Columbia University

Computer Science Department

COMS W4115 Programming Languages and Translators

MatCab: Matrix Manipulation Language

Author:
Cheng Xiang (cx2142)
Yu Qiao (yq2145)
Ran Yu (yr2239)
Tianchen Yu (ty2275)

Professor:
Stephen A. Edwards

October 31, 2012

Contents

1 Introduction

The MatCab is a programming language that simplifies and accelerates matrix manipulations. It is a C-style language that particularly aims at easier and faster matrix computing for programmers. It provides intuitive matrix related operators and will take the advantage of GPU to speed up the program run-time performance. This manual is inspired by the C reference manual [Dennis M. Ritchie, 1973]. and CLAM reference manual [Jeremy, Robert, Kevin and Yongxu, 2011].

2 Lexical Conventions

There are six kinds of tokens: comments, identifiers, keywords, constants, strings, and operators. Comments, blanks, tabs, tabs, new lines and are ignored. The token is recognized until the parser sees a separator.

2.1 Comments

A line that begins with a pound sign (#) is comments. The compiler will ignore the comments. Only single-line comment is supported.

#This is a line of comment.

2.2 Identifiers

An identifier is used to distinguish one variable or function to another. It is sequence of letters and numbers, and it must start with a letter. The identifiers are case-sensitive, i.e. "move" and "Move" are different variables.

Identifier → *[a-zA-Z]([a-zA-Z_] | [0-9])**

2.3 Keywords

A keyword is reserved and cannot be used as an identifier.

Keyword → *int* | *char* | *float* | *rowvec* | *colvec* | *matrix* | *return* | *break* | *continue* | *if* | *else* | *for* | *while* | *entry* | *true* | *false* | *import* | *export*

2.4 Constants

There are four kinds of constants in Matcab: numeric constants, string literal constants, vector constants, and matrix constants.

2.4.1 Numeric Constants

A numeric constant can be an integer constant, a float constant. The integer is a series of numbers. The float constant contains an integer part, a decimal point, a fraction part and a character 'f'.

Numeric Constant → Integer Constant | Float Constant
Integer Constant → [0-9]+
Float Constant → [0-9]+ . [0-9]+ 'f'

2.4.2 String Literal Constants

A string literal constant is anything inside a pair of double quote marks.

String Literal → "(any character)"

2.4.3 Vector Constants

A vector constant can be a row vector or a column vector. A pair of square brackets demarcates them both. A row vector is ended with a character 'râĂŹ while a column vector is ended with a character 'câĂŹ. For example, [3,4,5]r is a row vector.

Vector Constant → Row Vector Constant | Column Vector Constant Row Vector Constant → open_square_bracket close_square_bracket 'r' | open_square_bracket (Numeric Constant ,)+ Numeric Constant close_square_breacket 'r' Column Vector Constant → open_square_bracket close_square_bracket 'c' | open_square_bracket (Numeric Constant ,)+ Numeric Constant close_square_breacket 'c'

2.4.4 Matrix Constants

A matrix constant begins with an open square bracket and ends with a close square bracket, just like the vector constant. It can be filled with several row vectors, column vectors, or numbers. For example, [(1,2,3);(4,5,6);(7,8,9)] gives a 3*3 matrix with values specified; [c1;c2;c3] gives a matrix with column vector specified; [r1;r2;r3] gives a matrix with row vector specified.

Matrix Constants → open_square_bracket (comma separated numbers) close_square_bracket*

3 Syntax Notations

Because MatCab supports both number and matrix operations, we use uppercase letter and lowercase letter to distinguish them in the following discussion - Any uppercase character denotes a matrix, and any lowercase denotes a number.

4 Types

There are two types can either be an atomic type or a compound type.

4.1 Atomic Types

As a support or a complement language to other general programming languages, like C language, which focuses on matrix manipulation, MatCab supports a subset of arithmetic types defined in C programming language.

char
int
float

4.2 Compound Types

MatCab supports compound types built upon atomic types.

rowvec
colvec
matrix

Users cannot define new data types in MatCab.

5 Expressions

Here the definition of MatCab expressions follows the steps in C Language Reference Manual defining expressions.

5.1 Primary Expressions

An identifier, a constant, a string (an array of chars), an expression enclosed in parentheses, a primary expression followed by an expression in square brackets or a function call (a primary expression followed by parentheses containing possibly empty, as well as possibly comma separated list of expressions) could be a primary expression.

Identifier
Constant
String
(expression)
Primary-expression [expression]
Primary-expression (expression-list)

5.2 Unary operators

There are several kinds of unary operators in MatCab:

+ expression
- expression
! expression
expression ˜
expression ˜.
expression ’
| expression |
tr(expression)

! operator will make the entire expression of value 0, if the operand is of value other than 0; and will be of value 1 if the operand is of value 0.

˜ and ˜. operator performe matrix inverse calculation. As the convention of MatCab, if ˜. operator is chosen by the programmer, the calculation will use GPU to accelerate.

‘ operator gives the transpose of the original expression.

| A | operator calculates the determinant of the given expression.

tr() operator calculates the trace of the given expression.

5.3 Binary operators

expression + expression
expression - expression
expression * expression
expression / expression
expression
expression *. expression
expression ** expression
expression == expression
expression || expression
expression && expression

*. operator will performe matrix multiplication using GPU acceleration. ==, || and && operators are used in flow control.

5.4 Other operators

Besides unary and binary operators, MatCab supports some useful matrix calculations using such operators:

submat(expression, expression, expression, expression, expression)

sum(expression, expression, expression)

submat operator gives out a submatrix of the original expression according to the parameters assigned to. sum operator gives the sum of the elements in a row or a column of a given expression.

More matrix operators are to be added along the development of MatCab.

5.5 I/O Expressions

MatCab take in the input and give out the output using I/O expressions defined within the language itself.

Matrix-identifier = Import (filename)

Export (filename, matrix-identifier)

6 Declarations

Declarations are used to associate an identifier and its data type. It tells the interpreter how to treat each identifier properly. All variables must be firstly explicitly declared before it can be used in a statement. Declarations could appear at any place of a MatCab program. It has the form:

declaration:
    Type-specifier declarator-list
declarator-list:
    First-declarator
    First-declarator, declarator-list
first-declarator:
    Declarator initializer
declarator:
    Identifier

Declarator ()
Declarator [constant-expression]
Initializer:
Constant
constant-expression-list

7 Statements

Statements end with semicolons in MatCab, just like what it is like in C language. Most of the time, they are expression statements. If needed, statements can be grouped together using curly bracket pair. For flow control purpose, *if* statements, *while* statements, *for* statements are supported in MatCab. Their grammar rules are exactly the same as C language.

8 Program Definition

A MatCab program consists of several definition statements and exactly one entry point. The entry point itself is a function declaration with a specified form:

Program → EntryPoint FunctionDecl*
EntryPoint → **int entry** () Statement
FunctionDecl → Type **id** FormalList
FormalList → Type **id** FormalRest*
FormalRest →, Type **id**
Type → AtomicType
→ CompoundType
AtomicType → **char**
→ **int**
→ **float**
CompoundType → rowec<AtomicType>
→ columnvec<AtomicType>
→ matrix<AtomicType>
Statement → VarDecl*
→ Statement*
→ **if** (Exp) Statement **else** Statement
→ **while** (Exp) Statement
→ **for** (Exp ; Exp ; Exp) Statement*
→ **id** = Exp ;
→ **id** [Exp]= Exp ;
→ **id** = **import** (**id**);

→ **export** (**id**);
Exp → (Exp)
→ [Exp]
→ Exp BInOp Exp
→ LhsUnaryOp Exp
→ Exp RhsUnaryOp
→ | Exp |
→ | tr (exp) |
→ **submat** (Exp+)
→ **sum** (Exp+)
→ **Constant**
→ **true**
→ **false**
→ **id**
BinOp → +
→ -
→ *
→ /
→ %
→ *.
→ **
LhsUnaryOp → +
→ -
→ !
RhsUnaryOp → ˜
→ ˜.
→ ’
VarDecl → Type **id**

9 How Scope Rules Work in MatCab

Identifiers are only valid in the scope that enclosed by the nearest curly bracket pair. If an identifier is declared out of any curly bracket pairs, then it is a global variable that accessible from anywhere of the program.

10 Example

```
// Sample program Matrix Multiplication
int entry() {
```

```
    int THRES = 100;
    matrix A = import(``matrix1.txt'')f;
    matrix B = import(``matrix2.txt'');
    int ma, mb, na, nb;
    ma = rows_count(A);
    mb = rows_count(B);
    na = columns_count(A);
    nb = columns_count(B);
    if (na == mb)
    {
        matrix C = new matrix(ma, nb);
        If (ma > THRES || na > THRES || nb > THRES)
            C = A *. B;
        else
            C = A * B;
        export(``result.txt'', C);
    }
}
```

References

[Dennis M. Ritchie, 1973] Dennis M. Ritchie (1973). C Reference Manual. *Bell Telephone Laboratories*

[Jeremy, Robert, Kevin and Yongxu, 2011] Jeremy Andrus and Robert Martin and Kevin Sun and Yongxu Zhang (2011). CLAM: The Concise Linear Algebra Manipulation Language. *Columbia University*