CS315 Project 1 Report

Group No: 39

Deniz Yüksel 21600880 Section 1

Faruk Oruç 21601844 Section 1

Name of the Language: GraBot

BNF Description:

<program> -> programstart <stmts> <funcs> programend | <empty>

<funcs> -> <func> | <funcs> <func>

<func> -> <funcname> (<parameters>) <action>

<action> -> <stmts>

<stmts> -> <stmt> | <stmts> <stmt>

<stmt> -> <loop_stmt> (@@ <comment>) | <logic_stmt> (@@ <comment>) | <assign_stmt>; (@@ <comment>)

<comment> -> <term>

<term> -> <flo> | <int> | <bool> | <str>

<str> -> <char> | <str> <char>

<char> -> <alph> | <num>

<num> -> 0 | 1 | ... | 9

<alph> -> a | b | ... | z | A | B | ... | Z

<bool> -> true| false

<int> -> <num> | <num> <num>

<flo> -> <int> . <int>

<assign_stmt> -> <var> <assop> <expr> | <var> <assop> <term> | <var> <assop> <var>

<var> -> <alph> | <alph> <str>

<assop> -> =

<expr> -> <term> | <term> <matOperator> <term> | <cond>

<cond> -> <term> <condOperator> <term>| <bool>

<condOperator> -> <lesssign> | <lessoreqsign> | <greateroreqsign> | <greatersign> | <equalsign>

<equalsign> -> ==

<greatersign> -> >

<greateroreqsign> -> >=

<lessoreqsign> -> <=

<lesssign> -> <

<matOperator> -> <plus> | <minus> | <mult> | <divide> | <mod>

<mod> -> %

<divide> -> /

<mult> -> *

<minus> -> -

<plus> -> +

<logic_stmt> -> <if_stmt>

<if_stmt> -> if (<cond>) { <stmts> } endif | if (<cond>) { <stmts> } endif else { <stmts> } endelse

<loop_stmt> -> <for_loop_stmt> | <while_loop_stmt>

<while_loop_stmt> -> while (<cond>) { <stmts> } endwhile

<for_loop_stmt> -> for (<expr>; <cond>; <expr>) { <stmts> } endfor

<parameters> -> <parameter> | <parameters> <parameter>

<parameter> -> <direction> | <rotation> | <sensorID>

<sensorID> -> <frontSensor> | <backSensor>

<backSensor> -> B1 | B2 | B3 | B4

<frontSensor> -> F1 | F2 | F3 | F4

<rotation> -> left | right

<direction> -> forward | backward

<funcname> -> move | turn | grab | release | senseDist | switchLight | str

Non – Terminal Symbols:

<program>: Our program consists of statememts and functions. They are labeled as <stmts> and <funcs>. We do not have an obect oriented design. Our language just consists of statements and functions that can be given as an order to a simple robot with two arms on the back and forward in total. A following program is written in our language if the program is empty.

<funcs>: Functions are composed of one of many <func>.

<func>: A func is named after the traditional name, “function” in C and C++ language. They have to contain a function name, a list of parameters between left and right paranthesis, and an action in their function bodies.

<action>: Action name is given to this token because in the block of functions, actions are performed in general. It fits the purpose of readability and writability through common sense and intuition.

<stmts> Statements consist of one or many <stmt>.

<stmt>: A statement can either be a loop statement, logic statement, or a comment. The first three items can include comments after their termination in the same line. In order words, comments can be placed after the statements in the same line, so the comments do not change the meaning of the program.

<term>: It is either <flo>, <int>, <bool> or <str>. The definitions of these names are given below.

<str> -> A string is either a single character, or is composed of multiple characters.

<char> ->: A char is either an alphabetic letter or a digit.

<num> -> Num name is given for the numbers between 0 and 9 including 0 and 9. It's readability is a bit unsatisfied in terms of the dictionary language of number and digit. Naming this <digit> could be a better option because a digit only consists between numbers 0 and 9, both included. <num> can sound ambiguous but is still correct.

<alph> -> Alph is the abbreviation for alphabet. It is short yet is readable. This name represents the upper and lowercase letters of the alphabet.

<bool>: Bool name is given after Geoge Bool who literally found its name. We decided to respect George Bool and use his definition as other computer languages. The bool values, naturally are true or false.

<int>: Integer name is named "int" as the traditional way. It is composed of a number which corresponds to digit in our language, or multiple numbers. A number is represented as <num>.

<flo>: The float name type consists of two integers seperated with a dot. It is named "flo" for the ease of writability.

<assign_stmt>: Assignment statements consist of variables assigned to an expression with the assignment operator which is "assop" in our language. They can also be in the composition of a variable assigned to a termor a variable assigned to another variable. It is written as "assign_stmt" because it is a statement.

<var>: This name is given after the conventional variable, which can be composed of alphanumerics. However the first character cannot be a number.

<assop>: This name is given after the traditional assignment operator in C group languages. It is written as "assop" for writability, and yet is is readable.

<expr>: Expressions consist of <term> names which are float, integer, strings. Or, a term is used with math operators with association to another term.

<matOperator>: Mathematical Operators consist of <mod>, <divide>, <mult>, <minus>, <plus>

<mod>: The traditional mod operator (%) is used in our language. We decided to include it because it is an essential mathematical operation for a computer language.

<divide>: The traditional divide operator (/) is used intuivitely so that our language has common sense through writability.

<mult>: The traditional divide operator (*) is used intuivitely so that our language has common sense through writability.

<minus>: The traditional divide operator (-) is used intuivitely so that our language has common sense through writability.

<plus>: The traditional divide operator (+) is used intuivitely so that our language has common sense through writability.

<logic_stmt>: Logic statements are if statements either with or without an else statement.

<if_stmt>: The traditional if statement is named as <if_stmt> in order to depict it is “if” and it is a statement. The word “stmt” is chosen to recall the word statement. By this way, it is more writable, and still readable.

<cond>: Condition name is given for the occasions that occur inside if statements, or they can be boolean values which are true or false. The boolean expressions are used inside conditions.

<condOperator>: This name consists of <equalsign>, <greatersign>, <greateroreqsign>, <lessoreqsign>, <lesssign> .

<equalsign>: This token is named intuitively after the conventional less sign which is “==”.

<greatersign>: This token is named intuitively after the conventional less sign which is “<”.

<greateroreqsign>: This token is named intuitively after the conventional less sign which is “>=”.

<lessoreqsign>: This token is named intuitively after the conventional less sign which is “<=”.

<lesssign>: This token is named intuitively after the conventional less sign which is “<”.

<loop_stmt>: This name consists of for or while loop statements.

<while_loop_stmt>: This name is for the “while” of the while loop. It is named while_loop_stmt for readability because “while” is a reserved word for the while loop in our language, and while loops are statements.

<for_loop_stmt>: This name is for the “for” of the for loop. It is named for_loop_stmt for readability because “for” is a reserved word for the for loop in our language, and for loops are statements.

<parameters>: Parameters consist of one or more <parameter>. It is named parameter for readability purposes.

<parameter>: Parameter name is given for the proper use of functions. A value between left and right paranthesis is a parameter.

<sensorID>: Sensor ID is composed of <backSensor> and <frontSensor>.

<backSensor>: This name’s origin comes from designs of cars. A car has 4 back sensor that activates when the gear is R. It is in order to detect 180 degrees in the backside. It is named backSensor for readability purposes.

<frontSensor>: This name is also originated from the sensors of a car. A car has 4 front sensors which cover 45 degrees each to have a better view of the frontline. Although its name is long, it’s for the sake of readability.

<rotation>: This name is used for the action of turning of this robot. It is named rotation after the policy of readability. It is either left of right.

<direction>: This name is used for the direction of where the robot goes. It is named direction for readability purposes. It is either forward of backward.

<funcname>: This name is for the functions that are reserved which are move, turn, grab, release, senseDist, switchLight.

Non – Trivial Tokens:

<comment>: A comment is labeled with a @@ in our language. We decided not to use // like the traditional comment since we believed that // decreases readability in case when the user writes / at the beginning of the comment. @ symbol is much more rare in programming languages and comments compared to /.

Reserved Words:

for & endfor : These two indicate the start and the end of a for loop

if & endif : These two indicate the start and the end of a if statement

else & endelse: These two indicate the start and the end of a else statement which coexist with if statement

while & endwhile: These two indicate the start and the end of a while loop

true & false: These two indicate whether the variable or a case is true or false

; : Indicates end of a statement

forward & backward : These two indicate the direction the robot is moving (only used in move function)

move(<direction>) : This predefined function moves the robot 1mm in desired direction

left & right: These two indicate the rotation the robot will make in the turn function

turn(<rotation>): This predefined function turns the robot 1 degree in the desired way

switchLight(str): This predefined function takes a single str as parameter. If the str is “on” it will turn on the lights, if it is “off” it will turn of the lights, otherwise it will not do anything.

F1 & F2 & F3 & F4 & B1 & B2 & B3 & B4: Since the design of the robot has 8 sensors (4 on the front side and 4 on the back side) each of these reserved words indicate a single sensor on the robot.

senseDist(<backSensor> | <frontSensor>): This predefined function gets the distance from an object in millimeters as flo values (x.y).

Identifiers:

As predefined, identifiers, or variables in our language, are defined without specifying their type (str, int etc.). This causes program to be less readable since the reader wouldn’t be able to understand what is the type of the variable but it makes the program much more writable. They are simply given a name which starts with an alphabet letter and continues with any number or alphabet character for their definition. The convention for identifiers is the first word is all lowercase letters and the following words are written without space and their first letter is uppercase like in Java. For instance, in our language a variable is defined as whileCounter = 0;