Functional Programming languages and its applications in Mathematics - A Survey

Shobhit Mani Tiwari
Assistant Professor
Department of Computer Science & Engineering
Email: smtewari.u2@gmail.com

Alok Kr Gupta
Assistant Professor
Babu Banarasi Das Group of Education
Lucknow, Uttar Pradesh
Email:alokgupta7apr@gmail.com

Abstract—Functional programming languages were originally developed in late 1970's. With its root in Lambda calculus and combinatory calculus this proved to be the best choice for mathematicians,.Its applications were visible in areas varying from differential calculus to rough set theory. The main aim was to provide compact notion for writing program mainly for symbolic computation and applications based on list-processing. The main concern lies in functionality and not storage and assignment sequences, it also provides features like lazy evaluation, higher order functions and pattern matching. This paper is a brief survey on the impact of functional programming on current research in the field of mathematics. We took many papers and analyzed the importance of these languages in shaping the modern mathematics. .

I. Introduction

Programming languages were written keeping in mind the need for communication between machine and humans. From classical assembly languages to high level languages we had saw huge differences in the way we wrote their syntax. Earlier their used to be machine dependent syntax and operations which were replaced by architecture dependent syntax. Later we saw the languages which were platform and architecture independent as well. With the advent of C++ , i.e object oriented programming (OOP) we were able to write programs with lesser complicated and lesser confusing syntax.

Functional Programming have a well defined semantics , they are written clearly and have implicitly defined flow of control. It appears complicated to someone who is unaware of this concept but on the other hand it is concise and self explanatory for one who knows the ins and out of this model. The lazy evaluation techniques helps in describing mathematical objects which in turn helps in implementing things in quantum mechanics so that scientists can use them. This model has helped mathematicians believe that computers were made out of mathematics and now they can help mathematics to solve the problems which were once unsolvable. With intensive exploitation of packages provided by these languages , the programmer can easily write semantics for formula processing and numerical evaluations. . There exist numerous such packages like Symbolic packages, algebra topology packages etc. Which proves the above fact. .

A. Structure of Functional Programming language

1) Data Types: In any language, variable(data) should be declared before it can be used in program. Data types are the keywords, which are used for assigning a type to a variable.
Syntax for declaration of a variable:
datatype variablename;
Example: int a,b;

- string: A string is a data type used in programming, such as an integer and floating point unit, but is used to represent text rather than numbers. It is comprised of a set of characters that can also contain spaces and numbers. For example, the word "hamburger" and the phrase "I ate 3 hamburgers" are both strings. Even "12345" could be considered a string, if specified correctly. Typically, programmers must enclose strings in quotation marks for the data to recognized as a string and not a number or variable name.
- character: Use the Char data type when you need to hold only a single character and do not need the overhead of String. In some cases you can use Char(), an array of Char elements, to hold multiple characters. The default value of Char is the character with a code point of 0. It holds unsigned 16-bit (2-byte) code points ranging in value from 0 through 65535. Each code point, or character code, represents a single Unicode character.
- float: The Float data type is an alias for a Real or Precision data type, depending on the precision you specify.
- integer: Integer data type allows a variable to store numeric values. int keyword is used to refer integer data type. The storage size of int data type is 2 or 4 or 8 byte. It varies depend upon the processor in the CPU that we use. If we are using 16 bit processor, 2 byte (16 bit) of memory will be allocated for int data type. Like wise, 4 byte (32 bit) of memory for 32 bit processor and 8 byte (64 bit) of memory for 64 bit processor is allocated for int datatype.

2) Lists: A list is a sequential data structure. It differs from the stack and queue data, structures in that additions and removals can be made at any position

in the list.The List operations are Add : adds a new node Set : update the contents of a node Remove : removes a node IsEmpty : reports whether the list is empty IsFull : reports whether the list is full Initialize : creates/initializes the list Destroy : deletes the contents of the list (may be implemented by re-initializing the list)

3) Classes: When you define a class, you define a blueprint for an object. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.
4) Functions: A function is a group of statements that together perform a task. Every C program has at least one function, which is main(), and all the most trivial programs can define additional functions.
You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division usually is so each function performs a specific task.
The general form of a function definition in C programming language is as follows:
returntype functionname(parameter list)
{
body of the function
}
A function definition in C programming language consists of a function header and a function body. Here are all the parts of a function:
5) The Lambda calculus and computation theory:
The Lambda-calculus is the computational model the functional languages are based on. It is a very simple mathematical formalism. In the theory of the lambda calculus one is enabled to formalize, study and investigate properties and notions common to all the functional languages, without being burdened by all the technicalities of actual languages, useless from a theoretical point of view.
In a sense, the lambda calculus it is a paradigmatic and extremely simple functional language. The concepts the Lambda-calculus is based on are those that are fundamental in all functional programming languages:
variable (formalisable by x, y ,z,) abstraction (anonymous function) (formalisable by lambda.x.M where M is a term and x a variable) application (formalizable by MN, where M and N are terms)
We have seen that these are indeed fundamental notions in functional programming. It seems, however, that there are other important notions: basic elements, basic operators and the possibility of giving names to expressions. Even if it could appear very strange, this notions actually are not really fundamental ones.
The formalization of the notion of computation in Lambda calculus: The Theory of Beta reduction
In a functional language to compute means to evaluate expressions, making the meaning of a given expression more and more explicit. Up to now we have formalized in the lambda-calculus the notions of programs and data (the terms in our case). Let us see now how to formalize the notion of computation. In particular, the notion of "basic computational step". We have noticed in the introduction that in the evaluation of a term, if there is a function applied to an argument, this application is replaced by the body of the function in which the formal parameter is replaced by the actual argument.

II. Functional Programming -Features

In a functional programming , a function is a mapping taking one or more arguments and producing a single result, and is defined using an equation that gives a name for the function, a name for each of its arguments, and a body that specifies how the result can be calculated in terms of the arguments.When a function is applied to actual arguments, the result is obtained by substituting these arguments into the body of the function in place of the argument names. Some og the common features are enumerated belo

1) Concise program Syntax
2) Powerful type system
3) Polymorphism
4) List comprehensions
5) Recursive functions
6) Lazy evaluation
7) Overloading

III. Mathematics as Building block for Functional Programming

Mathematics has always been considerd as Qeen of all sciences. In case of Functional programming this statements holds true. There are several features which were inherited by modern languages from Mathematics. Many of the calculation strategies which are present in functional programming comes from trivial concepts like recursion, algebra, double recursion, coalgebra etc.

I.Recursion and Algebra Recursion holds a great importance in modern programming languages. Data structures like lists, and trees can be best implemented using this concept. If we consider mathematical operations , recursion is very useful. Several algorithms pertaining to mathematical foundations can be easily dealt with recursion. A concept called corecursion (dual recursion) is also useful in programming. Algebra and coalgerbra is used to describe the behavior of structures like automata.

II.Category Another such feature is category . A category c is defined as a mathematical structure containing of objects. Here we our concern lies both on objects and morphism. Objects are mapped and they are composable as well.These categories follow rules like transitivity and associativity. These categories are very useful in defining structures in a Functional Programmming languages.

IV. MILONGA- A modular Functional language

MILONGA stands for modular implementation language oriented to nonlinear geometry applications. This was designed specifically for geometrical interpretation and applications. The concept says that a line or a curve can be easily represented by using an equation, we have to use function instead of symbolic

expressions. It was clearly observed that Functional programming paradigm would be useful in such scenario. Solving a polynomial equation or a curve becomes easy by MILONGA by taking polynomials and integers as basic data structure , the MILONGA language also allows the implementation of most fundamental algorithms commonly used for polynomial equation solving. In particular, This language is expensive but still useful and expressive. It is also capable of solving elimination based algorithms. For its execution, a given MILONGA program is mapped into C++. The compiler for this was written in HASKELL which is a non structured programming language. Other non strict programming languages can also be used for creating such compilers. Thanks to this tool, namely Haskell s monads, it took only one man-month to develop the whole MILONGA compiler. MILONGA is based on abstract machine, which was basically an inspiration from G machine pf Johnsson. These kinds of languages allows the applications to be very specific , optimized amd performant. The compiler transforms the code into a highly optimized C++ code.

V. Concurrency and Parallelism

When multiple execution threads operate on same data it is known as concurrency while parallelism is the case where a computational task or tasks are divided into more than one communicating processes or processors. This concept is rapidly gaining importance in current computing scenario. There are several occasions where the need of such computations can be observed..The main reason is that rate of sped or performance gain in single-processor is not improving at the pace it used to. There is always a need for efficient programming language which can understand and utilize the architecture for gaining performance using multiple cores and multiple cpus. Most of the languages which are used for parallel and concurrent programming are error-prone and inefficient.

Functional programming can be thought of as a great alternative for these languages , the semantics that we write using concept of algebra and functions can be mathematically remodeled to implement concurrency so that parts of program executed by one thread can be executed simultaneously with parts executed with other threads. It must be impossible for one execution thread to modify data that another one is reading or writing at the same time. So, if several threads need to modify a data item, they must do so in a coordinated way and such coordination can be easily modeled by using functional programming strategies.

VI. Functional Programming in Soft Computing

Soft computing is a collection of techniques in computation which comprises of artificial intelligence, machine learning, fuzzy logic and traditional mathematics as well. Its an attempt to study and analyze very complex phenomena which were either left unsolved or were inefficiently solved by conventional methods. In the fields like rough sets, fuzzy sets or fuzzy rough sets, we need to derive fnctions for modeling data models based on which further data predictions can be made. In such situations functional programming is very useful. There are several packages which were specially designed for such purpose. Languages like Java or C were unable to work efficiently when data size were high. In case of machine learning the learning models can also be created and understood easily using functional programming. Functional programming provides simple, powerful , concise and persistent way to write code for such paradigms . In case of rough sets when the attributes are high in number we require storage structures to be efficient and conventional languages were unable to do that. With packages provided by functional languages one can operate on them easily.

VII. Data Management and FP

During design or production usually a large amount of data is generated, by large industrial projects. For instance if we consider a non linear model being converted into a linear model with several operting points will have its own analysis data. These data have complex interrelationships and they evolve dynamically and needs to be kept in a consistent state as changes are made to data. Managing this kind of data unaided is difficult and error-prone. These data are usually statistically very important. Other fields like stock market also produces large amount of data that should be managed in order to predict or analyze the situations. Databases and programming languages have evolved along different lines, mainly because they were developed for different purposes. Recent attempts have been made to integrate databases and programming languages in one uniform environment, Functional Programming has also given significant contribution to enhance the way huge databases are used , let us take an example PolyEX , this language supports semantic data modeling features and have powerful structuring facilities. It also have options to define new data types and storage structure for databases.

VIII. Functional Programming in Embedded Systems

Embedded Systems development is slow and costly because it is low level nature. The programming is close to machine. We need tools which can provide abstraction to those details and let the programmer write a high level program to work on those systems. The systems on the other hand should be reliable and fast. In some situations sensors are used and those sensors analyze the data using mathematical functions. Because of these requirements the complexity of these systems makes it even harder to develop them. We have seen a rapid increase in the number of devices using Linux environment. Development is no longer tied to hardware. Embedded applications should also use the operating system as their interface to the hardware. Those operating systems are usually real time OS and require mathematical libraries. Packages provided by functional languages lets the developer write codes for performing such mathematical calculations . There are several examples of functional programming languages , one such language is OCaml which is a general-purpose programming language that supports functional programming keeping in mind safety and reliability as well. Due to its high-performance nature this is widely used in current times. Other advatages of this language is that it supports object oriented and portability also. It consists of a huge mathematical library which allows user to write programs with less LOC.

IX. Conclusion

We have seen the huge and promising possibility of the use of functional programming in various domains which are dominated by imperative and data driven approaches which were implemented in low level languages. We then presented the examples demonstrating the areas where functional programming found its application which is not limited to mathematics and computer sciences.We also demonstrated its use in soft computing and saw how it helps when the data is big and important. Another thing that was discussed is that we can write scripts based on functional languages by converting the syntax into XML type semantics.

X. REFERENCES

1) PeterAczel and Non-Well-Founded Sets Number14inCSLILectureNotes.StanfordUniversity,1988
2) FlorentBalestrieri,Theundecidability of purestream equations.Draftpaper,2011.
3) Coalgebrasin, functional programming and type theory. VenanzioCapretta .Theoretical Computer Science412(2011)
4) M. Frigo and S. G. Johnson ”The Design and Implementation of FFTW3”, Proc. IEEE, vol. 93, no. 2, pp.216 -231 2005.
5) J. Karczmarczuk ”Scientific Computation and Functional Programming”, Computing in Science and Eng., vol. 1, no. 3, pp.64 -72 1999.
6) Bird R, Introduction to functional programming Prentice Hall 2000.
7) http://www.haskell.org