LUPA: Lenguaje Unificado de Procesos Administrativos

Emely Arráiz **Pedro R. Borges** **Roger V. Soler** **Cristina Zoltan**

Universidad Simón Bolívar
Dpto. de Computación y Tecnología
de la Información
Apartado Postal 1089
Caracas 1080, Venezuela
{arraiz,borges,soler,zoltan}@ldc.usb.ve

Abstract

The Unified Language for Administrative Processes (LUPA, in spanish) is a language oriented towards programming and communicating the various tasks associated to the precise treatment of business transactions. The syntactic objects of the language are regular expressions built from a small set of operators, which allow the denotation of the process associated to each transaction to be expressed in a simple and modular way. The semantic specification is given as interpreted Petri nets, and the consistency of the processes is shown.

Keywords: Formal methods, Regular expresions, Petri nets

Resumen

El Lenguaje Unificado de Procesos Administrativos (LUPA) es un lenguaje orientado a la programación y comunicación de las distintas tareas asociadas al tratamiento preciso de transacciones en la empresa. Los objetos sintácticos del lenguaje son expresiones regulares construidas a partir de un pequeño conjunto de operadores que permiten en forma sencilla denotar el proceso asociado a una transacción, haciendo uso de la modularidad (subexpresiones). La especificación semántica es dada a través de redes de Petri interpretadas, mostrando la consistencia de los procesos denotados.

Palabras claves: Métodos formales, Expresiones regulares, Redes de Petri

0 Introducción

La noción de transacción en una empresa es el intercambio de bienes o servicios que integran el "negocio" de esa empresa. El proceso asociado a cada transacción estará formado, por una parte, por el conjunto de tareas que se efectuan en la empresa con el objeto que dicha transacción sea satisfecha, esta idea de satisfacción viene dada por el hecho que la transacción finalice y se le haya dado complimiento a los eventos que genera esa transacción. Por otra parte, ese conjunto de tareas que conforman el proceso asociado a la transacción para darle cumplimiento tiene una estructura "natural" de red (grafo) que está dada por la "programación", es decir, el orden en que deben efectuarse dichas tareas dada su prelación. De esta forma, un proceso será las tareas y su programación. El lenguaje LUPA permite denotar estas tareas y su programación a través de expresiones regulares, es decir, la semántica de estas expresiones serán redes (redes de Petri interpretadas [2]) que modelan la transacción. Un aspecto importante es tener la noción del estado del proceso, es decir, la(s) tarea(s) que se efectúa(n) en un momento dado dentro del proceso. Una forma de presentar esta noción es a través de la marcación en las redes de Petri [6].
En la sección 1 se dará el uso de redes de Petri en la especificación de procesos. Luego, se presentarán las expresiones (la sintaxis) que caracteriza la programación de tareas en un proceso (sección 2). Finalmente se establece la semántica de dichas expresiones a través de las Redes de Petri Interpretadas (sección 3) y las propiedades de los procesos así definidos en la sección 4.
Para concluir esta introducción recordaremos que la descripción y flujo de las tareas (programación) de los procesos forma parte de lo que se conoce como sistemas y procedimientos, también se conoce al proceso como flujo de trabajo (ing.: workflow), en particular, cuando todas las tareas son automatizables.

1 Redes de Petri y procesos

En esta sección se dará una introducción a las redes de Petri clásicas a modo de una indicación de terminología y notación. Además se dará el uso de dichas redes en la descripción de procesos (sección 1.2).

1.1 Redes de Petri

Las redes de Petri son grafos bipartitos orientados con dos tipos de nodos llamados lugares, representados por círculos, y transiciones, representados por rectangulos. Los arcos conectan nodos de tipo diferente.

Definición 1.1 (Red de Petri) *Una red de Petri es una tripleta* (L, T, F)*:*

- L *es un conjunto finito de lugares*
- T *es un conjunto finito de transiciones* $(L \cap T = \emptyset)$
- $F \subseteq (L \times T) \cup (T \times L)$ *es un conjunto de arcos dirigidos (relación de flujo)*

Un lugar p es un *lugar de entrada* de una transición t si y solo si existe un arco dirigido del lugar p a la transición t. Un lugar p es un *lugar de salida* de una transición t si y solo si existe un arco dirigido de la transición t al lugar p. Usaremos $\bullet\, t$ para denotar el conjunto de lugares de entrada a la transición t. Usaremos la notación $\bullet$ antes o después de una transición o lugar para indicar los arcos que salen o entran a la misma, es decir:

$$\begin{array}{ll} \bullet\,\mathrm{t} = & \{p \mid p \in L \wedge < p, t > \in F\} \\ \mathrm{t}\bullet = & \{p \mid p \in L \wedge < t, p > \in F\} \\ \bullet\,\mathrm{p} = & \{t \mid t \in T \wedge < t, p > \in F\} \\ \mathrm{p}\bullet = & \{t \mid t \in T \wedge < p, t > \in F\} \end{array}$$

Una red de Petri posee un marcación en los lugares, y según sea este marcación se clasifican en tres tipos de redes [4]. El primer tipo posee marcas (ing.:tokens) no estructuradas en los lugares. En los lugares se tiene una interpretación booleana (presencia o ausencia de marcas). El segundo tipo es capaz de reconocer la cantidad de marcas no estructuradas existentes en un lugar, lo que equivale a que el marcación de una red es una función de los lugares a los naturales: $\mu \in L \rightarrow \mathcal{N}$. El tercer tipo posee marcas estructuradas y cada lugar puede albergar un multiconjunto de ellas.
Las transiciones son los elementos activos de la red, y la red cambia de marcación en función de las transiciones que se disparan. Una transición t se *habilita* si y solo si todos los elementos de $\bullet\, t$ tienen al menos una marca; abusando del lenguaje generalizaremos diciendo $\mu(\bullet\, t) > 0$. Una transición *habilitada* puede *dispararse*; si la transición t se dispara entonces t *consume* una marca de cada lugar de entrada p de t (i.e: $\bullet$t) y *produce* una marca en cada lugar de salida de t (i.e: t$\bullet$).

Se define una relación entre marcaciones, en base a que la marcación de una red puede transformarse en otra debido a transiciones que se disparan simultaneamente $\mu_1 \xrightarrow{\mathcal{T}} \mu_2$, donde la marcación μ_1 se transforma en la marcación μ_2 debido a que las transiciones del conjunto $\mathcal{T}$ ($\mathcal{T} \subseteq T$), se dispararon. Esta definición difiere de la dada en [3], ya que extendemos la relación entre marcaciones para un conjunto de transiciones que se disparan. Denotaremos por $\mu_1 \xrightarrow{*} \mu_2$ a la clausura transitiva de la relación.

Definición 1.2 (Estado alcanzable) *Usaremos (PN, μ_0) para denotar la red de Petri PN con una marcación inicial μ_0. Una marcación μ_a es un* estado alcanzable *de (PN, μ_0) si y solo si $\mu_0 \xrightarrow{*} \mu_a$.*

1.2 Uso de redes de Petri en la definición de procesos

Las redes de Petri han sido utilizadas por diversos autores —ver resumen en [7]— como representación de procesos, sin embargo los diversos modelos revisados no incluyen la totalidad de la información asociada al proceso en la misma red. Usaremos el ejemplo del proceso de reclamación usado en [3], que utiliza redes de tipo 3, para ver que hay elementos que no son explicitados en las WF-net, como son los relojes y los elementos de ingreso al proceso distintos del lugar de entrada.

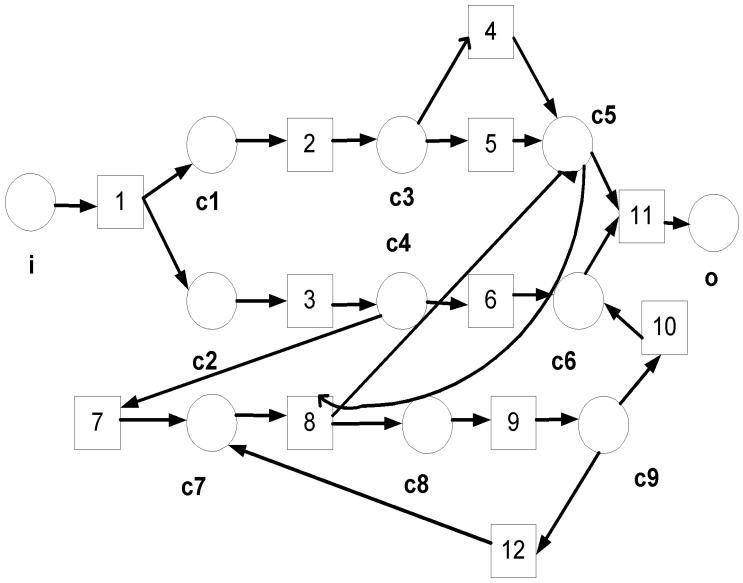

Figura 1: Una red para el proceso de reclamaciones

En la figura 1 se han numerado las tareas (transiciones) de la forma siguiente: (1) Registro del reclamo, (2) Envio de cuestionario, (3) Evaluación del reclamo, (4) Vencimiento del tiempo (time-out), (5) Procesamiento del cuestionario, (6) No procesar, (7) Se requiere procesamiento, (8) Procesamiento del reclamo, (9) Verificación del procesamiento, (10) Conformación del procesamiento, (11) Archivar, (12) No conformación del procesamiento.
La red de la figura 1 corresponde a la red de procesamiento de una reclamación, que incluye como tarea el envío de un formulario y la espera por la recepción del formulario llenado como se ve en la transición 5. La transición 4 posee un reloj, y el texto descriptivo del ejemplo se menciona que la espera por la recepción y procesamiento del cuestionario está limitada a dos semanas. Asimismo el disparo de la actividad 5 (procesamiento) depende de la llegada del exterior del proceso del cuestionario que fue enviado en la transición 2. El ejemplo muestra un proceso iterativo en la transición 9 donde se está permitiendo una cantidad no acotada de verificaciones del procesamiento hasta lograr conformarlo. La condición c9 es una condición de OR direccionando el flujo hacia la transición 12 o la transición 10 que son respectivamente las tareas de no conformación del procesamiento y la conformación para dar lugar al archivado de los documentos en la transición 11. Lo mismo sucede con la condición c4. El flujo en los dos sentidos entre el lugar c5 y la transición 8 conecta dos subprocesos internamente en el proceso de reclamación no quedando claro si esto es consecuencia de una optimización de la red o un diseño original.
A fin de sistematizar el sentido de los predicados asociados por [3], nosotros usaremos las redes de Petri interpretadas haciendo una modificación que completa el modelo de redes en función de establecer un lenguaje de comunicación que permiten definir la acción de una transición mediante un operador asociado a ella y

completando la semántica de la descripción de un proceso. Además se evitará el uso de flujos del interior de un subproceso al interior de otro.

2 Modelado de transacciones

Usando diversos tipos de redes, la literatura refleja gran actividad en la expresión de procesos mediante redes de Petri. Sin embargo las redes generadas requieren pasar por un proceso de verificación (manual o automático) de las bondades de la red obtenida.
Nuestra propuesta, a traés de LUPA, es la definición de un lenguaje suficientemente expresivo que permite modelar los procesos administrativos, los cuales se pueden traducir en Redes de Petri cuya estructura posee las propiedades deseables de construcción. En la siguiente sección se presentarán las expresiones de procesos (tareas), las cuales serán construidas inductivamente a través de una base de operadores (constructores) que permitirán dar una "sintaxis" a la programación en los procesos.

2.1 Expresiones de Procesos (EP)

Las expresiones de procesos (EP) tendrán la sintaxis siguiente: Sean $t_i \in T$ (conjunto finito de tareas) y $P \in EP$, entonces:

$$
\begin{array}{lll}
\mathrm{P} \rightarrow & t_i & \text{(1. tarea simple)} \\
& \mid P_1 \circ P_2 & \text{(2. secuenciación)} \\
& \mid P_1 \wedge P_2 & \text{(3. conjunción)} \\
& \mid P_1 \ \textcircled{\vee} P_2 & \text{(4. disyunción)} \\
& \mid P_1 \vee P_2 & \text{(5. anulación)} \\
& \mid P^{+} & \text{(6. iteración)}
\end{array}
$$

Entre paréntesis se ha colocado la noción que denota la expresión de proceso correspondiente a cada producción, así se tendrá:

1. el proceso $P = t_i$ denota la tarea simple (atómica) t_i, este proceso P finaliza cuando se ejecutó la tarea t_i;

2. el proceso $P = P_1 \circ P_2$ denota que primero se realiza el proceso P_1, cuando P_1 finaliza se realiza el proceso P_2, el proceso P finaliza cuando finaliza P_2;

3. el proceso $P = P_1 \wedge P_2$ denota que los procesos P_1 y P_2, se realizan simultaneamente, cuando ambos procesos han finalizado también finaliza P;

4. el proceso $P = P_1 \ \textcircled{\vee} P_2$ denota bajo una condición que se realizará solo uno de los procesos y cuando el que se realiza haya finalizado también finaliza P;

5. el proceso $P = P_1 \vee P_2$ denota que ambos proceso se realizan simultaneamente, cuando uno de ellos finaliza, también finaliza P, es decir el otro proceso se anula;

6. el proceso $P = P^{+}$ denota que el proceso P se realiza repetidamente en forma secuencial como en 2 hasta que se de una condición de finalización.

Con estas ideas se puede expresar la programación presentada en la red de la figura 1 de la manera siguiente:

$$((1 \circ ((2 \circ (4 \vee 5)) \wedge 3)) \circ (6 \textcircled{\vee} (7 \circ ((8 \circ 9)^{+} \circ 10)))) \circ 11 \tag{1}$$

El uso del operador de anulación ($\vee$) no está limitado a acotar tiempo, como la tarea 4 (fig 1). En ciertos procesos conviene disparar actividades en paralelo, sabiendo que alguna de ellas no van a ser concluídas. Tomando por ejemplo un proceso de cobro de dinero, las gestiones de cobranza constituyen un proceso, mientras que la preparación de la cobranza judicial requiere una variedad de actividades. En un instante del proceso puede dispararse la actividad de cobranza extra-judicial y la preparación del proceso de cobranza judicial. La cobranza extra-judicial poseerá un plazo para completarse. Si la cobranza extra-judicial prospera, debe darse por terminado el proceso de cobro de dinero, independientemente del avance que se haya logrado en la preparación de la cobranza judicial.

2.2 Axiomatización de las expresiones

En forma intuitiva podemos establecer para las expresiones de procesos la axiomatización siguiente:

$P \circ (Q \circ R)$	=	$(P \circ Q) \circ R$	$\circ$ asociativa
$P \wedge (Q \wedge R)$	=	$(P \wedge Q) \wedge R$	$\wedge$ asociativa
$P \vee Q \vee R$	=	$P \vee (Q \vee R)$	$\vee$ asociativa a la derecha
$P \Vee Q \Vee R$	=	$P \Vee (Q \Vee R)$	$\Vee$ asociativa a la derecha
$P \wedge Q$	=	$Q \wedge P$	$\wedge$ conmutativa
$P \vee Q$	=	$Q \vee P$	$\vee$ conmutativa
$P \Vee Q$	=	$Q \Vee P$	$\Vee$ conmutativa
$P \circ (Q \Vee R)$	=	$(P \circ Q) \Vee (P \circ R)$	distributiva izq $\circ$ y $\Vee$
$(P \Vee Q) \circ R$	=	$(P \circ R) \Vee (Q \circ R)$	distributiva der $\circ$ y $\Vee$

Por ejemplo, la expresión 1 que corresponde a la figura 1 será equivalente a:

$$(1 \circ (3 \wedge (2 \circ (5 \vee 4)))) \circ ((((7 \circ (8 \circ 9)^{+}) \circ 10) \circ 11) \Vee (6 \circ 11)) \tag{2}$$

La idea que la operación $\circ$ no distribuya sobre $\wedge$ y $\vee$ es que los procesos involucrados son "simultaneos" y no pueden modificar la misma "información". Con el operador $\Vee$ se ha elegido asociatividad a la derecha, haciendo asociación con la selección de casos en los lenguajes de programación. Con el operador $\vee$ se ha elegido asociatividad a la derecha, dandole preferencia al proceso más a la izquierda. La razón de que estos operadores no sean asociativos quedará aclarado en la sección 3.2, cuando se establezca la semántica

3 Semántica de las expresiones para procesos

En esta sección se presentará formalmente la semántica de los operadores a través de las redes de Petri interpretadas.

3.1 Redes de Petri interpretadas

A diferencia del enfoque [3] usaremos las redes de Petri interpretadas según [2], haciendo simplificaciones a la definición dada en los estandares, para adaptarla a las necesidades de las redes de flujo que usan parcialmente la potencialidad de las Redes de Petri interpretadas. Las redes que usamos caen dentro de la categoría 3 de la clasificación de redes de Petri según [4].
En el enfoque [3] asocia a cada lugar un predicado, utilizado como pre/post condiciones de las tareas que le permite modelar las redes del tipo AND/OR split y los AND/OR joint propuestas en [1]

Definición 3.1 (Redes de Petri interpretadas) *Las redes de Petri interpretadas están constituídas por una red de Petri clásica <L,T,F> y un ambiente (D, OP, PR) y dos funciones φ y ψ que sirven de enlace entre la parte de comandos (la red) y el ambiente, donde se registran las operaciones. Es decir ((L,T,F),((D,OP,PR)),φ,ψ)*

- *D es el conjunto de estados del ambiente*
- *$OP = \{op_1, op_2, \cdots, op_s\}$ es un conjunto de operadores*

$$op_i : D \rightarrow D$$

- *$PR = \{pr_1, pr_2, \cdots, pr_s\}$ es un conjunto de predicados sobre D*

$$pr_i : D \rightarrow \{cierto, falso\}$$

- *$\varphi : L \rightarrow OP$ asocia a cada lugar de la red un operador*
- *$\psi : T \rightarrow PR \times OP$ asocia a cada transición un par <predicado,operador> que corresponde a la tarea a realizar.*

El funcionamiento de estas redes de Petri interpretadas requiere, en función de los predicados asociados a las transiciones que el modo de atravesar las transiciones se modifique respecto a las redes de Petri clásicas. Las marcas de la marcación de la red poseerán dos estados: *disponible* y *no-disponible*. Las marcas pasarán al estado disponible, si se ha completado la operación asociada al lugar donde estas se encuentran. De igual manera una transición t para quedar habilitada debe tener $\mid \mu(\bullet t) \mid > 0$ y al menos una marca disponible

en cada uno de los elementos de $\mu(\bullet t)$. Una transición t se disparará si está habilitada y se verifica el predicado asociado a ella. Al dispararse una transición se ejecuta la operación asociada por ψ, el disparo de una transición produce marcas no disponibles en $t\bullet$.
De esta forma se define:

$$\mu\text{: L} \quad \rightarrow \quad \{\text{s,n}\}^* \quad \text{(i.e secuencias finitas de s -disponible- y n -no disponible-, utilizando secuencias finitas por multiconjuntos)}$$

$$\begin{array}{rrcl} & \mid \mu(P) \mid & = & \text{longitud de la secuencia} \\ \text{además,} & \mid \mu(P) \mid & = & \mid \mu(P) \mid_s + \mid \mu(P) \mid_n \\ \text{donde} & \mid \mu(P) \mid_s & = & \text{cantidad de s en la secuencia } \mu(P) \\ \text{y} & \mid \mu(P) \mid_n & = & \text{cantidad de n en la secuencia } \mu(P) \end{array}$$

también se define:

$$\mid \mu(\bullet t) \mid_s > 0 \quad \Leftrightarrow \quad (\forall p \in \bullet t)(\mid \mu(p) \mid_s > 0)$$

Ahora bien, a través de una transición t ∈ T, se pasa de un par <estado, marcación> a otro par <estado, marcación> de la forma siguiente:
Sea d el estado "actual" del ambiente y μ la marcación "actual" , entonces:

$$< d, \mu > \stackrel{t}{\rightarrow} < d^{'}, \mu^{'} >$$

$$\begin{array}{ll} \Leftrightarrow & \mid \mu(\bullet t) \mid_s > 0 \wedge (\psi(t) \downarrow 1)(d) \\ & \Rightarrow (\forall p \in \bullet t)(\mid \mu^{'}(p) \mid_s = \mid \mu(p) \mid_s - 1 \wedge \mid \mu^{'}(p) \mid_n = \mid \mu(p) \mid_n) \\ & \wedge\, d^{'} = (\psi(t) \downarrow 2)(d) \\ & \wedge (\forall p \in t\bullet)(\mid \mu^{'}(p) \mid_n = \mid \mu(p) \mid_n + 1 \wedge \mid \mu^{'}(p) \mid_s = \mid \mu(p) \mid_s) \end{array}$$

además se realizará la operación de transformar las marcas a disponibles, es decir:

$$n \in \mu^{'}(t\bullet) \quad \Rightarrow \quad \varphi(t\bullet)(d^{'}) \quad \wedge n \rightarrow s$$

En esta última implicación, hemos abusado del lenguaje ya que $\varphi(t\bullet)$ corresponde a un conjunto de operadores que se "activan". Obviamente esto crea problemas en el ambiente ya que se tendrán φ en paralelo y lo mismo sucede con los ψ. En la próxima sección se verán restricciones sobre los $op \in OP$ que permitirán darle solución al problema del ambiente. La solución es que los operadores actuén sobre conjuntos disjuntos. En lo que se refiere a los operadores asociados el φ no se pondrán restricciones ya que el operador asociado al φ es la identidad, salvo para el lugar i que estará "aislado", es decir:

$$\varphi : (P - \{i\}) \quad \rightarrow \quad \text{Id}$$

Esto hace que las marcas que llegan a los lugares quedan inmediatamente disponibles.

A continuación daremos una serie de definiciones sobre las redes de Petri interpretadas que nos permitirán manejar el ambiente y la noción de ejecución en la red.

Definición 3.2 (Estructura de los estados) *El conjunto de estados D del ambiente A que representará el "llenado" del formulario de la transacción, será de la forma:*

$$D = D_1 \times D_2 \times \ldots \times D_n = \prod_{k=1}^{n} D_i$$

Definición 3.3 (Firma de los operadores) *Para los operadores*

$$op_j \in OP(j = 1, \ldots, m_1)$$

se tendrá:

$$op_j : \prod_{k \in I_{j_d}} D_k \rightarrow \prod_{k \in I_{j_c}} D_k$$

$$\begin{aligned} donde, I_{j_d} &= Indices\_d(op_j)(i.e.\ Indices\ del\ dominio\ de\ op_j) \\ y\ I_{j_c} &= Indices\_c(op_j)(i.e.\ Indices\ del\ codominio\ de\ op_j) \end{aligned}$$

Para las funciones Id (Identidad en D) y Mk (constante booleana en D_k) se tendrá los índices siguientes:

- *si $op_j = Id$ entonces $I_{j_d} = I_{j_c} = \emptyset$*
- *si $op_j = Mk = \lambda d. < d \downarrow 1, \cdots, d \downarrow k-1, cierto, d \downarrow k+1, \cdots, d \downarrow n >$ entonces $I_{jd} = \emptyset \wedge Ijc = \{k\} \wedge D_k = Bool$*
- *si $op_j = \overline{Mk} = \lambda d. < d \downarrow 1, \cdots, d \downarrow k-1, falso, d \downarrow k+1, \cdots, d \downarrow n >$ entonces $I_{jd} = \emptyset \wedge Ijc = \{k\} \wedge D_k = Bool$*

Definición 3.4 (Firma de los predicados) *Para los predicados $pr_j \in PR$ $(j = 1, \ldots, m_2)$ se tendrá:*

$$pr_j : \prod_{k \in I_{j_d}} D_k \rightarrow \{cierto, falso\}$$

donde, I_{j_d} = Indices del dominio de pr_j y
I_{j_c} = Indices del codominio del predicado $pr_j = \emptyset$
también se tendrá:
si $op_j \equiv cierto \vee op_j \equiv falso$ (predicado constante)
entonces $I_{j_d} = I_{j_c} = \emptyset$

Definición 3.5 (Independencia) *Se dirá que dos subconjuntos θ_1 y θ_2 de predicados y operadores (i.e. $\theta_1, \theta_2 \subseteq OP \cup PR$) tienen la propiedad de ser idependientes (denotado por $\theta_1 \| \theta_2$) si se tiene:*

$$\theta_1 \| \theta_2 \Leftrightarrow (\forall < f, g > \in \theta_1 \times \theta_2) \quad ((I_{g_c} \cap (I_{f_d} \cup I_{f_c})) \cup (I_{f_c} \cap (I_{g_d} \cup I_{g_c})) = \emptyset)$$

es decir, si ninguna f modifica el dominio o codominio de alguna g y viceversa.

Definición 3.6 (Extensiones naturales de φ y ψ) *Las funciones φ y ψ las extenderemos naturalmente a conjuntos:*

$$\begin{array}{lllllllll} L' & \subseteq & L & \Rightarrow & \varphi(L') & = & \{\varphi(l) \mid l \in L'\} & \subseteq & OP \\ T' & \subseteq & T & \Rightarrow & \psi(T') & = & \{\psi(t) \mid t \in T'\} & \subseteq & PR \times OP \\ & & & \wedge & \psi(T') \downarrow 1 & = & \{\psi(t) \downarrow 1 \mid t \in T'\} & \subseteq & PR \\ & & & \wedge & \psi(T') \downarrow 2 & = & \{\psi(t) \downarrow 2 \mid t \in T'\} & \subseteq & OP \end{array}$$

Definición 3.7 (Secuencia de ejecución) *Es una secuencia de transiciones que se disparan, iniciadas por los lugares que habilitan la primera transición y sucedida por los lugares de llegada de la última transición de la secuencia de transiciones.*

Definición 3.8 (Marcación inicial μ_0) *Llamaremos marcación inicial μ_0 a la marcación siguiente:*

$$\mu_0(\mathbf{i}) =< n >$$

$$\forall l \in L - \{\mathbf{i}\} \quad (\mu(l) =<>)$$

3.2 Red de Petri interpretada asociada a una expresión de proceso

En esta sección se dará la semántica de las expresiones de procesos. Esta semántica vendrá dada por la Red de Petri interpretada asociada a la expresión. La asociación se contruirá inductivamente según los casos. Denotaremos por $\Re_E$ la red de Petri interpretada asociada a una expresión de proceso E, es decir:

$$\Re_E =<< L_E, T_E, F_E >, < D, OP, PR >, \varphi_E, \psi_E >$$

donde el ambiente $< D, OP, PR >$ es general durante la construcción de la red. Para los casos 2 a 6, supondremos que las redes a componer son $\Re_P$ y $\Re_Q$, ver figura 3, asociadas a los procesos P y Q respectivamente. También veremos a P como una sola transición, lo mismo para Q. En todos los casos debe verificarse:

$$\begin{aligned} L_P \cap L_Q &= \emptyset \\ T_P \cap T_Q &= \emptyset \end{aligned}$$

Los casos son:

Caso 1 Sea $E = t$ (El proceso es una transición -tarea- simple t)(ver figura 2)

se tendrá que $\Re_E$:

$$\begin{array}{lcl} L_E & = & \{\mathbf{i}, \mathbf{o}\} \\ T_E & = & \{t\} \\ F_E & = & \{<\mathbf{i}, t>, <t, \mathbf{o}>\} \\ \varphi_E(\mathbf{i}) & = & \varphi_E(\mathbf{o}) = I_d \\ \psi_E(t) \downarrow 1 & = & cierto \\ \psi_E(t) \downarrow 2 & = & op_k \in OP \end{array}$$

En este caso se tiene la siguiente secuencia de ejecución a partir de la marcación inicial μ_0:

$$< \mathbf{i}, t, \mathbf{o} >$$

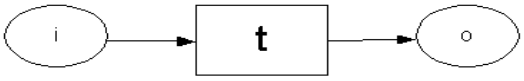

Figura 2: Tarea Simple

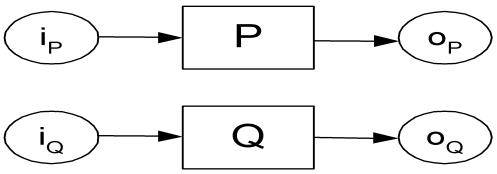

Figura 3: Redes a componer (casos 2 a 6)

Caso 2 Sea $E = P \circ Q$ (secuenciación)(ver figura 4)

Se tendrá las restricciones:

$$\begin{array}{lcl} L_P \cap L_Q & = & \emptyset \\ \{\varphi_P(\mathbf{i}_P)\} & \| & \{\varphi_Q(\mathbf{i}_Q)\} \end{array}$$

y $\Re_E$:

$$\begin{array}{lcll} L_E & = & L_P \cup L_Q & \text{donde: } \mathbf{i} = \mathbf{i}_P, \mathbf{o} = \mathbf{o}_Q, \mathbf{o}_P = \mathbf{i}_Q \\ T_E & = & T_P \cup T_Q & \\ F_E & = & F_P \cup F_Q & \\ \varphi_E(\mathbf{i}) & = & \varphi_Q(\mathbf{i}_Q) \circ \varphi_P(\mathbf{i}_P) & \\ \varphi_E(L_E - \mathbf{i}) & = & \{Id\} & \\ \psi_E(T_P) & \equiv & \psi_P(T_P) & (\text{i.e.}(\forall t \in T_P)(\psi_E(T) = \psi_P(T)) \\ \psi_E(T_Q) & \equiv & \psi_Q(T_Q) & (\text{i.e.}(\forall t \in T_Q)(\psi_E(T) = \psi_Q(T)) \end{array}$$

Es decir que se ha unificado $\mathbf{i}_P$ en $\mathbf{i}$, $\mathbf{o}_P$ e $\mathbf{i}_Q$, $\mathbf{o}_Q$ en $\mathbf{o}$. También hemos llevado las operaciones asociadas a $\mathbf{i}_P$ e $\mathbf{i}_Q$ a $\mathbf{i}$.

En este caso se tiene una secuencia de ejecución s a partir de la marcación inicial μ_0:

$$s =< \mathbf{i}, P, Q, \mathbf{o} >$$

Es decir primero se ejecuta P y luego Q.

Caso 3 Sea $E = P \wedge Q$ (conjunción)(ver figura 5)

Se tendrá las restricciones:

$$\begin{array}{rcl} (L_P \cup L_Q) \cap \{\mathbf{i}, \mathbf{o}\} & = & \emptyset \\ (T_P \cup T_Q) \cap \{Y_1, Y_2\} & = & \emptyset \\ (\varphi_P(L_P) \cup \psi_P(T_P) \downarrow 1 \cup \psi_P(T_P) \downarrow 2) & \| & (\psi_Q(T_Q) \downarrow 1 \cup \psi_Q(T_Q) \downarrow 2 \cup \varphi_Q(L_Q)) \end{array}$$

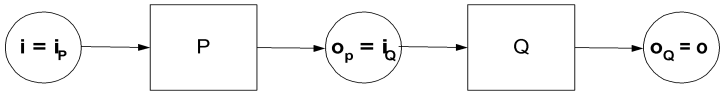

Figura 4: Secuenciación

y $\Re_E$:

$$\begin{array}{lcl}
L_E & = & \{\mathbf{i},\mathbf{o}\} \cup L_P \cup L_Q \\
T_E & = & T_P \cup T_Q \cup \{Y_1, Y_2\} \\
F_E & = & \{<\mathbf{i}, Y_1>, <Y_1, \mathbf{i}_P>, <\mathbf{o}_P, Y_2>, \\
 & & <Y_1, \mathbf{i}_Q>, <\mathbf{o}_Q, Y_2>, <Y_2, \mathbf{o}>\} \\
 & & \cup F_P \cup F_Q \\
\varphi_E(\mathbf{i}) & = & \varphi_Q(\mathbf{i}_Q) \circ \varphi_P(\mathbf{i}_P) \\
\varphi_E(L_E - \mathbf{i}) & = & \{Id\} \\
\psi_E(T_P) & \equiv & \psi_P(T_P) \text{ (i.e.}(\forall t \in T_P)(\psi_E(T) = \psi_P(T)) \\
\psi_E(T_Q) & \equiv & \psi_Q(T_Q) \text{ (i.e.}(\forall t \in T_Q)(\psi_E(T) = \psi_Q(T)) \\
\psi_E(Y_1) = \psi_E(Y_2) & = & <\text{cierto,ID}>
\end{array}$$

En este caso se tiene una secuencia de ejecución s, partir de la marcación inicial μ_0 donde P y Q se ejecutan simultaneamente:

$$s =< \mathbf{i}, Y_1, (P \| Q), Y_2, \mathbf{o} >$$

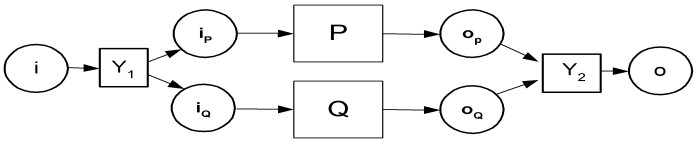

Figura 5: Conjunción

Caso 4 Sea $E = P \bigtriangledown Q$ (disyunción)(ver figura 6)

Se tendrá las restricciones:

$$\begin{array}{rcl}
(L_P \cup L_Q) \cap \{\mathbf{i},\mathbf{o},u,v\} & = & \emptyset \\
(T_P \cup T_Q) \cap \{C_1, C_2, C_3, C_4)\} & = & \emptyset \\
\varphi_P(L_P) & \| & \varphi_Q(L_Q)
\end{array}$$

y $\Re_E$:

$$\begin{array}{lcl}
L_E & = & \{\mathbf{i},\mathbf{o},u,v\} \cup L_P \cup L_Q \\
T_E & = & T_P \cup T_Q \cup \{c_1, c_2, c_3, c_4\} \\
F_E & = & \{<\mathbf{i}, C_1>, <C_1, u>, <u, C_2>, <C_2, \mathbf{i}_P>, \\
 & & <C_1, v>, <v, C_3>, <C_3, \mathbf{i}_Q>, \\
 & & <u, C_3>, <v, C_2>, <\mathbf{o}_P = \mathbf{o}_Q, C_4>, <C_4, \mathbf{o}>\} \\
 & & \cup F_P \cup F_Q \\
\varphi_E(\mathbf{i}) & = & \varphi_Q(\mathbf{i}_Q) \circ \varphi_P(\mathbf{i}_P) \\
\varphi_E(L_E - \{\mathbf{i}\}) & = & \{Id\}
\end{array}$$

$$\begin{array}{lcl}
\psi_E(T_P) & \equiv & \psi_P(T_P) \text{ (i.e.}(\forall t \in T_P)(\psi_E(T) = \psi_P(T)) \\
\psi_E(T_Q) & \equiv & \psi_Q(T_Q) \text{ (i.e.}(\forall t \in T_Q)(\psi_E(T) = \psi_Q(T)) \\
\psi_E(C_1) = \psi_E(C_4) & = & <\text{cierto,ID}> \\
\psi_E(C_2) & = & < pr_k, Id > \\
\psi_E(C_3) & = & < \neg pr_k, Id > \quad (i.e. : \psi_E(C_3) \downarrow 1 = \neg(\psi_E(C_2) \downarrow 1))
\end{array}$$

es decir, se han unificado $\mathbf{o}_p$ y $\mathbf{o}_Q$.

En este caso se tiene una secuencia s elegida del conjunto que se obtiene a partir de la marcación inicial μ_0:

$$s \in \{< \mathbf{i}, C_1, C_2, P, C_4, \mathbf{o} >, < \mathbf{i}, C_1, C_3, Q, C_4, \mathbf{o} >\}$$

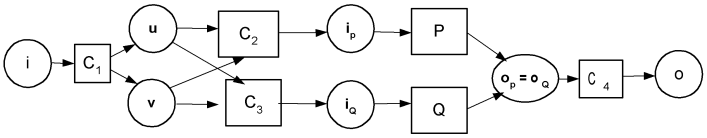

Figura 6: Disyunción

Caso 5 Sea $E = P \vee Q$ (anulación)(ver figura 7)

Se tendrá las restricciones:

$$\begin{array}{rcl}
(L_P \cup L_Q) \cap \{\mathbf{i}, \mathbf{o}, g_k, u, v, r, s\} & = & \emptyset \\
(T_P \cup T_Q) \cap \{K_1, K_2, K_3, K_4, K_5)\} & = & \emptyset \\
(\varphi_P(L_P) \cup \psi_P(T_P) \downarrow 1 \cup \psi_P(T_P) \downarrow 2) & \| & (\psi_Q(T_Q) \downarrow 1 \cup \psi_Q(T_Q) \downarrow 2 \cup \varphi_Q(L_Q)) \\
Mk_1 & \| & (\varphi_P(L_P) \cup \psi_P(T_P) \downarrow 1 \cup \psi_P(T_P) \downarrow 2) \\
Mk_1 & \| & (\psi_Q(T_Q) \downarrow 1 \cup \psi_Q(T_Q) \downarrow 2 \cup \varphi_Q(L_Q)) \\
Mk_2 & \| & (\varphi_P(L_P) \cup \psi_P(T_P) \downarrow 1 \cup \psi_P(T_P) \downarrow 2) \\
Mk_2 & \| & (\psi_Q(T_Q) \downarrow 1 \cup \psi_Q(T_Q) \downarrow 2 \cup \varphi_Q(L_Q)) \\
Mk_1 & \| & Mk_2 \ \text{(ver definición 3.3)}
\end{array}$$

y $\Re_E$:

$$\begin{array}{lcl}
L_E & = & \{\mathbf{i}, \mathbf{o}, g_k, u, v, r, s\} \cup L_P \cup L_Q \\
T_E & = & T_P \cup T_Q \cup \{K_1, K_2, K_3, K_4\} \\
F_E & = & \{< \mathbf{i}, K_1 >, < K_1, s >, < u, K_2 >, < K_1, \mathbf{i}_P >, < K_1, u >, \\
 & & < v, K_4 >, < K_1, \mathbf{i}_Q >, < u, K_3 >, < v, K_5 >, < \mathbf{o}_Q, K_3 >, \\
 & & < \mathbf{o}_P, K_2 >, < K_4, \mathbf{o} >, < s, K_4 >, < K_4, u >, < K_4, r >, \\
 & & < K_2, v >, < K_3, v >, < r, K_5 >, < K_5, g_k >\} \\
 & & \cup F_P \cup F_Q \\
\varphi_E(\mathbf{i}) & = & \varphi_Q(\mathbf{i}_Q) \circ \varphi_P(\mathbf{i}_P) \circ Mk_1 \circ Mk_2 \\
\varphi_E(L_E - \{\mathbf{i}\}) & = & \{Id\} \\
\psi_E(T_P) & \equiv & \psi_P(T_P) \ \text{(i.e.}(\forall t \in T_P)(\psi_E(T) = \psi_P(T)) \\
\psi_E(T_Q) & \equiv & \psi_Q(T_Q) \ \text{(i.e.}(\forall t \in T_Q)(\psi_E(T) = \psi_Q(T))
\end{array}$$

$$\begin{array}{lcl}
\psi_E(K_1) \downarrow 1 & = & \lambda d.d \downarrow k_1 \ \text{(único índice del codominio de } Mk_1) \\
\psi_E(K_1) \downarrow 2 & = & \overline{Mk_1} \\
\psi_E(K_2) \downarrow 1 & = & \psi_E(K_3) \downarrow 1 = \psi_E(K_4) \downarrow 1 = \psi_E(K_5) \downarrow 1 = \text{cierto} \\
\psi_E(K_2) \downarrow 2 & = & \lambda d. \begin{cases} \overline{Mk_2} \circ confirmar(P))(d) & \text{si } d \downarrow k_2 \\ Mk_2 \circ deshacer(P))(d) & \text{sino} \end{cases} \\
\psi_E(K_3) \downarrow 2 & = & \lambda d. \begin{cases} \overline{Mk_2} \circ confirmar(Q))(d) & \text{si } d \downarrow k_2 \\ Mk_2 \circ deshacer(Q))(d) & \text{sino} \end{cases} \\
\psi_E(K_4) \downarrow 2 & = & Id \\
\psi_E(K_5) \downarrow 2 & = & Mk_1
\end{array}$$

En este caso las secuencias de ejecución a partir de la marcación inicial μ_0 será uno de los dos pares siguientes:

$$\begin{array}{c}
s \in \{< \mathbf{i}, K_1, ((P, K_2, K_4, \mathbf{o}) \| (Q, K_3, K_5, g_K)) >, \\
< \mathbf{i}, K_1, ((Q, K_3, K_4, \mathbf{o}) \| (P, K_2, K_5, g_K)) >\}
\end{array}$$

Nótese que es aquí donde se fabrica la operación asociada al lugar $\mathbf{i}$ (dos Mk_j para cada operador $\textcircled{\vee}$). Además el operador se bloquea hasta que el proceso que terminó de último sea anulado. La función $confirmar$ realiza una modificación permanente del estado y $deshacer$ invalida las modificaciones asociadas al proceso que no será tomado en cuenta.

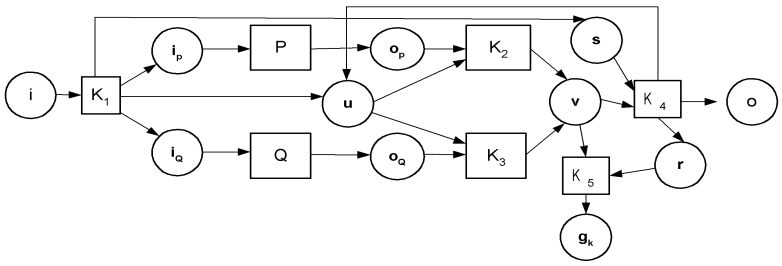

Figura 7: Anulación

Caso 6 Sea $E = P^+$ (Iteración)(ver figura 8)

Se tendrá las restricciones:

$$\begin{aligned} L_P \cap \{\mathbf{i}, \mathbf{o}, u, v\} &= \emptyset \\ T_P \cap \{R_1, R_2, R_3, R_4)\} &= \emptyset \end{aligned}$$

y $\Re_E$:

$$\begin{array}{lcl} L_E & = & \{\mathbf{i}, \mathbf{o}, u, v\} \cup L_P \\ T_E & = & T_P \cup \{R_1, R_2, R_3, R_4\} \\ F_E & = & \{< \mathbf{i}, R_1 >, < R_1, \mathbf{i}_P >, < \mathbf{o}_P, R_2 >, < R_2, u >, \\ & & < R_2, v >, < u, R_4 >, < v, R_4 >, < R_4, \mathbf{o} > \\ & & < u, R_3 >, < v, R_3 >, < R_3, \mathbf{i}_P >\} \\ & & \cup F_P \\ \varphi_E(\mathbf{i}) & = & \varphi_P(\mathbf{i}) \\ \varphi_E(L_E - \{\mathbf{i}\}) & = & \{Id\} \\ \psi_E(T_P) & \equiv & \psi_P(T_P) \text{ (i.e.}(\forall t \in T_P)(\psi_E(t) = \psi_P(t)) \\ \psi_E(R_1) = \psi_E(R_2) & = & < \text{cierto,Id} > \\ \psi_E(R_3) \downarrow 1 & = & pr_j \\ \psi_E(R_4) \downarrow 1 & = & \neg pr_j \quad (i.e. : \psi_E(R_4) \downarrow 1 = \neg(\psi_E(R_3) \downarrow 1)) \\ \psi_E(R_3) \downarrow 2 & = & \psi_E(R_4) \downarrow 2 = Id \end{array}$$

En este caso se tienen tres posibles secuencias de ejecución a partir de la marcación inicial μ_0 y son las siguiente:

$$\{< \mathbf{i}, R_1, P, R_2, R_4, \mathbf{o} >, < \mathbf{i}, R_1, (P, R_2, R_3)^+, P, R_2, R_4, \mathbf{o} >, \\ < \mathbf{i}, R_1, (P, R_2, R_3)^+ >\}$$

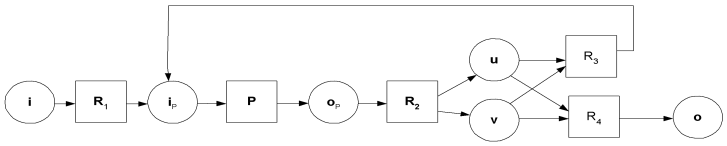

Figura 8: Iteración

4 Propiedades

En esta parte nos ocuparemos de analizar las redes de Petri interpretadas a la luz del comportamiento de las mismas como descriptores de procesos. El ambiente arrastra la información para el procesamiento de la transacción y la red establece el momento en que se puede realizar la transformación y/o creación de información. Entre las características más importantes de una red de procesos podemos señalar las siguientes:

- El proceso debe ser finito: Se debe asegurar que un token colocado en **i** llegue a **o** en un tiempo finito (condición que asegura la terminación de todo proceso) y estipulado por la evolución de la red y el token de entrada sea transformado por las operaciones asociadas a las transiciones que atraviesa, hasta su llegada a la salida (**o**).

- El estado inicial de la red será μ_0, el único token del formulario inicial en **i** y ningún otro token en los restantes estados.

- Las transiciones sereán a partir del par $< d_0, \mu_0 >$, donde d_0 corresponde al formulario inicial del ambiente y μ_0 corresponde al marcaje inicial con un token, aún no desponible, en **i**.

- El estado final $< d_f, \mu_f >$ tendrá un único token en el estado final **o**, que corresponde al formulario final(d_f) que se archiva al cerrar un proceso, más tokens a nivel de los g_k, que corresponden a los sumideros.

$$(i.e. \mu_f(\mathbf{o}) =< s >, \mid \mu_f(g_k) \mid \geq 0, \mu_f(L - (\{\mathbf{i}\} \cup \{k\}) =<>)$$

Nos referiremos primero a propiedades estructurales de las redes de Petri que manejamos en este trabajo, debido a la forma de construcción de las redes en base a la ley de composición (casos 2 a 4) de la base de redes. En [3], cuando se hace el análisis de la estructura de una buena red de flujo de trabajo (WF-net) se dice que necesariamente una red que incluye un **AND**-split debe recogerlo con un **AND**-joint. En los ejemplos de procesos mostrados anteriormente en 2.2, mostramos que hay circunstancias que justifican el disparar tareas en paralelo, esperando que sólo una de ellas concluya. Es por esto que el marcaje final solamente asegura que el lugar final **o**finaliza con un solo token, mientras que lugares intermedios pueden quedar con tokens que no serán procesados (estarán en sumideros).
A continuación revisamos parte de las propiedades de redes mencionadas en [3], extendiendolas al caso de redes de Petri interpretadas, que caracterizan redes de calidad expresando procesos. Veremos que las propiedades deseables que se mencionan [3] se aseguran básicamente en la redes que se manejan ya que su estructura está determinada por los elementos base de construcción propuestas en este trabajo.

Definición 4.1 (Fuertemente conexa) *Una red será fuertemente conexa sii para cualquier par de nodos x y y existe un camino de x a y.*

Definición 4.2 (Consistente) *Una red de Petri será consistente si cualquier estado μ alcanzable desde el estado inicial μ_0, existe una secuencia de disparos que llevan del estado μ al estado final de la red (un token en* **o***).*

Definición 4.3 (Acotada, segura) *Una red de Petri interpretada (PN, μ) es acotada sii para cada lugar p existe un natural n tal que para todo estado alcanzable el número de tokens habilitados en p es menor que n. La red es segura si en cada lugar el número máximo de tokens habiltados no excede a uno.*

Para las redes de Petri interpretadas que se han contruido en la sección 3.2, se tendrán los resultados siguientes:

Resultado 4.1 $\bullet\mathbf{i} = \emptyset \ \wedge \ \mid \mathbf{i}\bullet \mid = \ 1 \ \wedge \ \bullet\mathbf{i} = \emptyset \ \wedge \ \mid \mathbf{i}\bullet \mid = \ 1$
Con este resultado y agregando una flecha del lugar **o** *al lugar* **i** *podemos ver que las redes que se han contruido a partir de las expresiones de procesos son fuertemente conexas cuando el nodo de partida no pertenece al grupo sumidero en una subred* $\circledcirc$.

Resultado 4.2 *La red es consistente. Para los casos 2 a 5, está claro que $\mu_0 \rightarrow \mu_f$ (son consistentes). Para que las redes que utilizan el caso 6 sean consistentes hay dos mecanismos:*

1. *Verificar que el predicado en R_4 alcance el valor cierto, por ejemplo un número acotado de iteraciones.*

2. *Colocando a travéz del mecanismo de composición (caso 5) un reloj globalmente (en la red completa) o localmente, con la red que representa el caso 6 y no puede ser verificado. Es decir:*

$$t_1 \circ (t_2 \wedge t_3) \circ (t_4 \circledcirc (t_5 \circ t_6)^+)$$

para el caso global

$$(t_1 \circ (t_2 \wedge t_3) \circ (t_4 \circledcirc (t_5 \circ t_6)^+)) \vee r$$

y para el caso local

$$t_1 \circ (t_2 \wedge t_3) \circ (t_4 \circledcirc (t_5 \circ t_6)^+ \vee r)$$

Esta estructura, por un lado, asegura la consistencia de las redes de procesos, y por el otro lado, es natural dado que se está modelando procesos administrativos.

Resultado 4.3 *La red es segura, en el sentido que si μ es un marcaje que se obtiene a partir de μ_0 ($\mu_0 \rightarrow \mu$) tendremos:* $\mid \mu(L - \{g_{k_1}, \cdots\}) \mid \leq 1$

Resultado 4.4 *Es fácil demostrar las propiedades dadas en la sección 2.2.*

5 Conclusiones

El objetivo de este lenguaje LUPA, es modelar el proceso asociado a una transacción, muchas de las tareas que forman parte de las expresiones de procesos que hemos presentado en la sección 2, corresponden a comandos en aplicaciones administrativas de la empresa (sistemas contables-presupuestarios, procura, nómina, etc.). De esta forma, este lenguaje integra los sistemas administrativos. Está claro que algunas de las tareas en las expresiones de procesos necesitarán de la intervención humana, lo cual es facilmente modelable agregando al predicado de la transición correspondiente esa intervención.
La sintaxis que hemos utilizado para el lenguaje de procesos son las expresiones regulares de procesos, las cuales utilizando un número reducido de operadores representan en forma sencilla la modularidad en los procesos (i.e.:subprocesos), siendo estos operadores los conectores entre esos subprocesos. Un operador a destacar es la anulación que permite la anticipacón de ejecución de subprocesos que no necesariamente se utilizarán en el proceso final. En el modelo semántico para las expresiones de procesos que se ha establecido con las redes de Petri interpretadas, se asume un proceso para cada transacción. Esta restricción puede ser superada con la inclusión del identificador de transacción (χ_t), tanto en el ambiente, a nivel de los estados, como en la secuencia de marcas asociadas a los lugares ($\mu(l)$); es decir, coloreando la red con el identificador χ_t. Con esto se tiene un solo proceso para un tipo de transacción y no varias copias del proceso para transacciones simultaneas de un mismo tipo. Una restricción importante es la de auto-inhibición en la red asociada al operador de anulación ($\vee$), ver sección 3.2, caso 5. Esta restricción hace que dicho operador no sea asociativo cuando se encuentra dentro de una iteración ($+$). Una forma de solventar esta situación es creando una memoria local más grande en la red asociada al operador de anulación, con el objeto de manejar distintas pasadas (consecuencia de una iteración) de una misma transacción por ese operador.

El uso de redes de Petri como modelo semántico de las expresiones de procesos fue elegido, por una parte, por la sencillez de su funcionamiento, y por otra parte, dada la multiplicidad de herramientas disponibles para el análisis y presentación gráficas de dichas redes [5].

Referencias

[1] Commitee Draft ISO/IEC 15909. High-level petri nets - concepts, definitions and grafical notation. Disponible en
URL:http://www.daimi.au.dk/PetriNets/standardisation/, 2002.

[2] G.W. Brams. *Réseaux de Petri: Théorie et pratique.* Texte de l' Agence de l' Informatique. Masson, 1983.

[3] Will M.P.van der Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. *Lecture Notes in Computer Science, 1806*, 2000.

[4] Bernardiello F and DeCindio F. *A survey of Basic Models and Modular Net Classes.* LNCS vol 609. Springer Verlag, 1992.

[5] CNP group. Petri nets world. Disponible en
URL:http://www.daini.au.dk/PetriNets, 2002.

[6] Peterson J. Petri nets. *ACM Computing Surveys*, pages 223–252, Sept 1977.

[7] Gerrit K. Janssens, Jan Verelst, and Bart Weyn. Techniques for modelling workflows and their support of reuse. *Lecture Notes in Computer Science, 1806*, 2000.