

CNF adalah CFG dengan setiap produksinya berbentuk :

A -> BC

A -> a

Langkah-langkah pembentukan bentuk normal Chomsky secara umum sebagai berikut:

- Biarkan aturan produksi yang sudah dalam bentuk normal Chomsky
- Lakukan penggantian aturan produksi yang ruas kanannya memuat simbol terminal dan panjang ruas kanan > 1
- Lakukan penggantian aturan produksi yang ruas kanannya memuat > 2 simbol variabel
- Penggantian-penggantian tersebut bisa dilakukan berkali-kali sampai akhirnya semua aturan produksi dalam CNF
- Selama dilakukan penggantian, kemungkinan kita akan memperoleh aturan-aturan produksi baru, dan juga memunculkan simbol-simbol variabel baru

3. CYK

Cocke–Younger–Kasami-Algorithm (CYK atau CKY) adalah algoritma penguraian yang sangat efisien untuk *context free grammar*. Ini membuatnya ideal untuk menentukan masalah kata untuk tata bahasa bebas konteks, diberikan dalam bentuk normal Chomsky (CNF). Alat berikut dapat digunakan untuk memeriksa apakah kata tertentu $w \in \Sigma$ adalah bagian dari bahasa, diberikan dalam tata bahasa CNF.

Secara informal, algoritma bekerja sebagai berikut: Pada langkah pertama, tulis kata di baris pertama dan tambahkan setiap simbol non-terminal di baris di bawahnya yang menyimpulkan simbol terminal. Setelah itu, untuk setiap sel di grid mulai secara vertikal di atas dan turun menuju sel yang akan diperiksa dan sel kedua secara diagonal. Untuk setiap langkah tersebut, gabungkan sel dan periksa apakah kombinasi tersebut muncul di tata bahasa. Jika ya, tambahkan non-terminal sisi kiri ke sel kisi. Jika setelah semua langkah, simbol awal terdapat di baris terakhir, kata tersebut dapat diturunkan dengan tata bahasa yang diberikan.

BAB II FA DAN CFG

2.1 FA

['start', 'else', 'dead'],
['start', 'number', 'dead'],
['start', 'lowerCase', 'final'],
['start', 'upperCase', 'final'],
['start', 'specialSign', 'final'],
['final', 'else', 'dead'],
['final', 'number', 'final'],
['final', 'lowerCase', 'final'],
['final', 'upperCase', 'final'],
['final', 'specialSign', 'final'],
['dead', 'else', 'dead'],
['dead', 'number', 'dead'],
['dead', 'lowerCase', 'dead'],
['dead', 'upperCase', 'dead'],
['dead', 'specialSign', 'dead']

Pada representasi FA ini:
Setiap elemen pertama list melambangkan current state
Setiap elemen kedua list melambangkan input simbol yang diterima
Setiap elemen ketiga list melambangkan state yang dituju setelah FA menerima input simbol.

2.2 CFG

S -> IF_METHOD
S -> IF_METHOD CURFEW_OPEN
S -> ELSE_IF_METHOD
S -> ELSE_IF_METHOD CURFEW_OPEN
S -> ELSE_METHOD
S -> ELSE_METHOD CURFEW_OPEN
S -> WHILE_METHOD
S -> WHILE_METHOD CURFEW_OPEN
S -> FOR_METHOD
S -> FOR_METHOD CURFEW_OPEN
S -> BREAK_METHOD
S -> BREAK_METHOD CURFEW_CLOSE
S -> CONTINUE_METHOD
S -> CONTINUE_METHOD CURFEW_CLOSE

S -> FUNCTION_METHOD
S -> FUNCTION_METHOD CURFEW_OPEN
S -> RETURN_METHOD
S -> RETURN_METHOD CURFEW_CLOSE
S -> TRY_METHOD
S -> TRY_METHOD CURFEW_OPEN
S -> CATCH_METHOD
S -> CATCH_METHOD CURFEW_OPEN
S -> THROW_METHOD
S -> THROW_METHOD CURFEW_OPEN
S -> FINALLY_METHOD
S -> FINALLY_METHOD CURFEW_OPEN
S -> SWITCH_METHOD
S -> SWITCH_METHOD CURFEW_OPEN
S -> CASE_METHOD
S -> DEFAULT_METHOD
S -> LET_METHOD
S -> VAR_METHOD
S -> CONST_METHOD
S -> FUNCTION_CALLED
S -> CURFEW_OPEN
S -> CURFEW_CLOSE
S -> EXPRESSION
S -> UNER_OPERATION
UNER_OPERATION -> EXPRESSION_IN_PAREN UNER_OPERATOR
IF_METHOD -> IF IN_PAREN
ELSE_IF_METHOD -> ELSE IF IN_PAREN
ELSE_METHOD -> ELSE
WHILE_METHOD -> WHILE IN_PAREN
FOR_METHOD -> FOR IN_FOR_PAREN
BREAK_METHOD -> BREAK
CONTINUE_METHOD -> CONTINUE
FUNCTION_METHOD -> FUNCTION OBJECT IN_FUNC_PAREN
RETURN_METHOD -> RETURN
RETURN_METHOD -> RETURN EXPRESSION
TRY_METHOD -> TRY
CATCH_METHOD -> CATCH
THROW_METHOD -> THROW
THROW_METHOD -> THROW EXPRESSION_IN_PAREN
FINALLY_METHOD -> FINALLY

SWITCH_METHOD -> SWITCH IN_PAREN
CASE_METHOD -> CASE EXPRESSION_IN_PAREN COLON
DEFAULT_METHOD -> DEFAULT COLON
LET_METHOD -> LET LET_OBJECT
LET_OBJECT -> OBJECT ASSIGNMENT EXPRESSION_IN_PAREN
LET_OBJECT -> LET_OBJECT ASSIGNMENT LET_OBJECT
LET_OBJECT -> OBJECT
LET_OBJECT -> IN_PAREN
LET_OBJECT -> IN_PAREN OPERATOR LET_OBJECT
LET_OBJECT -> IN_PAREN COMMA LET_OBJECT
LET_OBJECT -> IN_PAREN ASSIGNMENT LET_OBJECT
VAR_METHOD -> VAR VAR_OBJECT
VAR_OBJECT -> PARAM
VAR_OBJECT -> PARAM ASSIGNMENT EXPRESSION_IN_PAREN
CONST_METHOD -> CONST PARAM ASSIGNMENT EXPRESSION_IN_PAREN
IN_FUNC_PAREN -> PAREN_OPEN PARAM PAREN_CLOSE
PARAM -> OBJECT
PARAM -> PARAM_ASSIGN
PARAM -> OBJECT COMMA PARAM
PARAM -> PARAM_ASSIGN COMMA PARAM
PARAM_ASSIGN -> DATA_TYPE ASSIGNMENT PARAM_ASSIGN
PARAM_ASSIGN -> DATA_TYPE ASSIGNMENT DATA_TYPE
IN_FOR_PAREN -> PAREN_OPEN EXPRESSION SEMICOLON EXPRESSION
SEMICOLON EXPRESSION PAREN_CLOSE
IN_PAREN -> PAREN_OPEN EXPRESSION_IN_PAREN PAREN_CLOSE
IN_PAREN -> PAREN_OPEN PAREN_CLOSE
IN_BRACKET -> BRACKET_OPEN EXPRESSION_IN_BRACKET BRACKET_CLOSE
IN_BRACKET -> BRACKET_OPEN BRACKET_CLOSE
EXPRESSION_IN_BRACKET -> EXPRESSION_IN_PAREN
EXPRESSION_IN_PAREN -> EXPRESSION
EXPRESSION_IN_PAREN -> EXP_COMPARATION
EXPRESSION -> EXP_ASSIGNMENT
EXPRESSION_IN_PAREN -> EXPRESSION SEPARATE_EXP
SEPARATE_EXP -> COMMA EXPRESSION_IN_PAREN
EXP_COMPARATION -> EXPRESSION COMPARATOR EXPRESSION
EXP_ASSIGNMENT -> EXPRESSION ASSIGNMENT EXPRESSION
EXP_ASSIGNMENT -> EXPRESSION ASSIGNOR EXPRESSION
EXPRESSION -> NUM
EXPRESSION -> BOOLEAN
EXPRESSION -> NUM DOT NUM

EXPRESSION -> IN_PAREN
EXPRESSION -> IN_BRACKET
EXPRESSION -> NULL
EXPRESSION -> OBJECT
EXPRESSION -> FUNCTION_CALLED
EXPRESSION -> OBJECT DOT FUNCTION_CALLED
EXPRESSION -> EXPRESSION COMPARATOR EXPRESSION
EXPRESSION -> EXPRESSION OPERATOR EXPRESSION
EXPRESSION -> LEFT_OPERATOR EXPRESSION
EXPRESSION -> CONDITION_TERNARY
CONDITION_TERNARY -> EXPRESSION TERNARY EXPRESSION COLON
EXPRESSION
DATA_TYPE -> OBJECT
DATA_TYPE -> NUM
DATA_TYPE -> BOOLEAN
DATA_TYPE -> NUM DOT NUM
DATA_TYPE -> FUNCTION_CALLED
DATA_TYPE -> NULL
FUNCTION_CALLED -> OBJECT IN_PAREN
FUNCTION_CALLED -> OBJECT IN_PAREN DOT FUNCTION_CALLED
IF -> 'IF'
ELSE -> 'ELSE'
WHILE -> 'WHILE'
FOR -> 'FOR'
BREAK -> 'BREAK'
CONTINUE -> 'CONTINUE'
FUNCTION -> 'FUNCTION'
RETURN -> 'RETURN'
TRY -> 'TRY'
CATCH -> 'CATCH'
THROW -> 'THROW'
FINALLY -> 'FINALLY'
SWITCH -> 'SWITCH'
CASE -> 'CASE'
DEFAULT -> 'DEFAULT'
LET -> 'LET'
VAR -> 'VAR'
CONST -> 'CONST'
TERNARY -> 'TERNARY'
CURFEW_OPEN -> 'CURFEW_OPEN'

```
CURFEW_CLOSE -> 'CURFEW_CLOSE'
PAREN_OPEN -> 'PAREN_OPEN'
PAREN_CLOSE -> 'PAREN_CLOSE'
BRACKET_OPEN -> 'BRACKET_OPEN'
BRACKET_CLOSE -> 'BRACKET_CLOSE'
SEMICOLON -> 'SEMICOLON'
COLON -> 'COLON'
COMMA -> 'COMMA'
DOT -> 'DOT'
BOOLEAN -> 'TRUE'
BOOLEAN -> 'FALSE'
NUM -> 'NUM'
OBJECT -> 'OBJECT'
NULL -> 'NULL'
COMPARATOR -> 'AND_LOP'
COMPARATOR -> 'OR_LOP'
COMPARATOR -> 'EQUAL_TO'
COMPARATOR -> 'VALUETYPE_EQUAL_TO'
COMPARATOR -> 'NOT_EQUAL_TO'
COMPARATOR -> 'VALUETYPE_NOT_EQUAL_TO'
COMPARATOR -> 'GREATER_EQUAL'
COMPARATOR -> 'GREATER'
COMPARATOR -> 'LESSER_EQUAL'
COMPARATOR -> 'LESSER'
ASSIGNMENT -> 'ASSIGNMENT'
ASSIGNOR -> 'PLUS_ASSIGNMENT'
ASSIGNOR -> 'MINUS_ASSIGNMENT'
ASSIGNOR -> 'MULTIPLY_ASSIGNMENT'
ASSIGNOR -> 'DIVIDE_ASSIGNMENT'
ASSIGNOR -> 'MODULO_ASSIGNMENT'
ASSIGNOR -> 'EXPONENT_ASSIGNMENT'
ASSIGNOR -> 'AND_ASSIGNMENT'
ASSIGNOR -> 'OR_ASSIGNMENT'
ASSIGNOR -> 'XOR_ASSIGNMENT'
ASSIGNOR -> 'AND_LOP_ASSIGNMENT'
ASSIGNOR -> 'OR_LOP_ASSIGNMENT'
ASSIGNOR -> 'NULLISH_ASSIGNMENT'
OPERATOR -> NORMAL_OP
OPERATOR -> BINARY_OP
UNER_OPERATOR -> 'INCREMENT_OP'
```

UNER_OPERATOR -> 'DECREMENT_OP'
NORMAL_OP -> 'PLUS_OP'
NORMAL_OP -> 'MINUS_OP'
NORMAL_OP -> 'MULTIPLY_OP'
NORMAL_OP -> 'DIVIDE_OP'
NORMAL_OP -> 'EXPONENT_OP'
NORMAL_OP -> 'MODULO_OP'
NORMAL_OP -> 'NULLISH'
LEFT_OPERATOR -> 'NEGATE_OP'
BINARY_OP -> 'SHIFTLEFT_OP'
BINARY_OP -> 'SHIFTRIGHT_OP'
BINARY_OP -> 'U_SHIFTRIGHT_OP'
BINARY_OP -> 'XOR_OP'
BINARY_OP -> 'AND_OP'
BINARY_OP -> 'OR_OP'
COMPARATOR -> 'AND_LOP'
COMPARATOR -> 'OR_LOP'
LEFT_OPERATOR -> 'NOT_LOP'

2.3 CNF dari CFG

S -> IF IN_PAREN
S -> ELSE_IF_METHOD0 IN_PAREN
S -> 'ELSE'
S -> 'ELSE'
S -> WHILE IN_PAREN
S -> FOR IN_FOR_PAREN
S -> 'BREAK'
S -> 'BREAK'
S -> 'CONTINUE'
S -> 'CONTINUE'
S -> FUNCTION_METHOD1 IN_FUNC_PAREN
S -> 'RETURN'
S -> 'RETURN'
S -> RETURN EXPRESSION
S -> 'TRY'
S -> 'TRY'
S -> 'CATCH'
S -> 'CATCH'
S -> 'THROW'
S -> 'THROW'

S -> THROW EXPRESSION_IN_PAREN
S -> 'FINALLY'
S -> 'FINALLY'
S -> SWITCH IN_PAREN
S -> CASE_METHOD2 COLON
S -> DEFAULT COLON
S -> LET LET_OBJECT
S -> VAR VAR_OBJECT
S -> CONST_METHOD10 EXPRESSION_IN_PAREN
S -> FUNCTION_CALLED35 FUNCTION_CALLED
S -> OBJECT IN_PAREN
S -> 'CURFEW_OPEN'
S -> 'CURFEW_CLOSE'
S -> EXP_ASSIGNMENT25 EXPRESSION
S -> EXP_ASSIGNMENT24 EXPRESSION
S -> 'NUM'
S -> 'FALSE'
S -> 'TRUE'
S -> PAREN_OPEN PAREN_CLOSE
S -> IN_PAREN21 PAREN_CLOSE
S -> BRACKET_OPEN BRACKET_CLOSE
S -> IN_BRACKET22 BRACKET_CLOSE
S -> 'NULL'
S -> 'OBJECT'
S -> FUNCTION_CALLED35 FUNCTION_CALLED
S -> OBJECT IN_PAREN
S -> CONDITION_TERNARY32 EXPRESSION
S -> EXP_ASSIGNMENT25 EXPRESSION
S -> EXP_ASSIGNMENT24 EXPRESSION
S -> 'NUM'
S -> 'FALSE'
S -> 'TRUE'
S -> PAREN_OPEN PAREN_CLOSE
S -> IN_PAREN21 PAREN_CLOSE
S -> BRACKET_OPEN BRACKET_CLOSE
S -> IN_BRACKET22 BRACKET_CLOSE
S -> 'NULL'
S -> 'OBJECT'
S -> FUNCTION_CALLED35 FUNCTION_CALLED
S -> OBJECT IN_PAREN

S -> CONDITION_TERNARY32 EXPRESSION
S -> LEFT_OPERATOR EXPRESSION
S -> EXPRESSION29 EXPRESSION
S -> EXPRESSION28 EXPRESSION
S -> EXPRESSION27 FUNCTION_CALLED
S -> EXPRESSION26 NUM
S -> EXPRESSION_IN_PAREN UNER_OPERATOR
ELSE_METHOD -> 'ELSE'
BREAK_METHOD -> 'BREAK'
CONTINUE_METHOD -> 'CONTINUE'
RETURN_METHOD -> 'RETURN'
TRY_METHOD -> 'TRY'
CATCH_METHOD -> 'CATCH'
THROW_METHOD -> 'THROW'
FINALLY_METHOD -> 'FINALLY'
LET_OBJECT -> 'OBJECT'
LET_OBJECT -> PAREN_OPEN PAREN_CLOSE
LET_OBJECT -> IN_PAREN21 PAREN_CLOSE
VAR_OBJECT -> 'OBJECT'
VAR_OBJECT -> PARAM_ASSIGN15 DATA_TYPE
VAR_OBJECT -> PARAM_ASSIGN14 PARAM_ASSIGN
VAR_OBJECT -> 'OBJECT'
VAR_OBJECT -> PARAM_ASSIGN15 DATA_TYPE
VAR_OBJECT -> PARAM_ASSIGN14 PARAM_ASSIGN
VAR_OBJECT -> PARAM13 PARAM
VAR_OBJECT -> PARAM12 PARAM
PARAM -> 'OBJECT'
PARAM -> PARAM_ASSIGN15 DATA_TYPE
PARAM -> PARAM_ASSIGN14 PARAM_ASSIGN
EXPRESSION_IN_BRACKET -> EXP_ASSIGNMENT25 EXPRESSION
EXPRESSION_IN_BRACKET -> EXP_ASSIGNMENT24 EXPRESSION
EXPRESSION_IN_BRACKET -> 'NUM'
EXPRESSION_IN_BRACKET -> 'FALSE'
EXPRESSION_IN_BRACKET -> 'TRUE'
EXPRESSION_IN_BRACKET -> PAREN_OPEN PAREN_CLOSE
EXPRESSION_IN_BRACKET -> IN_PAREN21 PAREN_CLOSE
EXPRESSION_IN_BRACKET -> BRACKET_OPEN BRACKET_CLOSE
EXPRESSION_IN_BRACKET -> IN_BRACKET22 BRACKET_CLOSE
EXPRESSION_IN_BRACKET -> 'NULL'
EXPRESSION_IN_BRACKET -> 'OBJECT'

EXPRESSION_IN_BRACKET -> FUNCTION_CALLED35 FUNCTION_CALLED
EXPRESSION_IN_BRACKET -> OBJECT IN_PAREN
EXPRESSION_IN_BRACKET -> CONDITION_TERNARY32 EXPRESSION
EXPRESSION_IN_BRACKET -> EXP_ASSIGNMENT25 EXPRESSION
EXPRESSION_IN_BRACKET -> EXP_ASSIGNMENT24 EXPRESSION
EXPRESSION_IN_BRACKET -> 'NUM'
EXPRESSION_IN_BRACKET -> 'FALSE'
EXPRESSION_IN_BRACKET -> 'TRUE'
EXPRESSION_IN_BRACKET -> PAREN_OPEN PAREN_CLOSE
EXPRESSION_IN_BRACKET -> IN_PAREN21 PAREN_CLOSE
EXPRESSION_IN_BRACKET -> BRACKET_OPEN BRACKET_CLOSE
EXPRESSION_IN_BRACKET -> IN_BRACKET22 BRACKET_CLOSE
EXPRESSION_IN_BRACKET -> 'NULL'
EXPRESSION_IN_BRACKET -> 'OBJECT'
EXPRESSION_IN_BRACKET -> FUNCTION_CALLED35 FUNCTION_CALLED
EXPRESSION_IN_BRACKET -> OBJECT IN_PAREN
EXPRESSION_IN_BRACKET -> CONDITION_TERNARY32 EXPRESSION
EXPRESSION_IN_BRACKET -> LEFT_OPERATOR EXPRESSION
EXPRESSION_IN_BRACKET -> EXPRESSION29 EXPRESSION
EXPRESSION_IN_BRACKET -> EXPRESSION28 EXPRESSION
EXPRESSION_IN_BRACKET -> EXPRESSION27 FUNCTION_CALLED
EXPRESSION_IN_BRACKET -> EXPRESSION26 NUM
EXPRESSION_IN_BRACKET -> EXP_COMPARATION23 EXPRESSION
EXPRESSION_IN_BRACKET -> EXP_ASSIGNMENT25 EXPRESSION
EXPRESSION_IN_BRACKET -> EXP_ASSIGNMENT24 EXPRESSION
EXPRESSION_IN_BRACKET -> 'NUM'
EXPRESSION_IN_BRACKET -> 'FALSE'
EXPRESSION_IN_BRACKET -> 'TRUE'
EXPRESSION_IN_BRACKET -> PAREN_OPEN PAREN_CLOSE
EXPRESSION_IN_BRACKET -> IN_PAREN21 PAREN_CLOSE
EXPRESSION_IN_BRACKET -> BRACKET_OPEN BRACKET_CLOSE
EXPRESSION_IN_BRACKET -> IN_BRACKET22 BRACKET_CLOSE
EXPRESSION_IN_BRACKET -> 'NULL'
EXPRESSION_IN_BRACKET -> 'OBJECT'
EXPRESSION_IN_BRACKET -> FUNCTION_CALLED35 FUNCTION_CALLED
EXPRESSION_IN_BRACKET -> OBJECT IN_PAREN
EXPRESSION_IN_BRACKET -> CONDITION_TERNARY32 EXPRESSION
EXPRESSION_IN_BRACKET -> EXP_ASSIGNMENT25 EXPRESSION
EXPRESSION_IN_BRACKET -> EXP_ASSIGNMENT24 EXPRESSION
EXPRESSION_IN_BRACKET -> 'NUM'

EXPRESSION_IN_BRACKET -> 'FALSE'
EXPRESSION_IN_BRACKET -> 'TRUE'
EXPRESSION_IN_BRACKET -> PAREN_OPEN PAREN_CLOSE
EXPRESSION_IN_BRACKET -> IN_PAREN21 PAREN_CLOSE
EXPRESSION_IN_BRACKET -> BRACKET_OPEN BRACKET_CLOSE
EXPRESSION_IN_BRACKET -> IN_BRACKET22 BRACKET_CLOSE
EXPRESSION_IN_BRACKET -> 'NULL'
EXPRESSION_IN_BRACKET -> 'OBJECT'
EXPRESSION_IN_BRACKET -> FUNCTION_CALLED35 FUNCTION_CALLED
EXPRESSION_IN_BRACKET -> OBJECT IN_PAREN
EXPRESSION_IN_BRACKET -> CONDITION_TERNARY32 EXPRESSION
EXPRESSION_IN_BRACKET -> EXP_ASSIGNMENT25 EXPRESSION
EXPRESSION_IN_BRACKET -> EXP_ASSIGNMENT24 EXPRESSION
EXPRESSION_IN_BRACKET -> 'NUM'
EXPRESSION_IN_BRACKET -> 'FALSE'
EXPRESSION_IN_BRACKET -> 'TRUE'
EXPRESSION_IN_BRACKET -> PAREN_OPEN PAREN_CLOSE
EXPRESSION_IN_BRACKET -> IN_PAREN21 PAREN_CLOSE
EXPRESSION_IN_BRACKET -> BRACKET_OPEN BRACKET_CLOSE
EXPRESSION_IN_BRACKET -> IN_BRACKET22 BRACKET_CLOSE
EXPRESSION_IN_BRACKET -> 'NULL'
EXPRESSION_IN_BRACKET -> 'OBJECT'
EXPRESSION_IN_BRACKET -> FUNCTION_CALLED35 FUNCTION_CALLED
EXPRESSION_IN_BRACKET -> OBJECT IN_PAREN
EXPRESSION_IN_BRACKET -> CONDITION_TERNARY32 EXPRESSION
EXPRESSION_IN_BRACKET -> LEFT_OPERATOR EXPRESSION
EXPRESSION_IN_BRACKET -> EXPRESSION29 EXPRESSION
EXPRESSION_IN_BRACKET -> EXPRESSION28 EXPRESSION
EXPRESSION_IN_BRACKET -> EXPRESSION27 FUNCTION_CALLED
EXPRESSION_IN_BRACKET -> EXPRESSION26 NUM
EXPRESSION_IN_BRACKET -> EXP_COMPARATION23 EXPRESSION
EXPRESSION_IN_BRACKET -> EXPRESSION SEPARATE_EXP
EXPRESSION_IN_PAREN -> EXP_ASSIGNMENT25 EXPRESSION
EXPRESSION_IN_PAREN -> EXP_ASSIGNMENT24 EXPRESSION
EXPRESSION_IN_PAREN -> 'NUM'
EXPRESSION_IN_PAREN -> 'FALSE'
EXPRESSION_IN_PAREN -> 'TRUE'
EXPRESSION_IN_PAREN -> PAREN_OPEN PAREN_CLOSE
EXPRESSION_IN_PAREN -> IN_PAREN21 PAREN_CLOSE
EXPRESSION_IN_PAREN -> BRACKET_OPEN BRACKET_CLOSE

EXPRESSION_IN_PAREN -> IN_BRACKET22 BRACKET_CLOSE
EXPRESSION_IN_PAREN -> 'NULL'
EXPRESSION_IN_PAREN -> 'OBJECT'
EXPRESSION_IN_PAREN -> FUNCTION_CALLED35 FUNCTION_CALLED
EXPRESSION_IN_PAREN -> OBJECT IN_PAREN
EXPRESSION_IN_PAREN -> CONDITION_TERNARY32 EXPRESSION
EXPRESSION_IN_PAREN -> EXP_ASSIGNMENT25 EXPRESSION
EXPRESSION_IN_PAREN -> EXP_ASSIGNMENT24 EXPRESSION
EXPRESSION_IN_PAREN -> 'NUM'
EXPRESSION_IN_PAREN -> 'FALSE'
EXPRESSION_IN_PAREN -> 'TRUE'
EXPRESSION_IN_PAREN -> PAREN_OPEN PAREN_CLOSE
EXPRESSION_IN_PAREN -> IN_PAREN21 PAREN_CLOSE
EXPRESSION_IN_PAREN -> BRACKET_OPEN BRACKET_CLOSE
EXPRESSION_IN_PAREN -> IN_BRACKET22 BRACKET_CLOSE
EXPRESSION_IN_PAREN -> 'NULL'
EXPRESSION_IN_PAREN -> 'OBJECT'
EXPRESSION_IN_PAREN -> FUNCTION_CALLED35 FUNCTION_CALLED
EXPRESSION_IN_PAREN -> OBJECT IN_PAREN
EXPRESSION_IN_PAREN -> CONDITION_TERNARY32 EXPRESSION
EXPRESSION_IN_PAREN -> LEFT_OPERATOR EXPRESSION
EXPRESSION_IN_PAREN -> EXPRESSION29 EXPRESSION
EXPRESSION_IN_PAREN -> EXPRESSION28 EXPRESSION
EXPRESSION_IN_PAREN -> EXPRESSION27 FUNCTION_CALLED
EXPRESSION_IN_PAREN -> EXPRESSION26 NUM
EXPRESSION_IN_PAREN -> EXP_COMPARATION23 EXPRESSION
EXPRESSION -> EXP_ASSIGNMENT25 EXPRESSION
EXPRESSION -> EXP_ASSIGNMENT24 EXPRESSION
EXPRESSION -> 'NUM'
EXPRESSION -> 'FALSE'
EXPRESSION -> 'TRUE'
EXPRESSION -> PAREN_OPEN PAREN_CLOSE
EXPRESSION -> IN_PAREN21 PAREN_CLOSE
EXPRESSION -> BRACKET_OPEN BRACKET_CLOSE
EXPRESSION -> IN_BRACKET22 BRACKET_CLOSE
EXPRESSION -> 'NULL'
EXPRESSION -> 'OBJECT'
EXPRESSION -> FUNCTION_CALLED35 FUNCTION_CALLED
EXPRESSION -> OBJECT IN_PAREN
EXPRESSION -> CONDITION_TERNARY32 EXPRESSION

DATA_TYPE -> 'OBJECT'
DATA_TYPE -> 'NUM'
DATA_TYPE -> 'FALSE'
DATA_TYPE -> 'TRUE'
DATA_TYPE -> FUNCTION_CALLED35 FUNCTION_CALLED
DATA_TYPE -> OBJECT IN_PAREN
DATA_TYPE -> 'NULL'
OPERATOR -> 'NULLISH'
OPERATOR -> 'MODULO_OP'
OPERATOR -> 'EXPONENT_OP'
OPERATOR -> 'DIVIDE_OP'
OPERATOR -> 'MULTIPLY_OP'
OPERATOR -> 'MINUS_OP'
OPERATOR -> 'PLUS_OP'
OPERATOR -> 'OR_OP'
OPERATOR -> 'AND_OP'
OPERATOR -> 'XOR_OP'
OPERATOR -> 'U_SHIFTRIGHT_OP'
OPERATOR -> 'SHIFTRIGHT_OP'
OPERATOR -> 'SHIFTLEFT_OP'
S -> IF_METHOD CURFEW_OPEN
S -> ELSE_IF_METHOD CURFEW_OPEN
S -> ELSE_METHOD CURFEW_OPEN
S -> WHILE_METHOD CURFEW_OPEN
S -> FOR_METHOD CURFEW_OPEN
S -> BREAK_METHOD CURFEW_CLOSE
S -> CONTINUE_METHOD CURFEW_CLOSE
S -> FUNCTION_METHOD CURFEW_OPEN
S -> RETURN_METHOD CURFEW_CLOSE
S -> TRY_METHOD CURFEW_OPEN
S -> CATCH_METHOD CURFEW_OPEN
S -> THROW_METHOD CURFEW_OPEN
S -> FINALLY_METHOD CURFEW_OPEN
S -> SWITCH_METHOD CURFEW_OPEN
UNER_OPERATION -> EXPRESSION_IN_PAREN UNER_OPERATOR
IF_METHOD -> IF IN_PAREN
ELSE_IF_METHOD -> ELSE_IF_METHOD0 IN_PAREN
ELSE_IF_METHOD0 -> ELSE IF
WHILE_METHOD -> WHILE IN_PAREN
FOR_METHOD -> FOR IN_FOR_PAREN

FUNCTION_METHOD -> FUNCTION_METHOD1 IN_FUNC_PAREN
FUNCTION_METHOD1 -> FUNCTION OBJECT
RETURN_METHOD -> RETURN EXPRESSION
THROW_METHOD -> THROW EXPRESSION_IN_PAREN
SWITCH_METHOD -> SWITCH IN_PAREN
CASE_METHOD -> CASE_METHOD2 COLON
CASE_METHOD2 -> CASE EXPRESSION_IN_PAREN
DEFAULT_METHOD -> DEFAULT COLON
LET_METHOD -> LET LET_OBJECT
LET_OBJECT -> LET_OBJECT3 EXPRESSION_IN_PAREN
LET_OBJECT3 -> OBJECT ASSIGNMENT
LET_OBJECT -> LET_OBJECT4 LET_OBJECT
LET_OBJECT4 -> LET_OBJECT ASSIGNMENT
LET_OBJECT -> LET_OBJECT5 LET_OBJECT
LET_OBJECT5 -> IN_PAREN OPERATOR
LET_OBJECT -> LET_OBJECT6 LET_OBJECT
LET_OBJECT6 -> IN_PAREN COMMA
LET_OBJECT -> LET_OBJECT7 LET_OBJECT
LET_OBJECT7 -> IN_PAREN ASSIGNMENT
VAR_METHOD -> VAR VAR_OBJECT
VAR_OBJECT -> VAR_OBJECT8 EXPRESSION_IN_PAREN
VAR_OBJECT8 -> PARAM ASSIGNMENT
CONST_METHOD -> CONST_METHOD10 EXPRESSION_IN_PAREN
CONST_METHOD9 -> CONST PARAM
CONST_METHOD10 -> CONST_METHOD9 ASSIGNMENT
IN_FUNC_PAREN -> IN_FUNC_PAREN11 PAREN_CLOSE
IN_FUNC_PAREN11 -> PAREN_OPEN PARAM
PARAM -> PARAM12 PARAM
PARAM12 -> OBJECT COMMA
PARAM -> PARAM13 PARAM
PARAM13 -> PARAM_ASSIGN COMMA
PARAM_ASSIGN -> PARAM_ASSIGN14 PARAM_ASSIGN
PARAM_ASSIGN14 -> DATA_TYPE ASSIGNMENT
PARAM_ASSIGN -> PARAM_ASSIGN15 DATA_TYPE
PARAM_ASSIGN15 -> DATA_TYPE ASSIGNMENT
IN_FOR_PAREN -> IN_FOR_PAREN20 PAREN_CLOSE
IN_FOR_PAREN16 -> PAREN_OPEN EXPRESSION
IN_FOR_PAREN17 -> IN_FOR_PAREN16 SEMICOLON
IN_FOR_PAREN18 -> IN_FOR_PAREN17 EXPRESSION
IN_FOR_PAREN19 -> IN_FOR_PAREN18 SEMICOLON

IN_FOR_PAREN20 -> IN_FOR_PAREN19 EXPRESSION
IN_PAREN -> IN_PAREN21 PAREN_CLOSE
IN_PAREN21 -> PAREN_OPEN EXPRESSION_IN_PAREN
IN_PAREN -> PAREN_OPEN PAREN_CLOSE
IN_BRACKET -> IN_BRACKET22 BRACKET_CLOSE
IN_BRACKET22 -> BRACKET_OPEN EXPRESSION_IN_BRACKET
IN_BRACKET -> BRACKET_OPEN BRACKET_CLOSE
EXPRESSION_IN_PAREN -> EXPRESSION SEPARATE_EXP
SEPARATE_EXP -> COMMA EXPRESSION_IN_PAREN
EXP_COMPARATION -> EXP_COMPARATION23 EXPRESSION
EXP_COMPARATION23 -> EXPRESSION COMPARATOR
EXP_ASSIGNMENT -> EXP_ASSIGNMENT24 EXPRESSION
EXP_ASSIGNMENT24 -> EXPRESSION ASSIGNMENT
EXP_ASSIGNMENT -> EXP_ASSIGNMENT25 EXPRESSION
EXP_ASSIGNMENT25 -> EXPRESSION ASSIGNOR
EXPRESSION -> EXPRESSION26 NUM
EXPRESSION26 -> NUM DOT
EXPRESSION -> EXPRESSION27 FUNCTION_CALLED
EXPRESSION27 -> OBJECT DOT
EXPRESSION -> EXPRESSION28 EXPRESSION
EXPRESSION28 -> EXPRESSION COMPARATOR
EXPRESSION -> EXPRESSION29 EXPRESSION
EXPRESSION29 -> EXPRESSION OPERATOR
EXPRESSION -> LEFT_OPERATOR EXPRESSION
CONDITION_TERNARY -> CONDITION_TERNARY32 EXPRESSION
CONDITION_TERNARY30 -> EXPRESSION TERNARY
CONDITION_TERNARY31 -> CONDITION_TERNARY30 EXPRESSION
CONDITION_TERNARY32 -> CONDITION_TERNARY31 COLON
DATA_TYPE -> DATA_TYPE33 NUM
DATA_TYPE33 -> NUM DOT
FUNCTION_CALLED -> OBJECT IN_PAREN
FUNCTION_CALLED -> FUNCTION_CALLED35 FUNCTION_CALLED
FUNCTION_CALLED34 -> OBJECT IN_PAREN
FUNCTION_CALLED35 -> FUNCTION_CALLED34 DOT
IF -> 'IF'
ELSE -> 'ELSE'
WHILE -> 'WHILE'
FOR -> 'FOR'
BREAK -> 'BREAK'
CONTINUE -> 'CONTINUE'

```
FUNCTION -> 'FUNCTION'
RETURN -> 'RETURN'
TRY -> 'TRY'
CATCH -> 'CATCH'
THROW -> 'THROW'
FINALLY -> 'FINALLY'
SWITCH -> 'SWITCH'
CASE -> 'CASE'
DEFAULT -> 'DEFAULT'
LET -> 'LET'
VAR -> 'VAR'
CONST -> 'CONST'
TERNARY -> 'TERNARY'
CURFEW_OPEN -> 'CURFEW_OPEN'
CURFEW_CLOSE -> 'CURFEW_CLOSE'
PAREN_OPEN -> 'PAREN_OPEN'
PAREN_CLOSE -> 'PAREN_CLOSE'
BRACKET_OPEN -> 'BRACKET_OPEN'
BRACKET_CLOSE -> 'BRACKET_CLOSE'
SEMICOLON -> 'SEMICOLON'
COLON -> 'COLON'
COMMA -> 'COMMA'
DOT -> 'DOT'
BOOLEAN -> 'TRUE'
BOOLEAN -> 'FALSE'
NUM -> 'NUM'
OBJECT -> 'OBJECT'
NULL -> 'NULL'
COMPARATOR -> 'AND_LOP'
COMPARATOR -> 'OR_LOP'
COMPARATOR -> 'EQUAL_TO'
COMPARATOR -> 'VALUETYPE_EQUAL_TO'
COMPARATOR -> 'NOT_EQUAL_TO'
COMPARATOR -> 'VALUETYPE_NOT_EQUAL_TO'
COMPARATOR -> 'GREATER_EQUAL'
COMPARATOR -> 'GREATER'
COMPARATOR -> 'LESSER_EQUAL'
COMPARATOR -> 'LESSER'
ASSIGNMENT -> 'ASSIGNMENT'
ASSIGNOR -> 'PLUS_ASSIGNMENT'
```

ASSIGNOR -> 'MINUS_ASSIGNMENT'
ASSIGNOR -> 'MULTIPLY_ASSIGNMENT'
ASSIGNOR -> 'DIVIDE_ASSIGNMENT'
ASSIGNOR -> 'MODULO_ASSIGNMENT'
ASSIGNOR -> 'EXPONENT_ASSIGNMENT'
ASSIGNOR -> 'AND_ASSIGNMENT'
ASSIGNOR -> 'OR_ASSIGNMENT'
ASSIGNOR -> 'XOR_ASSIGNMENT'
ASSIGNOR -> 'AND_LOP_ASSIGNMENT'
ASSIGNOR -> 'OR_LOP_ASSIGNMENT'
ASSIGNOR -> 'NULLISH_ASSIGNMENT'
UNER_OPERATOR -> 'INCREMENT_OP'
UNER_OPERATOR -> 'DECREMENT_OP'
NORMAL_OP -> 'PLUS_OP'
NORMAL_OP -> 'MINUS_OP'
NORMAL_OP -> 'MULTIPLY_OP'
NORMAL_OP -> 'DIVIDE_OP'
NORMAL_OP -> 'EXPONENT_OP'
NORMAL_OP -> 'MODULO_OP'
NORMAL_OP -> 'NULLISH'
LEFT_OPERATOR -> 'NEGATE_OP'
BINARY_OP -> 'SHIFTLEFT_OP'
BINARY_OP -> 'SHIFTRIGHT_OP'
BINARY_OP -> 'U_SHIFTRIGHT_OP'
BINARY_OP -> 'XOR_OP'
BINARY_OP -> 'AND_OP'
BINARY_OP -> 'OR_OP'
COMPARATOR -> 'AND_LOP'
COMPARATOR -> 'OR_LOP'
LEFT_OPERATOR -> 'NOT_LOP’

BAB III SPESIFIKASI TEKNIS PROGRAM

Program yang kami buat adalah program yang akan mengecek syntax bahasa Javascript berbasis Command Line Interface (CLI). Program kami menggunakan Context Free Grammar, Chomsky Normal Form, dan Cocke–Younger–Kasami serta FA yang telah dipelajari pada mata kuliah Teori Bahasa Formal dan Otomata. Kami telah mentranslasikan syntax Javascript dalam bentuk Context Free Grammar. Grammar tersebut kemudian diubah dalam bentuk CNF dengan menggunakan program <>. Setelah grammar dalam bentuk CNF dan algoritma CYK tersedia, maka proses parsing akan dimulai.

Program akan menerima sebuah file Javascript yang berisi kode-kode. Program kami akan membaca setiap line dalam kode tersebut dan mengecek apakah ada *error* dalam *syntax* dari kode-kode tersebut. Alur kerja program utama adalah sebagai berikut:

1. Pengecekan terhadap ada atau tidaknya file input sebelum dibuka, jika ada maka akan lanjut ke langkah 2
2. CFG akan di load dan diubah menjadi CNF lalu disimpan dalam suatu file txt.
3. File input akan per line dan setiap line akan diubah menjadi list berisi terminal
4. List terminal lalu dicek dengan menggunakan CYK
5. Jika CYK berhasil melakukan parsing terhadap list terminal, akan dilakukan handling terhadap kata-kata kunci keywords dan curly brackets dari codeline untuk menghindari kesalahan dalam sintaks antar line yang tidak dapat dikenali oleh CYK.
6. Jika file lulus uji sintaks, maka akan menampilkan pesan bahwa sintaks diterima. Sedangkan jika tidak, akan ditampilkan line letak kesalahan dan pesan bagaimana error terjadi.

BAB IV IMPLEMENTASI DAN PENGUJIAN

4.1 Fungsi-fungsi program dan headernya

4.1.1 Algoritma CYK

4.1.1.1 findRules(charItem)

Prosedur findRules mencari rules dari inputan string yang diterima

4.1.1.2 initArray(tWord)

Prosedur initArray menerima sebuah string (kalimat) dan membuat sebuah matriks berdasarkan jumlah kata

4.1.1.3 parseFirst(tWord,Array)

Prosedur parseFirst mengecek baris pertama dalam matriks

4.1.1.4 getDiag(array,possibleProductions,currentLength,currentHeight,length,i,diagList)

Prosedur getDiag mendapatkan item-item yang berada pada sel (dalam matriks) diagonal

4.1.1.5 getDown(array,possibleProductions,currentHeight,currentLength)

Prosedur getDown akan mendapatkan item-item menurun ke bawah mulai dari baris dimana pencacah berada

4.1.1.6
checkCombinations(array,possibleProductions,diagList,currentLength,currentHeight)

Prosedur checkCombination berfungsi untuk memproduksi antara diagList dengan possibleProduction

4.1.1.7 parse(tWord,array,i)

Prosedur parse akan mem-*parsing* matriks

4.1.2 Algoritma CFG menuju CNF

4.1.2.1 load_grammar_as_list(input string:filename)

Fungsi load_grammar_as_list menerima input nama file grammar yang akan dibaca. Setiap rule pada grammar di filename kemudian dibaca dan disimpan sebagai tuple dari left-hand side dan right-hand side. Tuple-tuple tersebut disimpan pada suatu list dan list inilah yang dikembalikan oleh fungsi ini.

4.1.2.2 is_terminal(input string:rule)

Fungsi is_terminal akan mengembalikan true jika rule berupa terminal dan false jika tidak.

4.1.2.3 insert_rule(rule)

Prosedur insert_rule akan memasukkan rule ke global dictionary CFG. Apabila LHS sudah ada pada dictionary CFG maka akan RHS akan ditambahkan sebagai value pada CFG. Jika LHS belum ada pada dictionary CFG, maka akan diinisialisasikan key LHS pada dictionary kemudian ditambahkan RHS sebagai value dari LHS.

4.1.2.4 prosedur load_grammar_as_cnf(input string:filename, <optional> input write_to_file)

Prosedur load_grammar_as_cnf menerima input filename. Grammar pada filename kemudian dibaca dan disimpan pada dictionary CFG sebagai global variable sesuai dengan Chomsky Normal Form, Apabila pengguna memasukkan write_to_file = True, maka hasil konversi grammar dalam bentuk CNF akan dituliskan dalam file .txt.

4.1.3 Algoritma FA

4.1.3.1 convertInputSymbol(char)

Menerima suatu karakter dan mengubahnya menjadi input simbol yang dapat diterima

oleh FA

4.1.3.2 transitions(currentState, inputSymbol)

Mengembalikan state yang dituju jika FA menerima currentState dan inputSymbol

4.1.3.3 checkWithDFA(varName, stringStack)

Mengecek apakah varName bisa diterima oleh suatu FA dan mengembalikan stringStack untuk mengecek apakah sedang di dalam string atau tidak

4.2 Pengujian

4.2.1 Testcase 1

```
// program to solve quadratic equation
let root1, root2;

// take input from the user
let a = prompt("Enterthefirstnumber");
let b = prompt("Enterthesecondnumber");
let c = prompt("Enterthethirdnumber");

// calculate discriminant
let discriminant = b * b - 4 * a * c;

// condition for real and different roots
if (discriminant > 0) {
    root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
    root2 = (-b - Math.sqrt(discriminant)) / (2 * a);

    // result
    console.log('Therootsofquadraticequationareroot1danroot2');
}

// condition for real and equal roots
else if (discriminant == 0) {
    root1 = root2 = -b / (2 * a);

    // result
    console.log("Therootsofquadraticequatioareroot1androot2");
}
```

Hasil Pengujian:

```
PS C:\Users\user\Documents\Kuliah\Semester 3 Jurusan\Teori Bahasa Formal & Automata\Tubes_TBFO2022\src> python main.py testcase2.js
Syntax accepted
```

4.2.2 Testcase 2

```
if (true) {
    print("salomo")
    let i = 0
    while (i < 2) {
        print(i)
        if (i == 3) {
            break
        }
        i++
    }
}
else {
    print("egi")
}
```

Hasil Pengujian:

```
PS C:\Users\user\Documents\Kuliah\Semester 3 Jurusan\Teori Bahasa Formal & Automata\Tubes_TBFO2022\src> python main.py testcase1.js
Syntax accepted
```

4.2.3 Testcase 3

```
try {
    switch (x) {
        case 1:
            print("oke")
            break
        case 2:
            print("notoke")
        default:
            print("baik")
            throw 10;
    }
}
catch
    print("this")
```

```
}
```

Hasil Pengujian:

```
PS C:\Users\user\Documents\Kuliah\Semester 3 Jurusan\Teori Bahasa Formal & Automata\Tubes_TBFO2022\src> python main.py testcase1.js
Syntax error at line 15
READED: CURFEW_CLOSE
Error : Curly bracket not closed
```

BAB V PENUTUP

1. Kesimpulan

Kami berhasil membuat program yang dapat mendeteksi kebenaran dari suatu sintaks bahasa pemrograman. Disini, kami membuat CFG berdasarkan sintaks bahasa pemrograman Javascript (Node.js) dan mengubahnya menjadi CNF. Setelah diubah menjadi CNF, bahasa tersebut di-*parse* dengan menggunakan CYK. Meskipun kompleksitas algoritma CYK membuat program bekerja lebih lambat, namun implementasinya sebenarnya lebih mudah untuk dimengerti dan dibuat. Banyak juga cara untuk merepresentasikan finite automata, salah satunya adalah dengan menggunakan array di dalam array seperti yang kami gunakan.

2. Saran

Diperlukan algoritma parsing yang lebih efektif dan efisien untuk mengecek sintaks dari suatu program dibandingkan CYK agar program utama dapat berjalan dengan lebih lancar. Selain itu, perlu pendalaman dan pencarian referensi juga agar dapat menghasilkan representasi FA yang lebih maksimal.

PEMBAGIAN TUGAS

NIM	Nama	Tugas
13521056	Daniel Egiant Sitanggang	CFG to CNF, Grammar
13521062	Go Dillon Audris	Main, FA, Grammar
13521063	Salomo Reinhart Gregory Manalu	CYK, Grammar

Link Github : https://github.com/GoDillonAudris512/Tubes_TBFO2022