Alternativa para especificación sintáctica: Gramática de sistemas de íconos generalizados-Gsig.

Juan Francisco Díaz Frías [*]
Escuela de Ingeniería de Sistemas y Computación, Universidad del Valle
Cali, Valle, Colombia
jdiaz@eisc.univalle.edu.co

and

Carlos Andrés Tavera Romero [**]
Escuela de Ingeniería de Sistemas y Computación, Universidad del Valle
Cali, Valle, Colombia
catavera@eisc.univalle.edu.co

Abstract

This paper presents a new mechanism of grammar specification for Visual Programming Languages named Generalized Icon System Grammar.

Keywords: Visual Programming Languages, Extended Positional Grammar, Textual specification.

Resumen

Este artículo presenta un nuevo mecanismo de especificación gramatical para Lenguajes de Programación Visual denominado Gramática de Sistemas de Íconos Generalizados.

Palabras claves: Lenguajes de Programación Visual, Gramática Posicional Extendida, Especificación textual.

[*]Docteur en Informatique, Universite de Paris XI; Director del grupo de investigación AVISPA, capítulo Univalle.
[**]Candidato al título de Doctor en Ingeniería, Universidad del Valle; Investigador del grupo AVISPA, capítulo Univalle.

1. INTRODUCCIÓN

La programación visual evoluciona al surgir la pregunta: porqué la insistencia de comunicarse con las computadoras usando lenguajes textuales cuando se podría tener mayor productividad y hacer más accesible este campo si se dibujaran las imagenes que llegan a la mente al considerar soluciones a algún problema? Las personas construyen imágenes mentales, además, se ha demostrado que los Lenguajes de Programación Textuales son difíciles de aprender para muchas personas (como en el estudio presentado en [8]). Estas ideas motivaron el estudio de Lenguajes de Programación Visual (VPL por su sigla en inglés). La investigación sobre la formalización de VPL se ha concentrado principalmente en dos líneas de investigación:

- Gramáticas para VPL, uno de sus grandes aportadores es el profesor Costagliola[1] cuyos trabajos más recientes se encaminan a la creación de Parsers para VPL [3] cuando estos utilizan un tipo de especificación gramátical denominada Gramáticas Posicionales Extendidas [2] (desde ahora GPE). La principal dificultad de esta propuesta radica en que los lenguajes diseñados siguiendo estas gramáticas deben ser simples, debido a que el análisis sintáctico propuesto requiere grandes cálculos y, como no está enfocado al almacenamiento, tampoco es fácil extraer un programa visual desde su representación textual; de igual manera, las demostraciones sobre propiedades formales del lenguaje exigen convertir el código en otro tratable matemáticamente.

- Semántica de VPL, un grupo de investigación sobresaliente en esta área es el del profesor Erwig[2] donde su desarrollo más novedoso acerca de los VPL es un estudio sobre la Inferencia Visual de Tipos [7], empleando un trabajo anterior sobre la Especificación Semántica de VPL [6] en el cual presenta la importancia de una especificación textual de características como la semántica, para poder efectuar demostraciones en algún momento.

Desde lo aplicativo, recientes investigaciones se han interesado en el paradigma de programación Spreadsheet (estilo hoja de cálculo) como el presentado en [9], donde la especificación gramatical se facilita por la manera en la que se presenta la información y porque está orientado a la programación funcional donde la sintáxis no es muy extensa. Al ver el panorama, no hay grandes cambios en las formas de especificación gramatical desde las GPE y la utilización de otra implica la creación de nuevas maneras de Parsing o la disminución de la expresividad del lenguaje; igualmente, los Programas Visuales por ser dibujos requieren una representación textual no sólo para su almacenamiento y posterior extracción, si no al realizar los procesos de los cuales sólo se conocen métodos textuales, esta necesidad no está claramente contemplada por las especificaciones gramaticales existentes.

Inicialmente, el grupo AVISPA[3] (capítulos Universidad del Valle y Universidad Javeriana de Cali-Colombia) empleó las GPE para la implementación de GraPiCO[4]; pero en el desarrollo se encontró que estos mecanismos no se ajustaban a los requerimientos, dado que representaban un análisis sintáctico complicado y el código de almacenamiento era de difícil tratamiento. Por todas las razones expuestas, se diseñó una forma de especificación para VPL denominada **Gsig**, que permitió: el empleo de dibujos, la utilización de mecanismos de Parsing conocidos y probados como el Descendente Predictivo para gramáticas LL(1) y la obtención de un código textual para almacenar los programas. De esta manera, los aportes que se presentarán de las **Gsig** son: un mecanismo de especificación gramatical que utiliza métodos de análisis sintáctico conocidos, flexibilidad en los íconos empleados y la

[1] http://www.dmi.unisa.it/people/costagliola/www/www/home/indice.htm

[2] http://web.engr.oregonstate.edu/∼erwig/

[3] Grupo de investigación en **A**mbientes **VIS**uales de **P**rogramación **A**plicativa.

[4] Lenguaje visual con base en el cálculo PiCO (π Calculus and Concurrent Objects, desarrollado por el grupo AVISPA).

obtención de un código textual para la representación de los Programas Visuales permitiendo desde su dibujo el almacenamiento y viceversa, y su tratamiento formal como por ejemplo, comprobación de la semántica y diseño de reglas de traducción.

Este artículo tratará el siguiente orden de ideas: 1) Introducción general, 2) Repaso de las Gramáticas Posicionales Extendidas, 3) Propuesta de las **Gsig**, 4) Referencia al Cálculo PiCO, base del Lenguaje GraPiCO y 5) Ejemplo de la utilización de las **Gsig** en el nuevo Lenguaje Visual GraPiCO.

2. ESPECIFICACIÓN SINTÁCTICA DE LENGUAJES VISUALES

En esta sección, se recuerda la definición de GPE, una especificación sintáctica para VPLs.

2.1. Gramáticas Posicionales Extendidas

Las GPE presentadas en [2] extienden las gramáticas libres de contexto para los lenguajes textuales hacia gramáticas para lenguajes visuales, mediante nuevas relaciones adicionales a la concatenación. Para establecer estas relaciones se requiere un conjunto de identificadores de relación y un evaluador pictórico para verificar su comportamiento.

Las GPE están compuestas por producciones de la forma:

$$A \rightarrow \overbrace{X_1\ R_1\ X_2\ R_2\ \cdots X_{m-1}\ R_{m-1}\ X_m}^{Lista\ de\ símbolos\ relacionados}\ \Delta, \Gamma \tag{1}$$

Donde:

- A: es un símbolo no_terminal.
- $Lista\ de\ símbolos\ relacionados$: es una sucesión de símbolos del alfabeto y de relaciones que un par de símbolos consecutivos cumplen, por ejemplo, $X_1\ R_1 X_2$ representa que X_1 está relacionado con X_2 mediante la relación R_1.
- Δ: es un conjunto de *reglas para sintetizar* los valores de los atributos sintácticos de A a partir de los atributos de $X_1,\ X_2,\ \cdots,\ X_m$.
- Finalmente, Γ: es un conjunto de *triplas de inserción* de la forma $(N, Cond, \Delta)$. Estas últimas, se emplean para insertar dinámicamente nuevos símbolos terminales en la frase de entrada durante la etapa de análisis sintáctico.
  Donde:
  - N: es el símbolo terminal a ser insertado.
  - $Cond$: es una precondición que se debe cumplir para insertar N.
  - Y, Δ: es la regla usada para calcular los atributos de N a partir de los atributos de $X_1,\ X_2,\ \cdots,\ X_m$.

Por ejemplo, en la figura 1 se presenta un objeto visual compuesto por un conjunto de íconos con etiquetas $Ic_1, Ic_2, ..., Ic_n$ dentro del ícono Ic'. Se desea definir una gramática que lo describa, además de un mecanismo para almacenar los atributos de los íconos. Una GPE que represente el conjunto de íconos presentados en la figura 1 se muestra en la figura 2.

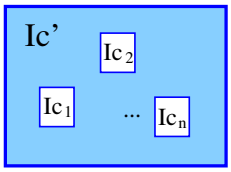

Fig. 1: Conjunto de íconos con etiquetas Ic_1, Ic_2, ... , Ic_n dentro del ícono Ic'.

1. $Ic' \rightarrow GrupoIc'$
2. $Ic \rightarrow Ic_1 \qquad \Delta(Ic = Ic_1)^a$
3. $Ic \rightarrow \vdots$
4. $Ic \rightarrow Ic_n \qquad \Delta(Ic = Ic_n)$
5. $GrupoIc' \rightarrow GrupoIc \, \langle \mathbf{Junto}^b \rangle \, Ic$

   $\Delta(GrupoIc' = f(GrupoIc,\ Ic))^c$

   $\Gamma : \{(CONTEXTO^d, |GrupoIc| > 0^e,$

   $CONTEXTO = g(GrupoIc,\ Ic)^f)\}$
6. $GrupoIc' \rightarrow GrupoIc \, \langle \mathbf{R}^g \rangle \, CONTEXTO$

   $\Delta(GrupoIc' = CONTEXTO)$
7. $GrupoIc' \rightarrow Ic$

   $\Delta(GrupoIc' = Ic)$

1. Un grupo de íconos es un conjunto de íconos Ic'.
2.- 4. Todo ícono Ic_i (del tipo i) es un ícono genérico Ic.
5. Un grupo de íconos al lado de un ícono es un grupo de íconos.
6. Un grupo de íconos y un CONTEXTO es un grupo de íconos.
7. Un ícono genérico es un grupo de íconos.

[a] Los atributos de Ic_1 son tomados por Ic.
[b] **Junto** en una relación cumplida por cualquier par de íconos que comparten la misma ventana.
[c] Los atributos de $GrupoIc'$ son calculados mediante la función f.
[d] CONTEXTO es un terminal abstracto insertado dinámicamente en la frase de entrada durante el análisis sintáctico.
[e] Condición que asegura que un grupo de íconos está compuesto por, al menos, un ícono.
[f] Los atributos de CONTEXTO son calculados mediante la función g.
[g] **R** es una relación que siempre es satisfecha por cualquier par de símbolos.

Fig. 2: Gramática Posicional Extendida de un grupo íconos: $Ic_1, Ic_2, \ldots, Ic_n$ dentro del ícono Ic'.

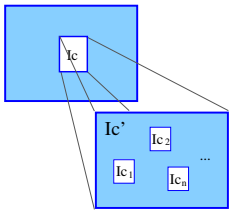

Fig. 3: Ícono $Ic^{\wr}$ (representa la figura en su totalidad): expansión del ícono etiquetado con Ic en los íconos con etiquetas Ic_1, Ic_2, ... , Ic_n dentro del ícono Ic'.

- $Ic^{\wr} \rightarrow Ic \ \langle \textbf{expansión} \rangle \ Ic'$

Fig. 4: Gramática Posicional Extendida del ícono $Ic^{\wr}$, compuesto de Ic y su expansión Ic'

Definición 1 *Se llamará* **Expansión** *a la acción resultante de activar un ícono para visualizar su contenido. En la mayoría de entornos gráficos una expansión se logra presionando una cantidad determinada de veces un botón del ratón. En adelante, se representará visualmente una expansión mediante el dibujo de la proyección del ícono expandido hacia el conjunto de íconos producto de la expansión, como se presenta en la figura 3.*

Si se quiere modelar el caso de la figura 3 en GPE, la creación de una nueva relación *Expansión* es requerida. Una gramática posicional para este caso es presentada en la figura 4.

En seguida, se presentará la nueva propuesta por la cual pueden modelarse los lenguajes visuales.

3. GRAMÁTICAS DE SISTEMAS DE ICÓNICOS GENERALIZADOS - GSIG

En esta parte se hará una introducción a la teoría de la composición de los Sistemas Icónicos y su mecanismo de especificación. Luego, se presentarán formalmente las **Gsig** y ejemplos de su uso.

3.1. Composición de los Sistemas Icónicos

Empleando la teoría presentada en [4], donde se introduce el término Sistema Icónico[5].

Todo Ícono Generalizado está compuesto de dos partes:

1. Parte Física: La información de la imagen como tal. Ejemplos de este tipo de datos son: la localización del archivo almacenando la gráfica y la localización en el espacio de trabajo.

2. Parte Lógica: El significado de la imagen. Aquí está comprendida información como: a qué clase de sintagma[6] corresponde el ícono y por cuáles íconos está compuesta la imagen.

Esta caracterización de la composición de los componentes visuales será utilizada en la definición formal de las **Gsig** presentada a continuación en la sección 3.2.

[5]Conjunto estructurado de Íconos Generalizados relacionados.

[6]Combinación ordenada de significantes que interactúan formando un todo con sentido, dentro de un conjunto de reglas y convenciones sintácticas.

3.2. Casos de especificación de constructores visuales

Existen dos clases de especificación de los constructores visuales:

1. Especificación de Constructor Visual Simple: Es la especificación que se efectúa a un ícono en particular sin tomar su entorno.

2. Especificación de Constructor Visual Compuesto: Es la especificación realizada al conjunto de Constructores Visuales Simples que lo conforman.

A continuación se presentan las **Gsig** mediante los mecanismos para lograr los dos tipos de especificaciones de constructores visuales.

3.2.1. Especificación de un Constructor Visual Simple.

La especificación de un Constructor Visual Simple X en términos de **Gsig** se logrará efectuando los siguientes pasos:

1. Abstrayendo su información Física como:

   - Una coordenada de referencia del ícono del constructor, expresado en términos de una pareja ordenada $(x - y)$.
   - El camino de la gráfica del ícono del constructor en el medio de almacenamiento. A este camino se seguirá denominando $Path(X)$.
   - La etiqueta dada por el usuario al ícono en el programa. A esta etiqueta se llamará $Name(X)$.

2. Abstrayendo su información Lógica o, lo que es lo mismo, la especificación de la expansión del constructor que se está especificando. En adelante, para referirnos a la expansión del ícono de un constructor GraPiCO etiquetado con X, emplearemos la notación $Ex(X)$. De igual forma, se utilizará $\mathcal{E}[\![X]\!]$ como la función semántica que entrega la especificación textual del constructor con etiqueta X ó la especificación del constructor mismo.

3. Y organizando la información obtenida en 1. y 2. entre un par de símbolos de delimitación $(Sd_1 Sd_2)$[7] como se muestra en la ecuación 2:

$$\mathcal{E}[\![X]\!] = S^{8}\ Sd_1\ \overbrace{(x-y)\ Path(X)}^{Parte\ Física}\ \sim\ \overbrace{Name(X)}^{Etiqueta}\ :\ \overbrace{\mathcal{E}[\![Ex(X)]\!]}^{Parte\ Lógica}\ Sd_2 \tag{2}$$

3.2.2. Especificación de un Constructor Visual Compuesto.

La especificación de un Constructor Visual Compuesto $X_1, \cdots, X_n$ dentro de un sintagma, se efectúa al listar las diferentes especificaciones de cada constructor, separados por un símbolo de sincronización $\widehat{Sc_i}$ identificando la relación i entre un par de constructores; lo anterior, encerrado entre un par de símbolos de delimitación Sd_1 y Sd_2, como se presenta en la ecuación 3:

$$\mathcal{E}[\![X_1, \cdots, X_n]\!] = Sd_1\ \mathcal{E}[\![X_1]\!]\ \widehat{Sc_1} \cdots \widehat{Sc_{n-1}}\ \mathcal{E}[\![X_n]\!]\ Sd_2 \tag{3}$$

[7] Ejemplos de Sd_1 y Sd_2 son { }, ( ), [ ], $\langle\ \rangle$.

[8] En ocasiones se requiere el empleo de un símbolo de sincronización S ó símbolos de delimitación para establecer condiciones en los constructores e identificarlos de manera única.

3.3. Definición del proceso de construcción de las Gsig a partir de la especificación de los constructores visuales

Luego de la definición del proceso de especificación de los constructores visuales, se muestra la caracterización de las **Gsig** mediante la definición 2.

Definición 2 *Una* **Gsig** *es una gramática independiente de contexto, donde las producciones pueden ser de tres formas:*

1. $X \rightarrow \mathcal{E}[\![Y]\!]$
2. $X \rightarrow \mathcal{E}[\![Ex(Y)]\!]$
3. $X \rightarrow (\ Coordenadas\ )\ Imagen$

1. *Generación de una especificación.*
2. *Generación de una Parte Lógica.*
3. *Generación de una Parte Física.*

Al emplear la denifinición 2 se construye la producción más general para un constructor visual, presentada en la ecuación 4.

$$X \rightarrow Y \ \sim \ Etiqueta \ : \ Z \tag{4}$$

De la ecuación 4, se observa que, en efecto, el símbolo $\rightarrow$, contiene dos connotaciones en su función semántica:

1. El constructor visual X se compone de: Y, no_terminal representado la Parte Física, el símbolo de sincronización '$\sim$', un símbolo de identificación denominado $Etiqueta$, el símbolo de sincronización '**:**' y finalmente, por Z, no_terminal que corresonde a su Parte Lógica.
2. La $Expansión$ del ícono representado por X tiene como resultado lo que produce Z.

De otro lado, gracias a la estructura de la producción mostrada en la ecuación 4 se modelan diversos casos de constructores visuales en donde alguno de sus elementos es nulo, estos casos se presentan en la tabla 1.

3.4. Ejemplo de utilización de las Gsig

En esta sección, se mostrará un ejemplo de especificación con las **Gsig**, retomando los ejemplos de los constructores visuales de la figuras 1 y 3.

3.4.1. Gramática de un Constructor Visual Compuesto.

- Al considerar el caracter '**,**' como el símbolo de sincronización separador de los íconos que comparten una ventana.
- Empleando la ecuación 3, la definición 2 y la ecuación 4.

  Se obtiene la gramática para el Constructor Visual Compuesto expuesta en la figura 1, presentada en la ecuación 5.

$$Ic' \rightarrow \sim Etiqueta\_De\_Ic' \ : \ \mathcal{E}[\![Ic_1]\!] \ , \ \cdots \ , \ \mathcal{E}[\![Ic_n]\!] \tag{5}$$

1. Constructor sin Parte Física, $Y \rightarrow \epsilon$. $X \rightarrow \sim\ Etiqueta\ :\ Z$	1. Elemento gráfico del que se conoce su ubicación por referencia a otro elemento o porque siempre está en el mismo lugar.
2. Constructor sin etiqueta. $X \rightarrow Y\ \sim : Z$	2. Objeto visual que no tiene asociada una etiqueta, ya sea porque el lenguaje le asigna una o porque no la requiere.
3. Constructor sin Parte Lógica, $Z \rightarrow \epsilon$. $X \rightarrow Y\ \sim\ Etiqueta\ :$	3. Ícono que internamente no está compuesto de otros elementos visuales; en otras palabras, la función *Expansión* no retorna elemento alguno. Este caso se emplea para modelar los literales visuales.

Tabla 1: Casos de constructores visuales modelados por las **Gsig**.

- Finalmente, al resolver las especificaciones de los íconos en la ecuación 5 y empleando los símbolos de delimitación '**(**' y '**)**' se tendría la gramática de la ecuación 6.

$$Ic' \rightarrow \sim Etiqueta\_De\_Ic'\ : \\ (\ ParteFísica\_De\_Ic_1\ \sim\ Etiqueta\_De\_Ic_1\ :\ ParteLógica\_De\_Ic_1\ ,\ \cdots\ ) \tag{6}$$

3.4.2. Gramática de un Constructor Visual Simple.

- A través de la ecuación 2, la definición 2 y la ecuación 4,

  Se consigue la gramática para el constructor visual mostrado en la figura 3, presentada en la ecuación 7.

$$Ic \rightarrow\ ParteFísica\_De\_Ic\ \sim Etiqueta\_De\_Ic\ :\ \mathcal{E}[\![\ Ex(Ic)\ ]\!] \tag{7}$$

- Y al emplear la producción que se indica en la ecuación 5,

  Se llegaría a la gramática de la ecuación 8.

$$Ic \rightarrow\ ParteFísica\_De\_Ic\ \sim Etiqueta\_De\_Ic\ :\ Ic' \tag{8}$$

3.4.3. Gramática de la composición de un Constructor Visual Simple y un Constructor Visual Compuesto.

Luego de la construcción de la gramática en la sección 3.4.1 del Constructor Visual Compuesto presentado en la figura 1; y la gramática en la sección 3.4.2, del Constructor Visual Simple que se

Programa Principal:	P	$\rightarrow$	Q	Programa P compuesto por el Proceso Q.
Proceso:	Q	$\rightarrow$	$Q \mid Q$	Proceso Q compuesto por procesos concurrentes.
	Q	$\rightarrow$	**clone** Q	Proceso con condición de replicación.
	Q	$\rightarrow$	Q	Proceso Q.

Fig. 5: Segmento de gramática del cálculo PiCO presentada en [1].

enmarca en la figura 3, se muestra en la ecuación 9 la gramática del par de constructores visuales, a partir de las gramáticas planteadas en las ecuaciones 6 y 8.

$$\begin{aligned} &Ic \rightarrow ParteFísica\_De\_Ic \sim Etiqueta\_De\_Ic : Ic' \\ &Ic' \rightarrow \sim Etiqueta\_De\_Ic' : \\ &\quad (\, ParteFísica\_De\_Ic_1 \sim Etiqueta\_De\_Ic_1 : ParteLógica\_De\_Ic_1 , \cdots) \end{aligned} \tag{9}$$

Posteriormente, una transformación que favorece la comprensión de la gramática presentada en la ecuación 9, es la de la inclusión de un no_terminal por cada ícono del Constructor Visual Compuesto $\{Ic_1, \cdots, Ic_n\}$, además de la adición de un identificador (que correspondería al símbolo de sincronización S mencionado en la ecuación 2) para cada categoría de íconos, así: 1 para Ic_1, 2 para Ic_2, ... y n para Ic_n.

La gramática final es expuesta en la ecuación 10.

$$\begin{aligned} &Ic \rightarrow ParteFísica\_De\_Ic \sim Etiqueta\_De\_Ic : Ic' \\ &Ic' \rightarrow \sim Etiqueta\_De\_Ic' : (Ic_1 , \cdots , Ic_n) \\ &Ic_1 \rightarrow \mathbf{1}\ ParteFísica\_De\_Ic_1 \sim Etiqueta\_De\_Ic_1 : ParteLógica\_De\_Ic_1 \\ &\quad \vdots \\ &Ic_n \rightarrow \boldsymbol{n}\ ParteFísica\_De\_Ic_n \sim Etiqueta\_De\_Ic_n : ParteLógica\_De\_Ic_n \end{aligned} \tag{10}$$

4. EJEMPLO: EL LENGUAJE GRAPICO

En el grupo de investigación AVISPA, capítulo Universidad del Valle, actualmente se implementa un nuevo cálculo computacional visual llamado GraPiCO, cálculo computacional de objetos concurrentes con restricciones basado en el cálculo PiCO, propuesto por el grupo AVISPA en [1]. En este desarrollo se utilizan las gramáticas **Gsig** para especificar un programa visual GraPiCO y despues, efectuar el proceso de compilación. Un segmento se presenta en la sección 4.1.

4.1. Programa Principal y procesos concurrentes en PiCO

En la figura 5 hay un fragmento de la gramática del cálculo PiCO introducido por el grupo AVISPA en [1].

Dado que el segmento de gramática de la figura 5 es una representación para su divulgación, se requiere el diseño de otra gramática en un formato que facilite emplear el cálculo PiCO como modelo para la especificación textual de GraPiCO.

En la figura 6 se encuentra una gramática equivalente al segmento del cálculo PiCO presentado en la figura 5 que cumple con las condiciones para LL(1), organizada de manera que cada una de las partes

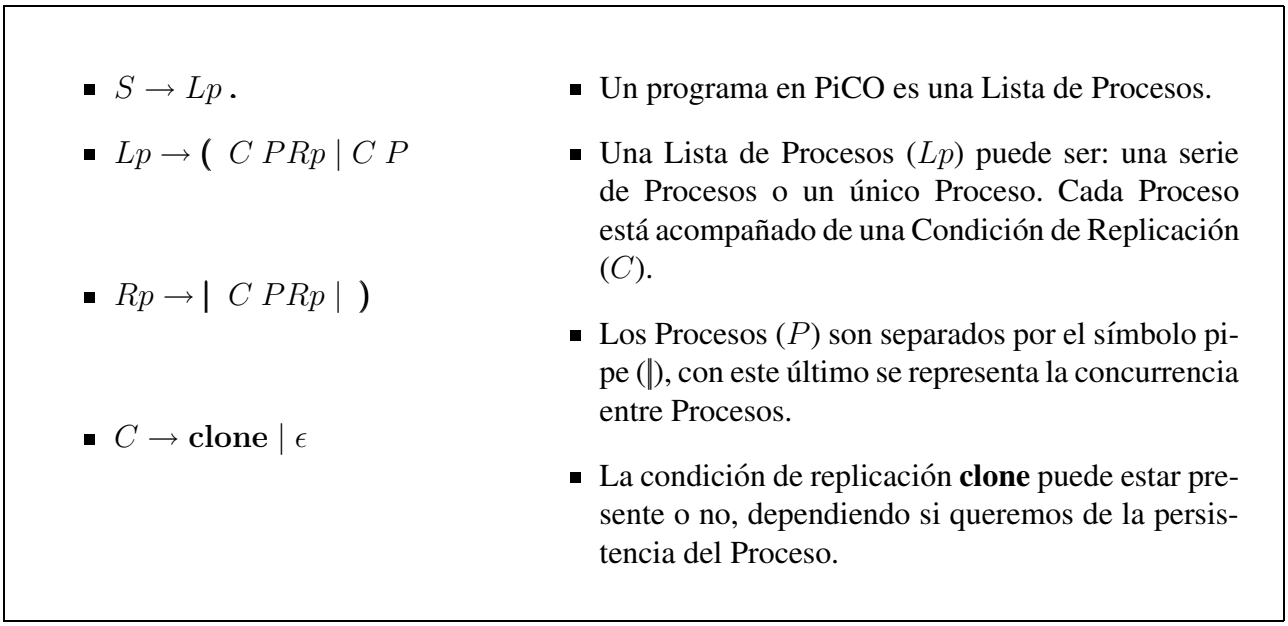

Fig. 6: Gramática LL(1) del cálculo PiCO.

izquierdas de las producciones configure una etiqueta de un sintagma del lenguaje; la organización también se efectuó con el fin de que cada una de las etiquetas que sean empleadas como parte izquierda de varias producciones conformen un paradigma[9] del lenguaje siguiendo la definición de los dos ejes de Saussure expuestos en [5].

4.2. Gramática Gsig de un segmento de GraPiCO

Finalmente, se encontrará la gramática **Gsig** en forma LL(1) del segmento de gramática del cálculo PiCO de la figura 6. Como se aprecia visualmente en la figura 7 y formalmente en las producciones de la figura 8 un programa está compuesto de una Parte Física, una imagen (referenciando el programa) con su respectiva localización, una etiqueta (identificación que el usuario le impone al ícono del programa) y una Parte Lógica, conformada por el cuerpo del programa que es, a su vez, un conjunto de procesos concurrentes. De esta forma, para representar gramaticalmente lo anterior, dado que la expansión del no_terminal $Programa$ debería mostrar su Parte Lógica, el símbolo $\rightarrow$ tiene dos significados: el usual en gramáticas BNF y el de la $Expansión$. De manera similar, para modelar la concurrencia (visualmente, procesos en una misma ventana) se emplea el operador |. La utilización del operador | como símbolo de sincronización obedece al uso que se hace de este caracter en la gramática original de PiCO y el deseo de continuar con la notación. No existe ambigüedad alguna entre este símbolo ni el usual en las gramáticas BNFE (para agrupación de varias partes derechas en una parte izquierda), porque al ser la gramática de GraPiCO LL(1) cuando se emplea | como símbolo de sincronización de especificaciones, siempre aparecerá como parte de PRIMERO[10] de una producción; mientras que, cuando | se emplea como separador de lados derechos, nunca se encontrará al inicio de una parde derecha de alguna producción. Y así, se ha obtenido una **Gsig** (presentada en la figura 8) para el segmento de Cálculo PiCO.

[9] Conjunto virtual de elementos de una misma clase gramatical que pueden aparecer en un mismo contexto.

[10] Conjunto de componentes léxicos que operan como los primeros símbolos de una o más cadenas generadas a partir de la producción.

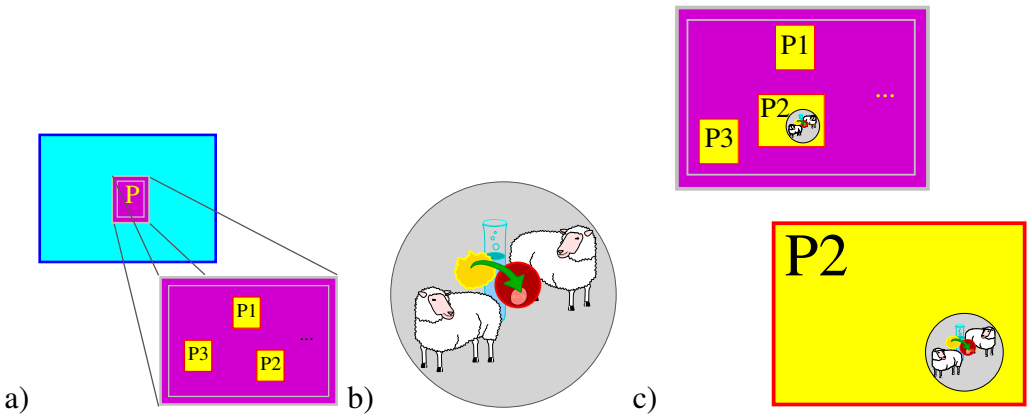

Fig. 7: Programa principal y Replicación de procesos: a)Programa GRAPICO, compuesto por los procesos concurrentes: P_1, P_2, P_3, ... b) Operador icónico para la Replicación, b) Proceso P_2 con replicación.

$Programa \rightarrow \{\ ParteFísica\ \sim\ Etiqueta\ :\ ParteLógicaPrograma\ \}$

- $ParteFísica \rightarrow (\ Coordenadas)\ Imagen$
- $ParteLógicaPrograma \rightarrow (\ Clonación\ Proceso\ ListaProcesos \mid Clonación\ Proceso$
  - $ListaProcesos \rightarrow \mid\ Clonación\ Proceso\ ListaProcesos\ \mid\ )$
  - $Proceso \rightarrow ParteFísica\ \sim\ Etiqueta\ :\ ParteLógicaProceso$
  - $Clonación \rightarrow *^a \mid \epsilon$

[a] Símbolo de sincronización y literal para identificar la clonación de procesos (equivalente a **clone** en PiCO).

Fig. 8: Sección de gramática para Programa.

5. CONCLUSIONES

- La forma de representación de las GPE requiere la inclusión de una gran cantidad de información operativa como la contenida en Δ y Γ, incluso en los programas más simples. Por esto, si se desea efectuar demostraciones matemáticas de propiedades lingüísticas, se requiere la modificación del código de almacenamiento a otro adecuado para el tratamiento formal. Todas **Gsig** tienen forma LL(1), por lo tanto, la especificación de un constructor visual posibilita una forma de análisis de lenguajes mediante conocidas técnicas para códigos textuales. De igual manera, su estructura permite efectuar una traducción dirigida por la sintaxis y los tratamientos matemáticos requeridos para demostrar características formales del lenguaje.

- El almacenamiento de código en las GPE, es un tema que no se tiene en la cuenta. Todavía no existe una relación simétrica de traducción entre el Programa Visual y su representación gramatical; pues se presenta una forma de plasmar el primero en el segundo, pero no es directa la operación inversa. Para las **Gsig** el código generado es una representación textual apta pa-

ra el almacenamiento, porque su estructura ayuda en la recuperación de los dibujos desde su especificación.

- En las GPE la etapa de Parsing requiere la implementación de un programa nada sencillo y las etapas posteriores, como el análisis semántico y la traducción de código, requieren soluciones que aún no se han descubierto. La creación de la especificación gramatical con **Gsig** puede ser vista como un procedimiento bastante mecánico si se cuenta con la gramática del cálculo computacional del Lenguaje de Programación Visual en diseño.

- Como las GPE emplean una notación LR(k) mientras que las **Gsig** una LL(1), las GPE especifican un superconjunto de los lenguajes de las **Gsig**, pero debido al nivel de generalidad de las GPE, en ocasiones, como el caso de GraPiCO; el esfuerzo que acarrea el empleo de una GPE, no es acorde a su utilidad.

REFERENCIAS

[1] G. Álvarez, J. F. Díaz, L. Quesada, F. Valencia, G. Tamura, C. Rueda, y G. Assayag. PiCO: A Calculus of Concurrent Constraint Objects for Musical Applications . *Workshop on Constraints Techniques for Artistic Applications, ECAI98*, 1998.

[2] G. Costagliola. Extended Positional Grammars. *Proceedings of the IEEE International Symposium on Visual Languages*, 103–110, 2000.

[3] G. Costagliola, V. Deufemia, F. Ferrucci, y C. Gravino, On the pLR Parsability of Visual Languages. *2001 IEEE Symposia on Human-Centric Computing*, Stresa, Italy, September 2001. Pages 48 to 49.

[4] Shi-Kuo Chang. *Principles of Visual Programming Systems*. Prentice-Hall, 1990.

[5] C. Bally y A. Séchehaye. Cours de linguistique générale. 1916.

[6] M. Erwig. Abstract Syntax and Semantics of Visual Languages. *Journal of Visual Languages and Computing*. Vol 9, No. 5, 461-483, 1998.

[7] M. Erwig. Visual Type Inference, Martin Erwig. *Journal of Visual Languages and Computing*, Vol. 17, No. 2, 161-186, 2006.

[8] M. Nadin. Visual Programming Languages - Efficiency of the Visual driving Technology. *Technical report in PROJECT 2 - EFFICIENCY OF THE VISUAL*, University of Texas at Dallas. http://www.utdallas.edu/∼pds017300/nadin/VPL.pdf (2004).

[9] M. Burnett, J. Atwood, R. Djang, H. Gottfried, J. Reichwein, S. Yang, Forms/3: A First-Order Visual Language to Explore the Boundaries of the Spreadsheet Paradigm. *Journal of Functional Programming*, March, 2001. Pages 155 to 206.