David Joshua Estrera

Hans Matthew Abello

CSC617M - Machine Project CFG

Wikang Sawa (Python-Inspired Filipino Language)

The grammar G = (V, T, S, P) of our interpreter is defined by:

1. Variables (V) and Terminals (T)

Set	**Elements**
V (Variables)	<program>, <top_level_list>, <top_level_item>, <import_stmt>, <const_decl>, <struct_decl>, <var_decl_list>, <var_decl>, <data_type>, <main_block>, <func_decl>, <return_hint>, <param_list>, <param>, <statement_list>, <statement>, <assignment_stmt>, <io_stmt>, <if_stmt>, <elif_blocks>, <else_block>, <while_stmt>, <repeat_until_stmt>, <for_stmt>, <return_stmt>, <control_stmt>, <func_call_stmt>, <expression>, <simple_expr>, <term>, <factor>, <variable_access>, <access_modifier>, <call_suffix>, <args_list>, <relop>, <addop>, <mulop>, <identifier>, <literal>, <filename>, <bool_expression>
T (Terminals)	gamitin, talaan, depinisyon, pangunahin, baryabol, numero, desimal, bolyan, salita, wala, kung, kung_sakali, kung_hindi, pasa, habang, para, sa, ulitin, hanggang, tigil, tuloy, at, o, hindi, ipakita, basahin, konstante, =, :, ->, ., ,, (,), [,], +, -, *, /, %, ==, !=, <, >, <=, >=
Special Tokens	NEWLINE, INDENT, DEDENT
S (Start)	<program>

2. Productions (P)

A. Program Structure & Global Definitions

Non-Terminal	**Production Rule**
<program>	::= <top_level_list>
<top_level_list>	::= <top_level_item> <top_level_list>
	| ε
<top_level_item>	::= <import_stmt>
	| <const_decl>
	| <struct_decl>
	| <func_decl>
	| <main_block>
	| NEWLINE
<import_stmt>	::= gamitin <filename> NEWLINE
<const_decl>	::= konstante <identifier> = <literal> NEWLINE

B. User-Defined Types (Records)

Non-Terminal	**Production Rule**
<struct_decl>	::= talaan <identifier> : NEWLINE INDENT <var_decl_list> DEDENT

C. Main Function

Non-Terminal	**Production Rule**
<main_block>	::= pangunahin : NEWLINE INDENT

	<statement_list> DEDENT

D. Function Declaration

Non-Terminal	Production Rule
<func_decl>	::= depinisyon <identifier> (<param_list>) <return_hint> : NEWLINE INDENT <statement_list> DEDENT
<return_hint>	::= -> <data_type>
	| ε
<param_list>	::= <param> , <param_list>
	| <param>
	| ε
<param>	::= <identifier> : <data_type>

E. Variable Declaration

Non-Terminal	Production Rule
<var_decl_list>	::= <var_decl> <var_decl_list>
	| ε
<var_decl>	::= baryabol <identifier> : <data_type> = <expression> NEWLINE
	| baryabol <identifier> : <data_type> NEWLINE
	| <identifier> = <expression> NEWLINE
<data_type>	::= numero | desimal | bolyan | salita | <identifier>

F. Statements

Non-Terminal	Production Rule
<statement_list>	::= <statement> <statement_list>
	| <statement>
<statement>	::= <assignment_stmt>
	| <io_stmt>
	| <if_stmt>
	| <while_stmt>
	| <repeat_until_stmt>
	| <for_stmt>
	| <return_stmt>
	| <control_stmt>
	| <func_call_stmt> NEWLINE
	| pasa NEWLINE
<assignment_stmt>	::= <variable_access> = <expression> NEWLINE
<io_stmt>	::= ipakita (<expression>) NEWLINE
	| basahin (<variable_access>) NEWLINE

G. Control Flow

Non-Terminal	Production Rule
<if_stmt>	::= kung <bool_expression> : NEWLINE INDENT <statement_list> DEDENT <elif_blocks> <else_block>

<elif_blocks>	::= kung_sakali <bool_expression> : NEWLINE INDENT <statement_list> DEDENT <elif_blocks>
	| ε
<else_block>	::= kung_hindi : NEWLINE INDENT <statement_list> DEDENT
	| ε
<while_stmt>	::= habang <bool_expression> : NEWLINE INDENT <statement_list> DEDENT
<repeat_until_stmt>	::= ulitin : NEWLINE INDENT <statement_list> DEDENT hanggang <bool_expression> NEWLINE
<for_stmt>	::= para <identifier> sa <expression> : NEWLINE INDENT <statement_list> DEDENT
<return_stmt>	::= ibalik <expression> NEWLINE
	| ibalik NEWLINE
<control_stmt>	::= tigil NEWLINE | tuloy NEWLINE

H. Expressions

Non-Terminal	**Production Rule**
<expression>	::= <simple_expr> <relop> <simple_expr>
	| <simple_expr>
<simple_expr>	::= <simple_expr> <addop> <term>
	| <term>
<term>	::= <term> <mulop> <factor>
	| <factor>

<factor>	::= <identifier> <call_suffix>
	| <variable_access>
	| <literal>
	| (<expression>)
	| hindi <factor>
<variable_access>	::= <identifier> <access_modifier>
<access_modifier>	::= [<expression>] <access_modifier>
	| . <identifier> <access_modifier>
	| ε
<call_suffix>	::= (<args_list>)
<args_list>	::= <expression> , <args_list>
	| <expression>
	| ε
<relop>	::= == | != | < | > | <= | >=
<addop>	::= + | - | o
<mulop>	::= * | / | % | at