COMS W4115

Programming Languages and Translators

ASML: White Paper

09/23/2003

Davidov, Olga
olga.davidov@riag.com

ASML:

ARRAY SORTING AND MANIPULATION LANGUAGE

Introduction

ASML is designed to be a simple, intuitive, and efficient language for performing mathematical manipulations on arrays. A number of existing broader domain languages and tools, such as C, C++, Java, Matlab, provide array manipulation functionality, yet many useful operations, such as sorting, shuffling, shifting are not given by these languages, and users need to either use libraries or define these operations themselves. Furthermore, such a universal tool as Matlab is oriented at a very broad domain – it deals with practically every possible aspect of mathematics. Consequently, the amount of information a user has to absorb to learn and use such a tool is huge. While being very effective in what it does and very useful in many circumstances, it is an overkill for a user who's interested only in array-related functionality. ASML will provide most of the array manipulation functionality given in Matlab, plus add more useful operations that deal with arrays.

Simple

ASML will be a simple, easy to learn, and intuitive language. It provides most of the intuitive and easy-to-understand functionality in Matlab dealing with arrays and adds a number of extra useful array-manipulation operations. ASML also provides a number of most common control flow constructs, such as loops and conditionals. ASML will be easy to learn, have easy, self-explanatory syntax. It should be readable and understandable to anyone familiar with array manipulation functionality.

Robust

Programs written in ASML are robust and reliable by virtue of simplicity of the language design. Namely, all numbers in the program are automatically treated as floating point numbers. So there are only two types of variables in the language: real numbers and arrays of real numbers. This allows preventing many error-prone situations due to type conversions. Additionally, there is extensive size checking in most of the operations. For example, if a user attempts to add two arrays of a different size, an error message will be issued.

Architecture Neutral and Portable

ASML will be an interpreted language, with its interpreter written in Java, which makes ASML architecture neutral. ASML programs can be easily seamlessly ported to any platform that has JVM running on it.

Interpreted

ASML will have a number of very high-level operation-intensive manipulations implemented as built-in functions. This justifies the need for this language to be interpreted. Another advantage of being interpreted is that commands can be evaluated and executed on the fly in the command-line interactive tool such as Matlab command-line interpreter. Commands in ASML can be executed either by calling ASML interpreter and passing it a text file with an ASML program or by calling the interpreter without arguments and executing each instruction one by one on the interpreter's command line.

High Level

ASML will implement a number of high-level operation-intensive built-in functions, such as array sorting via different algorithms, such as bubble sort, insertion sort, heap sort, selection sort, also random shuffling, shift, and other non-linear mathematical functions, such as logarithm and exponent. These high-level operations will be very useful to someone wishing to verify the end result of a complex operation implemented by hand, or say, compare the execution time of various sorting algorithms.

Model of computation

Programs written in ASML will be analyzed and interpreted by ANTLR, translated to Java code and then executed by JVM. Lexer, parser, and tree walker will be generated using ANTLR.

Data types

There are 2 data types in ASML: real numbers and arrays of real numbers. All numerals in the ASML program will be automatically cast to floating point numbers. The main concern of a user of ASML should be not on programming and typing issues, but on getting back the result of whatever operation the user is trying to perform.

Scope

Similar to typing decision, handling of scopes in ASML should be simple. As with scripting languages such as perl and shell, a variable is automatically declared the first time it is encountered in the program, and is seen from that point on. Obviously, this opens a door for hardly catchable debugging problems, but in this case simplicity of the program's syntax weighs more than ease of debugging.

Error Handling

Al the lexical and syntactic errors will be caught by the lexer and parser constructed by ANTLR from the lexer and parser grammars. The translator will check the sizes of the arrays used in per-element operations, such as addition of two arrays and report an error if the sizes do not match. Also, an error will be reported if a scalar is being used where an array variable is expected, and vice versa.

Control Flow Structures

ASML will support most common control flow structures: looping constructs – while loop and for loop; conditionals – if-then-else and (possibly) switch statement.

Example of Syntax

Here is a sample program in ASML:

```
a = [1, 2, 3, 4, 5, 6];
for (i=1; i<10; i++)
{
     a = a*i;
}
b = rand[1:6];
c = a.b;
shiftleft(c, 5);
quicksort(c);
c
```

In this program we first initialize array a. Then a is successively multiplied by numbers 1 through 9 and the result is assigned to a. Then we initialize array B with 6 random numbers. Array c is created by concatenating elements of a and b. Then we shift c left by 5 positions (circular shift). Finally, we sort elements of c using quicksort and print c.

Note that similar to Matlab, if an expression is followed by a semicolon, its value is not printed out, otherwise the value of expression is printed.

Possible extensions

Time permitting, the following extensions will be added to the language:

- I/O: users will be able to read input from text files and save output to files.
- User-defined functions: users will be able to define their own functions. This is an essential feature of a language, since it allows for extensibility, reusability, and modularity.

Conclusion

ASML will be a simple, efficient, narrow-domain language for array sorting and manipulation. Using intuitive, self-explanatory syntax it will allow users with little programming experience to write concise, high level programs that perform fairly complex mathematical manipulations on arrays.