MAlice Language Specification

Suhaib Sarmad, Alex Appetiti, Jamal Khan

October 2010

1 Lexical Structure

1.1 White Space and Line termination

White space is ignored and a new line is terminated with a full stop.

1.2 Keywords

The keywords that cannot be used in the program are:

was | a | number | became | too | found | then | there | drank | ate | but | and | letter

2 Backus-Naur Form Grammar

A context free grammar has terminal symbols on the left-hand side and sequences of nonterminals on the right-hand side.

2.1 Expressions

<statement-list>	::=	<statement> <statement-list> |
		<return>
<return>	::=	'Alice found' <exp> '.'
<exp>	::=	<exp> '|' <xors> | <xors>
<xors>	::=	<xors> '^' <ands> | <ands>
<ands>	::=	<ands> '&' <sums> | <sums>
<sums>	::=	<sums> '+' <terms> |
		<sums> '-' <terms> | <terms>
<terms>	::=	<terms> '*' <nots> |
		<terms> '/' <nots> |
		<terms> '%' <nots> | <nots>
<nots>	::=	'~' <nots> | <factor>
<factor>	::=	<const> | <ident>
<statement>	::=	<declaration> | <assignment> | <step>
<declaration>	::=	<ident> 'was a' <type> <seperator> |
		<ident> 'was a' <type> 'too' <seperator>
<assignment>	::=	<ident> 'became' <exp> <seperator> |
		<ident> 'became' <char> <seperator>
<step>	::=	<ident> 'ate' <seperator> |
		<ident> 'drank' <seperator>
<seperator>	::=	'and' | 'but' | 'then' | ',' | '.'
<type>	::=	'number' | 'letter'

2.2 Types

MAlice has two primitive types which are numbers and letters. Numbers are 32-bit whilst letters are 8-bit.

2.3 Semantics

A declaration, marked by the keywords 'was a', assigns a type to an identifier and allows for it to be assigned to a value.

An assignment, marked by the keyword 'became', assigns a value to a previously declared identifier.

All statements (i.e. declarations or assignments) must be separated by a separator (i.e. one of "but, and, then, '.', ',').

For a program in MAlice to be valid, the aforementioned BNF along with the following rules must be respected:

- The programmer may not cast from a number to a letter and viceversa, all operations involving a mix of the two will result in an error.
- Each program must include a return statement.
- Before applying any operations on any variables (including 'ate' or 'drank') said variables must have been previously declared and assigned an initial value.
- An assignment cannot occur unto a variable that has not been previously declared.
- The same variable cannot be declared twice.
- A declaration can be followed by the keyword 'too' only if a variable has been declared immediately before it, and said variable was of the same type.
- All identifiers must start with a letter and must contain only alphanumeric characters and may include the underscore.

When a MAlice program attempts to execute an invalid instruction (eg. division by zero) its subsequent behaviour is undefined.

2.4 Operators

The following are operators used by the MAlice Language in decrementing precedence.

	Operation	Symbol	Example	Level of Precedence
Unary:	Bitwise Negation	~	~213=42	0
Binary:	Modulus	%	87%45=42	1
	Integer Division	/	128/3=42	1
	Multiplication	*	6*7=42	1
	Subtraction	-	47-5=42	2
	Addition	+	23+19=42	2
	Bitwise AND	&	187&110=42	3
	Bitwise XOR	^	153&179=42	4
	Bitwise OR	|	34|8=42	5

As mentioned above, all operations are done in 32-bits, but depending on the operating environment, the return value of a MAlice program may be truncated. For example, on a 32-bit Unix system, only the last 8 bits (the 8 LSBs) are returned.

As a consequence, the maximum value that MAlice can return is equal to $2^s - 1$ where 's' is equal to the number of bits used by the operating environment as an exit code. Any attempt to go over this value or "under 0" will result in the answer being truncated (which mathematically is equivalent to a $\%2^s$ operation). As a result, MAlice's return values do not support negative numbers.