Anteproyecto de grado

# Teoría de bisimilaridad para el Cálculo ntcc


Investigador
Luis Fernando Pino Duque

Director de Trabajo de Grado
Juan Francisco Díaz Frias, Ph.D.
Profesor titular
Escuela de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Valle

Co-Director
Frank D. Valencia, Ph.D.
Científico Investigador del CNRS en LIX
École Polytechnique de Paris
en el INRIA equipo COMÈTE

Santiago de Cali, 26 de agosto de 2009

# Índice

# 1. Introducción

En el mundo actual la tecnología constituye uno de los pilares más importantes para el desarrollo de la sociedad. Este protagonismo ha hecho que el acceso a los dispositivos de alta tecnología sea cada día mas frecuente. Elementos como la Web, las redes inalámbricas, los computadores portátiles de alta capacidad, los dispositivos móviles, entre otros, han permitido que se avance hacia la globalización de la información, pero trayendo consigo nuevos retos y problemas que se deben solucionar para asegurar un servicio confiable, correcto y seguro.

Las ciencias de la computación ofrecen un marco en el cual se puede dar una formalización de las tecnologías informáticas, permitiendo establecer las condiciones sobre las cuales funcionan de manera correcta. Existen diversos factores que afectan la correctitud de dichas tecnologías, uno de los más recientes y desafiantes es la concurrencia, esta consiste en la cantidad de usuarios y procesos que hacen uso de manera simultanea de un sistema. Es aquí donde un área de las ciencias de la computación denominada teoría de la concurrencia entra en acción.

En dicha teoría se resaltan los cálculos de procesos, cuya intención es modelar y razonar acerca de sistemas concurrentes. Tales cálculos tienen la capacidad de expresar de manera formal los sistemas, por tanto es posible razonar sobre ellos para obtener resultados correctos. Existen diversos ejemplos tales como CCS [1], $\pi$-cálculo [2], s$\pi$-cálculo ($\pi$-cálculo para razonar sobre seguridad), entre otros, los cuales se han especializado para resolver problemas específicos debido a que cada uno de ellos cuenta con un enfoque para el modelamiento y una expresividad asociada (cosas que se pueden modelar con el). El cálculo que será objeto de estudio se denomina cálculo `ntcc`, cuya base se encuentra en el uso de la lógica y las restricciones, junto con la expresión del comportamiento de los procesos en el tiempo.

Este proyecto busca desarrollar una teoría de bisimilaridad para el cálculo `ntcc`, la cual permitirá analizar el comportamiento de los procesos y facilitar la implementación de herramientas asociadas a este cálculo. También se busca dotar a `ntcc` con un prototipo inicial de verificación de procesos mediante la equivalencia de bisimilaridad, permitiendo abrir las puertas para el desarrollo de futuras aplicaciones que ayuden a automatizar actividades relacionadas con el uso del cálculo `ntcc`.

## 1.1. Planteamiento del Problema

La teoría de la concurrencia busca analizar aquellos sistemas en los que actúan diversos procesos de manera simultánea, y realizar algún tipo de razonamiento sobre los mismos para obtener conclusiones acerca de la correctitud, seguridad y otros aspectos importantes.

Para ello se utilizan los cálculos de procesos, los cuales permiten realizar el modelamiento de dichos procesos en sistemas concurrentes, y dependiendo de su especialización

[1] Calculus of Communicating Systems - Cálculo de Sistemas Comunicantes mas información en: http://es.wikipedia.org/wiki/Cálculo_de_sistemas_comunicantes

[2] Mas información en: http://en.wikipedia.org/wiki/Pi-calculus

son capaces de expresar, hasta cierto punto, una serie de acciones que pueden ser objeto de estudio para un posterior razonamiento.

Cálculos como el CCS y el $\pi$-cálculo han desarrollado una noción de equivalencia denominada bisimilaridad. Esta noción es muy fuerte, ya que permite razonar de manera sencilla acerca de la semejanza en el comportamiento de los procesos, con lo cual se pueden realizar todo tipo de labores con dicha noción tales como verificación, razonamiento, entre otros.

La programación por restricciones concurrente (CCP [3] ) es un formalismo con el que se pueden realizar razonamientos acerca de sistemas concurrentes utilizando las restricciones y la lógica. Este formalismo ha sido extendido y particularizado, con el fin de obtener la capacidad de modelar otros aspectos importantes. En especial el comportamiento asíncrono y no determinista, teniendo en cuenta el tiempo en el que se ejecutan los procesos. Esta extensión ha sido llamada `ntcc`.

Dicho cálculo se ha venido desarrollando durante la última década y tiene una base teórica muy robusta. Pero hay un problema, este cálculo carece de una equivalencia de comportamiento como la noción de bisimilaridad que es tan fuerte e importante en la teoría de la concurrencia. Es aquí donde este proyecto pretende aportar al cálculo `ntcc`, dotándolo con una teoría de bisimilaridad que permita determinar cuando los procesos se comportan equivalentemente según la definición desarrollada.

Después de haber definido dicha teoría, es necesario desarrollar técnicas de verificación, cuyo objetivo es encontrar una manera de determinar si dos procesos son bisimilares. Mediante el uso de estas técnicas será posible tener en cuenta otro problema importante, el cual consiste en la falta de herramientas prácticas sobre las cuales se pueda modelar y razonar acerca de las equivalencias entre procesos escritos en este cálculo. Por tanto, a través del uso de las técnicas desarrolladas será posible hacer una implementación que brindará la posibilidad de establecer si dos procesos son bisimilares o no.

Es por ello que otro de los objetivos es implementar un prototipo inicial para el calculo `ntcc` cuyo propósito es utilizar de manera practica la noción de bisimilaridad en el modelamiento de los sistemas concurrentes, este será muy especifico de un subconjunto del cálculo (debido a su complejidad) y permitirá abrir la brecha en el uso de la bisimilaridad para el mismo.

[3] Concurrent constraint programming - mas información en:
http://en.wikipedia.org/wiki/Concurrent_constraint_logic_programming

# 2. Objetivos

## 2.1. Objetivo General

Proponer una teoría de bisimilaridad para el cálculo `ntcc`.

## 2.2. Objetivos Específicos

- Definir el concepto de bisimilaridad para el cálculo `ntcc`.
- Desarrollar un conjunto de axiomas que conformarán la base de la teoría de bisimilaridad.
- Establecer las propiedades que se derivan de la teoría de bisimilaridad las cuales permitirán la caracterización de la misma.
- Definir las técnicas de verificación que brindarán la posibilidad de determinar la bisimilaridad entre los procesos escritos en el cálculo `ntcc`.
- Implementar un prototipo que permita escribir procesos en el cálculo `ntcc` y verificar si estos son bisimilares.

# 3. Justificación

La teoría de bisimilaridad es una de las equivalencias más representativas e importantes en la teoría de la concurrencia. Como se ha descrito en secciones anteriores, los cálculos mas importantes hacen uso de esta noción para diversos tipos de aplicaciones (verificación formal, simulación, entre otros).

Por su parte, el cálculo `ntcc` carece de una equivalencia de comportamiento como bisimilaridad, por lo que no es posible realizar aplicaciones que la requieran. Es por ello que la importancia de este proyecto radica en el desarrollo de una teoría de bisimilaridad para el calculo `ntcc`, la cual será novedosa y traerá consigo nuevas formas de aplicar el cálculo a problemas reales.

Además, como complemento al desarrollo teórico se busca abrir la brecha en la verificación de procesos con el cálculo `ntcc`. Esto se verá reflejado en la implementación del prototipo inicial que utilizará la noción de bisimilaridad.

Se ha escogido el cálculo `ntcc` debido a su importancia en el grupo AVISPA, ya que este ha decido robustecer este cálculo a través de su proyecto REACT [4], el cual esta en la etapa de conclusión y tendrá su continuación en el proyecto REACT+. La idea principal es fortalecer el cálculo `ntcc` para aplicarlo a problemas de la vida real. Por tanto, la teoría de bisimilaridad contribuirá en la consolidación de dicho cálculo, brindando nuevas posibilidades de aplicación.

[4] Robust theories for Emerging Applications in Concurrency Theory

En conclusión, mediante la realización de este proyecto se reforzará al cálculo `ntcc` con una noción de bisimilaridad, y con un prototipo inicial que hace uso esta noción de manera práctica.

# 4. Marco de Referencia

## 4.1. Marco Teórico

### 4.1.1. Cálculo de procesos [15]

Los cálculos de procesos son una amplia familia de aproximaciones relacionadas al modelamiento formal de los sistemas concurrentes. Estos proveen una herramienta para la descripción de interacciones, comunicaciones y sincronizaciones de alto orden entre un conjunto de agentes (procesos) independientes.

Existen diferentes cálculos de procesos en la literatura principalmente de acuerdo en su énfasis sobre el álgebra. Los mas representativos con CCS [4], CSP [5] y el álgebra de procesos ACP [6, 7]. Las diferencias entre estos cálculos surgen en aspectos como la construcción de los procesos (lenguaje de procesos), los métodos usados para dar significado a los términos de los procesos (la semántica de los procesos), y los métodos para razonar acerca del comportamiento de los procesos (por ejemplo, equivalencia de procesos o lógica de procesos). Otros aspectos considerados en la teoría de estos cálculos son su expresividad, y el análisis de sus equivalencias de comportamiento. Se describirán entonces algunas de las cuestiones nombradas anteriormente.

**El lenguaje de procesos.** Una característica común de los lenguajes de procesos es que le prestan mucha atención a la economía. Esto significa que hay pocos operadores o combinadores (combinators), cada uno con un papel diferente y fundamental. Los cálculos de procesos usualmente proveen los siguientes combinadores:

- *Acción - Action*, para representar la ocurrencia de acciones atómicas.
- *Producto - Product*, para expresar composición paralela.
- *Suma - Summation*, para expresar un curso alternado en la computación.
- *Restricción - Restriction* (o *Ocultamiento - Hiding*), para delimitar la interacción entra los procesos.
- *Recursión - Recursion*, para expresar comportamiento infinito.

**Ejemplo.** Sea $\mathcal{N}$ un conjunto de *nombres - names* $a, b, ...$, y sea $\overline{\mathcal{N}}$ su respectivo conjunto de *co-nombres - co-names* $\overline{\mathcal{N}} = \{\overline{a} | a \in \mathcal{N}\}$ disyunto de $\mathcal{N}$. El conjunto de *etiquetas - labels*, que contienen los $a$ y $\overline{a}$, es $\mathcal{L} = \mathcal{N} \cup \overline{\mathcal{N}}$. El conjunto de *acciones - actions Act*, que contiene los $\mathbf{a}$ y $\mathbf{b}$ (en negrita) extendiendo a $\mathcal{L}$ con un nuevo símbolo $\tau$. La acción $\tau$ se dice que es *silenciosa - silent* (interna o no observable). Las acciones $a$ y $\overline{a}$ son concebidas como complementarias, entonces $\overline{\overline{a}} = a$. La sintaxis de los procesos esta dada por:

$$P, Q, ... ::= 0 \mid \mathbf{a}.P \mid P + Q \mid P \mid\mid Q \mid P \backslash a \mid A \langle b_1, ..., b_n \rangle$$

**Descripción intuitiva.** El significado intuitivo de los términos de este lenguaje de procesos es el siguiente. El proceso 0 no hace nada. El proceso $\mathbf{a}.P$ realiza una acción atómica $\mathbf{a}$ y luego se comporta como $P$. La suma $P+Q$ es un proceso que se puede comportar como $P$ o como $Q$. $P \mid\mid Q$ representa la composición paralela de $P$ y $Q$. Ambos $P$ y $Q$ pueden actuar independientemente pero ellos también se pueden sincronizar si realizan acciones complementarias. La restricción $P\backslash a$ se comporta como $P$ excepto que no puede realizar las acciones $a$ o $\overline{a}$. Los nombres $a$ y $\overline{a}$ se dice que están *ligados - bound* en $P\backslash a$. $A\langle b_1, ..., b_n\rangle$ denota el llamado a una definición única recursiva de la forma $A(a_1, ..., a_n) \stackrel{def}{=} P_A$ donde todos los nombres no ligados del proceso $P_A$ están en $\{a_1, ..., a_n\}$. Obviamente $P_A$ puede contener llamados a $A$. El proceso $A\langle b_1, ..., b_n\rangle$ se comporta como $P_A[b_1, ..., b_n/a_1, ..., a_n]$, es decir $P_A$ con cada $a_i$ reemplazado por $b_i$, renombrando las variables ligadas cuando sea necesario para evitar ambigüedades.

**Semántica de Procesos.** Los métodos mediante los cuales se dota a los procesos con un significado pueden involucrar al menos tres tipos de aproximaciones: *semántica Operacional, Denotacional y Algebraica*. Tradicionalmente, CCS y CSP enfatizan el uso del método operacional y denotacional, respectivamente, mientras que el énfasis de ACP es sobre el método algebraico. Para este trabajo es importante describir la semántica operacional y las equivalencias de comportamiento, mas específicamente la bisimilaridad.

**Semántica Operacional.** Una semántica operacional interpreta un termino que representa un proceso al usar transiciones (etiquetadas o no) especificando sus pasos computacionales. Una transición etiquetada $P \xrightarrow{a} Q$ especifica que $P$ realiza la acción $\mathbf{a}$ y entonces se comporta como $Q$. Las relaciones $\xrightarrow{a}$ están definidas para ser las mas pequeñas las cuales cumples las reglas de la Tabla 1. En estas reglas la transición debajo de la línea se infiere a partir de aquellas que están arriba.

### 4.1.2. Bisimilaridad [16]

Ya que se ha definido el concepto de semántica operacional, se puede introducir entonces las nociones típicas de equivalencia. De especial importancia para este trabajo la noción de bisimilaridad asociada con el cálculo CCS.
Se requiere una pequeña notación para introducir el concepto: La secuencia vacía es denotada por $\epsilon$. Dada una secuencia de acciones $s = \mathbf{a}_1.\mathbf{a}_2.... \in Act^*$, se define $\xrightarrow{s}$ como

$$(\xrightarrow{\tau})^* \xrightarrow{a_1} (\xrightarrow{\tau})^*...(\xrightarrow{\tau})^* \xrightarrow{a_n} (\xrightarrow{\tau})^*$$

Note que $\stackrel{\epsilon}{\Longrightarrow} = \xrightarrow{\tau}{}^*$. Se usa $P \stackrel{s}{\Longrightarrow}$ para decir que existe un $P'$, tal que $P \stackrel{s}{\Longrightarrow} P'$ y similarmente para $P \xrightarrow{s}$.

**Bisimilaridad Fuerte (Strong Bisimilarity),** Intuitivamente, $P$ y $Q$ son fuertemente bisimilares si siempre que $P$ realice una acción $\mathbf{a}$ y evolucione a $P'$ entonces $Q$ puede también realizar $\mathbf{a}$ y transformarse en un $Q'$ que es fuertemente bisimilar a $P'$, además debe suceder similarmente si $P$ y $Q$ se intercambian.

La intuición arriba nombrada puede ser formalizada de la siguiente manera. Una relación simétrica $B$ entre procesos se dice que es una bisimulación fuerte si y solo si para todo $(P, Q) \in B$,

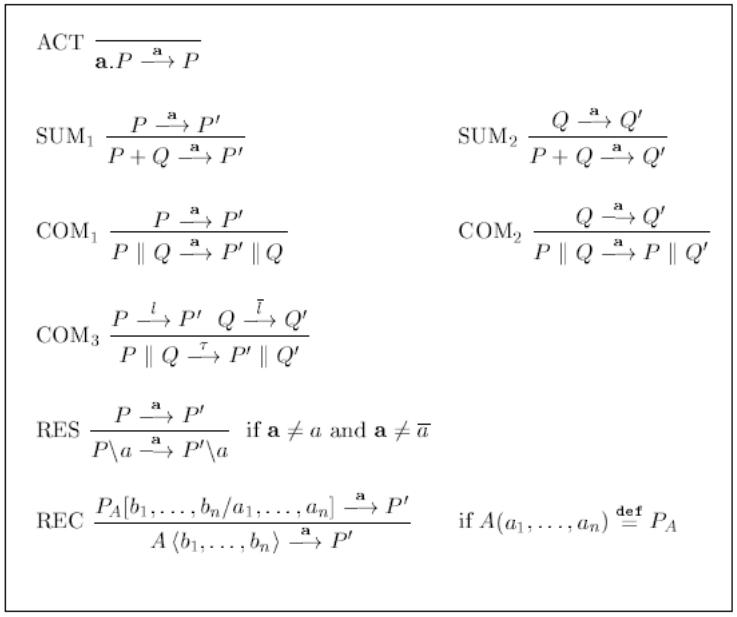


Figura 1: Un ejemplo de semántica operacional para un calculo de procesos como CCS (tomado de [18], Pág. 35)

$$\text{si } P \xrightarrow{a} P' \text{ entonces para algún } Q', Q \xrightarrow{a} Q' \text{ y } (P', Q') \in B$$

Se dice que $P$ es fuertemente bisimilar a $Q$, escrito $P \sim Q$ si y solo si existe una bisimulación fuerte que contenga la pareja $(P, Q)$.

**Bisimilaridad Débil (Weak Bisimilarity)** Esta equivalencia es similar a la anterior pero intenta no tener en cuenta las acciones silenciosas en el momento que se comparan los procesos. Esta bisimilaridad puede ser obtenida al reemplazar las transiciones $\xrightarrow{a}$ de arriba con las (secuencias de acciones observables) transiciones $\stackrel{s}{\Longrightarrow}$ donde $s \in \mathcal{L}^*$. Se utiliza $\approx$ para denotar bisimilaridad débil. Note que $P \not\sim \tau.P$ pero $P \approx \tau.P$.

### 4.1.3. CCP - Programación por restricciones concurrente [16]

En su tesis doctoral [1], Saraswat propuso la programación por restricciones concurrente (CCP, por su sigla en ingles) como un modelo de concurrencia basado en la comunicación a través de variables compartidas y algunas ideas sencillas tomadas de la lógica. Descrito informalmente, el modelo ccp combina elegantemente conceptos de la lógica y mecanismos de concurrencia.

**El modelo ccp**: Un sistema concurrente se especifica mediante el modelo ccp en términos de *restricciones* sobre las variables del sistema. Una restricción es una formula en lógica de primer orden que representa *información parcial* acerca de los valores de las variables. Por ejemplo, para un sistema con variables $x$ y $y$ cuyos dominio son los números naturales,

la restricción $x + y > 16$ especifica valores posibles para $x$ y $y$ (aquellos que satisfacen la inecuación). El modelo ccp esta parametrizado con un *sistema de restricciones*, el cual especifica las restricciones pertinentes para el tipo de sistema en consideración, y una relación de derivación (entailment) entre las restricciones (por ejemplo, $x+y > 16 \models x+y > 0$).

Durante una computación en ccp, el estado del sistema es especificado por una entidad llamada el *almacén*(*store*) en donde reside la información acerca de las variables del sistema. El *almacén* es representado como una restricción, y así se puede proveer solo información parcial acerca de las variables. Conceptualmente, el *almacén* en ccp es el *medio*(*medium*) por el cual los agentes interactúan entre si.

Un proceso en ccp puede actualizar el estado del sistema solo añadiendo (o *diciendo - telling*) información al almacén. Esto es representado como la conjunción lógica del almacén, el cual representa el estado previo, y la restricción que esta siendo añadida. Por tanto, al actualizar no se cambian los valores de las variables como tal, sino que restringe algunos de los valores previamente posibles.

Además, los procesos en ccp pueden sincronizarse al consultar (o *preguntar - asking*) información del almacén. Las preguntas están bloqueadas hasta que haya suficiente información en el almacén para *derivar (entail)* (responder afirmativamente) la consulta, es decir, la operación de pregunta determina si la restricción que representa el almacén deriva la consulta.

Una computación en ccp termina cuando llegue a un punto, llamado *de descanso - resting* o *inactivo - quiescent*, en el cual no se puede agregar mas información al almacén. La salida de la computación esta definida como el punto final del almacén, también llamado *almacén inactivo - quiescent store*.

**Ejemplo.** (tomado de [16]) Para hacer la descripción del modelo ccp mas clara, considere el escenario ilustrado en la figura 2. Se tienen cuatro agentes (o procesos) que desean interactuar a través del medio (inicialmente vacío). Nombrándolos, empezando desde el proceso en la esquina izquierda-arriba y siguiendo en el sentido de las manecillas del reloj, $A_1$, $A_2$, $A_3$ y $A_4$ respectivamente. Suponga que se han programado para ejecutar en el mismo orden en el que fueron nombrados.

De esta manera $A_1$ actúa primero y le dice a los demás a través del medio que la temperatura es mayor a 42 grados pero sin especificar un valor exacto. En otras palabras $A_1$ da a los otros información parcial acerca de la temperatura. Esto hace que se añada la restricción "temperatura $> 42$" al almacén previamente vacío.

Ahora $A_2$ pregunta si la temperatura es exactamente 50 grados, y si es así el desea ejecutar un proceso $P$. Sin embargo, de la información actual en el almacén no se puede determinar cual es el valor exacto de la temperatura. El agente $A_2$ queda entonces bloqueado así como el agente $A_3$, ya que tampoco se puede deducir si la temperatura esta entre 0 y 100 grados.

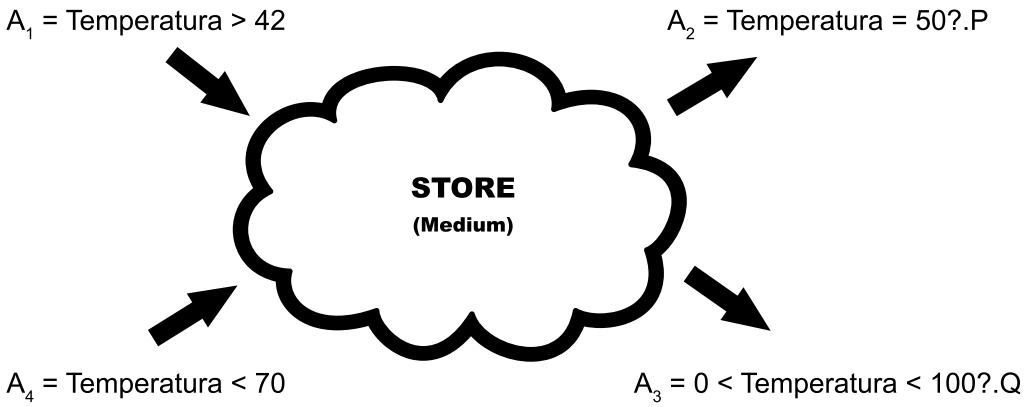


Figura 2: Un escenario simple en ccp

El turno es ahora para $A_4$, el cual dice que la temperatura es menor que 70 grados. El almacén se convierte en "temperatura $> 42 \wedge$ temperatura $< 70$". Ahora el proceso $A_3$ puede ejecutar $Q$ ya que su consulta puede ser deducida por medio de la información en el almacén. El otro agente que pregunta $A_2$ esta condenado a estar bloqueado por siempre, a menos que $Q$ añada suficiente información al almacén de tal forma que la consulta pueda ser respondida.

**El Lenguaje de Procesos CCP.** En el ámbito de los cálculos de procesos, el lenguaje de procesos en el modelo ccp esta dado por un pequeño numero de operadores sencillos. En vez de dar la sintaxis concreta del lenguaje, se brindará la descripción de la intuición básica que cada constructor trae consigo. Entonces, en ccp se tienen:

- La *acción decir - tell*, que añade restricciones al almacén (como $A_1$ en el ejemplo anterior).
- La *acción preguntar - ask (o acción prefija - prefix)*, para expresar una pregunta que le precede la continuación de otro proceso (como $A_2$).
- *Composición Paralela*, la cual combina procesos concurrentemente. Por ejemplo, el escenario anterior donde $A_1$, $A_2$, $A_3$ y $A_4$ corrían concurrentemente.
- *Ocultamiento - Hiding (o localidad - locality)*, para expresar variables locales que delimitan la interfaz sobre la que los procesos interactúan con otros.

- *Sumatoria - Summation*, que expresa una combinación disyunta de agentes para permitir acciones alternadas.
- *Recursión - Recursion*, para definir comportamiento infinito.

Vale la pena decir que sin la sumatoria, el modelo ccp es determinista en el sentido que el almacén final es el mismo, independientemente del orden de ejecución de los componentes paralelos [2].

**Sintaxis de CCP**, *Los procesos P,Q,... en CCP son construidos a partir de restricciones en el sistema de restricciones subyacente y cumpliendo la siguiente sintaxis*:

$$P, Q := \textbf{tell}(c) \mid \textbf{when } c \textbf{ do } P \mid P \mid\mid Q \mid (\textbf{local } x)P \mid q(x)$$

El proceso **tell**$(c)$ añade la restricción $c$ al almacén. El proceso **when** $c$ **do** $P$ pregunta si $c$ puede ser deducida del almacén. Si se puede, se comporta como $P$. En caso contrario, espera hasta que el almacén contenga la suficiente información para deducir $c$. La composición paralela de $P$ y $Q$ se representa con $P \mid\mid Q$. El proceso (**local** $x$)$P$ se comporta como $P$, excepto que toda la información en $x$ producida por $P$ puede ser vista únicamente por $P$ y la información en $x$ producida por otros procesos no puede ser vista por $P$. El proceso $q(y)$ es un *identificador* con aridad $|y|$. Se asume que cada identificador tiene una definición (recursiva) única, de la forma

$$q(x) \stackrel{def}{=} Q$$

con un $x$ diferente y $|x| = |y|$. El proceso $q(y)$ se comporta entonces como $Q[y/x]$.

### 4.1.4. Cálculos basados en CCP y el Cálculo ntcc [17]

En la literatura se han estudiado diferentes extensiones de los constructores básicos presentados anteriormente con el fin de proveer ajustes para la programación y especificación de sistemas con el enfoque declarativo de la programación por restricciones concurrente. Para este proyecto es importante describir el cálculo `ntcc`, el cual es una extensión al modelo tcc, incluyendo comportamiento asíncrono y no determinista.

**Temporal CCP**(`tcc`) El modelo `tcc` considera la computación reactiva como el proceder *deterministicamente* en unidades de tiempo discretas (o intervalos de tiempo). En otras palabras, el tiempo es conceptualmente dividido en intervalos discretos. En cada uno de ellos, un proceso deterministico en CCP recibe un estimulo (es decir, una restricción) del ambiente, se ejecuta con este estimulo como el *almacén inicial* y cuando este llega al punto inactivo, se responde al ambiente con el almacén final. Además, el punto inactivo determina un proceso residual, que será ejecutado en el siguiente intervalo de tiempo. El calculo `tcc` introduce constructores para (1) *retrasar - delay* la ejecución de un proceso. Y para dar un (2) *tiempo muerto - time-out* a las operaciones, que esperan durante el intervalo de tiempo actual para encontrar presente una fracción de información Si no lo esta, se dispara un proceso en el *siguiente intervalo de tiempo.*

**Sintaxis de `tcc` deterministico** Los procesos $P, Q, ...$ en CCP son construidos a partir de restricciones en el sistema de restricciones subyacente y cumpliendo la siguiente sintaxis:

$$P, Q := \mathbf{skip} \mid \mathbf{tell}(c) \mid \mathbf{when}\ c\ \mathbf{do}\ P \mid P \mid\mid Q \mid (\mathbf{local}\ x)P \mid \mathbf{next}P \mid \mathbf{unless}\ c\ \mathbf{next}\ P \mid\ !P$$

Los procesos $\mathbf{tell}(c)$, $\mathbf{when}\ c\ \mathbf{do}\ P$, $P \mid\mid Q$ y $(\mathbf{local}\ x)P$ son similares a aquellos en CCP. El proceso $\mathbf{next}P$ retrasa la ejecución de $P$ al siguiente intervalo de tiempo. El *tiempo muerto* $\mathbf{unless}\ c\ \mathbf{next}\ P$ es también un retraso, pero $P$ es ejecutado en la siguiente unidad de tiempo si y solo si $c$ no puede ser deducida por el almacén final en la fracción de tiempo actual. Finalmente, la *replicación* $!P$ significa $P \mid\mid \mathbf{next}P \mid\mid \mathbf{next}^2P \mid\mid ...$, es decir, libera una copia de $P$ por cada unidad de tiempo.

**Cálculo `ntcc`**. Este cálculo extiende a `tcc` para poder expresar comportamiento no determinista y asincronía.

**Sintaxis de `ntcc`**. Los procesos en `ntcc` resultan de añadir a la sintaxis de `tcc` los siguientes constructores:

$$\sum_{i \in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i \mid\ \star P$$

La elección mediante guardas $\sum_{i \in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i$ donde $I$ es un conjunto finito de índices, representa un proceso que, en el intervalo de tiempo actual, debe escoger no deterministicamente uno de los $P_j$ $(j \in I)$ cuya correspondiente guarda (restricción) $c_j$ es deducida por medio del almacén. La alternativa escogida, si la hay, hace que el resto no sean tenidas en cuenta. Si no se puede escoger ninguna opción entonces la sumatoria permanece bloqueada hasta que se agregue mas información al almacén. El operador "$\star$" permite expresar comportamiento asíncrono a través de los intervalos de tiempo. Intuitivamente, el proceso $\star P$ representa $P + \mathbf{next}P + \mathbf{next}^2P + ...$, es decir, un retraso de tiempo arbitrario pero finito para la activación del proceso $P$.

### 4.1.5. Estado del Arte

Este proyecto busca desarrollar una teoría de bisimilaridad para el calculo `ntcc`, por tanto es importante resaltar que otras teorías de bisimilaridad se han desarrollado para los cálculos de procesos mas representativos, como CCS y $\pi$-cálculo. Además, que herramientas se han desarrollado que hagan uso de dichas teorías. Se presentan entonces los aspectos arriba nombrados.

1. **Antecedentes Científicos**

   - **Bisimilaridad en CCS.[18]** Como se había descrito anteriormente (pero con mas detalle) se puede definir bisimilaridad de la siguiente manera:

     **Definición.** *Una relación simétrica $B$ entre procesos se dice que es una bisimulación fuerte si y solo si para todo $(P, Q) \in B$,*

$$\text{si } P \xrightarrow{a} P' \text{ entonces para algún } Q', Q \xrightarrow{a} Q' \text{ y } (P', Q') \in B$$

Se dice que $P$ es fuertemente bisimilar a $Q$, escrito $P \sim Q$ si y solo si existe una bisimulación fuerte que contenga la pareja $(P, Q)$.

- **Bisimilaridad en $\pi$-calculo.[18]** Para el $\pi$-cálculo la definición de bisimilaridad es análoga, la diferencia radica en que aquí los procesos se comportan de manera diferente a CCS, ya que tienen otra seria de posibles acciones lo que hace que aplicar la bisimilaridad sea un poco mas laborioso que en CCS.

  **Definición.** Una relación binaria $\mathcal{S}$ sobre los procesos definidos en $\pi$-calculo es una simulación fuerte,

  $$\text{si } P \xrightarrow{\alpha} A \text{ entonces existe un } B \text{ tal que } A\mathcal{S}B \text{ y } Q \xrightarrow{\alpha} B$$

  Si $\mathcal{S}$ y su inversa son simulaciones fuertes entonces $\mathcal{S}$ es una bisimulación fuerte. Dos agentes $A$ y $B$ son bisimilares o fuertemente equivalentes, escrito $A \sim B$, si la pareja $(A, B)$ esta en alguna bisimulación fuerte.

2. **Antecedentes Tecnológicos**

   - **The Edinburgh Concurrency Workbench (CWB)**[5] Este workbench es una herramienta automatizada que sirve para la manipulación y el análisis de sistemas concurrentes. En particular, el CWB permite utilizar diferentes semánticas de procesos para comparar a través de diferentes equivalencias y verificar algún tipo de modelo.

     Por ejemplo, con CWB es posible:

     - Definir comportamientos en CCS, y realizar análisis de comportamiento variados.
     - Definir proposiciones en una lógica modal potente y verificar si un proceso satisface una especificación formulada en la lógica.
     - Derivar automáticamente formulas lógicas que distinguen procesos no equivalentes.
   - **The Mobility Workbench (MWB)** [6] MWB es una herramienta similar a la anterior, pero su aplicación tiene como foco principal $\pi$-calculo en vez de CCS. De manera análoga este workbench es utilizado para modelar sistemas concurrentes y permite razonar acerca equivalencias, comportamientos, entre otras funcionalidades.

## 4.2. Marco Conceptual

### 4.2.1. Proceso [8]

Un proceso es una instancia de un programa de computador que esta siendo ejecutado por un sistema de cómputo el cual tiene la habilidad de correr diferentes programas concurrentemente.

[5] http://homepages.inf.ed.ac.uk/perdita/cwb/summary.html
[6] http://www.it.uu.se/research/group/mobility/mwb

#### 4.2.2. Cálculo [9]

Es un sistema de símbolos no interpretados, es decir, sin significación alguna, en el que se establecen mediante reglas estrictas, las relaciones sintácticas entre los símbolos para la construcción de expresiones bien formadas (EBF), así como las reglas que permiten transformar dichas expresiones en otras equivalentes; entendiendo por equivalentes que ambas tienen siempre y de forma necesaria el mismo valor de verdad.

#### 4.2.3. Concurrencia [10]

Es la propiedad de los sistemas que permiten que múltiples procesos sean ejecutados al mismo tiempo, y que potencialmente puedan interactuar entre sí.

#### 4.2.4. Axioma [11]

Es una "verdad evidente" que no requiere demostración, pues se justifica a sí misma, y sobre la cual se construye el resto de conocimientos por medio de la deducción;

#### 4.2.5. Teorema [12]

Es una afirmación que puede ser demostrada como verdadera dentro de un marco lógico.

#### 4.2.6. Restricción [13]

Es una relación entre una o varias variables en la cual se establece una condición.

#### 4.2.7. Semántica [14]

Se refiere a los aspectos del significado, sentido o interpretación del significado de un determinado elemento, símbolo, palabra, lenguaje o representación formal.

### 4.3. Marco Contextual

Actualmente esta en curso un proyecto de investigación denominado REACT (*Robust theories for Emerging Applications in Concurrency Theory*) el cual es un esfuerzo conjunto entre AVISPA [7] (grupo de investigación de la Universidad del Valle en convenio con la Universidad Javeriana Cali *-del cual el investigador es miembro-*), el equipo de representación musical en el IRCAM [8] (Institut de Recherche et Coordination Acoustique/Musique) y el equipo comète en el LIX [9] (Laboratoire d'Informatique de l'Ecole Polytechnique en Paris, Francia), este proyecto busca aplicar la teoría de la concurrencia a problemas específicos de la vida real. Mas específicamente, los cálculos de procesos (como `ntcc`) aplicados en áreas como protocolos de seguridad, sistemas biológicos e interacción multimedia.

Como se puede observar la aplicación de los cálculos de procesos como `ntcc` es importante en diferentes campos. Este proyecto será de gran ayuda para el desarrollo e

[7] http://cic.puj.edu.co/wiki/doku.php?id=grupos:avispa:avispa
[8] http://www.ircam.fr/
[9] http://www.lix.polytechnique.fr/

implementación de nuevas herramientas que se basan en dichos cálculos, puesto que esta teoría de equivalencia puede servir de apoyo para la determinación de similitudes entre los procesos de una manera diferente a las ya existentes en el calculo `ntcc`.

De esta manera se obtiene como principal ventaja una teoría que permitirá realizar la verificación automática de procesos, lo cual es ampliamente usado en las áreas de aplicación anteriormente nombradas. Además, proveer a `ntcc` con un prototipo inicial para el modelamiento de procesos utilizando la noción de bisimilaridad.

Una ventaja adicional de este proyecto es afianzar las relaciones entre AVISPA y el Ecole Polytechnique en Paris, por medio del profesor Frank Valencia (Co-Director del proyecto), ya que este proyecto nutre los intereses que se han edificado en conjunto.

# 5. Requerimientos del Proyecto

Básicamente se requiere un estudiante de pregrado dedicado a tiempo completo en el proyecto, además es necesario un director de proyecto que guié el desarrollo con el fin de realizar las actividades encaminadas a lograr los objetivos. Un experto en el área que para este caso es co-director del proyecto, con el cual se pueda tener comunicación constante para de esta manera tener la información oportunamente.

En cuanto a recursos tecnológicos, es necesario contar con un equipo de cómputo con capacidades y herramientas suficientes para trabajar cómodamente, además este equipo debe tener acceso a Internet para obtener la información requerida para el desarrollo del proyecto.

Se ilustra un presupuesto básico en la figura:

| **Item** | **Costo** |
|---|---|
| Investigador | 5'000.000 |
| Equipos de Computo | 2'000.000 |
| Libros, revistas, fotocopias | 400.000 |
| Gastos de Alimentación y Transporte | 1'000.000 |
| Gastos no previstos | 250.000 |
| Total | 8'650.000 |

# 6. Metodología

La realización de este proyecto se resume en un conjunto de cinco actividades globales, que encapsulan el desarrollo de cada uno de los objetivos a cumplir. Dichas actividades se enumeran de la siguiente manera:

- **Revisión Bibliográfica** En esta etapa se debe realizar una revisión de la bibliografía concerniente al proyecto, es decir, el propósito es profundizar en conceptos como: bismilaridad para CCS y $\pi$-cálculo, un amplio conocimiento del cálculo `ntcc`

y técnicas de verificación. Para ello se realizar consultas en artículos, libros, Internet y entrevistas con expertos en el tema. Como resultado se debe obtener una base teórica robusta que permitirá el desarrollo del proyecto.

- **Definición de la noción de bisimilaridad para el cálculo `ntcc`** Para esta fase se dedicará la mayor parte del tiempo en la definición formal del concepto de bisimilaridad, básicamente se hará una revisión formal del concepto y una definición de los requisitos que debe cumplir la noción, para que posteriormente se desarrolle dicha definición de forma clara y concreta. Aquí se obtendrá como resultado la definición formal de bisimilaridad.

- **Propiedades de bisimilaridad** En esta etapa se realizará la definición de las propiedades que cumple la noción antes desarrollada. Para ello se tomará la base formal y se realizarán demostraciones que permitan obtener las propiedades esenciales que cumple dicha noción. El resultado será una serie de propiedades de la bisimilaridad.

- **Técnicas de Verificación y Prototipo** Para esta etapa se definirán las técnicas que se utilizaran para realizar verificación sobre procesos definidos mediante el cálculo `ntcc`. El propósito de esta fase es obtener una serie de técnicas que se usen en la implementación del prototipo, ya que se pretende que este sea capaz de utilizar la noción de bisimilaridad al comprobar si dos procesos en `ntcc` son bisimilares o no. Con dichas técnicas, se procederá a la implementación del prototipo, cuya función será (como se nombró anteriormente) la de realizar verificación bajo el concepto de bisimilaridad. Como resultado se obtendrá un prototipo que haga uso de la noción desarrollada.

- **Pasantia e Informe Final** En la etapa final se realizaran los últimos ajustes a la teoría, para ello el investigador realizará una pasantia en el École Polytechnique (Francia), en donde se revisará a fondo la teoría y la implementación desarrolladas. Una vez realizada esta labor, se procederá a la realización del informe final del proyecto para evaluar los resultados obtenidos y las contribuciones realizadas.

| **Etapa** | **Tiempo Estimado** |
|---|---:|
| Revisión Bibliográfica | 3 Meses |
| Definición de bisimilaridad | 1 Mes |
| Propiedades de bisimilaridad | 3 Meses |
| Técnicas de Verificación y Prototipo | 3 Meses |
| Pasantia e Informe Final | 2 Meses |
| Total Estimado | 12 Meses |

# Referencias


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

[2] V. Saraswat, M. Rinard, y P. Panangaden. *The semantic foundations of concurrent constraint programming*. In *POPL '91*, pages 333-352, 21-23 January 1991.

[3] R. Milner. *Operational and Algebraic Semantics of Concurrent Processes*, pages 1203-1241. Elsevier, 1990.

[4] R. Milner. *Communication and Concurrency. International Series in Computer Science*. Prentice Hall, 1989. SU Fisher Research 511/24.

[5] C. A. R. Hoare. *Communications Sequential Processes*. Prentice-Hall, Englewood Cliffs (NJ), USA, 1985.

[6] J.A. Bergstra and J.W. Klop. *Algebra of communicating processes with abstraction*. Theoretical Computer Science, 37(1):77-121, 1985.

[7] J. C. M. Baeten and W. P. Weijland. *Process Algebra*. Cambridge University Press, 1990.

[8] Colaboradores de Wikipedia. *Proceso (informática)* [en línea]. Wikipedia, La enciclopedia libre, 2009 [fecha de consulta: 13 de abril del 2009]. http://en.wikipedia.org/wiki/Process_(computing).

[9] Colaboradores de Wikipedia. *Cálculo* [en línea]. Wikipedia, La enciclopedia libre, 2009 [fecha de consulta: 13 de Abril del 2009]. http://es.wikipedia.org/wiki/Cálculo.

[10] Alegsa, diccionario informático. *Definición de concurrencia*. 2009. [fecha de consulta: 13 de Abril del 2009]. http://www.alegsa.com.ar/Dic/concurrencia.php.

[11] Colaboradores de Wikipedia. *Axioma* [en línea]. Wikipedia, La enciclopedia libre, 2009 [fecha de consulta: 13 de Abril del 2009]. http://es.wikipedia.org/wiki/Axioma.

[12] Colaboradores de Wikipedia. *Teorema* [en línea]. Wikipedia, La enciclopedia libre, 2009 [fecha de consulta: 13 de Abril del 2009]. http://es.wikipedia.org/wiki/Teorema.

[13] Colaboradores de Wikipedia. *Constraint* [en línea]. Wikipedia, La enciclopedia libre, 2009 [fecha de consulta: 13 de Abril del 2009]. http://en.wikipedia.org/wiki/Constraint_(mathematics).

[14] Colaboradores de Wikipedia. *Semántica* [en línea]. Wikipedia, La enciclopedia libre, 2009 [fecha de consulta: 13 de Abril del 2009]. http://es.wikipedia.org/wiki/Semántica.

[15] Catuscia Palamidessi y Frank D. Valencia. *Languages for concurrency*. EATCS. 2006.

[16] Frank Valencia. *Decidability of Infinite-State Timed CCP Processes and First-Order LTL*. Theor. Comput. Sci. 330(3): 577-607. Elsevier. 2005

[17] Carlos Olarte, Camilo Rueda y Frank Valencia. *Concurrent Constraint Programming: Calculi, Languages and Emerging Applications*.

[18] R. Milner. *Communicating and Mobile Systems: the $\pi$-calculus*. Cambridge University Press, 1999.