Towards the Development of a Token Recognizer for a Yorùbá-based Programming Language

Ezekiel K. Olatunji[1*], John. B. Oladosu[2], Stephen O. Olabiyisi[2]
[1]Department of Mathematical and Computing Sciences, Thomas Adewumi University, Oko, Kwara state, Nigeria
[2]Department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Nigeria

ezekiel.olatunji@tau.edu.ng, johnoladosu@gmail.com, soolabiyisi@lautech.edu.ng

Abstract

Experimental development of a programming language that uses the lexicons of the Yoruba language has been attempted. This was done in the belief that an indigenous African language-based programming language would enhance the comprehension of computer-based problem-solving processes by indigenous students and teachers. However, the modern orthography of the base language was not considered in the implementation of the programming language. The orthography of a standard Yoruba language consists of the basic alphabets of the language as well as some diacritical marks whose presence or otherwise indicate the pronunciation of an alphabet. The present study is on the development of a token recognizer for a Yoruba-based programming language that takes into cognizance the modern orthography of the base language. Yoruba is one of the indigenous African languages, being the first language of more than 30 million people in the southwest of Nigeria.; and spoken by more than 100 million people globally. Appropriate regular expressions were designed to specify the lexical structure of the Yoruba-based programming language, a state transition diagram was also drawn to show how the basic tokens of the language in a given source program will be recognized, while implementation of the system would be carried out using Python programming language. In readiness for the actual implementation of the system, the expected output of a correct modern Yoruba token recognizer is presented in this paper.

Keywords: Lexicon, Orthography, Programming language, Token, Yoruba language

1. Introduction

Programming languages (PLs) are notations for the description of algorithms and data structures for computer machines and people. The role of a PL is that of enabling a programmer to express abstract ideas in a language that a computer machine can understand (Olatunji, 2019). The list of valid words or strings in a given language is called the language's lexicons.

Compilers and interpreters are the common processors developed for the implementation of a PL (Olatunji, 2006). A PL's token recognizer (also called a scanner or lexical analyzer) is usually the first routine or subsystem of most PL processors. A token of a

programming language is the smallest unit of a source program, that is, a program in a non-machine language. Examples of PL tokens are keywords, identifiers, literals, and so on (Olatuniji, 2014; Aho et al, 2007). In the parlance of formal grammar from whose theories the study derives its philosophy, tokens of a programming language are the terminal symbols of the language.

A token recognizer conventionally scans each source program statement, character by character, in order to build up and classify the tokens of the source language in the program. In addition, a token recognizer eliminates 'redundancies' in the source program, such as comments and blanks characters) from a source statement. Errors in the source program statements, such as illegal characters are also reported (Olatunij 2019).

It has been observed that most real-life PLs borrowed their lexical items from the Asian and European languages, however, there is a paucity of research information on the development of a PL with lexicons of an African indigenous language like Yoruba. Experimental development of an African native language-based PL, using Yoruba as the base language, was attempted by (Olatunji et al, 2019, 2021) in the belief that such a PL would facilitate the comprehension of computer-based problem-solving processes by indigenous learners and instructors. This belief has also been corroborated by other research studies (UNESCO, 2007; Pflepson, 2011; Silva et al, 2020).

However, the work of Olatuni (2019) did not consider the modern orthography of the standard Yoruba language. Words in the standard Yoruba language do not only consist of the language's alphabet but also some diacritical marks whose presence or absence on an alphabet denotes the way the alphabet is to be pronounced. The standard Yoruba is the one that is studied in formal schools and spoken during news broadcasts on television and radio. Yoruba is one of the indigenous African languages, the mother tongue of more than 30 million people in the southwest of Nigeria.; and is spoken by more than 100 million people globally (Eludiora et al, 2015).

This present study aims at developing a token recognizer for a PL that uses standard Yoruba lexicons with the correct tonal or diacritical marks. Successful development of a token recognizer for the Yoruba-based PL will facilitate the remaining effort in the design and implementation of a full-scale Yoruba-based PL. Adoption of the PL, when fully implemented, will definitely increase the functional load of the Yoruba language and thus serve as an effective strategy that will:

- enable indigenous learners and teachers to quickly grasp the intricacies involved in problem-solving processes that are computer-based.
- prevent the predicted genocide of indigenous African languages, including Yoruba (Awobuluyi, 2014; Azeez, 2013)
- ensure the continuity and relevance of Yoruba language and other African languages in the age of ICT and increasing globalization; and also
- bridge the digital divide between the developed and underdeveloped/ developing countries of the world (Olatunji, 2019)

2. Review of Literature

2.1 The Yoruba Language

Yorùbá is a tonal language (Olumuyiwa, 2013), having three contrastive tones: h(igh), l(ow), and m(id) (the default tone). Every syllable must have at least one tone. Tones distinguish the meaning of words and are indicated by the use of the acute accent for high tone (⟨á⟩, ⟨ń⟩), the grave accent for low tone (⟨à⟩, ⟨ǹ⟩); mid is unmarked. Examples are:

i. Kọ (h) - to build
ii. Kọ (m) - to sing
iii. Kọ (l) - to reject

Due to colonization by the British, some Yoruba words were borrowed from English. Examples are Ṣọ̀ọ̀sì (church), télọ̀ (tailor) and béba (paper). Some Yoruba words were also borrowed from Arabic as a result of influence of Islam. Examples are àdúrá from "al-du'a" (prayer) and Sérià from "Shari'ah" (punishment).

The present orthography of Yorùbá makes use of the Latin script, modified by the use of the digraph and certain diacritics (Olumuyiwa, 2013). A digraph is a pair of characters used to represent one phoneme (distinct sound) or a sequence of the phoneme that does not correspond to the normal sound values of the two characters combined. The only example is 'gb'. A diacritic is a mark or sign added to a letter in order to change or distinguish its sound values. For example, À and Á are respectively pronounced with low and high tones.

The alphabet of the standard Yoruba orthography consists of 25 uppercase letters and 25 lower case letters. The upper case letters of the Yoruba alphabet are:

A B D E Ẹ F G GB H I J K L M N O Ọ P R S Ṣ T U W Y

The lowercase symbols of the Yoruba alphabet are:

a b d e ẹ f g gb h i j k l m n o ọ p r s ṣ t u w y

Three of each of the upper case and lower case letters carry an under-dot sign. These are Ẹ, Ọ and Ṣ for the upper case letters and ẹ ọ and ṣ for the lower case letters. These letters are also pronounced differently from their equivalent without the under-dot sign. The tonal signs are usually placed on the vowels in the alphabet, with the exception of the letter i. The vowels in the Yorùbá upper case letters, in addition to I, are:

A, E, Ẹ, O, Ọ and U.

The pronunciation of the letters without diacritics corresponds more or less to the phonetic notation based primarily on the Latin alphabet.

2.2 Related works

A design and implementation of an indigenous African language-based programming language were carried out by Olatunji et al (2021). The Yoruba language was used as the case

study in their work. The study was embarked upon with the belief that the availability of a programming language in one's mother tongue will enhance the teaching and learning of computer programming, especially at the primary and junior secondary school levels. However, as noted earlier, in their work, the modern orthography of the Yoruba language was not considered. This is the major gap in their work that the present study is trying to address.

Qalb is a functional PL that was developed by Nasser (2012). The PL was based on the lexicons of the Arabic language. It does not make use of the ASCII character set for its encoding. The PL can be used to write meaningful computer programs in the Arabic language. The development of Qalb was motivated by a strong desire to challenge the tradition in which the design of most popular programming languages is predominantly based on the lexicons of the English language. Such a tradition, as noted by McAllister (2013), makes learning programming very difficult for students whose native language does not use the Latin alphabet as does the English language. The token recognizer in the present study makes use of the Yoruba alphabet from which its lexical items are formed.

Ajayi, Adagunodo and Aderounmu (2007) developed an interpreter that translated a sequence of instructions written in the Yoruba language to machine-understandable instructions. The interpreter, according to Ajayi *et al*. (2007), is a simple language having its features derived from the more commonly interpreted languages like BASIC. The work of Ajayi *et al* (2007) is similar in a number of respects to this study, however, their approach is quite different from ours. While for implementation, they employed interpretation; in this research, compilation is the approach to be used for implementation. The present study is a subsystem of a compiler.

Dolittle is a PL whose keywords are based on the Japanese language. It was developed in 2000 (Yoo et al, 2006) as an object-oriented educational programming language that is suitable for children in both elementary and secondary schools. The programming language was written in Java and does not require elaborate declarations. The programming language whose token recognizer is being considered in this study will be a structured PL that uses Yoruba words and is designed for programming by primary and junior secondary school children.

Other native language-based programming languages include the Hindawi programming system, Ezhil and Rappira (Olatunji, 2019). The Hindawi programming language enabled programs to be written in Indic languages like Hindi and Bangla (Kumar, 2011). Ezhil is an interpreted Tamil-based programming language that was targeted at junior high school students. The language was developed by Annamalai (2013). Rapirra consisted of Russian-based keywords (Dadhich, 2006). Most of these native language-based programming languages, like the one being considered in this study, were developed for pedagogical reasons.

3. Methods and Materials

The character set and reserved words of the programming language (PL) were respectively formed from the basic Yoruba alphabet and standard Yoruba words. The lexical items of a subset of the Yoruba-based programming language were specified by defining necessary regular (type-3) grammars using the Backus-Naur Form (BNF) notations. The lexical structure of a PL determines which group of symbols or characters are treated as identifiers, reserved words, operators, and so on. A subset of the design of the PL that is relevant to this

present study is shown in Figure 1. The BNF definitions for variables (Feriebu) and numbers (Nomba) are the ones given in Figure1.

For the implementation of the Token Recognizer, a state transition diagram, shown in Figure 2, was drawn to recognize the basic tokens/lexical items of the PL. In the diagram of Figure 2, D stands for digits (0-9), L stands for letters of the Yoruba alphabet (a-y, A-Y), DELIM represents single character delimiters [(,), +, -, /, *, <,>, = comma], excluding the single quote (‘) and exclamatory mark (!). EOL stands for end of line character, VC stands for any valid character among digits, letters, delimiters and the blank character; IVC stands for any character not in the set of VC, while ‘Others’ mean any character that is not part of the lexeme for a particular token category. INT, RWD, IDN, DEL and SLT respectively stand for internal representations of integer, reserved word, identifier, delimiter and string literal.

Definition of < Nọmba> (Number) and < Feriebu> (Variable)

< Nọmba> → <Dijiti> [{ <Dijiti> }^{4}]
< Feriebu> → <ABD> [{ <ABD> | <Dijiti>}^{6}]
<Dijiti> → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<ABD> → A | B | D | E | Ẹ | F | G | ‘GB’| H | I | J | K | L | M | N | O | Ọ| P| R | S | S | T | U | W | Y |
a | b | d | e | ẹ | f | g | ‘gb’| h | i | j | k | l | m | n | o | ọ | p | r | s | ṣ |
t | u | w | y | <ABDTona>

<ABDTona> → á | à | é | è | ó | ò | | ù | ú |

À | Á | É | È | Ó | Ò | Ú | Ù

Note:
The definition of <Nọmba> implies it cannot be greater than 99999, while a
< Feriebu> which is an identifier cannot be more than 7 characters long.

Figure 1: Production Rules for <Nomba> and <Feriabu

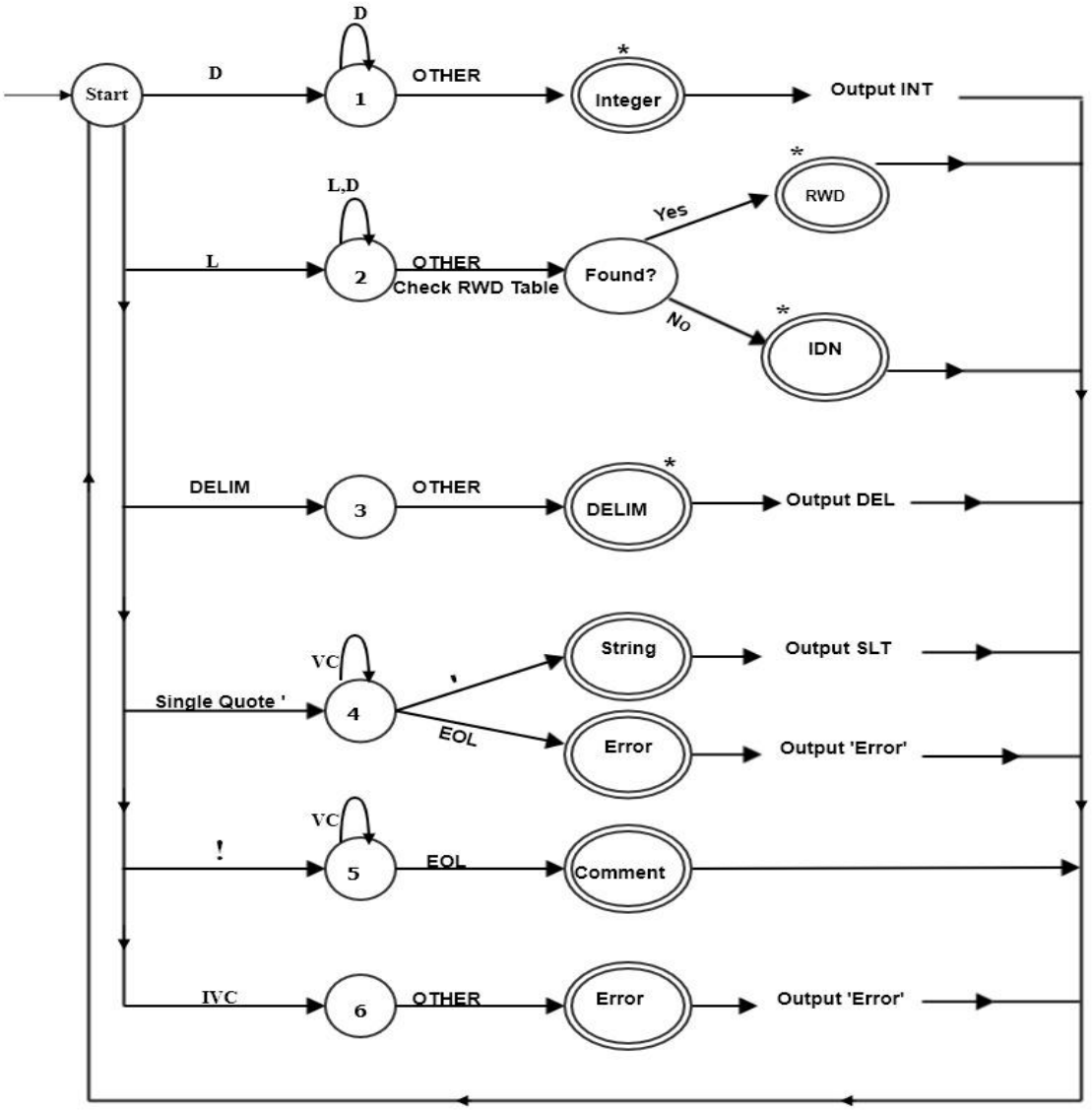

Figure 2: State Transition Diagram of the Token Recognizer

The Python Programming language will be used for the programming of the Token recognizer. Furthermore, input to the system includes;

a) Yoruba-based Source program statements
b) Table of terminal symbols (that is, the programming language's reserved words, operators and delimiters).

While the output that will be produced by the system (the token recognizer) includes:

a) File of Tokens – which consists of uniform symbols (tokens) found in the source program

b) Table of uniform symbols for each statement in the source program. Each entry in the table consists of the actual token and its category (such as an identifier or a literal). A sample is shown in the result section of this paper.
c) Comment and Error File. The file contains comments on each source statements in the program along with detected errors in the program. The comment indicates the lexical correctness or otherwise of each source statement.

A text editor, such as a notepad, will be used for the entry of source statements in the Yoruba-based PL. A sample fictitious source program is given in Figure 3. Each source statement is contained in one line.

```
Dèbó = Adé
Adé = 5
Jẹ́ki abd45yu = 55W / Q4 '- ) ihjkl + 'mnop' * ! rst < u1 W
Abd = 4 (+
!Abd = 4 (E+ma) >Z – Gbógó / Q2
Abd = 4 (E+ma) >Z – Gbógó / Q2
```

Figure 3: Source Program in Yoruba for Testing the System

4. Result and Discussion

Figure 3 is a fictitious source program in Yoruba that can serve as input to the system. If the system is tested with the source program in Figure 3, a sample output that will be produced for statement 3 in the source program is as shown in Figure 4. Figure 5 is the output that will be produced for the entire source program of Figure 1.

Figure 4 contains the different tokens recognized in the source statement 3 along with their token categories (reserved word, identifiers and so on). In the test data, an exclamatory mark (!) precedes a comment statement, while a string literal is enclosed within two single quotes. Figure 5 shows that characters Q and Z in statement 3 of Figure 3 are correctly recognized as being invalid characters in the alphabets of the Yoruba language.

OUTPUT OF THE TOKEN RECOGNIZE FOR THE YORUBA BASED PROGRAMMING LANGUAGE

Statement 3 : Jẹ́kí Abd45yu = 56W / Q4’ -) ilijkl + ‘mnop’ * ! rst < u1w

S/N	Actual Token	Token category Internal Repr
1.	Jẹ́kí	Reserved word
2.	Abd45yu	Identifier
3.	=	Operator/Delimiter
4.	56	Integer
5.	W	Identifier
6.	/	Operator/Delimiter
7.	4	Integer
8.	-) ihjkl +	String Literal
9.	Mnop	Identifier

Statement 3 : Jẹ́kí abd45yu = 56W / Q4 ‘ -) ihjkl + ‘mnop’ * !rst < u1 w has been processed

Press any key to continue / F1 ka tẹ́ Bọtini kan lati tẹ́siwaju

Figure 4: System’s Output for Statement 3 in the Source program of Figure 3.

System's Produced Comments & Error File

Àbájàde Ṣiṣe Ayẹ̀wo Síntáasí Akọ́kọ́ Ninù Purogiràamù Yí
= = ==========
Àsìṣé ni ilà 3 : Q jẹ́ Kàràkitá Àjeji
Àṣisé ni ilà 3 * | rst < u1 w Literal Ọlọ́rọ́ọ yii Ko dara to o
Ọrọ̀ Alaye ni ilà 5 : !Abd – 4 (E+ma) >z – Gbògo / Q2
Àsisé ni ilà 6 : Z jẹ Kàràkitá Ajeji
Àsisé ni ilà 6 : Q jẹ Kàràkitá Ajeji

Lexical analysis of the program is completed with Errors!!
Siṣe àyẹ̀wo Síntáasi Àkọ́kọ́ ninù Purogiràamù yi ti pari –
SUGBON Wàhálà diẹ wa !

Figure 5: System's Output for the programmer (using Figure 3)

5. Conclusion and future work

Successful development of a token recognizer for a programming language (PL) with the lexicons and orthography of a standard Yoruba language will fast tract the full development of a compiler for such a Yoruba-based PL. Such a PL will provide an indigenous platform for teaching computer programming to indigenous pupils of junior secondary schools. This study is on the development of a token recognizer and not on the implementation of a complete compiler for the Yoruba-based PL. The next step in this study is to embark on the full implementation of the system using Python PL or any other suitable PL.

Acknowledgement

The authors are grateful to Mr. J. O. Ogunleye of TAICO and Mr. Taiwo Timothy of Thomas Adewumi University (TAU), Oko, Kwara State, Nigeria for their assistance in putting the corresponding author through on the appropriate tonal signs for some used Yoruba words.

References

Aho, A. V.; Lam, M. S.;Sethi, R. and Ullam, J.D (2007), *Compilers – Principles, Techniques and Tools*, Pearson Education/ Addison Wesley, New York, 2nd ed; pp 1040

Ajayi, A. O.,Adagunodo, E.R. and Aderounmu, G. A. (2007); An object-oriented Approach to Crafting an Interpreter: Yoruba Language as a Case Study, *Journal of Computer Science and its Application, Nigeria Computer Society*, ; Vol. 14, No. 2, pp 60-72

Annamalia, M. (2013). Invitation to Ezhil: A Tamil Programming Language for Early Computer Science Education. [Online]. Available: http://ezhillang.org Retrieved on 17-03-2015

Awobuluyi, O (2014). Yoruba Must Not Die Out, A Lecture delivered at the faculty of Arts, Obafemi Awolowo University, Ile-Ife, Nigeria, Feb. 2014.

Azeez, A.S (2013). The Yoruba language and Literature 21st Century and beyond, *Covenant Journal of Language Studies (CJLS),* Vol 1, No 2, pp 148-159, Dec 2013

Dadhich, B.S. (2006). Needed: A good "Hindawi Programming Language". [Online]. Available: www.localizationlabs.balendu.com/ Retrieved on 15-06-2015

Eludiora, S. I.; Agbeyangi, A.O. and O.I. Fatusin (2015); "Development of English to Yoruba Machine Translation System for Yoruba verbs' Tone Changing", *International Journal of Computer Applications,* vol. 129, No. 10, 2015

Kumar, L (2011). Hindawi: Write Computer Programs in Indic Languages. [Online]. Available: http://techwelkin.com/hindawi-now-write-computer-programs-in-indic-langaugeso Retrieved on 18-03-2015

Nasser, R. (2012). Qalb,[Online]. Available: http://nas.sr/%D9%82%D9%84%D8%A8 Retrieved on 18-03-2015

Olatunji, E. K., Oladosu, J. B., Odejobi, O. A. and Olabiyisi, S. O. (2021). Design and Implementation of an African Native Language-based Programming Language. *International Journal of Advances in Applied Sciences (IJAAS*), 10(2); 171-177
. https://doi.org/10.11591/ijaas.v2.i2.pp171-177

Olatunji, E. K. (2019). Development of a Programming Language with Yoruba Lexicons
. Unpublished PhD Thesis, Ogbomoso-Nigeria, Ladoke Akintola University of Technology.

Olatunji, E. K., Oladosu, J. B., Odejobi, O. A. and Olabiyisi, S. O. (2019a). Design of an

African Native Language-based Programming Language. *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)*, 3(1); 72-78

Olatunji, E. K (2006). *Compiler Techniques – An Introductory Text on Concepts and Principles*, Igbagbogbemi Publisher, Ilorin – Nigeria, 2ed, pp 6, 7,8, 63. ISBN 978-36062-7-1

Olatunji, E.K. (2014), *Programming Languages – Introductory Text on Concepts and Principles*, Igbagbogbemi Publishers, Ilorin – Nigeria, ISBN 978-187-458-9, pp 8, 2014.

Pflepsen, A. (2011). Improving Learning Outcomes through Mother Tongue-Based Education, , MTB-MLE Network. [Online]. Available: https://www.eddataglobal.org Last Accessed on 30-07-2014, stored online as eddata_ii_mother_tongue_instruction.pdf Vol 1, No 2, pp 148-159, Dec 2013.

Silva, G., Santos, G., Canedo, E.D., Rissoli, V., Praccano, B. and Andrade, G. (2020) Impact of Calongo Language in an Introductory Computer Programming Course. *In 2020 IEEE Frontiers of Education Conference,* pp 1-9. DOI: https://doi.org/10.1109/FIF44824.2020.9274150

UNESCO (2007). Mother Tongue-based Literacy Programmes: Case Studies of Good Practice in Asia. Bangkok: UNESCO Bangkok, Thailand, {Online}. ISBN 92-9223-113-8 Available: www.unesdoc.unesco.orgRetrieved on 30-07-2014

Yoo, S. W., Kim, K., Kim, Y., Yeun, Y., Kanemune, S & Lee, W (2006). Empirical Study of Educational Programming languages for K-12: Between Dolittle and Visual Basic; *in International Journal of Computer Sciences and Networks Security*, vol, No 6, 2006