Árbol ancestral *

*Note: Sub-titles are not captured in Xplore and should not be used

1st Peña Andia Victor Angelo
Ciencia de la computación
UTEC
Lima, Peru
victor.pena@utec.edu.pe

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—En este proyecto que se implemento en c++ un traductor de términos ancestrales. Para esto se creo una gramática capaz de detectar cada una de las palabras y poder luego traducirlas.**

***Index Terms*—gramática, C++, traductor, términos ancestrales**

I. Introduction

Los lenguages son un conjunto de simbolos definidos por $\sum$, donde con los elementos en sigma es posible crear cadenas de simbolos finitas. Pero, para conocer si una determinada oracion o conjunto de palabras estan formadas correctamente se usa el concepto de gramática. Este concepto ayuda a definir correctas cadenas que puedan ser reconocidas por un automata [1].

Estos conceptos ayudan a definir un determinada maquina capaz de resolver uno o unos ciertos problemas, dependiendo de sus estados [2].

Para este problema se uso los conceptos anteriores para si conseguir que una cadena de palabras sea reconocida. Luego, esta cadenas sea inspeccionada si es correcta. Finalmente, si es correcta se muestra la traduccion de la frase, caso contrario se presenta un error y se interrumpe el proceso.

Por otro lado, se creo una representacion intermedia que expresa la relacion de padre o madre respecto a un funcion.

II. Planteamiendo del problema

Para resolver el problema existe un conjunto infinito A1 que tiene los elementos ancestrales a traducir $A1 = \{mother, father, grandmother, greatgrandfather, ...\}$, detro de este conjunto se identifico las palabras con las que la gramatica trabaja: father, mother, great y grand.

Con estos datos se analizo la formacion de una traduccion intermedia de tipo: $mother = mo()ofather = fa()$ o, $grandmother = g(mo())$.

Al tener las palabras pertenecientes a la gramatica y la conversion de palabra a representacion intermedia, es posible disear la gramatica adecuada para este lenguage. La gramatica diseada seria de este modo:

$$S- > GM|GF$$

$$G- > RD$$

$$R- > RA$$

$$F- > father$$

$$M- > mother$$

$$D- > grand$$

$$A- > great$$

Al tener las reglas por las cuales la gramatica se forma es posible comenzar de derecha a izquierda a analizar la palabra.

En la segunda parte del problema se pide analizar la respuesta de la primera parte y traducirlo. Para esto la funcion debe hacer uso de una gramatica similar a la usada para los imput's

$$S- > GM|GV$$

$$G- > RD$$

$$R- > RA$$

$$V- > vatter$$

$$M- > mutter$$

$$D- > gross$$

$$A- > ur$$

Con la gramatica anterior y el resultado de la funcion se logro traducir la frase y encontrar una representacion intermedia.

Finalmente, se recibe una oracion cullas palabras estan dentro del conjunto A3, siendo $A3 = \{The, mother, father, of, Mary, John\}$. Esta oracion pasara traducirce al lenguage A4, donde $A4 = \{Die, ein, mutter, vatter, von, Maria, John\}$

La gramatica del lenguage A3 es:

$$Sentence- > InitSentence2$$

Identify applicable funding agency here. If none, delete this.

$$Sentence2- > SecondSentence2|ONombre$$

$$Nombre- > Mary|John$$

$$Init- > Themother|Thefather$$

$$Second- > ofthemother|ofthefather$$

$$O- > of$$

La gramatica del lenguage A4 es:

$$Sentence2- > Second.Sentence2|GONombre$$

$$Nombre- > Mary|John$$

$$Second- > ur$$

$$O- > von$$

$$G- > gross$$

Con estas gramaticas creadas solo es necesario realizar un codigo que al leer las frases verifique que sean correctas, mientras se encarga de traducirlas.

III. Resolucion

El problema lo dividimos en dos pricipales partes, donde la primera se usa dos clases y en la segunda dos funciones. Se usa la funcion

$$string\ middleInterpretation(string\ word)$$

donde recibe como parametro $word$ que es la palabra que pasara a ser una funcion.

```
// functions.cpp
string middleInterpretation(string word){
  bool controlFoM = false, controlG = false;
  string partialWord;
  string middle;
  for (auto letter = word.rbegin(); letter !=
      word.rend() ; ++letter) {
    partialWord.insert(partialWord.begin(),*letter);
    if (partialWord == "father" and
        !controlFoM){
      middle.insert(0,"fa(");
      partialWord = "";
      controlFoM = true;
    }else if(partialWord == "mother" and
        !controlFoM){
      middle.insert(0,"mo(");
      partialWord = "";
      controlFoM = true;
    } else if(partialWord == "grand" and
        !controlG){
      middle.insert(0,"g(");
      middle.insert(middle.end(),')');
      partialWord = "";
      controlG = true;
    }else if(partialWord == "great" and
        controlG){
      middle.insert(0,"g(");
      middle.insert(middle.end(),')');
      partialWord = "";
    }else{
      if(partialWord.size() > 6 ){
        cout<<"Error en la frase:
            "<<partialWord<<endl;
        return "error";
      }
    }
  }
  if(!partialWord.empty()){
    cout<<"Error en la frase:
        "<<partialWord<<endl;
    return "error";
  }
  return middle;
}
```

En esta funcion, la variable locales controFoM se encarga de controlar si la frase ya recibio ”father” o ”mother”, ya que esto se debe recibir solo una vez. Por otro lado, controlG se encarga de asegurarse que se reciba solo una vez ”grand” y los demas sean ”great”. Luego, partialWord es la palabra que se forma concatenando cada caracter de derecha a izquierda. Los casos posibles en que la palabra parcial sea distinta a lo esperado se retorna ”error”. Si todo finalizaba como lo esperaba se devolvia la representacion intermedia.

Para la segunda parte del problema se uso una funcion que recibia la representacion intermedia y luego la traducia a aleman. Para esto se uso la funcion:

$$string\ finalInterpretation(string\ word)$$

. La funcion recibe el string $word$ que es la representacion intermedia a traducir.

```
string finalInterpretation(string word){
int count = 0;
bool controlFoM = false, controlG = false;
string partialWord;
string middle;
for (auto letter = word.rbegin(); letter !=
    word.rend() ; ++letter) {
partialWord.insert(partialWord.begin(),
    *letter);
if(partialWord == ")") {
count += 1;
partialWord = "";
}else if (partialWord == "fa(" and
    !controlFoM and count>0){
middle.insert(0,"vatter");
partialWord = "";
count -=1;
controlFoM = true;
}else if(partialWord == "mo(" and !controlFoM
    and count>0){
middle.insert(0,"mutter");
partialWord = "";
controlFoM = true;
count -=1;
} else if( partialWord == "g(" and controlFoM
    and !controlG and count>0){
middle.insert(0,"gross");
partialWord = "";
controlG = true;
count -=1;
}else if (partialWord == "g(" and controlFoM
    and controlG and count>0){
```

```
middle.insert(0,"ur");
partialWord = "";
count -=1;
}else{
if(partialWord.size() > 3 or count == 0){
cout<<"Error en la frase:
    "<<partialWord<<endl;
return "error";
}
}
}
return middle;
}
```

IV. Concluciones

References

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.