ChaScript: Breaking Language Barrier using a Bengali Programming System

Arman Kamal[1], Md. Nuruddin Monsur[2], Syed Tanveer Jishan[3] and Nova Ahmed[4]

[1,2,3,4] Department of Electrical and Computer Engineering, North South University
Plot-15, Block-B, Bashundhara, Dhaka 1229, Bangladesh

[1]arman1790@gmail.com; [3]tanveer_syed@live.com; [4]nova.ahmed.2011@gmail.com

Abstract—**As technology is evolving as a driving force in our society, the need for computer programming is arising. However, since most of the computer programming languages are English-based, they can act as barriers for learning computer programming for people who are not adequate in English. In this paper we present a Non-English based educational programming system which was build with the goal to help Bengali speaking people learn and write computer programs in Bengali. We have also discussed how the system was build and why it will be useful for Bengali students to use as a platform to learn computer programming.**

Index Terms—**Non-English Based Programming Language, Educational Programming Language, ECMAScript, Jison, JavaScript, Learning Programming;**

I. Introduction

According to HOPL [8] online database of languages there are about over 8500 programming languages which are recorded and more than one third of them are build with English keywords. The prevalence use of English keywords in programming language is mainly because it is the most common language in the world. However, people who are not properly exposed to English Language may face challenges in learning and practicing computer programming using programming languages where English keywords are in use.

It is imminent that in order to understand and practice computer programming in greater depth students need to use the most common programming languages. Hence, Non-English based educational programming language will act like a bridge that will introduce users to the world of logic by solving existing problems they have probably solved in math classes or similar problem solving classes. Once the concept of logic is clear, the transition towards a full-fledged programming language would be easier.

In this paper we are presenting a Bengali educational programming language called ChaScript[1]. ChaScript was build in order to create a platform for Bengali speaking people learn computer programming without the need for English Language. To understand the need of Bengali programming language first we need to know about Social Cognitive Theory[4][6]. Social Cognitive Theory states that, learning from other people by means of observing them is an effective way of gaining knowledge and altering behaviour. In case of common programming languages it is a tough task. Not only does these programming languages have very complicated structures but their documentations are also a huge barrier for people with limited knowledge of English. This is where ChaScript[1] comes in handy. It gives a gateway for new programmers to start coding and getting the hang of it through observation and do it so without any particular need of supervision. Another aspect of Social Cognitive Theory states that a learner improves his newly learned knowledge or behaviour by putting it into practice. In common programming language cases, this might not always be easy because learners find it hard to apply them in real life scenarios. ChaScript, on the other hand, is designed in a way that allows students to use it easily for solving computational and mathematical problems.

In order to build the programming system, the grammar of the scripting language ECMAScript[9] was used and the grammar was parsed using a JavaScript parser generator, Jison. Since ChaScript contains modified parser syntax and it can be considered as a translated version of ECMAScript.

The paper is organized as follows, in section II, we present related works, section III presents implementation of the programming system, syntax of the programming system is illustrated in section IV followed by the user experiment results on section V and then the conclusion.

II. Related Works

In order to demonstrate the need for a Non-English based educational programming language, we are going to show other programming languages of similar nature as well as we explored and also explore the domains beyond the realm of computer science. We are going to present some works where it is discussed how language plays an important role in the efficiency of learning of students.

Hindawi[11] is a free, open-source, complete non-English based systems programming platform that supports all paradigms of programming languages such as assembly language, logic programming and functional programming. It takes away language barrier allowing non-English literates to take up computer sciences and learn computer programming in their mother tongue.

Mallet[10] has discussed about how a course like mathematics can be difficult to understand in a lecture conveyed in certain language that the students are not as familiar with. Mallet has also emphasized how a person who cannot understand the spoken form of the language of instruction quickly, will have difficulty understanding what is being taught without the facility of additional learning support. Novak and Langerova[2] have considered the problems arising when teaching mathematics with the aid of tools such as computer algebra systems that are not available in the students' native language. This is a problem encountered in the Czech Republic-based classroom. The authors suggested the use of a multilingual dictionary of mathematical terminology which will help overcome such issues to an extent. Bretag et al.[3] has addressed the problem of the consistent lower-than-average achievement of international students from non-English speaking backgrounds in information systems courses in South Australia through an action research project aimed at facilitating improved learning outcomes for these students. Cuevas[7] had discussed that learning of mathematics needs a variety of linguistic skills of the language with which lessons are conveyed. Students who are not habituated with the language may take considerably more time to master it.

RoboMind[12] is a programming environment with its own integrated scripting language which is available in seven different language. The goal of this system is to help people learn programming without much hurdle and regardless of language and without much hurdle.

DoLittle[5] is a Japanese based educational programming language developed by Kanemune et al. This programming language fully supports Objects Oriented Paradigm. Moreover, easy syntax are provided for Japanese people to learn programming smoothly.

III. Implementation of the Programming System

We chose ECMAScript as our intermediate language to interpret and we used Jison, a JavaScript parser generator to parse the ECMAScript grammar. We used Jison to generate a JavaScript file from the ECMAScript grammar. That JavaScript file was used to parse the intermediate language. The file was used as a parser and the intermediate file was given to it as input. During the parsing, an Abstract Syntax Tree was generated. The Abstract Syntax Tree was used to maintain the semantic information of the program properly. If the parser successfully parses the file, an ECMAScript program generated from the semantic information found in the Abstract Syntax Tree is given as output which is then evaluated by the JavaScript interpreter. If the parsing procedure cannot successfully parse the file, it raises an exception and gives detail information about the parsing error and the line number where the error has occurred.

Interpreting a language gives implementations some additional flexibility over compiled implementations. Features that are often easier to implement in interpreters than in compilers include: Platform independence: ECMAScript not only gives us platform independence, but also certain level of portability. You can run an ECMAScript (as well as our code) from any platform as long as you have a web browser in it. Reflection and reflective use of the evaluator: ECMAScript has a first order eval function which lets us compile and run the code on client side without needing a backend support. Dynamic typing: Another advantage of ChaScript is we don't need to define a variable data type thanks to dynamic typing of ECMAScript. Dynamic scoping: When a symbol is referenced, the ChaScript will walk up the symbol-table stack to find the correct instance of the variable to use. This is extremely helpful for inexperienced users.

When designing the interpretation process we wanted to make the system as modular as possible so that future addition and development to the language can be easily facilitated. We also tried to maintain a certain level of standardization because we want this software to be available for future open source development.

We start by declaring 4 string arrays. These arrays contain,

- Bengali Keywords: The Bengali keywords of our programming language. E.g. যদি, নাহলে
- English Keywords: Contains the corresponding JavaScript keyword. E.g. If, Else
- Bengali Numbers: Contains Bengali Numerals(০-৯)
- Symbols: All general symbols used in ChaScript.

We take the whole code in a string and start processing. We start the interpretation process by replacing all the Bengali numerals. We do it simply by searching for each Bengali numeral and replacing it with the array index of that number. Then we look for the general symbols in the code and add whitespace to each of the symbols for tokenization purposes. We start with an empty string as our final JavaScript code. Step by step we interpret ChaScript[1] and parse it to the JavaScript code. The Interpretation process can be divided into 6 main steps. They are

a) String Handling: For string handling we keep a string stack. Than we split each line of code into string arrays based on quotation mark ("). Theoretically, every even number element of that array should be a string. We replace that string with the keyword "str" and push the string into the string stack. This is done to avoid complication in the keyword and variable handling process. After the variable handling process, we search for the keyword "str" and every time it's found, we pop an element from the string stack and parse it to the JavaScript code. The process is shown in Figure 1.

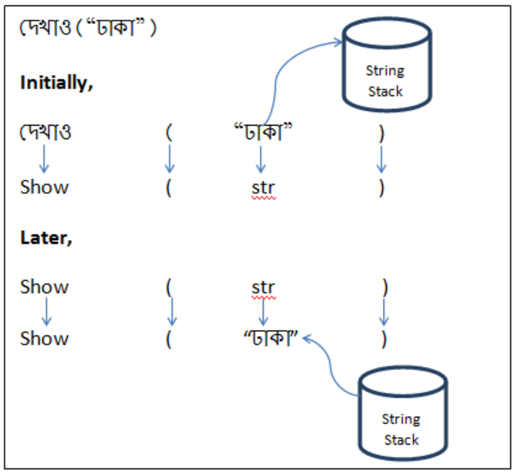


Fig. 1 String Handling Method

b) Tokenization: In this stage we simply split the whole code into several strings based on line breaks. Then each of those strings again broken down into tokens based on whitespaces. Each token is then analyzed and handled accordingly.

c) Number Handling: For number Handling we simply take each token and check all the characters within that token are a number and then parse the number to our code. If not, we go to the next stage.

d) Keyword Handling: The token later gets searched in the Bengali Keyword array, and if found the corresponding JavaScript keyword is parsed into the code. If not found in the Keyword array we go to variable handling.

e) Variable Handling: For variable handling keep two stacks. The first one keeps all the Bengali variables and the second one keeps track of their English counterpart. Whenever we find a token which is neither a keyword nor a number. We look for the token in the Bengali variable stack and if found we simply parse its corresponding English variable to our code. If not found we push the new variable to our Bengali

Variables stack. Then we randomly generate an English variable and push that in the English variable stack. Parsing that variable ends the process. The whole variable handling process is illustrated in Figure 2.

f) Input/output Handling: After the code is generated we check which platform and browser the code is running on. And implement proper JavaScript input/output function for that platform.

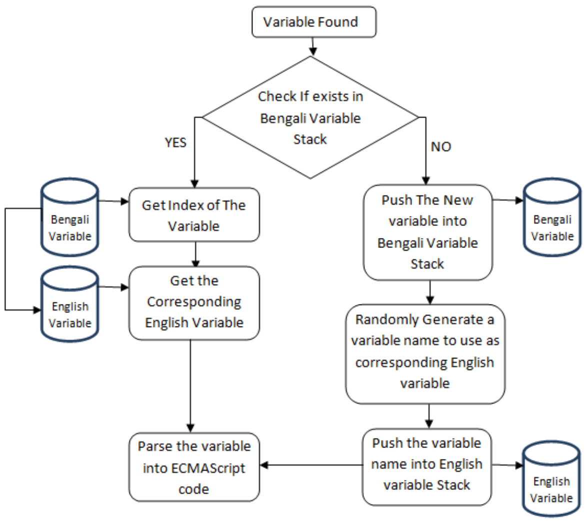

Variable Found
Check If exists in Bengali Variable Stack
YES
NO
Bengali Variable
English Variable
Get Index of The Variable
Get the Corresponding English Variable
Push The New variable into Bengali Variable Stack
Bengali Variable
Randomly Generate a variable name to use as corresponding English variable
Push the variable name into English variable Stack
English Variable
Parse the variable into ECMAScript code

Fig 2. Variable Handling System

IV. Syntax of the Programming System

Designing ChaScript syntax our prime focus was to make the language as easy and accessible as possible. We also wanted to have maximum amount of output with minimum amount of syntax. And of course consisting with common tongue so that people new to programming language will not have much problem understanding it. If we look at the core components of our programming language we can see how easy it is compared to a generic C language .Let us start with a simple "Hello World" program which is shown in Table I.

TABLE I. Hello World Program

ChaScript	C

দেখাও("হ্যালো ওয়ার্ল্ড")	#include"stdio.h"
	int main(){
	
	printf("Hello World");
	
	return 0;
	}

Easy variable declaration is a feature of ChaScript. There is no need for variable type declaration since type declaration is implicitly handled. If we look at the figure we can see how easily we can manipulate variables. If we tried to write the same code in C or Java, it would take a much larger set of codes and a lot of supervision. The conditional statements are almost the same as any other C-alike programming languages. However, when designing loop statement we wanted to make the concept of loop accessible to new programmers. So, instead of using conditional loop we simply run a loop and want the users to stop it manually. Although this may make the program vulnerable to infinite loop, designing the loop structure in primitive fashion will help the students to learn the logic of loop structures.

TABLE II. Fibonacci Number Generator

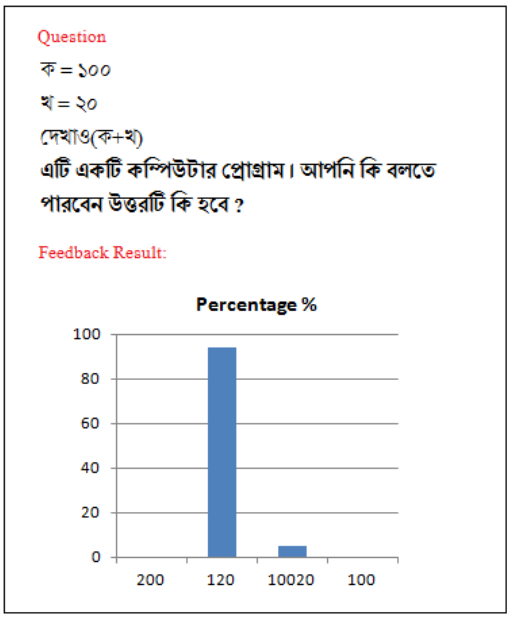
Using Recursion	Using Loop

ফাংশন ফিবোনাচি(ক)
{
যদি(ক == ০)
 উত্তর (০)
যদি(ক ==১)
 উত্তর (১)
উত্তর (ফিবোনাচি(ক - ১) +
ফিবোনাচি(ক -২))
}
 	
দেখাও(ফিবোনাচি(৩))
	ফাংশন ফিবোনাচি(ম){
	ক = ০
	খ = ১
	প=০
	গ=০
	চলবে{
	    যদি (প <= ১){গ = প}
	    নাহলে{
	      গ = ক + খ
	      ক = খ
	      খ = গ
	    }
	প=প+১
	যদি(প==ম){
	থামো
	}
	  }
	উত্তর গ
	}
	
	দেখাও(ফিবোনাচি(৩))

In the Table II, we are representing a recursive and a loop based approach to write a Fibonacci Number generation program in ChaScript. This example highlights the syntax of this programming language where we can notice the difference in loop structure with other conventional programming languages.

V. Experiment on Non Programmers

We commenced an online survey as well as a paper based survey with approximately 110 participants. Most of these participants have not been properly exposed to computer programming before. We tested them with two simple programming problems written in ChaScript and asked them to select the right answers. Our goal was to test whether the participants can understand the programs written in ChaScript.



Question

ক = ১০০

খ = ২০

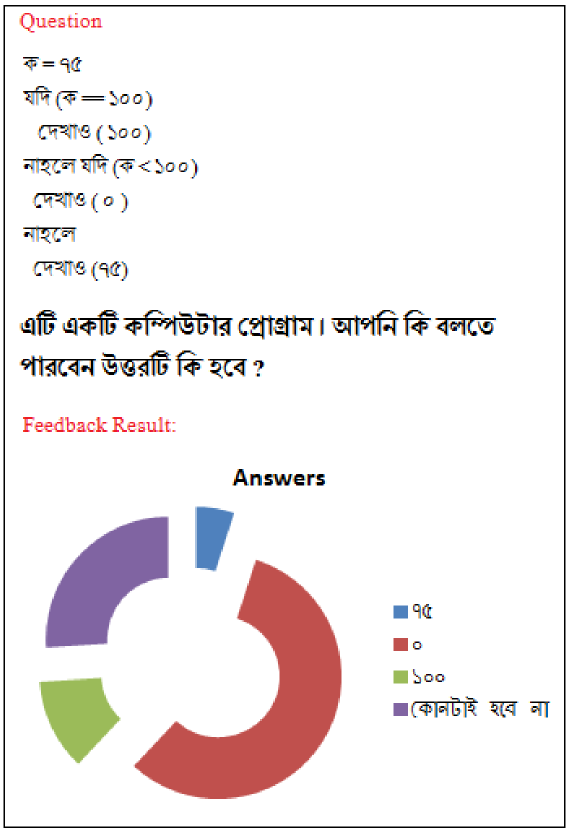
দেখাও(ক+খ)

এটি একটি কম্পিউটার প্রোগ্রাম। আপনি কি বলতে পারবেন উত্তরটি কি হবে ?

Feedback Result:

Fig. 3 User feedback on the programming question

Figure 3 shows the question that is being asked to the participants. Under the questions, a graph is presented showing the participant's responses to the multiple choice question in terms of percentages. It is evident from the graph that almost 95% of the participants chose the correct answer.



Question

```
ক = ৭৫
যদি (ক == ১০০)
  দেখাও ( ১০০)
নাহলে যদি (ক < ১০০)
  দেখাও ( ০ )
নাহলে
  দেখাও (৭৫)
```

এটি একটি কম্পিউটার প্রোগ্রাম। আপনি কি বলতে পারবেন উত্তরটি কি হবে ?

Feedback Result:

Fig 4. User feedback on another programming question

We asked the participants to answer another programming problem (Figure 4) which is slightly more difficult in nature compared to the first question. This program was also written in ChaScript and is based on conditional statements. As we can notice from the figure 4, almost 60% of the participants managed answer correctly, given that the correct answer was 0. Provided that the experiment was conducted mostly on non-programmers, getting a higher percentage of correct answers indicates that this platform can be used effectively to teach students how to program.

We would like to cordially thank the participants who agreed to take part in our survey.

VI. Conclusions

In this paper, we have introduced a programming language based on a native language that may break the initial barrier of learning the concept of programming. We have developed a full-fledged running version of this language and made it available through a web portal and the language has received positive responses from many native speakers.

As our description of ChaScript is restricted to only four pages we could not provide with full details. Although ChaScript is a well translated programming language, a lot more work is required to make it more refined and , hence, amiable to Bengali speaking individuals who are curious to learn programming but are confined because of their lack of knowledge in English language.

References

[1] ChaScript-Home, ChaScript, [online] 2014, http://www.chascript.com (Accessed:15 June 2014).
[2] M. Novák, and P. Langerová, "Raising efficiency in teaching mathematics in non-English speaking countries: an electronic bilingual dictionary of mathematical terminology", in Proceedings of 3rd International Conference on the Teaching of Mathematics at the Undergraduate Level, 2006.
[3] T. Bretag, S. Horrocks,and J. Smith, "Developing classroom practices to support NESB students in information systems courses: Some preliminary findings", International Education Journal, Vol. 3, (4), pp.57-69, 2012.
[4] A. Bandura, "Social cognitive theory: An agentic perspective" Annual review of psychology, Vol.52, (1), pp.1-26, 2001.
[5] S. Kanemune, T. Nakatani, R. Mitarai, S. Fukui, and Y. Kuno, (2004). "Dolittle-Experiences in Teaching Programming at K12 Schools", in C5 , 2004.
[6] D. Compeau, A. H. Christopher and H. Sid, "Social cognitive theory and individual reactions to computing technology: A longitudinal study.", Mis Quarterly, Vol.23, (2), 1999.
[7] G. J. Cuevas, "Mathematics learning in English as a second language.", Journal for Research in Mathematics Education, 1984, pp.134-144.
[8] *D. Piggott, HOPL: An interactive roster of programming languages.,* Perth: Murdoch University, 2006.
[9] ECMAScript, E. C. M. A., and European Computer Manufacturers Association, "ECMAScript Language Specification", 2011.
[10] D. G. Mallet, "Walking a mile in their shoes: Non-native English speakers' difficulties in English language mathematics classrooms.", Journal of Learning Design, Vol.4, (3), 2011.
[11] Hindawi-Hindi Programming System, Hindawi, [online] 2004, http://hindawi.in/en_US/ (Accessed:15 June 2014).
[12] RoboMind.net - Welcome to RoboMind.net, the new way to learn, [online] 2005, http://www.robomind.net/en/index.html (Accessed:15 June 2014).