LanPro

Edgar Uriel Domínguez Espinoza

4 de julio de 2013

Resumen

Este documento presenta un primer diseño de LanPro (LANguage PROcessing), un lenguaje formal cuyo objetivo es el de describir lenguas humanas, útil en cualquier nivel de análisis y fácil de aprender.

1. Introducción

En 1956 John McCarthy (1927-2011) crea el concepto lógico-matemático de *Inteligencia Artificial*, y años más tarde propone la utilización de de expresiones simbólicas y el primer diseño del lenguaje formal LISP. A partir de entonces el concepto de *Inteligencia Artificial* se comercializa y grandes avances científicos se dan a conocer, las computadoras hablan, juegan ajedrez, pero estarán siempre limitadas por ser máquinas, aunque sean más rápidas que el cerebro humano nunca podrán mover el corazón, respirar, escribir, ver, oír, sentir y pensar al mismo tiempo; la computadora esta destinada a la personalización no a la imitación.

Sin embargo, si podemos procesar el lenguaje de forma efectiva. Con esta visión, los ingenieros en computación toma la información generada por los lingüistas pero al desconocer los fenómenos de ésta ciencia crean muchas herramientas que si bien muestran el resultado esperado, son ineficientes y el conocimiento lingüístico se pierde: sus códigos no reflejan lo que el análisis lingüístico muestra y los lingüistas se ven incapaces de corregirlos porque no saben leer esos códigos. De este modo, la ciencia ha sido detenida porque los lingüistas han sido incapaces de crear una notación clara que permita comunicar su conocimiento con otros científicos. Los modelos matemáticos, las ecuaciones químicas y los diagramas de flujo, utilizados en numerosas ciencias permiten modelar fenómenos científicos aprendiendo tan solo unas cuantas reglas y a partir de las estructuras más sencillas se describen los sistemas más complejos.

Es momento entonces de crear un modelo flexible que permita describir fenómenos lingüísticos sin importar el punto de vista teórico del lingüista[1], que permita por primera vez, distinguir entre lo lineal y lo no lineal, lo relacional y lo jerárquico, lo cuantitativo y lo cualitativo. En este breve trabajo retoma la

[1] A principios del siglo XX, se crea la física cuántica, que daría un giro total de la visión que se tenía de ésta ciencia hasta el momento, sin embargo Max Planck nunca tuvo que cambiar los modelos matemáticos.

idea de McCarthy[2] y basados en dos tipos de datos y dos estructuras se propone crear una herramienta que permita explicar fenómenos lingüísticos y hacerlos comprencibles para otros científicos interesados.

Cualquier lingüista puede aprender LanPro en un día, existen dos tipos de datos: átomos y símbolos; existen dos tipos de estructuras: pares de celda y listas. A partir de estos se pueden expresar y procesar fenómenos en diferentes niveles de análisis lingüístico usando una sola notación. Aún más allá, abre las puertas para hacer nuevos tipos de análisis, dejar las estructuras gráficas arbóreas binarias por estructuras que son más prometedoras como los gráficos ponderados, algunos análisis cualitativos lentos con métodos cuantitativos más eficaces[3], porque cuando usan los criterios correctos el análisis se sostiene por si mismo.

Hasta el momento LanPro es un dialecto de LISP, una virtualización del lenguaje, por lo que puede representar cualquier lengua de forma idealizada, este es un primer ensayo, hasta no estar completo se debe aclarar al lector que esto es solamente teoría.

2. Unidades fundamentales

2.1. Elementos de Descripción

2.1.1. Átomos

Un átomo es una combinación de los símbolos básicos del Alfabeto Fonético Internacional en conjunto con los modificadores de los mismos. Los números también son átomos y la combinación de números y letras pero esta combinación no puede iniciar con un dígito. Tampoco son átomos aquellas combinaciones que comienzan con modificadores.

Son ejemplos de átomos:

(1) L5

(2) 2343

(3) hola

(4) kɘ'βʌhk^{h}

(5) t͡ʑ^{ʸ}fs

Son ejemplos de no átomos:

(6) * 54dg
Inicia con un dígito

(7) * 'kasas
Inicia con un modificadores

[2]Idea que en aquellos años se gestaba entre los científicos del MIT, podemos relacionar fácilmente a Chomsky con McCarthy, aunque no puedo afirmar que uno haya leído al otro.

[3]Los criterios cualitativos que Porto Dapena señala en su Manual de Técnica Lexicográfica son equivalentes al Criterio de Costos de Michael P. Oakes.

(8) * ola mundo

Existe un espacio en medio

Al usar LinPro distinguiremos varios tipos de átomos dependiendo del nivel de análisis lingüístico y el fenómeno que nos ocupe en ese momento, así tendremos átomos fonémicos, silábicos, morfémicos, léxicos, etc. Para realizar análisis desde el punto de vista cuantitativo como puede suceder en el terreno fonético se usan los átomos numéricos. Sin embargo siempre tendremos a nuestra disposición algunos tipos de átomos:

Simbólico: Cuya característica principal es que nunca existirán dos símbolos iguales. Los símbolos sirven para asociar valores y tienen un carácter estrictamente funcional para el sistema descriptivo de la lengua.

Cadenas: En caso de que se requiera en un átomo un elemento que no lo sea, por ejemplo que tenga un espacio en blanco, este puede introducirse como cadena. Las cadenas siempre serán escritas entre comillas dobles “ ”.

***t* y *nil*:** Estos dos son átomos simbólicos estándar. Sirven para distinguir el final de una lista, como resultado de los operadores lógicos, en la definición de procesos, entre otras cosas porque tienen la característica de que *siempre se evalúan a si mismos*.

2.1.2. Listas

El segundo tipo de dato con el que trabaja LanPro son las listas. Una lista puede ser una secuencia de átomos separados por un espacio y encerrados entre paréntesis, incluyendo la posibilidad de que una lista contenga una sublista que cumple con las mismas características.

Definimos como **TÉRMINO** de una lista a un elemento de la misma, ya sea o no un átomo, por lo que una lista se define:

$$(\text{término}_1\ \text{término}_2\ \ldots \text{término}_k)$$

donde k es el número de elementos de la lista

Son ejemplos de listas:

Lista	Número de términos	Términos
(lista)	1	lista
((esto es) (una lista))	2	(esto es), (una lista)
(esto es una lista)	4	esto, es, una, lista

2.2. Elementos de Estructura

2.2.1. Apuntadores

Existen solo dos estructuras de datos necesarias y suficientes para el funcionamiento de LanPro, la primera estructura puede considerarse como un arreglo o una lista de símbolos (desde ahora denominaremos *diccionario* a esta estructura), cada símbolo tiene asociados dos apuntadores *V* y *O*.

El primer apuntador *V* asocia un valor al símbolo y el segundo apuntador *O* asocia una operación al símbolo.

2.2.2. Símbolos

Como ya fue mencionado, los símbolos son átomos con carácter funcional. Un símbolo tendrá tres funciones dentro de LanPro:

1. Evalúa al valor que tiene el apuntador *V*
2. Si es el primer elemento de una lista, evalúa el valor del apuntador *O*
3. Si el símbolo es el primer argumento de la operación **SETQ** el símbolo es usado pero no se evalúa ningún apuntador.

2.2.3. Par de celda

Un par de celda es la segunda estructura de datos necesaria en LanPro, de ella derivan todas las operaciones y es la forma básica de organización de información de la lengua que se busca describir. Un par de celda es como su nombre indica un par de datos, definido de la siguiente manera:

(fst . snd)

donde fst funcionará como discriminaste para una búsqueda eficiente.

Debe ponerse especial atención al espacio que hay entre los elementos y el punto, esta notación permite diferenciar un par de celda con la notación lingüística estándar de la sílaba que algún análisis podría requerir. A pesar de que un par de celda por si solo sirve para mantener orden entre pares de datos simples, lo más frecuente es anidar un par con otro para construir estructuras más complejas como árboles, los pares se anidan de la siguiente manera:

(a . (b . c))

Si el elemento snd del último par de celda anidado es nil entonces la estructura puede representarse en forma de lista, es decir:

(a . (b . (c . nil))) = (a b c)

2.2.4. Operadores

Un operador es un símbolo que en el diccionario tiene el apuntador *O* asociado a una estructura de pares de celda. A continuación un ejemplo básico.

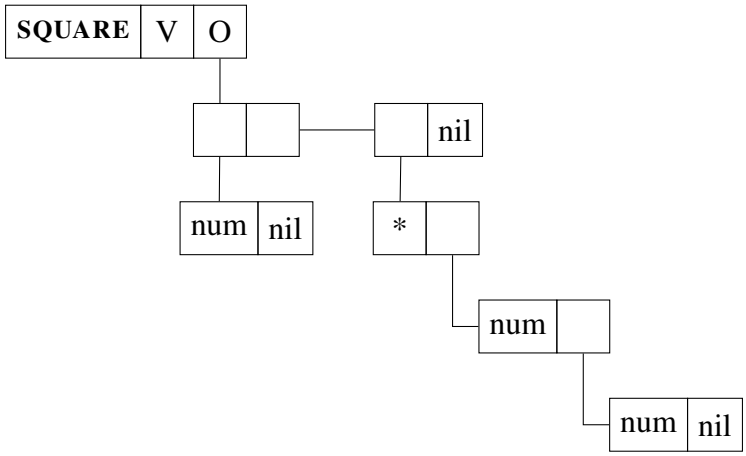

Figura 1: Ejemplo de un operador

El operador **SQUARE** de la figura 1 recibe como argumento un número y lo multiplica por si mismo, escrito en LanPro, el operador es el siguiente:

```
1   (DOP square (num)
2     "SQUARE recibe un numero y lo multiplica por si mismo"
3     (* num num))
```

Un ejemplo de uso es el siguiente:

```
1   (SQUARE 5)
2   25
```

Hay varios detalles que deben mencionarse a pesar de la sencillez del ejemplo, el primer detalle es el uso del operador **DOP** para definir **SQUARE**, este primer operador es parte de LanPro.

En la línea 2 de la definición del operador tenemos un comentario que describe el operador que estamos escribiendo, este comentario es optativo pero útil porque hará que la descripción lingüística sea auto-documentada, es decir, para entender LanPro solo basta con conocer la estructura básica de este documento porque cualquier particularidad lingüística o de cualquier otra naturaleza, sin importar la complejidad de la misma puede ser documentada en el lugar preciso.

Para cerrar los detalles sobre esta función, cabe mencionar que existen estructuras arbóreas asociadas frecuentemente a LanPro, estos árboles se recorren en preorden (como en gran parte de las teorías lingüísticas que se apoyan en árboles) y por lo tanto la notación de escritura, tal como se ve en el ejemplo, es prefija[4]. La explicación para la obtención de estos árboles se encuentra en la sección 3. La figura 2

[4]La evaluación de los operadores se realiza de izquierda a derecha comenzando por las listas más internas de la expresión.

muestra la estructura arbórea usada en el operador **SQUARE**, la estructura es un producto aritmético.

Figura 2: Estructura arbórea utilizada en la figura 1

Existen operadores fundamentales para el funcionamiento de LinPro, estos símbolos se pueden describir como *palabras reservadas* escritas en mayúsculas y siempre al inicio de una lista. Son operadores fundamentales:

Operador: QUOTE

Número de argumentos: 1

Argumentos: Un término cualquiera

Resultado: El argumento

Ejemplo: (**QUOTE** (a c)) $\rightarrow$ (a c)

Operador: FST

Número de argumentos: 1

Argumentos: Un par de celda o una lista no vacía

Resultado: El primer término de la lista o el primer elemento del par

Ejemplo: (**FST** (**QUOTE** (a c))) $\rightarrow$ a

Operador: SND

Número de argumentos: 1

Argumentos: Un par de celda o una lista no vacía

Resultado: El resto de la lista después de borrar el primer término o el segundo elemento del par

Ejemplo: (**SND** (**QUOTE** (a c))) $\rightarrow$ c

Operador: PAR

Número de argumentos: 2

Argumentos: Dos términos cualquiera

Resultado: Crea un par de celda tal que su fst es el primer argumento y su snd es el segundo argumento

Ejemplo 1: (**PAR** (**QUOTE** a) (**QUOTE** c)) → (a . c)

Ejemplo 2: (**PAR** (**QUOTE** a) (**QUOTE** (b c))) → (a b c)

Operador: ATOM

Número de argumentos: 1

Argumentos: Un término cualquiera

Resultado: Retorna ***t*** si el argumento es un átomo; ***nil*** en otro caso

Ejemplo: (**ATOM** (**QUOTE** (a c))) → ***nil***

Operador: EQ

Número de argumentos: 2

Argumentos: Dos átomos

Resultado: Retorna ***t*** si los átomos son iguales; ***nil*** en otro caso

Ejemplo: (**EQ** (**QUOTE** a) (**QUOTE** a)) → ***t***

Operador: NULL

Número de argumentos: 1

Argumentos: Un término cualquiera

Resultado: Retorna ***t*** si el argumento es una lista vacía; ***nil*** en otro caso

Ejemplo: (**NULL** (**QUOTE** ())) → ***t***

Con base en estos operadores se pueden construir operadores más complejos sin embargo existen una serie de operadores ya definidos en la lengua que facilitan la estandarización de las descripciones.

3. Escritura de Descripción Lingüística

3.1. Equivalencia de estructuras arbóreas binarias

Para ejemplificar el uso de LanPro para describir una lengua, nos basaremos en modelos con los que posiblemente este familiarizado el lector, primero tomaremos la estructura básica de la teoría X-Barra y en seguida mostramos la misma estructura representada por un árbol de pares de celda:

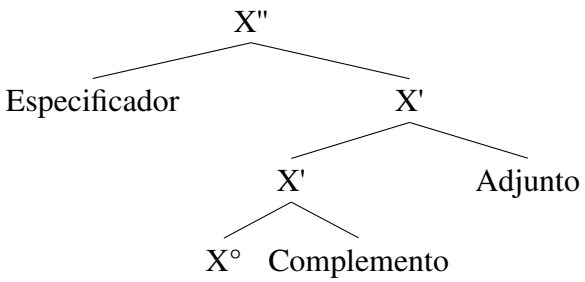

Figura 3: Estructura arbórea de un sintagma, teoría X-barra

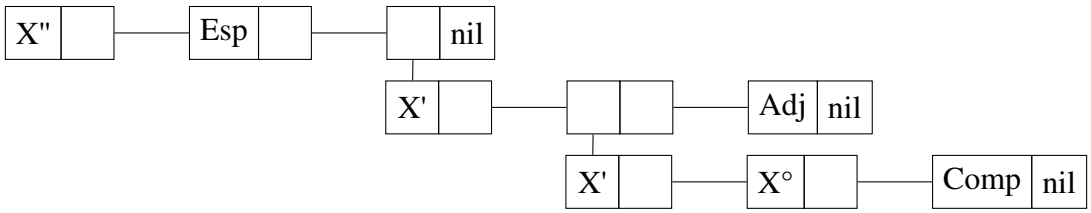

Figura 4: Estructura arbórea de un sintagma, LanPro

La estructura arbórea de la figura 4, ya no es propiamente un árbol binario en LanPro (si lo fuera, tendríamos que conservar únicamente las hojas del árbol) pero si es la estructura que posee los mismos elementos y características que el original[5] además puede representarse con pares de celdas para obtener una secuencia plana de la siguiente manera:

(9) (X" . (Esp . ((X' . ((X' . (X° . (Comp . nil))) . (Adj . nil))) . nil)))

Ahora por partes iremos simplificando (9) en listas para quitar los puntos y paréntesis de algunos pares de celda. La primera de ellas son los tres elementos más profundos en el árbol:

Figura 5: Primera parte de la figura 4

(10) (X' . (X° . (Comp . nil))) = (X' X° Comp)

[5]El objetivo de este apartado no es crear una teoría nueva a través de LanPro solo mostrar una equivalencia de nomenclatura con la teoría X-barra, por supuesto, LanPro puede formalizar cualquier teoría actual o futura que sea computable.

Continuamos sustituyendo la lista obtenida en la estructura de pares de celda inicial y repetimos la misma simplificación:

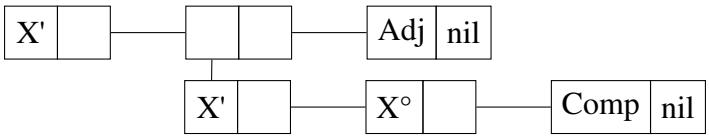

Figura 6: Segunda parte de la figura 4

(11) (X' . ((X' X° Comp) . (Adj . nil))) = (X'(X' X° Comp) Adj)

Hacemos este procedimiento por última vez para obtener una estructura simplificada de todo el árbol donde la estructura de pares anidada ya no es evidente:

(12) (X" . (Esp . ((X' (X' X° Comp) Adj) . nil))) = (X" Esp (X' (X' X° Comp) Adj))

3.2. Estructuras arbóreas en Fonología

El modelo autosegmental-jerárquico ha usado una estructura arbórea no binaria para explicar diversos fenómenos de la lengua y el diseño de LanPro puede simplificar cualquier estructura compleja con relativa sencillez, la figura 7 ilustra una estructura arbórea básica y en 13 la estructura plana equivalente en LanPro.

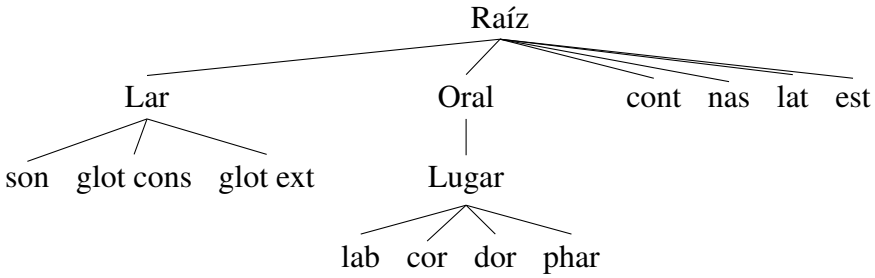

Figura 7: Árbol de rasgos según el modelo autosegmental-jerárquico

(13) (Raiz (Lar son g.cons g.ext) (Oral (Lugar lab cor dor phar)) (cont nas lat est))

(14) ((reson cons aprox) (Lar son g.cons g.ext) (Oral (Lugar lab cor dor phar)) (cont nas lat est))

Para completar al máximo esta estructura general, podemos decir que el nodo raíz de la figura 7 esta compuesto por los rasgos resonante, consonántico y aproximante por lo que podemos sustituir claramente el símbolo "/" con una lista con estos elementos como en (14) y podemos hacer lo mismo expandiendo las hojas para poder colocar los rasgos pertinentes.

Ahora en base a esta lista podemos representar los sonidos de una lengua basados la lista ordenada de rasgos:

(15) ((+cons) (Lar -son) (Oral (Lugar lab cor)) (-cont))
Es una consonante bilabial oclusiva sorda (p)

(16) ((-reson -cons) (Lar g.ext) (+cont))
Es la glotal fricativa sorda (h)

(17) ((-reson +cons) (Lar -son g.ext) (Oral (Lugar lab)) (+cont))
Es una consonante bilabial oclusiva sorda glotalizada (p^h)

(18) ((+reson +cons) (Lar +son) (Oral (Lugar cor)) (+cont))
Es una consonante coronal, vibrante múltiple (r)

(19) ((+reson +cons) (Lar +son) (Oral (Lugar cor)) (-cont))
Es una consonante coronal, vibrante simple (ɾ)

3.3. Asignación de valor

Como ya se dijo en la sección 2.2.2 para asociar formalmente el símbolo con un valor se usa el operador **SETQ**. Entonces el [x] está definido como una lista de la forma de (14). En LanPro esto se escribe:

```
1  (SETQ r (QUOTE ((+reson +cons) (Lar +son) (Oral (Lugar cor)) (+cont))))
```

En la expresión anterior observamos que el primer argumento de **SETQ** es el símbolo destino de la asignación y el segundo argumento es una expresión cuyo resultado se asignará al símbolo.

4. Ejemplo de descripción de un fenómeno

La descripción de un fenómeno en LanPro se ejecuta normalmente de forma interpretativa e interactivamente. En su forma más sencilla, un operador o un proceso lingüístico se presenta como una expresión completamente puesta entre paréntesis con todos los operadores necesarios en la forma prefija. Todas las variables tienen valores átomos o listas. A continuación iremos desarrollando la estructura de la descripción del *Principio de Contorno Obligatorio (PCO)* que es tan importante en Fonología porque ayuda a entender muchos comportamientos en las lenguas.

El PCO establece:

(20) Se prohíben elementos idénticos adyacentes [Núñez et al., 1999]

Entonces el primer paso natural es saber si dos elementos adyacentes violan este principio, para eso examinaremos los efectos de el *PCO* sobre la palabra [karreta]. Estos efectos son conocidos ya que de dos segmentos idénticos e individuados obtendremos la unión múltiple de dos posiciones en el esqueleto con un segmento [Núñez et al., 1999].

Para saber si dos elementos adyacentes incumplen (20) necesitamos primero definir los segmentos a probar, en este caso probaremos [r] que aparece en la palabra [karreta].

Este segmento ya lo hemos definido anteriormente en la sección 3.3 por lo que procedemos a realizar un operador sencillo el cual tomará como argumentos dos segmentos y los comparará, si incumple el *PCO* regresará *t*, *nil* en otro caso:

```
(DOP pco (arg1 arg2)
  "Prueba si se incumple el principio de contorno obligatorio"
  (COND ((EQUAL arg1 arg2))))
```

En el operador definido arriba, **COND** se traduce como una sentencia if con la que se dispone los lenguajes formales y **EQUAL** es la igualdad, no se usa el símbolo de igual (=) para poder hacer la distinción con las operaciones aritméticas.

La realización en nuestra descripción es la siguiente:

```
(PCO r r)
t
```

Hasta el momento, el operador **PCO** funciona solo para identificar segmentos que incumplen (20). Ahora necesitamos que el operador **PCO** no solo identifique sino también que haga la asociación de los segmentos:

```
(DOP pco (arg1 arg2)
  "Realiza el principio de contorno obligatorio"
  (COND ((EQUAL arg1 arg2) arg1)
        ((LIST arg1 arg2))))
```

El nuevo operador **PCO** devolvera un solo segmento si los argumentos incumplen (20) pero si no es el caso devolvera los dos argumentos.

```
(PCO r r)
((+reson +cons) (Lar +son) (Oral (Lugar cor)) (+cont))
(PCO (QUOTE r) (QUOTE r))
r
(PCO r h)
(((+reson +cons) (Lar +son) (Oral (Lugar cor)) (+cont)) ((-reson -cons) (Lar g\.ext) (+cont)))
(PCO (QUOTE r) (QUOTE h))
(r h)
```

Arriba evaluamos varias veces **PCO** con resultados diferentes. En la línea 1 evaluamos los fonemas y el resultado es el valor del fonema resultante. En la línea 3 aunque hacemos lo mismo, en el argumento señalamos que no evalúe el símbolo al final. En la línea 5 y 7 colocamos ejemplos de dos segmentos que no incumplen (20) para que se haga un contraste entre resultados de nuestro operador.

Además hasta ahora el operador definido cumple bien (20) en la lengua española porque evaluará los

segmentos [i] y [y] como diferentes a pesar de que ambas son coronales, esto permite que la palabra silla se realice como [siya] tal y como aparece en varios dialectos hispanoamericanos [Núñez et al., 1999].

Sin embargo, Nuñez también señala que en tigrinia, una lengua Eritrea[6] la forma /baräk-ka/ "bendito seas" se realiza como [baräx-ka], pero las formas tautosilábicas surgen inalteradas como es el caso de /fäkkärä/ que se realiza como [fäkkärä]. Nuñez señala que el motivo de la inaplicación es "la distinción fonológica que subyace estas secuencias de segmentos". Lo que si es un hecho es que el *PCO* debe realizarse de manera diferente en tigrinia que en español y debemos modificar el operador **PCO** para que cumpla con lo que sucede en la lengua.

```
(DOP pco (arg1 arg2 &optional arg3)
  "Realiza el principio de contorno obligatorio"
  (COND ((NULL arg3) (COND ((EQUAL arg1 arg2) arg1)
                     ((LIST arg1 arg2))) )
        ((COND ((EQUAL arg1 arg2) (LIST arg3 arg2))
               ((LIST arg1 arg2))))))
```

Esta última versión de **PCO** agrega un argumento opcional que permite describir que es lo que pasa en tigrinia respecto al *PCO* únicamente y no a los contextos en los que se evita el principio porque Nuñez no da detalles suficientes como para definir las caracteristicas de estos segmentos tal y como hicieramos con los fonemas. De cualquier modo, esas definiciones no afectan **PCO** sino los contextos en los cuales podemos aplicarlos.

```
(pco k k x)
(((-reson +cons) (Lar -son) (Oral (Lugar dor)) (+cont)) ((-reson +cons) (Lar -son) (Oral (Lugar dor)) (-
    cont)))
(pco (QUOTE k) (QUOTE k) (QUOTE x))
(x k)
```

5. Presentación del documento

Finalmente debemos resumir como sería la descripción limpia del *PCO*. Esta descripción consta de tres apartados, el primer apartado contendrá todas las declaraciones de los fonemas usados en la descripción, de hecho este primer apartado debe obedecer completamente a una descripción de los fonemas de la lengua con capacidad para ser reutilizado en diversos estudios.

```
;; Fonemas existentes en la lengua
(SETQ r (QUOTE
```

[6]El Estado de Eritrea es un país situado al noreste de África. Limita al norte y al oeste con Sudán; al sur con Etiopía y Yibuti; el este del país posee una extensa costa con el mar Rojo. Su nombre proviene del griego "eritros", que quiere decir "rojo". Se independizó en 1993, lo que lo convierte en uno de los estados más jóvenes del mundo. Su capital y ciudad más poblada es Asmara.

```
          ((+reson +cons)
           (Lar +son)
           (Oral
            (Lugar cor))
           (+cont))))

(SETQ h (QUOTE
          ((-reson -cons)
           (Lar g.ext)
           (+cont))))


(SETQ k (QUOTE
          ((-reson +cons)
           (Lar -son)
           (Oral
            (Lugar dor))
           (-cont))))

(SETQ x (QUOTE
          ((-reson +cons)
           (Lar -son)
           (Oral
            (Lugar dor))
           (+cont))))

[...]

(PROVIDE (QUOTE fonemaEsp))
```

El segundo apartado contendrá las operaciones utilizadas en el análisis, en este caso constará únicamente de **PCO** en la última realización obtenida, ya que aunque así podrá ser utilizada también en caso de realizar un análisis sobre la lengua tigrinia.

```
(DOP pco (arg1 arg2 &optional arg3)
  "Realiza el principio de contorno obligatorio"
  (COND ((NULL arg3) (COND ((EQUAL arg1 arg2) arg1)
                     ((LIST arg1 arg2))) )
        ((COND ((EQUAL arg1 arg2) (LIST arg3 arg2))
               ((LIST arg1 arg2))))))
```

```
7
8  (PROVIDE (QUOTE pco))
```

Estos dos primeros apartados facilitan la organización, reutilizamiento, procesamiento y lectura de la información, a su vez que hacen la función de librerías dentro de un lenguaje formal.

Finalmente esta el apartado inmediato para el lector en el que se detalla al lector el análisis de lengua. Los comentarios sobre la lengua estarán precedidos por un punto y coma (;) distinguiendolos así de los ejemplos funcionales. La descripción comenzará mencionando el nombre de los otros apartados a los que debe referirse el lector para mayor detalle aunque también pueden aparecer en otros lugares a lo largo del análisis.

```
1  ; Apartados necesarios
2  (REQUIRE (QUOTE pco))
3  (REQUIRE (QUOTE fonemaEsp))
4
5  ; El Principio  de Contorno Obligatorio establece  el impedimento de la
       existencia de elementos
6  ; iguales adyacentes.
7
8  ; En  nuestra lengua,  los dos  segmentos de  la  palabra [karreta]  se
       realizan  como uno  solo
9  ; [kareta] de la siguiente manera:
10
11 (PCO (QUOTE r) (QUOTE r))
12 r
13 [...]
14
15 ; Sin embargo en otras lenguas como el tigrinia se realiza de la siguiente
       manera:
16
17 (PCO (QUOTE k) (QUOTE k) (QUOTE x))
18 (x k)
19
20 [...]
```

6. Conclusión

Aunque este primer diseño de LanPro puede ser insuficiente para una implementación completa, si detalla la base teórica del procesador y sus capacidades. El número limitado de estructuras básicas

sencillas permiten comprender fácilmente al lector como utilizar LanPro para realizar análisis flexibles y detallados de las lenguas.

Además, la forma en la que está organizada la información permite una lectura más sencilla al lector y por supuesto una implementación fácil para un análisis lingüístico con ayuda de Tecnologías de Información, lo cual equivale a darle al lingüista una herramienta de desarrollo tecnológico y científico potente, capaz de ahorrarle horas de trabajo y precisión en su análisis.

LanPro es capaz de describir cualquier estructura de datos que pueda ser útil al lingüista (no solo árboles binarios) y permite realizar análisis cuantitativos, cualitativos o bien combinarlos lo cual, desde el punto de vista del autor, es necesario para poder dar cuenta de muchos fenómenos lingüísticos.

Sin embargo, no todo es positivo, es necesario realizar una lista más amplia de operaciones fundamentales, además de la conveniencia de dejar el lenguaje de expresión simbólica (expresiones-S) a metaexpresiones (expresiones-M) para flexibilizar aun más la estructura del procesador.

Referencias

[Bell, 2004] Bell, M. (2004). *Understanding English Spelling*. Pegasus Educational.

[Bromley and Lamson, 1987] Bromley, H. and Lamson, R. (1987). *LISP Lore: A Guide to Programming the LISP Machine*. Springer.

[Glickstein, 2010] Glickstein, B. (2010). *Writing GNU Emacs Extensions: Editor Customizations and Creations with Lisp*. O'Reilly Media.

[Holm, 2008] Holm, N. M. (2008). *Sketchy LISP: An Introduction to Functional Programming in Scheme*. Lulu.

[McCarthy et al., 1965] McCarthy, J., of Technology. Computation Center, M. I., and of Technology. Research Laboratory of Electronics, M. I. (1965). *Lisp one five programmer's manual*. Massachusetts Institute of Technology.

[McCarthy, 1960] McCarthy, J. L. (1960). Recursive functions of symbolic expressions and their computation by machine, part I. *Communications of the ACM*, 3(4):184–195.

[Núñez et al., 1999] Núñez, R., Front, A., Vives, P., and Hualde, J. (1999). *Fonología generativa contemporánea de la lengua española*. Georgetown University Press.