Abstract — The knowledge-driven economy uses technology, thereby increasing the demand for language tools and resources to acquire and distribute the knowledge. Such tools and resources are scarce for the under resourced, spoken Bantu languages. This paper develops a computational grammar for the Ekegusii language in the Grammatical Framework (GF) to bridge the gap. The grammar development uses a bottom-up and modular-driven approach. A machine translation experiment was set up to evaluate the grammar resulting in BLEU and PER of 55.95% and 19.49%, respectively. This work contributes by providing computational grammar for an under-resourced language, thus providing a platform for analysis and synthesis, plus a machine translation within the GF ecosystem.

Index Terms — Ekegusii, Grammar Engineering, grammar, Grammatical Framework, Under resourced language.

I. INTRODUCTION

The technology knowledge-driven economies demand natural language processing (NLP) tools and resources to acquire and distribute knowledge and information [1]. However, few NLP tools and resources exist for under resourced languages and more so for the spoken Bantu languages; thus, language technology digital divide. One way to solve the challenge is Grammar engineering (GE) for under resourced languages. GE is creating computational grammar based on formal grammar that machines can parse or generate [2]. This paper performs grammar engineering for the Ekegusii language, an under-resourced language resulting in computational grammar using the grammatical framework (GF). The work is a significant milestone towards creating a standard Basic Language Resource Kit (BLARK) [3] for such a language.

Ekegusii is an agglutinative and tonal Bantu language. Guthrie [4] classify this great lake Bantu language [5] as E42. There are two main dialects of Ekegusii, mainly Maate and Rogoro dialects [6]. This research has used the Rogoro dialect by virtual of being the standard dialect.

GF is a grammar toolkit, a logic framework of syntaxes and a grammar formalism grounded on categorical formalism [7]-[9]. It has a single abstract syntax that defines a set of categories (Cat) of trees, a set of functions (Fun) to implement those trees plus their type and start category [10]. Furthermore, it has many parallel concrete syntaxes, one for each language grammar. These syntaxes define linearization of both the categories (lincat) and the function (lin) stated in the abstract syntax as exemplified using category Noun (N) with string “house” below [7].

Abstract syntax
Cat: N
Fun House: N

Concrete syntax
lincat N = Str
lin House = "house"

The parallel concrete syntaxes are parallel multiple context-free grammars (PMCFG) [11] and reside in the GF resource grammar library (RGL), where the syntactic and morphological properties of a specific language are captured and form the multilingual grammars ecosystem [12]. The RGL is subdivided into three primary modules: lexical, morphology and syntax. Morphology is implemented via paradigms (low and smart), which are functions that take lexeme word form(s) and generates the complete word forms (inflection table). Detrez et al. [13] define a smart paradigm as a Meta paradigm that inspects the given base form of a lexeme and tries to infer which low paradigm applies. Parsing provides a means of transforming language-specific strings to abstract trees, while linearization is a composition of homomorphic mapping from common abstract tree structure to specific language concrete syntax [8]. Machine translation would then be achieved by first parsing the source language's string to abstract trees then linearizing the abstract trees to strings in the target language.

Features in GF formalism are provided via parameters and are defined using the keyword param and mostly used in table types. For example, the noun in Bantu languages has a parameter number with values: singular and plural; therefore, the definition would look like:

\[
\text{param} \ 
\begin{array}{c}
\text{Number} = \text{Singular} \mid \text{Plural}
\end{array}
\]

A category may have more than one parameter. In such a case, a data structure record is used to gather them. For example, the category noun in Ekegusii has additional parameter gender apart from the number; therefore, it is defined as:

\[
N = \{s: \text{Number} \Rightarrow \text{Str}; g: \text{gender}\}
\]

The above is a table from number to strings and has an inherent feature of gender (functions over parameters) [14]. GF distinguishes the function \text{fun} used in abstract syntax and the function operation \text{oper} used to implement inflection paradigms. Operation is used to implement the regular pattern in grammars to avoid redundancy of repetition. The keyword \text{oper} is usually of the form:
One name can be used for different paradigms in the same category through operation overload.

Some Bantu language grammars have been developed in GF, mainly: Kikamba [15], Swahili [16] and Runyankore and Rukiga [17]. Furthermore, Ekegusii has other language resources such as an Interlingua rule-based machine translation between Swahili and Ekegusii [18] and morphology analyzer [19]. An online dictionary, little parallel corpus (some bible segments) and some monolingual corpus [20]. The above survey demonstrates that little work has been done to develop NLP tools and resources for this language; hence, this computational grammar will be a significant effort.

II. **EKEGUSII DESCRIPTIVE GRAMMAR**

Four descriptive grammar sources [6], [21]-[23] were used in computational grammar development. Here, we shall explain the descriptive grammar, starting with morphology followed by syntax.

A. **Morphology**

The morphology uses the nominal1 class system [24] based on affixes to a noun root [22]. Arguments have been forwarded whether the nominal class system should be referred to as gender or noun class see [15]. In this case, we adopt the view that a pair of noun classes (singular and plural) are to be regarded as gender, as shown in Table 1.

<table>
<thead>
<tr>
<th>TABLE I: EKEGUSII GENDERS (NOUN CLASSES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genders (classes)</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>omo_aba</td>
</tr>
<tr>
<td>omo_eme</td>
</tr>
<tr>
<td>eri_ama</td>
</tr>
<tr>
<td>ege_ebi</td>
</tr>
<tr>
<td>e_ci</td>
</tr>
<tr>
<td>oro_ci</td>
</tr>
<tr>
<td>aka_emi</td>
</tr>
<tr>
<td>obo_ama</td>
</tr>
<tr>
<td>oko_ama</td>
</tr>
<tr>
<td>ama_ama</td>
</tr>
<tr>
<td>aa</td>
</tr>
</tbody>
</table>

The regular noun’s structure consists of a prefix that is a noun class number (a subset of gender) and the noun root. The structure is exemplified by Example 1, where c means class and the number represent the class number as stated in Table 1.

**EXAMPLE 1: NOUN STRUCTURE**

<table>
<thead>
<tr>
<th>Singular</th>
<th>Plural</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eke-rogo (gender ege_ebi)</td>
<td>Ebi-rogo (gender ege_ebi)</td>
</tr>
<tr>
<td>c7-stem (gender ege_ebi)</td>
<td>c8-stem (gender ege_ebi)</td>
</tr>
<tr>
<td>Chair</td>
<td>Chairs</td>
</tr>
</tbody>
</table>

An adjective is a noun modifier inflect for number and class (a subset of gender). The regular adjective structure has an obligatory prefix (concord), which agrees with the nominal class [6], [21]-[23]. Example 2 shows a phrase using gender omo_eme (class 3 and 4). The bold shows the prefix of the gender used. The root of the adjective is represented by

**EXAMPLE 2: REGULAR ADJECTIVE STRUCTURE**

<table>
<thead>
<tr>
<th>Singular</th>
<th>Plural</th>
</tr>
</thead>
<tbody>
<tr>
<td>Omo_eme – gender nke – Adjroot</td>
<td>Omo_eme – gender nke – Adjroot</td>
</tr>
<tr>
<td>Small tree</td>
<td>Small trees</td>
</tr>
</tbody>
</table>

Verbs have a complex morphology with much prefixing and suffixing plus infxing for extensional morphology. Its declension involves several morphemes (several prefixes, root, extensional suffix, and final vowels representing mood) plus some grammar features such as person, number, gender, tense, polarity, etc. The morphemes of verbs embody all the constituents needed to make a sentence. Hence a verb can act in place of a sentence. Table II [6], [21]-[23] summarizes all the prefixes, suffixes, roots and extensions needed to form verbs in this language. The subject marker represents positive polarity, while the negation morpheme is negative polarity.

Both have grammar features of gender, number, and person that form the agreement parameter. It is essential to note that the following fields are usually not obligatory: object marker, infinitive, and extension. The focus morpheme cannot exist with negation [23]. The following notations are used: Fs for focus, Neg for negation, Agr for the subject marker, root for the root, Tns for tense. Asp for aspect, Fw for the final vowel and Aux for the auxiliary verb.

<table>
<thead>
<tr>
<th>TABLE II: VERB STRUCTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
</tr>
<tr>
<td>Prefixes</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Subject marker</td>
</tr>
<tr>
<td>Object marker</td>
</tr>
<tr>
<td>Root</td>
</tr>
<tr>
<td>Extension</td>
</tr>
<tr>
<td>Suffix</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Static</td>
</tr>
<tr>
<td>Final vowel</td>
</tr>
</tbody>
</table>

The Ekegusii tenses are marked by a tense morpheme or no morpheme at all. Three points are needed to mark different tenses, as argued by Reichenbach [25]. These points are the speech point, the reference point and the event point in relation to time, while time is based on speech point [26]. The coincidence of the three points results in the present tense. When the speech point is after the other two points, then past tense occurs. Future tense occurs when the speech point is before other points. Finally, when the reference time proceeds to event time, the resultant is perfect tense.

The aspect gives a view of the verb’s action, such as beginning, continuing, or ended [26]. Most of the time, tense and aspect are combined. Several tenses exist [6], [21]-[23]. This discussion focuses on the present, future and past tenses.

The Future tense is marked by the suffix “et” [27] though Ongorora [23] argues that the morpheme “e” does not necessarily represent tense. The Future tense is exemplified by Example 3.

---

1 https://glossary.sil.org/term/noun-class

DOI: http://dx.doi.org/10.24018/ejers.2021.6.3.2382
B. Syntax

The dominant topology for Ekegusii is subject-verb-object (SVO) [22], [28], whereby the subject is a noun phrase, followed by a verb phrase. The verb phrase is made of a verb and object complement that can be a verb phrase, noun phrase or both. The object’s presence is influenced by the verb valence (univalent, diveral, and trivalent). For example, for the univalent verb, the topology becomes SV because the one place verb does not require arguments. The syntactic agreement is via concord agreement within the lexical items mainly influenced by the gender [22].

A noun phrase (NP hereafter) is made of a noun and its modifiers that include adjectives (Adj), Determiners (Det), both possessives (Poss) and demonstratives (Dem) and numbers (Num). Definition 1 how the noun modifies follow each other to form the NP's structure [22], [29].

**Definition 1 NP structure**

[dem] [Noun] [Det <Poss> <Dem>] [Num] [Adj]

The structure represents a complex NP, while a simple NP can only be formed either by a noun or a pronoun. It is also possible to form a complex noun using post-modifiers to the noun phrase - mainly interrogative and past participle of a verb. The verb phrase structure is the same as a verb and carries all parameters that are integral to verbs.

III. IMPLEMENTING EKEGUSII GRAMMAR IN GF

The computational grammar development employed an experimental research design. The grammar development adopted the GF morphology-driven strategy and modular-driven development, a bottom-up method. It involves first defining the lexicon, smart paradigms based on the regular expression and their respective linearization categories before working on the syntax [8]. The evolutionary prototype model [30] was used because each function developed had to be iteratively tested to ensure its work before moving to the next function. GF provides text output in the command prompt. However, to visualize the parse trees from production rules or paradigms for the grammar, the Graphviz tool was used. It takes simple texts as input and converts them into diagrams.

A. Morphology

Table III represents the coding of genders in Table I. The coding is of GX, where G stands for gender and X is a number starting from one. Each gender combined two nominal classes based on parameter number (singular and plural) and separated by an underscore. The genders are coded in the resource grammar using the parameter Cgender as per Definition II.

---

2 https://graphviz.org/
TABLE III: GF CODING OF GENDERS

<table>
<thead>
<tr>
<th>GF coding</th>
<th>Genders</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>omo_abu</td>
</tr>
<tr>
<td>G2</td>
<td>e_ci</td>
</tr>
<tr>
<td>G3</td>
<td>eri_ama</td>
</tr>
<tr>
<td>G4</td>
<td>ege_ebi</td>
</tr>
<tr>
<td>G5</td>
<td>oro_ci</td>
</tr>
<tr>
<td>G6</td>
<td>aka_ebi</td>
</tr>
<tr>
<td>G7</td>
<td>obo_ama</td>
</tr>
<tr>
<td>G8</td>
<td>oko_ama</td>
</tr>
<tr>
<td>G9</td>
<td>a_f</td>
</tr>
<tr>
<td>G10</td>
<td>a_h</td>
</tr>
<tr>
<td>G11</td>
<td>a_i</td>
</tr>
</tbody>
</table>

**Definition II Gender coding in GF**

\[ \text{param} \\
\text{Cgender} = \text{G1}(\text{G2})(\text{G3})(\text{G4})(\text{G5})(\text{G6})(\text{G7})(\text{G8})(\text{G9})(\text{G10}) \mid \text{G11}; \]

These genders influence concordial agreements with part of speech tags; hence agreement is implemented using parameter \( \text{Agr} \) and its composition consist of gender, number and person as per Definition III. To minimize over generation during inflection, especially for verbs, optimization was done so that Person one (P1) and two (P2) are only applicable to gender one (G1) because animate (human) belongs here. The function \( \text{toAgr} \) translates each person level to the right agreement as exemplified in definition III.

**Definition III. Agreement definition**

\[ \text{param} \]
\[ \text{Agr} = \text{AgP1 Number} \mid \text{AgP2 Number} \mid \text{AgP3 Cgender Number} ; \]
\[ \text{oper} \]
\[ \text{toAgr} : \text{Cgender} \rightarrow \text{Number} \rightarrow \text{Person} \rightarrow \text{Agr} = \text{lg}, \text{n}, \text{p} \rightarrow \]
\[ \text{case p of} \{ \]
\[ \text{P1} \Rightarrow \text{AgP1 a} ; \]
\[ \text{P2} \Rightarrow \text{AgP2 n} ; \]
\[ \text{P3} \Rightarrow \text{AgP3 g n} \} ; \]

Generally, lexeme definition for linearization of each category followed a similar structure and involved the following, as exemplified by Example 6:

- Definition of the linearization category;
- The low-level paradigm;
- The lexeme for the category;
- Parameter for the category (some had others did not have).

### EXAMPLE 6: LEXICON DEFINITION

<table>
<thead>
<tr>
<th>Ekugesi Languages</th>
<th>Gloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>woman_N = regN ”omosubati” omo_abu;</td>
<td>Woman</td>
</tr>
<tr>
<td>small_A = regA “nike”;</td>
<td>Small</td>
</tr>
<tr>
<td>play_V = regV “chesa”;</td>
<td>Play</td>
</tr>
<tr>
<td>we_Pron = mkPron “intwe” “ito” G1 Pl P1 ;</td>
<td>We</td>
</tr>
<tr>
<td>very_AdA = mkAdA “mono” ;</td>
<td>Very</td>
</tr>
</tbody>
</table>

Using the example drawn from Example 6. \( \text{regN} \) is the paradigm for noun where a woman belongs, “omosubati” is the lexeme for a woman in Ekugusi language, while oma_abu is the parameter gender in which the noun belongs

\[ \text{woman_N} = \text{regN} “omosubati” omo_abu; \]

#### a) Noun

The noun inflects for parameter number that has values singular(sg) and plural (pl) and gender. Therefore, its linearization is modeled as a number to string as below.

\[ N = \{ s : \text{Number} \Rightarrow \text{Str} ; g : \text{Cgender} \} ; \]

Four paradigms (regular expression) are used to model noun inflection and Table IV explains these paradigms for nouns.

### TABLE IV: NOUN PARADIGMS

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mkN ( \text{Str} \rightarrow \text{Cgender} \rightarrow \text{N} )</td>
<td>Function ( \text{regN} ) takes in a string and gender and returns regular words forms</td>
<td></td>
</tr>
<tr>
<td>mkN ( \text{(man,men ; N)} \rightarrow \text{Cgender} \rightarrow \text{N} )</td>
<td>Function ( \text{compoundN} ) takes in two strings of nouns and gender and generate compound noun forms</td>
<td></td>
</tr>
<tr>
<td>mkN ( V \rightarrow \text{N} )</td>
<td>Function ( \text{verb2noun} ) takes in a verb and generates a noun in the gender G1.</td>
<td></td>
</tr>
<tr>
<td>mkN ( \text{(man,men ; N)} \rightarrow \text{Cgender} \rightarrow \text{N} )</td>
<td>Function ( \text{treng} ) takes in two strings of nouns and gender, then assigns one as singular and the other plural</td>
<td></td>
</tr>
</tbody>
</table>

An extra paradigm \( \text{mkNoun} \) (make noun) is used to assign all forms of inflections to the right parameter number. We illustrate noun paradigms using \( \text{regN} \) in Fig. 1. The function \( \text{PrefixPNom} \) represents the gender morpheme.

\[ \text{regN} : \text{Str} \rightarrow \text{Cgender} \rightarrow \text{Noun} = \{ w, g \rightarrow \text{let wpl} = \text{case g of} \{ \]
\[ \text{G1} \Rightarrow \text{case w of} \{ \]
\[ "omwo" + _ _ \Rightarrow "aba" + \text{Predef.drop 3 w} ; \]
\[ "omw" + _ _ \Rightarrow "ab" + \text{Predef.drop 3 w} ; \]
\[ _ _ \Rightarrow \text{PrefixPNom G1} + \text{Predef.drop 3 w} ; \]
\[ \text{G2} \Rightarrow \text{case w of} \{ \]
\[ "omwo" + _ _ \Rightarrow "emi" + \text{Predef.drop 3 w} ; \]
\[ _ _ \Rightarrow \text{PrefixPNom G2} + \text{Predef.drop 3 w} ; \]
\[ \text{G3} \Rightarrow \text{"chi"} + \text{Predef.drop 1 w} ; \]
\[ \text{G4} \Rightarrow \text{case w of} \{ \]
\[ "ri" + _ _ \Rightarrow "ama" + \text{Predef.drop 2 w} ; \]
\[ _ _ \Rightarrow \text{PrefixPNom G4} + \text{Predef.drop 1 w} ; \]
\[ \text{G10} \Rightarrow \} ; \]
\[ \text{G11} \Rightarrow \} ; \]
\[ \text{let wpl g} ; \]

Fig. 1. \( \text{regN} \) paradigm.

The noun morphological inflection table consists of a maximum of two-word forms for each number. An example of the noun “tree” is highlighted below

\[ \text{lang} \rightarrow \text{linearise} \rightarrow \text{table tree}_N \]
\[ s \text{ Sg} : \text{omote} \rightarrow \text{gloss tree} \]
\[ s \text{ Pl} : \text{emote} \rightarrow \text{gloss trees} \]

### b) Adjective

Adjective inflects for number and gender. Therefore, the parameter \( \text{AForm} \) was used to represent the above two-variable features(parameters) of concordial agreements. The parameter is defined below with values \( \text{AAdj} \) and \( \text{Advv} \) for positive and adverbs adjectives.

\[ \text{param} \]
\[ \text{AForm} = \text{AAdj Cgender Number} | \text{Advv} ; \]

Three paradigms were used to model adjective inflection. The paradigm \( \text{regA} \) takes a string and generates an inflection table. Colour adjectives except red, white and black are generated uniquely using the \( \text{cregA} \) paradigm. Its structure consisted of a prefix specific for each number and gender, the same as the prefixes of pronouns; hence the same operation used in pronouns are used here. The prefix is followed by a string “color of”, then the color lexeme as illustrated below.

DOI: http://dx.doi.org/10.24018/ejers.2021.6.3.2382
The irregA paradigms deal with irregular adjectives and just assign the listed strings in the lexicon definition.

cregA: Str-> { s: AForm => Str } = \{seo -> { s = table [ Adj g Str = PronsgPrefix g ++ "eragi ya" ++ seo; Adj g Pl= PronPlPrefix g ++ "eragi ya" ++ seo; Adv=> [] } } ;

Numeral definition and parameter

CardOrd = NCard | Nord;
DForm = unit | teen | ten | hund ;

The numeral and digits categories have gender and CardOrd as variable parameters and number as an inherent parameter. The values of parameter CardOrd are cardinal (Ncard) and ordinal (Nord) numerals. The numeral one is the only one that has the value of a number as singular; all others are plural. The values for parameter DForm were unit, teen, ten, and hund. The unit is for numerals between 0 to 9. The teen is between 11 to 19, while ten is for multiples of ten and hund for multiples of hundreds. GF4 implements numbers ranging from 0-999,999.

There was gender agreement (concord) for the cardinal numeral one to five in building the numeral. For numerals, six to eight and their multiples are constructed by recursion between one and five. For example, eighty would be constructed as (fifty thirty), as exemplified by Fig. 2.

Cardinal numerals between 1-9 and their multiples of ones, tens, teen, and hundreds had a specific paradigm to model them. The ordinal numeral was modeled by adding a disjunctive prefix of singular “of” in the specific language, while digits were modeled using a similar function for all languages. The function IDig, which took argument digit, returned digits 0 to 9, while function IIDig which took argument digit, followed by digits, returned numeral with at least two digits. The operation mk3Dig created the cardinal digits and ordinal digits by attaching the cardinal digits’ disjunctive prefixes. Fig. 3 exemplifies cardinal numeral in digits and words for gender GI1(omo_ab) in Ekegusii. For gloss “four hundred and eighty-two”.

Fig. 2. Word alignment.

d) Numerals

Numerals are either cardinal or ordinal. Cardinal describes quantity while ordinal shows order and represented in digits or words, for example, 12 and twelve respectively. Both formats are supported in GF. The GF numeral implementation is based on Hammarström and Ranta’s work. The numeral linearization type implementation is exemplified below and numeral parameters.

---

3 https://www.englisch-hilfen.de/en/grammatik/zahlen.htm

4 https://www.grammaticalframework.org/lib/doc/gfdoc/Numeral.html

DOI: http://dx.doi.org/10.24018/ejers.2021.6.3.2382
e) Pronoun

Personal and possessive pronouns are the two forms of pronouns implemented in GF. The personal pronoun acts as a noun phrase, thus requires an agreement in terms of gender, number, and person. The possessive pronoun in GF is implemented as a quantifier. Thus, a determiner hence inflects for gender and number. The parameter PronForm with values Pers for the personal pronoun and Poss for possessive pronoun are used to model in the paradigm mkPron that takes five arguments inputs (two strings—one for personal and a stem for the possessive, gender, number, and person in that order) as exemplified by the lexicon of the pronoun “he” below. The output is two sets of strings, namely: the personal pronouns that act as a noun phrase and the inflection table for the possessive pronoun based on gender and number.

```
he_Pron = mkPron "ere" "je" G1 Sg P3 ;
```

**Pronoun paradigm**

```
mkPron: (i, mine : Str) -> Cgender -> Number -> Person ->
{s: PronForm => Str ; a: Agr} = i,mine, g,n,p ->
{s = table |
Pers => i;
Poss n g => case <n,g> of {
<Sg ...> => ProunSgprefix g + mine ;
<Pl ...> => ProunPlprefix g + mine} ;
a = toAgr g n p ;
```

Specific preposition can be infused or not (true value meaning fused and vice versa).

```
mkPrep = overload {
mkPrep : Str -> Bool -> Prep = \str,bool ->
ln Prep {s = \n,g => str ; s1 = infusedstring;isFused = bool } ;
mkPrep : (Number => Cgender => Str) -> Bool -> Prep = \l,bool ->
ln Prep {s = t ; s1 = infusedstring ; isFused = bool} ;};
```

g) Other categories

Quantifier was defined as strings that inflect for gender and number. Determiners show an indefinite number of people or objects [32]. They include but are not limited to every, much, all, etc. and inflect for gender plus an inherent number. Moreover, some come before the noun they modify while others come after the noun. To show the determiner's position in relation to the noun it modifies, isPre a Boolean parameter is used. True indicates it comes before and false shows it comes. Adverbs do not inflect; hence are mere strings in their definition. However, on adjectives, there were adverbs formed out of adjectives. Therefore, to accommodate them at the syntax phase, since adjective inflects for gender and number, the adverb was configured to inflect for agreement (gender, number and person). Person three(P3) was used as a constant.

B. Syntax

The syntax is implemented using the dominant SVO topology and the parameters were exchanged among the categories in order to ensure syntactic agreement (concord agreements). The V topology is also implemented primarily where personal pronouns are used as the subject(S), thus pro-drop of the subject since it’s represented in the verb using the subject marker. Finally, SV is implemented where the verb does not have a compliment.

The CN production rules were implemented in different ways. First, using the noun of either valency one or two, then
NP and a relational noun or usage of “of” together with a noun phrase. Finally, a CN was modified by an adjective, relative clause, adverb, sentence and noun phrase, respectively. Overall, nine production rules for CN were implemented.

Syntactically, determiners phrase can be formed from numerals and quantifiers, with the latter been the central and the former optional. Here the focus is constructing a determiner from more than one category (speech tags). Two production rules were modeled. In the first one, the determiner is formed from quantifier and numeral, while in the second rule, there is the addition of ordinal numeral. The determiner is a post modifier of a noun hence the reason the Boolean isPre is true. Figure 4 shows an example of rule one for the two-grammar using gloss “these seven”.

![Fig. 4. Determiner example.](image)

Adjective phrases (AP) are formed in several ways. One rule implemented simple positive degree AP while the Comparative degree AP was implemented using positive adjective plus noun phrase and the “than” string. AP rules were also made from a relational adjective. AP rules were formed by AP been modified by NP, reflexive pronoun, adverbs, sentence complement and noun phrases. Figure 5 exemplified AP formed by a comparative adjective with NP as its modifier, for the gloss better than some students.

![Fig. 5. Example of AP parse tree.](image)

The NP was implemented from the common noun, proper names, determiners, pronouns, and recursion of NP with adverbs, pre-determiners, and determiners. NP. The implementation used two parameters: case and agreement (concord), which are needed when combining the NP with a verb phrase. The case values were nom for the nominative case, while NPos was introduced to cater to noun phrases from personal and possessive pronouns. Since personal pronouns are NP and can be inferred from the verb’s subject marker morpheme, the field isPron was used to store the information on whether the current NP is a pronoun or not to enable future pro-drop when needed. Complex are formed using a pre- or post-modifier of NP. The pre-modifiers are pre-determiner and determiner, while post-modifiers are past participle verbs, relative clauses, and adverbs. In total, ten productions of NP were implemented. Figure 6 shows word alignment for NP “all my three black eyes” in English. The NP. consists of a predetermined, possessive pronoun, cardinal numeral, adjective and a noun.

![Fig. 6. Noun phrase word alignment.](image)

The Verb phrase implementation mirrored the structure of verb implementation. Therefore, its linearization is the same as that of the verb and the VP morphology paradigm regVP uses similar strings as those of the verb with only two addition, compl for the verb’s object and inf for infinitive verbs. The subcategorization of verbs was taken care of through compl (one place, two-place, and three-place verb). The VP complements that are listed below. In total, 21 syntax rules were implemented:

- Use of the verb or the verb phrase.
- Use of verb to be and its complements (auxiliary verbs).
- Use of adverbs complements.
- Verb passivation of the verb.
- Reflexive complement.

Fig. 7 shows an example of VP for the gloss “I read the best book”.

![Fig. 7. Egekusii VP parse tree.](image)

The next category to model was clauses: declarative, question and relative clauses. All clauses have undetermined polarity, tense and anteriority, which is fixed at the sentence level. The question clause uses the parameter QForm with values QDir and QIndir for direct and indirect questions.
The direct question clause is formed by changing the declarative clause's tone to high or using a question mark. Declarative clauses are formed using the SVO topology where the S is a noun phrase, while V is the s field of the Verb phrase and O is the compl field of the Verb Phrase. Figure 8 is an example of a clause for “oronṣana robwate emete ya eragi ya machani emenene na chinyoni chigotera ororo” with gloss “The forest has big green trees and birds sing there”. The parse tree shows the combination of NLP and VP makes both clauses. The VP in the first clause consists of the two-place verb with an NP as an object, while the second clause is made of a one-place verb and adverb.

The two ways used to implement question clauses (QCl) were through the yes or no answer questions or through Interrogative. These interrogatives are: interrogative Pronouns (IP), interrogative Adverbs (IAdv), Interrogative Quantifiers (IDet), copula interrogative complement (IComp) and their modifiers. The relative clause (RCI) was formed in three ways. The basic way is to use a clause. The second and third involve verb phrases and a sentence that lacks a noun phrase been modified by a relative pronoun (RP). In total 19 production rules were implemented.

The sentences were primarily implemented by fixing the tense, anteriority, polarity at the question, declarative and relative clauses. Other ways include the use of embedded sentences such as question sentences and infinitive verb phrase. An adverb can modify a sentence with a comma or not. Finally, sentences were constructed using the subjunctive, relative clause and imperative verbs. The utterance was not implemented from sentences, questions and imperatives but also using one word, especially where it is an answer to a question in the following categories: noun phrases, interrogative adverb, interrogative pronouns, common nouns, numerals, verb phrases, adjective phrase, adverbs, and interjections. Fig. 9 exemplifies one-word utterance.

Fig. 8. Clause/Sentence parse tree.

Fig. 9. Utterance Examples.
The phrase is the start category and three productions were implemented by prefixing and suffixing utterance with phrasal category and Noun phrase as a vocative (Voc), respectively [8].

Finally, production rules for coordination were implemented for the following categories: sentences, adverbs, interrogative adverbs, noun phrases, adjectives, relative sentences, common nouns and determiner phrases.

In total, at the syntax phase, the Ekegusii computational grammar had 163 production rules.

IV. RESULTS AND DISCUSSION

GF is a multilingual ecosystem; therefore, a machine translation from English to Ekegusii was set up to evaluate the grammar precision. Consequently, the Bilingual Evaluation Understudy (BLEU), Word Error Rate (WER) and Position Independent Error Rate (PER) metrics, which are commonly used metrics for evaluating machine translation [33], were used. BLEU (ranges from 0 to 1 or expressed as a percentage) demonstrated a good correlation of machine translation to human judgment [34]. PER and WER based on Levenshtein distance [35] are excellent metrics to investigate Ekegusii errors since this language has a lot of nasal insertion, deletion and substitution, especially the joining of morphemes at the word level. Besides, all-inclusive error analysis taxonomy [36] by comparing hierarchical [33], [37] and linguistic [36] taxonomies was used to analyze the errors resulting from the machine translation manually.

A 100-sentence test suite (in the English language) developed specifically for Bantu languages [15] was used to perform the machine translation. An Ekegusii expert translated the test suite into the Ekegusii language to act as the gold standard. The human translation was subjected again to another expert to confirm the translation correctness. The 100 English sentences were parsed (strings to abstract trees) and linearized (abstract trees to strings) to Ekegusii using the developed Ekegusii grammar. This machine translation output formed the candidate or target language translation. The machine translations, Human translation (gold standards) and the source language (English) were in text files. The three sets were compared using the online Tilde software to extract the BLEU score where the first two were of the same phrase length(n grams) are scored. To addressing fluency of the translation since GF is known to over generate, the longer n-gram (4-gram) is used. The Ekegusii 55.95% BLEU score is encouraging for a language with much morphophonological transformation. The metrics PER and WER were used to investigate errors because the former does not penalize position while the latter does and this had a huge effect on accuracy, especially where two consecutive adjectives were used in a sentence as illustrated by Fig. 9, where a sentence has correct translation. However, due to two consecutive adjectives: red(chimbarirri) and small (chinke) are interchanged in the candidate translation, it results in 50% WER and 0% PER. Besides, the implication is high on the BLEU score since it scores partly 22.59%. Table 5.5 shows 19.49% and 23.90% for PER and WER scores. An in-depth analysis of the errors by manual annotation using the comparative taxonomy is shown in Fig. 10.

Table V presents the extracted English to Ekegusii machine translation metrics score.

<table>
<thead>
<tr>
<th>TABLE V: TRANSLATION METRICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metrics</td>
</tr>
<tr>
<td>1-gram</td>
</tr>
<tr>
<td>2-gram</td>
</tr>
<tr>
<td>3-gram</td>
</tr>
<tr>
<td>4-gram</td>
</tr>
<tr>
<td>PER</td>
</tr>
<tr>
<td>WER</td>
</tr>
</tbody>
</table>

The BLEU score measures the similarity index by comparing the same phrase length(n-gram) between the target(candidate) and reference (gold standard) sentences. Though, 1, 2, 3 and 4 grams (phrase length) are scored. To...
near, far, remote). When all the variants are implemented in GF due to the large verb inflection table, the compiler took a long time (more than 8 hours) to process. In fact, in most of the cases, the process was killed. Therefore, for testing purposes, we only used one tense in each category and this led to the scenario shown in Fig. 11 where the human translation is in the remote past tense while the machine translation is in the far past tense. Therefore, coping with this limitation of time complexity in GF led to several errors.

![Fig. 11. Example of verb tenses error.](image)

Semantic errors occurred when action had to be explained by more than one word or the word used in a specific context never made sense and Fig. 11 shows an example of the former. Verbs contributed most of the errors through verb tenses and phonology vowel change (morphophonological transformation).

V. CONCLUSION

In this paper, we have formalized the Ekegusii grammar in the interlingual GF ecosystem. In extending the GF resource grammar, we have provided a computational grammar with a BLEU score of 55.95% that is a significant step toward creating a basic language resource kit (BLARK) for this under-resourced language. The grammar provides the platform of building controlled applications on top of it besides generating bilingual corpus, which can be used to experiment with data driven approaches. The future work proposed is to work on the morphophonological transformation in the verb phrase to increase accuracy.

REFERENCES