Modelos para la Computación Móvil

Juan Francisco Díaz
Universidad del Valle
jdiaz@univalle.edu.co
Camilo Rueda
Universidad Javeriana-Cali
crueda@atlas.puj.edu.co

Resumen

En este artículo presentamos un análisis comparativo de algunos cálculos de computación móvil propuestos recientemente. Al tiempo que se describen el $\pi-$ cálculo propuesto por Milner en [RMW92], un cálculo de ambientes propuesto por Cardelli y Gordon en [CG98] y $PiCO$, un cálculo de objetos y restricciones propuesto por el grupo AVISPA [1] en [ADQ+00], se comparan entre ellos y se comentan sus principales características asociadas a la computación móvil. Otros cálculos como MCC ([R97]) y $DyTyCO$ ([VLSF99]) también son analizados.

1 Introducción

La formidable expansión de Internet en la última década ha venido cimentando la idea de la red como un vasto espacio computacional. De hecho, los diferentes protocolos de red establecen ya distintos tipos de *estructuras globales de información* ([Car97]), que van desde sistemas globales de archivos (FTP) hasta hipertextos globales (HTTP). Adicionalmente, la noción de *interfaces de aplicación* (API) del lenguaje *Java* ha mostrado en la práctica la conveniencia de aplicaciones o herramientas de software que puedan desplazarse (más que de computación, se trata de desplazamiento de código fuente serializado, en este caso) para encontrar el recurso de cómputo que garantice su eficiencia, confiabilidad o seguridad.

El futuro desarrollo de nuevos y más eficientes protocolos de comunicación para la red mundial conducirá seguramente a explorar la consecuencia natural de esta primera idea: La transmisión de computación puede en muchos casos aventajar a la transmisión de datos de un sitio a otro. Si consideramos la red mundial como un enorme conglomerado de dispositivos de cálculo ligados por "buses" de comunicación, esto es, como un computador con posibilidades de paralelismo masivo, podría pensarse en comprender su funcionamiento a partir de formalismos ya establecidos en este dominio. Además de las diferencias evidentes en los tiempos de comunicación entre componentes, la asimilación resultaría incompleta en varios aspectos: La heterogeneidad de los dispositivos de cómputo, la necesidad de encapsular diferentes modalidades de seguridad y confiabilidad, la pertinencia de los lenguajes de programación existentes y, muy especialmente, las nuevas modalidades de interacción existentes para la red. Para un usuario de la red la seguridad de los sitios y su relativo desempeño son propiedades *visibles* en las interacciones. Por lo tanto deberían, al menos en principio, hacer parte de los formalismos de computación.

Cuando la base tecnológica sobre la que se implanta el software cambia de manera tan radical, es prudente construír diferentes perspectivas rigurosas de lo que significa, en sus fundamentos, una computación sobre este nuevo entorno.

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

La computación tradicional en la década de los ochenta supo hallar en el λ-cálculo el fundamento para expresar alcances y propiedades de la programación. Sin embargo, ya hacia el final de esa década, la *práctica* del desarrollo de software sobre lenguajes orientados-objeto ponía en entredicho ese modelo al dar preponderancia (si bien no formal) a las nociones de *interacción* y *concurrencia*. En su discurso de premio Turing, Milner[Mil93] expresa la necesidad de fundamentar formalmente la programación orientada-objetos en un *cálculo* de la concurrencia, construído sobre el tratamiento riguroso de la noción de *nombre* o *referencia*. Este cálculo, llamado π-cálculo [RMW92], constituye el punto de partida obligado de la mayoría de los formalismos propuestos para los nuevos esquemas de computación.

En este artículo presentamos un análisis comparativo de varios de los cálculos de computación móvil propuestos recientemente.

2 El π-cálculo

El funcionamiento de los procesos secuenciales puede entenderse de manera apropiada como una actividad de cálculo de unos valores de salida a partir de ciertos valores de entrada. La noción matemática de *función* describe con exactitud esa relación de entrada-salida. La manera exacta en que los procesos secuenciales interactúan para lograr un resultado puede definirse de manera rigurosa y elegante mediante el λ-cálculo, formalismo cuya base es precisamente la noción de función.

Cuando se consideran procesos concurrentes la situación es diferente. Dos procesos concurrentes pueden interactuar de formas que no resulta natural reducir a una operación funcional de entrada-salida. Por ejemplo, el resultado de la interacción de dos procesos interconectados puede ser el que uno de ellos se conecte a otro proceso diferente. Si la noción de función no es pertinente, qué puede constituír entonces una base para pensar la concurrencia? La respuesta del π-cálculo es la noción de *nombre*. Un proceso debe ser identificable y capaz de mantener su identidad a través de una computación. Debe además poder identificar otros procesos con los que desea comunicarse. En el π-cálculo un *nombre* identifica un *canal* de comunicación. Dos procesos que se asocian a un mismo canal pueden comunicarse. Si el propósito fundamental de la identificación de un proceso es el de poder comunicarse con él, no resulta necesario asociar nombres a los procesos mismos puesto que siempre se puede acceder a un proceso a través del canal al cual está ligado. Desde luego que debe ser posible definir también procesos que actúen funcionalmente, pero este será un comportamiento derivado de nociones más fundamentales.

2.1 Sintaxis

La sintaxis de los procesos del π-cálculo (monádico) es la siguiente:

$$\boxed{P \ ::= \ \textstyle\sum_{i\in I} \pi_i.P_i \mid P|Q \mid !P \mid (\nu\, x)P}$$

en donde P, Q son procesos, x es un nombre y π un *prefijo*. El prefijo denota una *acción atómica* previa al proceso que le sigue. Puede ser de dos formas:

$x(z)$	significa:	Leer un dato z a través del canal x. El nombre z se liga en el proceso
$\overline{x}d$	significa:	Enviar un dato d a través del canal x. El nombre d no se liga

El proceso $P|Q$ denota la composición paralela de los subprocesos P, Q. Además de las acciones individuales de cada subproceso, una interacción posible del proceso $P|Q$ es que los subprocesos P y Q se comuniquen.

Por ejemplo: $x(y).\overline{y}z \mid \overline{x}(w).0$ denota la interacción de un proceso que envía un dato w por el canal x con un proceso que lee ese dato (llamándolo y) por el mismo canal. Esta acción de comunicación del dato w se considera atómica. El primer proceso de la composición continúa después con el envío de un dato z a través del canal w. La definición formal de este comportamiento se establece mediante *reglas de reducción*, que se presentan más adelante.

El proceso $!P$ representa la *replicación* de P. Es decir, produce tantas copias de P como se quiera (un número finito, si deseamos computaciones finitas!):

$!P = P|P|P|...$

Un nombre puede estar restringido al contexto de un proceso. Este es el propósito de $(\nu\, x)P$. El nombre x es local a P y solamente visible dentro de él. La localidad de nombres permite restringir comunicaciones a ciertos contextos.

El último tipo de procesos es la *suma* $\sum_{i\in I} \pi_i.P_i$, que representa escogencia no determinística. La escogencia no está controlada por el mismo proceso *suma* sino que depende de la comunicación. Los procesos P_i que intervienen en la suma deben ser procesos *normales*, definidos por la sintaxis:

$M, N ::= \pi.p \mid 0 \mid M + N$

El proceso 0 denota inacción. El proceso suma es *pasivo* hasta tanto no exista otro proceso externo capaz de comunicarse con alguno de los sumandos. Es en este caso cuando se escoge alguno para realizar la comunicación. Los no escogidos se descartan. Por ejemplo, en

$(x(y).\overline{z}y.0 + z(y).0 + x(w)w(z).0) \mid \overline{x}r.0$
el primero o el tercer sumando pueden escogerse para la comunicación. Si se escoge el tercero, el resultado es el proceso $r(z).0$.

2.2 Semántica

La semántica del cálculo se establece mediante una relación de congruencia y unas reglas de reducción. La relación de congruencia abstrae suficientes detalles como para considerar iguales ciertos procesos que no son sintácticamente idénticos. Por ejemplo, dos procesos que difieren únicamente en sus nombres ligados son realmente el mismo. Otro caso evidente es la conmutatividad de procesos en suma y en composición. Menos evidentes, quizá, son las siguientes:

$!P \equiv P\|P$
$(\nu\, x)0 \equiv 0,\ (\nu\, x)(\nu\, y)P \equiv (\nu\, y)(\nu\, x)P$
$(\nu\, x)(P\|Q) \equiv P\|(\nu\, x)Q$ si x no está libre en P

La relación de reducción es el mecanismo de operación del cálculo: Define cómo transformar unos procesos en otros. Se especifica mediante las siguientes reglas:

1. COM: $(... + x(y).P) \mid (... + \overline{x}z.Q) \rightarrow P\{z/y\}|Q$
2. PAR: $(P \rightarrow P') \ \Rightarrow\ (P|Q \rightarrow P'|Q)$
3. RES: $P \rightarrow P' \ \Rightarrow\ (\nu\, x)P \rightarrow (\nu\, x)P'$
4. STRUCT: $(Q \equiv P\ P \rightarrow P'\ P' \equiv Q') \ \Rightarrow\ (Q \rightarrow Q')$

Ejemplos de reducciones son:

$\overline{x}y.0 \mid x(u).\overline{u}v.0 \mid \overline{x}z.0 \;\rightarrow\; \overline{x}y.0 \mid \overline{z}v.0 \mid 0$

$(\nu\, x)(\overline{x}y.0 \mid x(u).\overline{u}v.0 \mid \overline{x}z.0) \;\rightarrow\; 0 \mid \overline{y}v.0 \mid \overline{x}z.0$

Se puede ver que la regla de COM permite reducir un sumando únicamente mediante comunicación. También que no hay reducción definida para procesos con prefijo, aparte de la comunicación. Las reglas PARy RES establece que los procesos pueden reducir bajo composición y restricción. La regla STRUCT dice que los procesos congruentes tienen las mismas reducciones.

El π-cálculo restringe la comunicación de datos exclusivamente a nombres. Qué es entonces en este contexto *movilidad*? La respuesta está en lo que es *observable* de un proceso: Sus posibilidades de comunicarse con otros a través de canales. Como los canales son nombres, pueden intervenir en una comunicación de tal manera que se modifique la interconexión de los procesos. Un proceso asociado a un canal de comunicación a puede reasociarse a otro canal b, como se muestra en el siguiente ejemplo (adaptado de [Mil91]). Supongamos un proceso CARRO con dos comportamientos básicos: *Hablar*, que consiste en comunicarse con procesos asociados a se canal y *cambiar*, que modifica su canal para comunicarse en el futuro con otros procesos diferentes. Supongamos también un proceso SISTEMA que produce procesos que pueden comunicarse con CARRO y que eventualmente puede ordenar a éste cambiar su canal de comunicación. Los procesos se muestran a continuación:

$$\begin{aligned} CARRO \quad \equiv^{def} \quad & (\nu\, carro)\, !carro(hable, cambie). \\ & (hable.\overline{carro}\; hable\; cambie \\ & + cambie(hable', cambie').\overline{carro}\; hable'\; cambie') \end{aligned}$$

$$SISTEMA \quad \equiv^{def} \quad (\nu\, h\, c\, h'\, c')\; (\overline{carro}\; h\; c \mid \overline{h\prime}.P \mid \overline{h}.Q \mid \overline{c}\; h'\; c')$$

La definición de CARRO es un proceso replicado (o sea, persistente) con escogencia de los dos comportamientos mencionados, cada uno identificado por un canal particular. El proceso SISTEMA se supone que opera en paralelo con CARRO. Además de definir nombres locales para los canales, SISTEMA crea un carro que transmite por los canales h y c; un subproceso "habla" con ese carro por el canal h (su continuación es Q) y otro subproceso lo hace por el canal h' (su continuación es P). Este último solamente puede hablar después de que el subproceso más a la derecha ha enviado al carro el mensaje de "cambio" correspondiente. El resultado es el "movimiento" del proceso CARRO: Su canal de comunicación ya no es el mismo.

2.3 Observabilidad

Puede inferirse del ejemplo descrito en la sección anterior que en cada momento la operación del cálculo está suscrita a las posibilidades de comunicación entre los procesos. Lo que es importante observar de los procesos es entonces el conjunto de canales que les son accesibles. Es decir, un proceso solamente es observable a trvés de su canal de comunicación. Formalmente, un proceso P es *observable* en α, escrito $P \downarrow_\alpha$, si algún $\alpha.A$ ocurre *sin guarda* en P. Un agente (proceso) ocurre sin guarda si no está bajo algún perfijo. Por ejemplo, $P = (\nu\, x)(a(y).Q \mid x(z).R)$ es observable en $a(y)$, pero no en $x(z)$. La forma débil de esta relación de observabilidad es permite reducir antes de observar: Un proceso P es *eventualmente observable* en α, escrito $P \Downarrow_\alpha$, si $P \longrightarrow^* Q$ y $Q \downarrow_\alpha$.

La noción de observabilidad establece una manera elegante de definir equivalencias entre procesos. Dos procesos P y Q son (contextualmente) *equivalentes* si siempre eventualmente observan lo mismo, para todo contexto de procesos en el que el uno o el otro pueda existir. Es decir, si dos procesos tienen eventualmente exactamente las mismas posibilidades de comunicación, entonces son indistinguibles.

La simplicidad y elegancia del π-cálculo para representar procesos concurrentes tiene su precio en ciertas limitaciones. Por un lado, si bien el uso de canales permite a un proceso solicitar servicios de varios otros, no es fácil para ese proceso restringir el envío de un mensaje a *algunos* de los otros procesos que utilizan

su mismo canal. Extendiendo la analogía utilizada por Milner, un proceso puede gritar "ayuda!", pero no puede impedir que el mensaje llegue también a quien busca hacerle daño. Por otro lado, como se verá en la presentación del cálculo de *ambientes*, existen particularidades de los sitios en Internet que no pueden representarse adecuadamente mediante la simple movilidad de canales de comunicación de los procesos.

3 Un cálculo de *ambientes*

Como ya se dijo, la seguridad de un sitio o su disponibilidad son propiedades observables en la interacción con Internet. En el π-cálculo la noción de *sitio* no es fácilmente representable mediante un proceso ligado a un canal. Conocer los "nombres" (direcciones IP) asociadas con un sitio no necesariamente garantiza su acceso o, si lo permite, no necesariamente deja a libre disposición todos sus recursos. Cada sitio está en general organizado jerárquicamente en términos de los diferentes recursos visibles. Existen, por lo demás, fronteras claras entre los "paquetes" de recursos accesibles a diferentes usuarios, que son fruto de políticas independientemente administradas en cada sitio. En [CG98] se denominan *ambientes* los componentes de la jerarquía y se formaliza, mediante un cálculo, la noción de computaciones que se desplazan a través de las fronteras que los delimitan. Un aporte fundamental de este cálculo es el modelamiento de movilidad tanto de los agentes (procesos) que interactúan en un ambiente, como de los ambientes mismos. Esto permite representar el escenario de un usuario que se desplaza a un punto distante geográficamente y que encuentra en su destino el mismo ambiente de computación (procesos en ejecución y datos, por ejemplo) que dejó en su lugar de origen. La dificultad esencial en el modelamiento de este escenario es el tratamiento de los aspectos de seguridad. No cualquier agente debe poder atravesar cualquier frontera. Las políticas de control deben estar claramente representadas y las propiedades de seguridad deben ser demostrables en el cálculo.

3.1 Sintaxis

La sintaxis del cálculo de ambientes de Cardelli se muestra en la tabla 1.

n		nombres
P, Q	$::=$	Procesos
	$(\nu\, n)P$	restricción
	0	inactividad
	$P\|Q$	composición
	$!P$	replicación
	$n[P]$	ambiente
	$M.P$	acción
M	$::=$	Capacidades
	$in\, n$	puede entrar en n
	$out\, n$	puede salir de n
	$open\, n$	puede abrir n

Tabla 1: Sintaxis del cálculo de ambientes

La idea fundamental es que los nombres, a diferencia del π-cálculo, no identifican procesos sino ambientes. Aún más, la facultad de movilidad está asociada también exclusivamente con los ambientes. Se supone además que los nombres que identifican los ambientes no son públicos. Los procesos tienen *capacidades* con relación a los ambientes (salir de ellos, entrar, abrirlos), pero de esas capacidades no puede reconstituírse el nombre del ambiente sobre el que operan. Algunos de los procesos del cálculo de ambientes ya se describieron en el π-cálculo. La operación de los otros se describe a continuación.

3.2 Semántica

El proceso $n[P]$ representa un ambiente de nombre n. Sus servicios o recursos están representados por el proceso P en el interior del ambiente. El proceso $M.P$ ejecuta una acción regida por una capacidad M. Como en los procesos con prefijo del π-cálculo, P es la continuación de esa acción de entrada, salida o apertura de un ambiente. La relación de congruencia y de reducción se muestra en las tablas 2 y 3. En estas tablas se incluyen únicamente reglas que involucran procesos cuyo funcionamiento difiere de los ya vistos para el π-cálculo.

$P \equiv Q \;\Rightarrow\; n[P] \equiv n[Q]$
$P \equiv Q \;\Rightarrow\; M.P \equiv M.Q$
$(\nu\, n)(m[P]) \equiv m[(\nu\, n)P]$ si $n \neq m$

Tabla 2: Congruencia en el cálculo de ambientes

$n[in\ m.P \mid Q] \mid m[R] \longrightarrow m[n[P \mid Q] \mid R]$	(Red in)
$m[n[out\ m.P \mid Q] \mid R] \longrightarrow n[P \mid Q] \mid m[R]$	(Red out)
$open\ n.P \mid n[Q] \longrightarrow P \mid Q$	(Red open)
$P \longrightarrow Q \;\Rightarrow\; n[P] \longrightarrow n[Q]$	(Red Amb)

Tabla 3: Relación de reducción en el cálculo de ambientes

En la tabla 3 la regla *Red in* establece la entrada de un ambiente en otro. El proceso que solicita entrada a un ambiente m lo hace dentro de otro ambiente n. Su ingreso a m incluye el ambiente en el que estaba. La movilidad entonces, como ya se dijo, se refiere a ambientes completos, no a procesos o a sus canales de comunicación. Similarmente, la regla *Red out* representa un ambiente que atraviesa la frontera de otro para salir de él. La regla *Red open* ofrece la posibilidad de "borrar" la frontera de un ambiente. Cualquiera de estas reglas requiere que los ambientes involucrados existan, de lo contrario el proceso en cuestión se bloquea. Por ejemplo, $n[in\ m.P \mid Q]$ se bloquea si no hay un ambiente m al mismo nivel que el ambiente n. Es interesante apreciar que estas reglas, junto con *Red Amb*, simulan ambientes que se desplazan mientras sus cómputos (es decir, en este contexto, sus reducciones) continúan. Hay movilidad real de la *computación*.

El siguiente ejemplo (adaptado de [CG98]) modela el acceso de un agente (ambiente) a servicios a través de un "cortafuegos" (*firewall*). El ambiente que representa el *firewall* debe mantener su nombre en secreto (conocerlo implica poder atravesar el *firewall*). Todo proceso que desee obtener acceso al *firewall* debe primero autenticarse. Una vez permitido el acceso a un agente, se debe garantizar que sus actividades permanecen controladas, esto es, que se desarrollan en un ambiente local en el interior del *firewall*.

En la figura 1 se ilustra este proceso. Un agente (ambiente identificado por una palabra clave p) solicita entrar al *firewall*, cuya identificación w permanece oculta para todo proceso exterior a él. El ambiente w envía un proceso guía g que conoce la palabra clave p. Este guía cruza la frontera del *firewall* y se introduce en el ambiente del agente p. El agente rompe entonces la frontera del guía (que deja así de ser un ambiente) para hacer que sus procesos hagan parte de p. Uno de estos procesos, que conoce el nombre w, introduce su ambiente (en este caso p) dentro del *firewall*. Finalmente, otro proceso dentro de w rompe a su vez la frontera de p y de sus subprocesos encapsulados y deja que se ejecuten dentro del ambiente w. La definición formal de los ambientes y las reducciones que simulan este proceso se muestran a continuación:

$firewall =^{def}\ (\nu\, w)w[g[out\ w.in\ p.in\ w] \mid open\ p.open\ p'.F]$

$Agente =^{def}\ p[open\ g.p'[Q]]$
Interacción:
$Agente \mid firewall \equiv$

$p[open\ g.p'[Q]] \mid (\nu\, w)w[g[out\ w.in\ p.in\ w] \mid open\ p.open\ p'.F]$

$\longrightarrow^{*}\ (\nu\, w)p[open\ g.p'[Q] \mid g[in\ w]] \mid w[open\ p.open\ p'.F]$

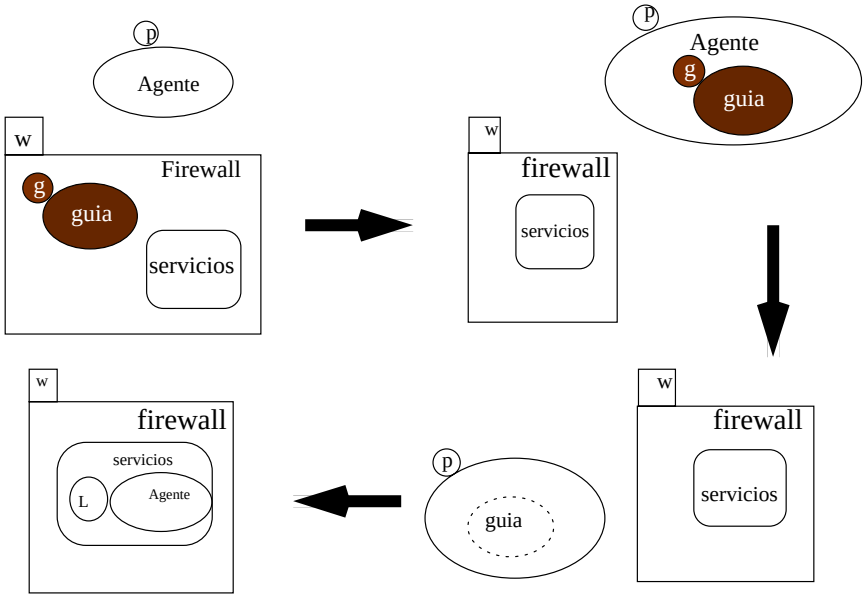

Figura 1: Cruce de un *firewall*

$\longrightarrow (\nu\, w)p[p'[Q] \mid in\ w] \mid w[open\ p.open\ p'.F]$

$\longrightarrow (\nu\, w)w[p[p'[Q]] \mid open\ p.open\ p'.F]$

$\longrightarrow^* (\nu\, w)w[Q \mid F]$

Como en el π-cálculo, la noción de lo que es observable permite definir de manera simple equivalencia entre procesos. A continuación se define formalmente observabilidad para el cálculo de ambientes.

3.3 Observabilidad

Lo que es importante observar de un proceso son los nombres de los ambientes de nivel superior que lo componen puesto que son ellos los que determinan su operación (sus reducciones posibles). Formalmente, un proceso P *exhibe un ambiente* n, escrito $P \downarrow_n$, si P es un proceso que contiene un ambiente de nivel más superior llamado n. Más precisamente,

$P \downarrow_n =^{def} P \equiv (\nu\, m_1 ... m_k)(n[Q] \mid R)$, donde $n \neq \{m_1 ... m_k\}$

La versión débil de la relación es: *P exhibe eventualmente un ambiente n,* escrito $P \Downarrow_n$, si P exhibe n posiblemente después de realizar reducciones. Formalmente:

$P \Downarrow_n =^{def} P \longrightarrow^* Q$ y $Q \downarrow_n$

Como en el π-cálculo, dos procesos son contextualmente equivalentes si observan eventualmente exactamente los mismos ambientes, bajo cualquier contexto.

A diferencia del π-cálculo, no hay primitivas de comunicación de datos en el cálculo de ambientes. En [CG98] se demuestra que este cálculo de ambientes es Turing-completo, de modo que en principio tales primitivas no son necesarias. Sin embargo, en la versión extendida del cálculo que se incluye en ese mismo artículo, se agregan mecanismos para la comunicación de *secuencias* de capacidades y se extiende la sintaxis de las capacidades para incluír variables, nombres y secuencias. La idea es facilitar el modelamiento de situaciones usuales tales como el envío de un paquete de datos un sitio a otro a través de enrutadores. La secuencia de capacidades modela en este caso el camino particular que se establece para el envío. Como las capacidades son

el objeto de las comunicaciones, tales caminos pueden pasarse a los procesos como argumentos, incrementando así significativamente, en la práctica, el poder expresivo del cálculo.

3.4 Consideraciones

El cálculo de ambientes es una contribución importante a la formalización de movilidad de la computación contexto reales, como *internet.* Es simple, elegante, expresivo y suficientemente próximo del π-cálculo como para poder heredar sus formalismos (equivalencia contextual, por ejemplo). La idea fuerte de esta similitud es abstraer la noción de proceso en la de ambiente, también identificados por nombres. La reducción de toda computación a interacciones de entrada, salida y apertura de ambientes permite controlar la complejidad del cálculo. Es lo que Milner ([Mil91]) denomina la deseable *parsimonia* de un formalismo de la concurrencia.

El precio de la parsimonia son algunas dificultades en expresar mecanismos concretos de interacción. Por ejemplo, es natural pensar que la comunicación de mensajes entre procesos involucre un agente local de un ambiente de origen, un agente local de un ambiente de destino, un enrutador (o varios), unos datos e información sobre su provenencia. Adicionalmente, una computación (ambiente) que se desplaza tiene necesidad de que se le garantice el reenvío de mensajes aún no recibidos y que ya fueron enviados a su sitio anterior ([AP97]). Aunque esta situación es *a priori* modelable en el cálculo, cabe preguntarse qué tanto complicaría el cálculo (la prueba de sus propiedades fundamentales) agregar estructura al dominio de nombres para representar naturalmente este tipo de situaciones.

Un aspecto más importante que resta sin modelar en el cálculo es el encapsulamiento de nombres que deben permanecer secretos. En el ejemplo del *firewall,* se *asume* que ciertos nombres no son conocidos al exterior del ambiente. Cómo probar, en el propio cálculo, que estas propiedades se mantienen? Una estrategia ([CGG00]) consiste en extender el tipamiento de nombres en el π-cálculo a *Grupos* que representan conjuntos de nombres privados cuya difusión debe estar confinada a ciertos procesos. La extensión de la noción de declaración local de nombres en el π-cálculo a declaración local de grupos permite garantizar esta privacidad. Una estrategia similar para el cálculo de ambientes parece enteramente posible.

4 PiCO : Un cálculo de Objetos y Restricciones

Como se mencionó anteriormente, si bien el uso de canales en el $\pi-$cálculo permite a un proceso solicitar servicios de varios otros, no es fácil para ese proceso restringir el envío de un mensaje a *algunos* de los otros procesos que utilizan su mismo canal.

La introducción de la noción de *restricción* combinada ortogonalmente con la noción de *objeto* y *paso de mensajes* como mecanismo de comunicación, son las innovaciones principales de $PiCO$, un cálculo concurrente (inspirado en el $\pi-$cálculo) de Objetos y Restricciones, diseñado como cálculo de base para construir herramientas de ayuda a la composición musical ([ADQ$^+$00, ADQ$^+$98].

La combinación mencionada se realiza de manera que las restricciones se utilizan como mecanismo de sincronización del paso de mensajes entre objetos.

$PiCO$ presenta dos características novedosas que lo distinguen de otros cálculos de objetos y de restricciones. Una es la noción de *objeto localizado en una restricción.* La otra es el concepto de *delegación* de mensajes.

En el π-cáclulo, la comunicación entre un proceso emisor y uno receptor se produce cuando ellos utilizan un mismo canal (denotado por un *nombre*). Si se mira cada nombre como un tipo, se puede decir que los procesos se comunican exactamente cuando el recpetor y el emisor utilizan el mismo tipo (único) de *comunicación.*

Por otro lado, en el ρ-cálculo ([Smo94]), los agentes emisor y receptor están "localizados" en variables. La comunicación se produce cuando se puede inferir, a partir de información global en forma de restricciones,

que las dos variables son iguales. Es decir, cuando se deduce, que los dos procesos tiene el mismo tipo (no necesariamente único) de comunicación.

En *PiCO* cada objeto receptor define explicitamente su tipo de comunicación "localizándose" a sí mismo en una restricción $\phi(\mathbf{sender})$, parametrizada en una variable **sender**. Esta restricción define, como tipo de comunicación, al conjunto de posibles demandantes de servicios del receptor (es decir, posibles emisores de mensajes al receptor).

Por su lado, los objetos emisores de mensajes se "localizan" en una variable (o en un nombre). Las restricciones sobre esta variable definen sus posibles valores. El tipo de comunicación del emisor es el conjunto de estos valores.

Entonces, un emisor y un receptor se pueden comunicar si, de la información global almalmacenada en forma de restricciones, se puede inferir que el tipo de comunicación del emisor de un mensaje es un *subtipo* del tipo de comunicación del objeto receptor. En caso contrario, la comunicación no se puede producir. Entonces, una forma de incrementar el conjunto de potenciales receptores de un mensaje consiste en "mover" la localización del emisor a un subtipo (es decir, agregar restricciones sobre su localización). De la misma manera, un receptor puede aumentar el conjunto de posibles demandantes de sus servicios "moviéndose " a un *supertipo* (es decir, añadiendo restricciones sobre las variables, distintas de **sender**, de su restricción de localización).

Nótese que los receptores con tipos de comunicación unitarios se comunican de igual forma que los procesos de lectura del $\pi-$cálculo. Y, los emisores que restringen su tipo de comunicación a lo que los receptores esperan, se comunican de la misma manera que las abstracciones del $\rho-$cálculo. Por tanto, la comunicación definida en *PiCO* es al menos tan expresiva que la del $\pi-$cálculo y la del $\rho-$cálculo.

En cuanto a la segunda característica mencionada, todo objeto receptor de mensajes en *PiCO* puede determinar un tipo de comunicación para *delegación* para responder aquellos mensajes que, aunque son enviados por emisores que tienen un tipo de comunicación aceptado por el receptor, solicitan un servico no provisto por el receptor. Este tipo de comunicación para delegación también se define por medio de una restricción. Delegar consiste entonces en "mover" el tipo del emisor del mensaje al tipo delegado.

4.1 Sintaxis

La sintaxis de *PiCO* se describe en la tabla 4. Existen tres procesos básicos: *Mensajes*, *Objetos* y *Restricciones*. Otros procesos como el proceso nulo, la replicación, la composición y la definición de una variable o un nombre local cumplen el mismo papel de los procesos análogos descritos en los cálculos anteriores.

El proceso $(\phi_{\mathbf{sender}}, \delta_{\mathbf{forward}}) \triangleright M$ representa un objeto (receptor de mensajes). El propósito de un objeto es responder a mensajes enviados por otros procesos. Un objeto responde a un mensaje remplazándose por otro proceso o *método*. El mensaje debe contener la información para escoger el proceso remplazante entre la colección de métodos del objeto.

El proceso $I \triangleleft m$ **then** P permite emitir mensajes. Todo proceso emisor de mensajes está *localizado* en un identificador (nombre, variable o valor). La restricción $\phi_{\mathbf{sender}}$ de cada proceso receptor (objeto) determina si un proceso emisor puede comunicarse con él. Si la localización del emisor satisface la restricción del receptor, el mensaje es recibido. Por ejemplo, el objeto $(\mathbf{sender} \in \{a, b\}, \mathbf{forward} \in \{b, c\}) \triangleright M$ acepta mensajes provenientes del proceso $a \triangleleft m[]$ **then** Q porque se cumple $a \in \{a, b\}$.

En caso que el mensaje recibido (m) no haga parte de los servicios ofrecidos por el objeto receptor ($m \notin M$), el mensaje es reenviado por medio de otra localización determinada por la restricción $\delta_{\mathbf{forward}}$, (llamada *condición de delegación*). Es decir, el proceso original $a \triangleleft m[]$ **then** Q se remplaza por un nuevo proceso emisor, $J \triangleleft m[]$ **then** Q, localizado en una nueva variable J la cual es restringida imponiendo la restricción δ. De esta manera un proceso receptor (el delegado) cuya restricción de localización sea satisfecha por J puede responder al mensaje m.

N	$::=$	Procesos Normales
	O	Inactividad
	$I \triangleleft m$ **then** P	Mensaje
	$(\phi_{\mathbf{sender}}, \delta_{\mathbf{forward}}) \triangleright M$	Objeto
R	$::=$	Procesos Restricciones
	tell ϕ **then** P	Imposición
	ask ϕ **then** P	Inferencia
P, Q	$::=$	Procesos
	local x **in** P	Variable Local
	local a **in** P	Nombre Local
	$P \mid Q$	Composición
	N	Proc. Normal
	clone P	Replicación
	R	Restricción
I, J, K	$::=$	Ids. Objetos
	a, l	Nombres
	v	Valores
	x	Variables
M	$::=$	Métodos
	$[l_1 : (\widetilde{x_1})P_1 \& \ldots \& l_m : (\widetilde{x_m})P_m]$	
m	$::=$	Mensajes
	$l : [\widetilde{I}]$	

Tabla 4: Sintaxis de $PiCO$

Por último, el comportamiento de los procesos de restricciones (imposición e inferencia) depende de un *almacén* global de información especificada en forma de restricciones. Este almacén es usado por $PiCO$ para controlar toda comunicación posible. El proceso de imposición **tell** ϕ **then** P significa "añadir ϕ al almacén y luego, activar P". El proceso de inferencia **ask** ϕ **then** P significa "Activar (destruir) P si la restricción ϕ ($\neg\phi$) es una consecuencia lógica de la información en el almacén. En caso contrario, suspender el proceso hasta que el almacén contenga sufiente información para decidir qué hacer".

4.2 Semántica

Al igual que en los cálculos descritos, la semántica de $PiCO$ se especifica mediante una relación de congruencia y una de reducción. Las tablas 5 y 6 incluyen únicamente reglas que involucran procesos cuyo funcionamiento difiere de los ya vistos para los cálculos anteriores (la notación $\phi \models\!\mid_{\Delta} \psi$ denota la equivalencia lógica de las restricciones ϕ y ψ).

$(\phi_{\mathbf{sender}}, \delta_{\mathbf{forward}}) \triangleright M \equiv (\phi_{\mathbf{sender}}, \delta_{\mathbf{forward}}) \triangleright M'$	si M' es una permutación de M.
tell ϕ **then** $P \equiv$ **tell** ψ **then** Q	
ask ϕ **then** $P \equiv$ **ask** ψ **then** Q	si $\phi \models\!\mid_{\Delta} \psi$ y $P \equiv Q$

Tabla 5: Congruencia en $PiCO$

La relación de reducción no se define directamente sobre los procesos sino sobre configuraciones. Una configuración es una pareja $\langle P; S\rangle$, donde P es un proceso y S un almacén.

En $PiCO$ el comportamiento de un proceso P se define por transiciones sobre la configuración inicial $\langle P; \top\rangle$. Una transición, $\langle P; S\rangle \longrightarrow \langle P'; S'\rangle$, significa que $\langle P; S\rangle$ puede ser transformada en $\langle P'; S'\rangle$ en un solo paso.

La regla *Comm* describe el resultado de la interacción entre el proceso emisor $I' \triangleleft l : [\widetilde{J}]$ **then** Q y el proceso receptor (objeto) $(\phi_{\mathbf{sender}}, \delta_{\mathbf{forward}}) \triangleright [l : (\widetilde{x})P \& \ldots]$. El almacén es utilizado para decidir si efectivamente se

$\dfrac{S\vdash_\Delta \phi[I'/\mathbf{sender}] \qquad \lvert\widetilde{K}\rvert=\lvert\widetilde{x}\rvert}{\left\langle I'\lhd l{:}[\widetilde{K}]\ \mathbf{then}\ Q \mid (\phi_{\mathbf{sender}},\delta_{\mathbf{forward}})\rhd[l{:}(\widetilde{x})P\&...];S\right\rangle \longrightarrow \left\langle Q \mid P\{\widetilde{K}/\widetilde{x'},I'/\mathbf{sender}\};S\right\rangle}$	Comm.
$\dfrac{S\vdash_\Delta \phi[I'/\mathbf{sender}] \quad S\sqcup\delta[I'/\mathbf{forward}]\vdash_\Delta \bot \quad l\notin Labels(M)}{\left\langle \begin{pmatrix} I'\lhd l:[\widetilde{K}]\ \mathbf{then}\ Q \mid \\ (\phi_{\mathbf{sender}},\delta_{\mathbf{forward}})\rhd M \end{pmatrix};S\right\rangle \longrightarrow \left\langle \begin{pmatrix} \mathbf{local}\ J\ \mathbf{in\ tell}\ \delta[J/\mathbf{forward}] \\ \mathbf{then}(J\lhd l:[\widetilde{K}]\ \mathbf{then}\ Q) \mid \\ (\phi_{\mathbf{sender}},\delta_{\mathbf{forward}})\rhd M \end{pmatrix};S\right\rangle}$	Del.
$\langle \mathbf{tell}\ \phi\ \mathbf{then}\ P;S\rangle \longrightarrow \langle P;S\wedge\phi\rangle$	Tell
$\dfrac{S\vdash_\Delta \phi}{\langle \mathbf{ask}\ \phi\ \mathbf{then}\ P;S\rangle\longrightarrow\langle P;S\rangle}$, $\dfrac{S\vdash_\Delta \neg\phi}{\langle \mathbf{ask}\ \phi\ \mathbf{then}\ P;S\rangle\longrightarrow\langle O;S\rangle}$	Ask

Tabla 6: Relación de reducción en $PiCO$

puede establecer una comunicación entre el emisor y el receptor. De ser así el proceso asociado al mensaje del emisor (en este caso P) es activado (haciendo las substituciones del caso), en paralelo con el proceso definido como continuación del proceso emsior (en este caso Q).

La regla *Del* describe formalmente cómo se delega el procesamiento de los mensajes (proceso descrito anteriormente). Las condiciones impuestas para la delegación evitan que el procesamiento del mensaje lo haga un proceso idéntico al original. Así se evitan ejecuciones infinitas por delegación.

Las reglas *Ask* y *Tell* describen la interacción entre los procesos de restricciones y el almacén. La regla *Tell* permite añadir información al almacén. Aumentar la información en el almacén es el prinicpal mecanismo para lograr que un proceso afecte el comportamiento de otros procesos en el sistema [Sar93]. La regla *Ask* permite consultar información del almacén y actuar en consecuencia. Cuando la consulta queda suspendida porque no se puede responder ni positiva ni negativamente, el proceso queda *Suspendido*. Un proceso suspendido puede ser despertado a posteriori, cuando se añada mayor información al almacén.

4.3 Movilidad en $PiCO$

El ejemplo de movilidad mostrado en la sección 2.2 es tratado en [ADQ+00] de la siguiente manera.

Suponga que un CENTRO está en contacto permanente con dos estaciones BASE, cada una en una parte distinta del país. Suponga también que un CARRO con un teléfono celular se mueve a lo largo del país, debiendo estar siempre en contacto con una BASE. Si el CARRO se aleja tanto de su BASE actual que se sale de su área de cubrimiento, un procedimiento que mantenga la comunicación se inicia, dando como resultado que el CARRO cancela el contacto con una BASE pero queda conectado con la otra. Una solución a este problema en $PiCO$ es (tomado de [ADQ+00]):

$industria\_carros \rhd [cree\_carro : (carro)$
$\quad carro \rhd [hablar : (m)\mathbf{local}\ base\ \mathbf{in}$
$\quad\quad central \lhd get\_base[base] | base \lhd com[m] |$
$\quad\quad industria\_carros \lhd cree\_carro[carro]]] |$
$base_i \rhd [com : (m)\ P(m)] |$
$\mathbf{local}\ c\ \mathbf{in}$
$\quad cellmaker \lhd create[c, 0] |$
$\quad \mathbf{clone}\ central \rhd [get\_base : (base)c \lhd get[base]\ \mathbf{then}\ c \lhd set[(base + i)\%2]]$

El objeto $industria\_carros$ se utiliza par crear procesos CARRO; los objetos $base_i, i = 1, 2$ procesan los mensajes enviados por cada CARRO; y el objeto $central$ decide cuál base debe recibir los mensajes. Note que $central$ simplemente intercambia la base que recibe el mensaje, tal como se hace en [Mil91].

En una situación más real, cada país contaría con un conjunto de centros de comunicación, contando cada centro con un conjunto de bases asociadas independientes. Cada una de estas bases puede procesar un mensaje solo si su distancia al CARRO es menor o igual a un valor dado. El papel de la base central es escoger alguna base de un centro de comunicación diferente al actual, cuando los mensajes de un carro no pueden ser trasmitidos por ninguna de las bases pertenecientes a éste. Para modelar este problema en $PiCO$ se utilizan los objetos "localizados" en restricciones:

$$
\begin{array}{l}
industria\_carros \triangleright [cree\_carro : (carro, pos) \\
\qquad\qquad carro \triangleright [hablar : (m) pos \triangleleft com[m] | carro \triangleleft cree\_carro[carro, pos] \& \\
\qquad\qquad\qquad get : (x)\textbf{tell}\ pos = x | industria\_carros \triangleleft cree\_carro[carro, pos] \& \\
\qquad\qquad\qquad set : (x) carro \triangleleft cree\_carro[carro, x]]] | \\
\phi_i(sender) \triangleright [com : (m)\ P(m)] | \\
\Psi(sender) \triangleright [com : (m)\ \textbf{local}\ base\ \textbf{in} \\
\qquad\qquad\qquad central \triangleleft get\_base(sender, base) | base \triangleleft com[m]] | \\
\textbf{clone}\ central\ \triangleright [get\_base : (pos, base) P(pos, base)]
\end{array}
$$

Donde, $\phi_i(p) = distancia\_a\_B_i(p) \leq d_i,\ i = 1, 2$ y $\Psi(p) = \neg\phi_1(p) \wedge \neg\phi_2(p)$.

Además del método de comunicación *hablar*, un CARRO cuanta con los métodos *get* y *set* para conocer o modificar su atributo posición. Las BASES han sido "localizadas" en las restricciones ϕ_i. Un objeto adicional "localizado" en Ψ se utiliza para llamar a la *central* cuando el CARRO no se puede comunicar con ninguna base de su centro actual.

4.4 Consideraciones

La integración de *Objetos* y *Restricciones* de la manera definida en $PiCO$ permite modelar más naturalmente situaciones en las que ciertos servicios (representados mediante objetos) deben ser provistos a colecciones de solicitantes (emisores de mensajes). Frecuentemente estas colecciones son definidas por comprensión (por medio de alguna propiedad o relación) y no por extensión (nombradas explíctamente); en estos casos la movilidad permitida por la "localización" en restricciones es un mecanismo de gran ayuda para el modelaje. Como se vió en el ejemplo, esta característica de $PiCO$ facilita la definición de automatismos en los procesos móviles (movilidad automática regida por la distancia).

Por otro lado, la integración de la orientación objeto en el cálculo facilita su utilización. Adicionalmente, el mecanismo de delegación permite realizar representaciones simples de múltiples esquemas de herencia.

5 Modelos de distribución

Otros autores han abordado el problema de la movilidad desde el punto de vista de la modelación de procesos distribuidos en diferentes *sitios*. Por ejemplo, en [R97] se encuentra la descripción de MCC, un cálculo concurrente por restricciones que integra la comunicación por agentes *ask* y *tell* y la comunicación asincrónica por paso de mensajes. En [VLSF99] se describe un modelo formal de distribución para objetos móviles llamado $DyTyCO$ como una extensión a $TyCO$, un cálculo de objetos concurrentes propuesto en [Vas94]. A continuación presentamos los aspectos más relevantes de estos cálculos en el contexto de la computación móvil.

5.1 *MCC*

La idea clave en *MCC* consiste en construir un modelo de procesamiento distribuido que, contando con la potencialidad de los lenguajes concurrentes con restricciones, integre (aunque parezca contradictorio en principio) las nociones de *cálculo local*, *almacén local* y *sitio* evitando la necesidad de recurrir a realizar resolución distribuida de restricciones.

En *MCC*, además de los agentes de comunicación *ask* y *tell* tomados del paradigma de programación concurrente con restricciones, se definen dos agentes primitvos adicionales de comunicación: *send* y *receive*. Los primeros son usados para la comunicación local en un sitio, mientras los segundos son usados para la comunicación entre sitios. Los agentes *ask* y *tell* funcionan igual que los agentes análogos mencionados en *PiCO*, pero localmente. Por su parte, los agentes *send* y *recieve* permiten comunicar restricciones de un sitio a otro; estas restricciones se envían de manera abstracta por medio del agente *send* y se reciben y se aplican a las variables determinadas por el agente *receive*.

5.2 *DyTyCO*

TyCO es un cálculo de objetos concurrentes definido en [Vas94], que cuenta con los procesos básicos definidos en el $\pi-$cálculo pero cuyo mecanismo de comunicación está basado en el paso de mensajes entre objetos y no en comunicación a través de canales. Por ello, además de los procesos básicos de composición, replicación y restricción de un nombre a un contexto, aparecen dos proceso básicos para crear objetos y para enviar mensajes. A diferencia de *PiCO*, en *TyCO* los objetos se localizan en nombres y los mensajes especifican con el nombre el objeto al cua'l se envían.

En *DyTyCO* aparece una nueva categoría léxica adicional a los nombres: los identificadores de *sitios*. Los nombres pueden estar o no prefijados por un sitio. En caso de estarlo, se denominan nombres remotos; sino, nombres locales. De igual manera los procesos ahora deben estar *localizados* en un identificador de sitio y pueden definirse plantillas de procesos. Un sitio es, entonces, un conjunto de procesos localizados y de plantillas de procesos, y una red es un conjunto de sitios ejecutándose concurrentemente, con la posibilidad de tener nombres locales. Las reglas de la congruencia estructural y de la relación de reducción de *TyCO* son extendidas para tener en cuenta la localización de los nombres y de los procesos. Según estas reglas, los objetos y los mensajes migran hacia el sitio donde están localizados sus nombres; por su lado,las plantillas de procesos son "bajadas" (downloaded) desde el sitio donde fueron definidas e instanciadas localmente.

6 Conclusiones

En este artículo se han presentado varios de los formalismos propuestos recientemente para modelar movilidad de procesos, cercanos todos, en mayor o menor grado, del π-cálculo. Mientras en *DyTyCO* y en el cálculo de ambientes la noción de seguridad en *sitios* de la red se trata mediante tipamiento de grupos de nombres, en *PiCO* se representaría mediante la restricción de localización de un objeto. Sin embargo, el requerimiento de un *almacén* global de restricciones en *PiCO* dificulta modelar fallas en *sitios*, lo que es, por el contrario, natural en *MCC*. En el caso de la red mundial, sin embargo, como lo sugiere Cardelli, la noción de *falla* es bien particular: Su efecto es solamente perceptible en el tiempo que tarda un sitio en garantizar accesibilidad.

Cabe preguntarse cuál es, realmente, la idiosincracia de cada modelo. Es decir, qué formalismo de formalismos permitiría precisar sus diferencias?. Un trabajo futuro especialmente interesante sería identificar para cada cálculo instancias de *estructuras de acción* ([Mil96]) que lo definan. De esta manera sus diferencias se apreciarían con precisión en los diferentes tipos de operaciones sobre estas estructuras de acción que cada uno requiera.

Otra cuestión que se plantea es la posibilidad de encontrar un cálculo unificado para la movilidad. Creemos

que en el estado actual de las investigaciones esto es prematuro y quizá inconveniente. La complejidad de las interacciones en internet hace que múltiples puntos de vista sobre ella sean no sólo posibles sino tal vez indispensables.

Referencias

[ADQ+98] G. Alvarez, J. F. Diaz, L. Quesada, F. Valencia, C. Rueda, and G. Assayag. Pico: A calculus of concurrent constraint objects for musical applications. In *Workshop on Constraints and the Arts, ECAI98*, Brighton, England, 1998.

[ADQ+00] G. Alvarez, J. F. Diaz, L. Quesada, F. Valencia, C. Rueda, G. Tamura, and G. Assayag. Integrating constraints and concurrent objects in musical applications: A calculus and its visual language. *Constraints*, 2000. To appear.

[AP97] R. Amadio and S Prasad. Modeling ip mobility. Technical Report 3301, INRIA, Sophia Antipolis, November 1997.

[Car97] L. Cardelli. Global computation. *Sigplan Notices*, 32(1):66–68, 1997.

[CG98] L. Cardelli and A. Gordon. Mobile ambients. *in Foundations of Software Science and Computational Structures, Maurice Nivat (Ed.), Lecture Notes in Computer Science*, 1378:140–155, 1998.

[CGG00] L. Cardelli, G. Ghelli, and A. Gordon. Secrecy and group creation. In *Proceedings of CONCUR'2000*, 2000. To appear.

[Mil91] R. Milner. The polyadic π-calculus: a tutorial. Technical Report ECS-LFCS-91-180, Laboratory for Foundations of Computer Science,, October 1991. Also in *Logic and Algebra of Specification*, ed. F. L. Bauer, W. Brauer and H. Schwichtenberg, Springer Verlag , 1993.

[Mil93] R. Milner. Interaction. *Communications of the ACM*, 36(1):79–89, 1993.

[Mil96] R. Milner. Calculi for interaction. *Acta Informatica*, 33(8):707–737, 1996.

[RÓ7] J.H. Réty. Un langage distribué concurrent avec contraintes. JFPLC-97, 1997.

[RMW92] J. Parrow R. Milner and D. Walker. A calculus of mobile processes, Parts I and II. *Journal of Information and Computation*, 100:1–77, September 1992.

[Sar93] V. A. Saraswat. *Concurrent Constraint Programming.* The MIT Press, Cambridge, MA, 1993.

[Smo94] G. Smolka. A calculus for higher-order concurrent constraint programming with deep guards. Research Report RR-94-03, Deutsches Forschungszentrum für Künstliche Intelligenz, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany, February 1994.

[Vas94] V. T. Vasconcelos. Typed concurrent objects. In M. Tokoro and R. Pareschi, editors, *Proc. of 8th European Conference on Object-Oriented Programming (ECOOP'94)*, volume 821 of *Lecture Notes in Computer Science*, pages 100–117. Springer-Verlag, 1994.

[VLSF99] V.T. Vasconcelos, L. Lopes, F. Silva, and A. Figueira. Dityco: an experiment in code mobility from the realm of process calculi, 1999.