Semántica de Sistemas Reactivos. Un preorden entre procesos basado en la ST-bisimulación.

Juan V. Echagüe[1]
echague@fing.edu.uy

Gabriel Fialco[1,2]
gfialco@fing.edu.uy

[1] InCo
Fac. de Ingeniería
U. de la República
Montevideo, Uruguay

[2] ESLAI
Universidad de Luján
Buenos Aires, Argentina

Resumen

En este trabajo definimos un preorden $\sqsubseteq_{ST}$ entre programas inspirado en la ST-bisimulación, donde $G_1 \sqsubseteq_{ST} G_2$ puede interpretarse naturalmente como "G_2 es una implementación paralela de G_1" sobre un modelo semántico "truly concurrent" de sistemas reactivos los Sistemas de Transiciones Asíncronos.

Este preorden es decidible para los sistemas finitos, esta contenido en las equivalencias semánticas que no consideran la información de paralelismo y es compatible con la ST-bisimulación.

Se muestra también que el preorden definido es compatible con una amplia familia de combinadores de procesos, aquellos que pueden definirse en el formato SOS-básico.

Finalmente comparamos $\sqsubseteq_{ST}$ con otras propuestas de la literatura.

Introducción

En el desarrollo de programas paralelos muchas veces poseemos una descripción adecuada del comportamiento deseado del programa, sin que esta incluya información sobre el paralelismo. A la vez, disponemos de hardware paralelo donde implementar dicho programa. Hay en general buenas razones para usar este hardware: en algunos casos, como en el de los protocolos de comunicación, la distribución forma parte del problema, en otros, deseamos simplemente acelerar nuestros cálculos. Surge entonces el propósito de paralelizar un programa (o una especificación). Esto es, a partir de un programa queremos construir otro con "el mismo comportamiento" pero "más paralelo". Este problema ha sido encarado desde múltiples perspectivas, desde la creación de amplias

áreas de estudio, como la de programación paralela, hasta la propuesta de refinadas técnicas de extracción automática de paralelismo.

En este trabajo estudiamos la siguiente pregunta:

¿ Cómo definir cuándo un programa es una implementación paralela de otro?

Nuestra perspectiva es la de la semántica formal (en [Lee90] puede encontrarse una visión de conjunto sobre este tema) buscando, para un cierto modelo semántico de programas que permite representar el paralelismo ("truly concurrent"), una relación $\sqsubseteq$ entre programas tal que $G_1 \sqsubseteq G_2$ pueda interpretarse naturalmente como: G_2 *es una implementación paralela de* G_1. Nos abstraemos así de todo lenguaje de programación, o de especificación o de cualquier arquitectura en particular.

Nuestro objetivo es sentar una base teórica para los Métodos Formales y la Ingeniería de Software necesarias en la construcción de programas paralelos.

Una respuesta a nuestra pregunta desde la semántica formal fue dada por Aceto en [Ace89]. Allí define un preorden $\leq$, "menos paralelo que", inspirado en la bisimulación [Par81] sobre los términos de una versión restringida de CCS [Mil80, Mil89]. Esencialmente, $p \leq q$ cuando todo pomset [Pra86] que p puede ejecutar, puede ser imitado por q con el mismo pomset o, eventualmente, con uno "más paralelo". Recíprocamente, toda ejecución de q puede ser imitada por p, eventualmente, con un pomset "menos paralelo". En todos los casos, la imitación lleva a estados relacionados por $\leq$. Luego de presentar su preorden, Aceto se interesa en las nociones de comporatamiento que este preserva, y que nos dicen en qué sentido un programa es implementación del otro: muestra entonces que los términos relacionados por el preorden son bisimilares. Finalmente provee una axiomatización completa de $\leq$ sobre los términos sin recursión.

Una aproximación más general es presentada por Yankelevich en [Yan93]. Allí propone una técnica para obtener preórdenes entre procesos a partir de preórdenes entre las observaciones de dichos procesos, utilizando distintas formas de la bisimulación. Trabajando con una variante de CCS, el autor muestra varias aplicaciones de esta técnica, entre ellas una para un preorden "más paralelo que", estrictamente más fino que el propuesto por Aceto. Se presentan también axiomatizaciones completas para estos preórdenes, parametrizadas por las axiomatizaciones sobre las observaciones.

Una propuesta similar a la presentada en este trabajo, pero reduciendo el problema al caso determinístico, es abordada por el autor en [Fia95].

En este trabajo abordamos al problema sobre un dominio semántico particular, los *Sistemas de Transiciones Asíncronos* (STA) [Shi85, Bed87]. Los STA son una generalización simple de los Sistemas de Transiciones [Kel76] donde las transiciones están etiquetadas por eventos, sobre los que se define una relación de independencia. Intuitivamente, dos eventos independientes pueden ejecutarse "en paralelo".

La principal contribución de este trabajo es la propuesta de un preorden $\sqsubseteq_{ST}$ sobre los STA, tal que $G_1 \sqsubseteq_{ST} G_2$ puede interpretarse como "G_2 es una implementación

paralela de G_1". Este preorden, inspirado en la ST-bisimulación [GV87], una relación de equivalencia que, al igual que la bismimulación, nos permite formalizar una fina noción de "igual comportamiento", teniendo en cuenta la capacidad de los procesos de ejecutar acciones simultáneamente. Esta definición de ser "más paralelo que" es decidible cuando el STA es finito. Mostramos que $\sqsubseteq_{ST}$ es un preorden compatible con la ST-bisimulación. Establecemos luego que el preorden está contenido en las equivalencias semánticas de la literatura que no consideran el paralelismo, debido a que $\sqsubseteq_{ST}$ sólo compara sistemas bisimilares [Par81, Mil83], esto justifica que lo usemos como noción de "implementación de".

Comparamos este preorden con otros que surgen en la literatura. Demostramos que en el caso determinista es equivalente al presentado en [Fia95], que en el caso no-determinista es estríctamente más grueso que el propuesto en [Yan93], e incomparable con el introducido en [Ace89].

Finalmente, $\sqsubseteq_{ST}$ es una precongruencia para una amplia familia de operadores sobre procesos, todos aqellos que puedan especificarse mediante una semántica operacional estructurada básica.

La organización de este trabajo es la siguiente. En la sección 1 presentamos los STA y la noción de ST-estado (que también llamamos diamante). En la sección 2 presentamos la definición de bisimulación clásica, y la de ST-bisimulación. La sección 3 está dedicada a definir el preorden $\sqsubseteq_{ST}$, y a estudiar sus propiedades. La sección 4 presenta un teorema general de precongruencia para $\sqsubseteq_{ST}$. En la sección 5 comparamos nuestro trabajo con otros que aparecen en la literatura. Concluimos en la sección 6 con una discusión sobre trabajos en curso y futuros.

1 Sistemas de Transiciones Asíncronos

Los *Sistemas de Transiciones Asíncronos (STA)*, introducidos en forma independiente por Shields y Bednarczyk [Shi85, Bed87], son una generalización simple de los Sistemas de Transiciones (ST). Los ST son el modelo mejor conocido de sistemas reactivos [Pnu85]. Son grafos orientados donde los vértices representan los *estados* del sistema (entre ellos generalmente se distingue uno, el *inicial*) y las flechas las *transiciones* entre los estados, *etiquetadas* sobre un alfabeto. Una ejecución de un ST es un camino del grafo que parte de un estado dado, y la observación de esta ejecución es la secuencia de etiquetas que se encuentran sobre las flechas.

Los STA generalizan a los ST: las etiquetas (llamadas *eventos*) poseen una *relación de independencia*. Intuitivamente, dos eventos independientes no comparten recursos y pueden por ello ejecutarse en paralelo. La relación de independencia es una relación binaria, simétrica e irreflexiva I, tal que para toda pareja de eventos $(a, b) \in I$, las transiciones satisfacen:

- *Estabilidad hacia adelante:* Si un estado es origen de dos transiciones etiquetadas

a y b, entonces lo es de dos secuencias etiquetadas $\langle a, b\rangle$ y $\langle b, a\rangle$ que convergen a un mismo estado.

- *Conmutatividad:* Si un estado es origen de una secuencia de transiciones etiquetada $\langle a, b\rangle$, también lo es de una secuencia $\langle b, a\rangle$, y ambas convergen a un mismo estado.

Resumiendo: cuando dos transiciones etiquetadas por eventos independientes parten de un mismo estado o están en secuencia, ellas forman un *diamante*. Esto da nacimiento a la noción de *ST-estado* [GV87], una noción matemática elegante que generaliza las de estado, transición y la de un conjunto de transiciones etiquetadas por eventos independientes. Las transiciones también satisfacen:

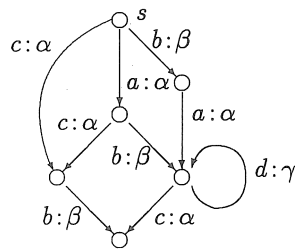
- *Determinismo:* De cada estado parte a lo sumo una transición etiquetada con un evento dado.

Definición 1 *Un* Sistema de Transiciones Asíncrono *(STA) etiquetado sobre un alfabeto de acciones* Act $= \{\alpha, \beta, \gamma, \ldots\}$ *es una estructura* $G = \langle Q, E, I, \rightarrow, l\rangle$, *donde*

- $Q = \{i, s, u, \ldots\}$ *es un* conjunto de estados, *a veces distinguimos a un estado* i *al que llamamos* estado inicial.
- $E = \{a, b, e \ldots\}$ *es un* conjunto de eventos.
- $I \subseteq E \times E$ *es la* relación de independencia, *irreflexiva y simétrica.*
- $\rightarrow \subseteq Q \times E \times Q$, *es la* relación de transición *(escribimos* $s \xrightarrow{e} s_1$ *cuando* $(s, e, s_1) \in \rightarrow$*) que cumple:*
  - *Determinismo. Si* $s \xrightarrow{e} s_1$ *y* $s \xrightarrow{e} s_2$ *entonces* $s_1 = s_2$.
  - *Estabilidad hacia adelante. Si* aIb, $s \xrightarrow{a} s_1$ *y* $s \xrightarrow{b} s_2$ *entonces* $\exists u. s_1 \xrightarrow{b} u \wedge s_2 \xrightarrow{a} u$.
  - *Conmutatividad. Si* aIb *y* $s \xrightarrow{a} s_1 \xrightarrow{b} u$ *entonces* $\exists s_2.\ s \xrightarrow{b} s_2 \xrightarrow{a} u$.
- $l : E \mapsto$ Act *es la función de etiquetado.*

Notamos $\mathcal{A} = \{G, G_1, G_2 \ldots\}$ *la clase de los STA.*

Ejemplo 2 *Mostramos aquí la representación gráfica de un STA sobre el alfabeto* $\{\alpha, \beta, \gamma\}$ *con 6 estados* $E = \{a, b, c, d\}$ *e* $I = \{(a, b), (b, a)\}$.



G

Los ST-estados (o diamantes) son una representación de la ejecución en paralelo de varios eventos independientes a partir de un estado.

Definición 3 (ST-estado) *Sea* $G = \langle Q, i, E, I, \rightarrow \rangle$. *Un* ST-estado *(o* diamante*) de* G *es una pareja* $\langle s, \{e_1, \ldots, e_n\} \rangle \in Q \times \mathcal{P}(E)$ *tal que para todos los* $e_j, e_k \in \{e_1, \ldots e_n\}$, $s \xrightarrow{e_j}$ *y si* $j \neq k$ *entonces* $e_j I e_k$. *Anotamos* $s + \sum_J e_j$ *para referirnos al diamante* $\langle s, \{e_1, \ldots, e_n\} \rangle$, *donde* $J = \{1, \ldots, n\}$ *es el conjunto de índices.*

El destino *de un ST-estado* $\langle s, \{e_1, \ldots e_n\} \rangle$ *(al cual notamos* $\lceil s + \sum_J e_j \rceil$ *siendo* $J = \{1, \ldots, n\}$ *el conjunto de índices) es el estado* u *tal que* $s \xrightarrow{e_1} \ldots \xrightarrow{e_n} u$. *Anotamos* $\mathrm{Diam}(G)$ *el conjunto de ST-estados de* G.

Los ST-estados son una representación abstracta de los estados (por ejemplo $i = \langle i, \emptyset \rangle$), las transiciones (por ejemplo $i \xrightarrow{b} s = \langle i, \{b\} \rangle$) y los diamantes engendrados por eventos independientes (por ejemplo $\langle i, \{a, b\} \rangle$). Nótese que cuando un STA es finito, también lo es el conjunto de sus ST-estados.

2 Bisimulación y ST-bisimulación

Lo que sigue es una presentación algo inusual de la bisimulación clásica de Milner y Park. La presentación apunta a señalar la estrecha relación entre esta definición y las relacionadas a la ST-bisimuación que aparecen más adelante, y sólo difiere de la usual en establecer relaciones entre las transiciones (diamantes de dimensión 1) además de hacerlo, como es usual, sobre los estados (diamantes de dimensión 0).

Esta definición nos dice que dos STA son bisimilares cuando pueden imitarse estados y transiciones.

Definición 4 (Bisimulación) *Sea* $G = \langle Q, E, I, \rightarrow, l \rangle$ *un STA. Una relación* $\mathcal{R} \subseteq Diam(G) \times Diam(G) \times \mathcal{P}(E \times E)$ *es una bisimulación en* G *sii para todo* $(s + \sum_J a_j, r + \sum_K b_k, h) \in \mathcal{R}$ *se cumple que*

1. *(buena formación de* h*)* $h : \{a_j : j \in J\} \rightarrow \{b_k : k \in K\}$ *es una biyección que preserva el etiquetado, i.e.* $l(h(a)) = l(a)$.

2. *(simulación a la derecha hasta una dimensión) si* $J = K = \emptyset$ *y* $s + a \in \mathrm{Diam}(G)$ *entonces* $\exists b \in E$ *tal que* $(r + b \in \mathrm{Diam}(G)$ *y* $(s + a, r + b, \{(a, b)\}) \in \mathcal{R}$.

3. *(simulación a la izquierda hasta una dimensión) si* $J = K = \emptyset$ *y* $r + b \in \mathrm{Diam}(G)$ *entonces* $\exists a \in E$ *tal que* $(s + a \in \mathrm{Diam}(G)$ *y* $(s + a, r + b, \{(a, b)\}) \in \mathcal{R}$.

4. *(culminación)* $\forall i \in J$, $(\lceil s + a_i \rceil + \sum_{J \setminus \{i\}} a_j, \lceil r + h(a_i) \rceil + \sum_{K, b_k \neq h(a_i)} b_k, h\,|_{\{a_j : j \in J, j \neq i\}}) \in \mathcal{R}$

Para todo $s, r \in Q$ *escribimos que* $\mathcal{R} : s \underline{\leftrightarrow} r$ *sii* $\mathcal{R}$ *es una bisimulación de* G *y* $(s, r, \emptyset) \in \mathcal{R}$.

Anotaremos $s \underline{\leftrightarrow} r$ *sii existe* $\mathcal{R}$ *tal que* $\mathcal{R} : s \underline{\leftrightarrow} r$.

Lema 5 *Para todo* $s, r \in Q$, *s es bisimilar con r según la definición clásica de bisimulación [Mil83, Par81] sii existe una bisimulación* $\mathcal{R}$ *(según la Definición 4 tal que* $\mathcal{R} : s \underline{\leftrightarrow} r$.

Puede verse que en la definición anterior sólo son trascendentes los diamantes de dimensión 0 y 1 (estados y transiciones), que son los que "deben ser imitados". Podemos hacer la definición más estricta solicitando que la imitación del crecimiento siga hasta cualquier dimensión. Esto nos da exactamente la ST-bisimulación.

Definición 6 (ST-bisimulación [GV87]) *Sea* $G = \langle Q, E, I, \rightarrow, l \rangle$ *un STA. Una relación* $\mathcal{R} \subseteq \mathrm{Diam}(G) \times \mathrm{Diam}(G) \times \mathcal{P}(E \times E)$ *es una ST-bisimulación en G sii para todo* $(s + \sum_J a_j, r + \sum_K b_k, h) \in \mathcal{R}$ *se cumple que*

1. *(buena formación de h)* $h : \{a_j : j \in J\} \rightarrow \{b_k : k \in K\}$ *es una biyección que preserva el etiquetado, i.e.* $l(h(a)) = l(a)$.

2. *(simulación a la derecha hasta cualquier dimensión) si* $s + \sum_J a_j + a \in \mathrm{Diam}(G)$ *entonces* $\exists b \in E$ *tal que* $r + \sum_K b_k + b \in \mathrm{Diam}(G)$ *y* $(s + \sum_J a_j + a, r + \sum_K b_k + b, h \cup \{(a, b)\}) \in \mathcal{R}$.

3. *(simulación a la izquierda hasta cualquier dimensión) si* $r + \sum_K b_k + b \in \mathrm{Diam}(G)$ *entonces* $\exists a \in E$ *tal que* $s + \sum_J a_j + a \in \mathrm{Diam}(G)$ *y* $(s + \sum_J a_j + a, r + \sum_K b_k + b, h \cup \{(a, b)\}) \in \mathcal{R}$.

4. *(culminación)* $\forall i \in J$, $(\lceil s + a_i \rceil + \sum_{J \setminus \{i\}} a_j, \lceil r + h(a_i) \rceil + \sum_{K, b_k \neq h(a_i)} b_k, h\,|_{\{a_j : j \in J, j \neq i\}}) \in \mathcal{R}$

Para todo $s, r \in Q$ *escribimos que* $\mathcal{R} : s \equiv_{ST} r$ *sii* $\mathcal{R}$ *es una ST-bisimulación de G y* $(s, r, \emptyset) \in \mathcal{R}$.

Diremos que s y r son ST-bisimilares, lo que anotaremos $s \equiv_{ST} r$ *sii existe* $\mathcal{R}$ *tal que* $\mathcal{R} : s \equiv_{ST} r$.

3 ST-Prebisimulación

En la sección anterior hemos presentado dos equivalencias sobre los STA. Ambas pueden verse como un elaborado juego de simulación entre dos procesos, donde cada uno debe poder imitar al otro, y el rol de imitador e imitado pueden intercambiarse en cada momento. La diferencia entre esas dos equivalencias reside en cuáles son las cosas que deben imitarse: la bisimulación es, en ese sentido, menos exigente que la ST-bisimulación, ya que pide solamente imitar estados y transiciones, esto es, ST-estados de dimensión 0 y 1. La ST-bisimulación exige que esta imitación se mantenga a cualquier dimensión.

En esta sección presentaremos la primera contribución de nuestro trabajo: definiremos la ST-prebisimulación sobre los STA. Intuitivamente la ST-prebisimulación es también un juego de simulación donde los roles de imitador e imitado pueden intercambiarse en cada instante. Pero los jugadores tendrán diferente exigencia respecto a aquello que debe imitarse. El proceso "más paralelo" debe imitar todo aquello que el "menos paralelo" pueda hacer, pero la exigencia para éste es más débil: basta que pueda imitar, como en la bisimulación clásica, los estados y las transiciones.

Puede pensarse que es un juego de imitación, con intercambio de roles, entre niños de diferente edad: aquel que es mayor podrá imitar todo lo que el más pequeño haga, y éste aquellos gestos más simples de su compañero mayor.

Luego de presentar la definición de la ST-bisimulación presentaremos algunas de sus propiedades esenciales, entre otras, que forma un preorden sobre los STA, y que tiene buenas relaciones con la bisimulación y la ST-bisimulación.

Definición 7 (ST-prebisimulación) *Sea* $G = \langle Q, E, I, \rightarrow, l\rangle$ *un STA. Una relación* $\mathcal{R} \subseteq Diam(G) \times Diam(G) \times \mathcal{P}(E \times E)$ *es una ST-prebisimulación de* G *sii para todo* $(s + \sum_J a_j, r + \sum_K b_k, h) \in \mathcal{R}$ *se cumple que*

1. *(buena formación de h)* $h : \{a_j : j \in J\} \rightarrow \{b_k : k \in K\}$ *es una biyección que preserva el etiquetado, i.e.* $l(h(a)) = l(a)$.

2. *(simulación a la derecha hasta cualquier dimensión) si* $s + \sum_J a_j + a \in \mathrm{Diam}(G)$ *entonces* $\exists b \in E$ *tal que* $r + \sum_K b_k + b \in \mathrm{Diam}(G)$ *y* $(s + \sum_J a_j + a, r + \sum_K b_k + b, h \cup \{(a, b)\}) \in \mathcal{R}$.

3. *(simulación a la izquierda hasta una dimensión) si* $J = K = \emptyset$ *y* $r + b \in \mathrm{Diam}(G)$ *entonces* $\exists a \in E$ *tal que* $(s + a \in \mathrm{Diam}(G)$ *y* $(s + a, r + b, \{(a, b)\}) \in \mathcal{R}$.

4. *(culminación)* $\forall i \in J, (\lceil s + a_i \rceil + \sum_{J\setminus\{i\}} a_j, \lceil r + h(a_i) \rceil + \sum_{K, b_k \neq h(a_i)} b_k, h\,|_{\{a_j : j \in J, j \neq i\}}) \in \mathcal{R}$

Para todo $s, r \in Q$ *escribimos que* $\mathcal{R} : s \sqsubseteq_{ST} r$ *sii* $\mathcal{R}$ *es una ST-prebisimulación de* G *y* $(s, r, \emptyset) \in \mathcal{R}$.

Diremos que s *y* r *son ST-prebisimilares, lo que anotaremos* $s \sqsubseteq_{ST} r$ *sii existe* $\mathcal{R}$ *tal que* $\mathcal{R} : s \sqsubseteq_{ST} r$.

Lema 8 *Sean G un STA y $\mathcal{R}_1$ y $\mathcal{R}_2$ dos ST-prebisimulaciones de G, entonces $\mathcal{R}_1 \cup \mathcal{R}_2$ es una ST-prebisimulación de G.*

Corolario 9 *Sea G un STA, entonces existe $\mathcal{R}$ la mayor ST-prebisimulación de G.*

La composición de dos ST-prebisimulaciones se puede definir de la siguiente manera:

Definición 10 *Sean $\mathcal{R}_1$ y $\mathcal{R}_2$ dos ST-prebisimulaciones de un STA G dado, entonces*

$$\mathcal{R}_2 \circ \mathcal{R}_1 = (s + \sum_J a_j, r + \sum_K b_k, h_2 \circ h_1) \mid \textit{exiten}$$
$$(s + \sum_J a_j, t + \sum_I c_i, h_1) \in \mathcal{R}_1 \ y \ (t + \sum_I c_i, r + \sum_K b_k, h_2) \in \mathcal{R}_2$$

Se cumple entonces el siguiente lema:

Lema 11 *Sean G un STA dado y $\mathcal{R}_1$ y $\mathcal{R}_2$ dos ST-prebisimulaciones en G, entonces $\mathcal{R}_2 \circ \mathcal{R}_1$ también es una ST-prebisimulación en G.*

La composición de una ST-bisimulación con una ST-prebisimulación , y la de una ST-prebisimulación con una ST-bisimulación se pueden definir en forma totalmente análoga a la Definición 10.

Se cumplen entonces los siguientes dos lemas:

Lema 12 *Sean G un STA dado, $\mathcal{R}_1$ una ST-prebisimulación en G y $\mathcal{R}_2$ una ST-bisimulación en G, entonces $\mathcal{R}_2 \circ \mathcal{R}_1$ es una ST-prebisimulación en G.*

Lema 13 *Sean G un STA dado, $\mathcal{R}_1$ una ST-bisimulación en G y $\mathcal{R}_2$ una ST-prebisimulac en G, entonces $\mathcal{R}_2 \circ \mathcal{R}_1$ es una ST-prebisimulación en G.*

Por último, presentaremos algunas propiedades de $\sqsubseteq_{ST}$.

Lema 14 *$\sqsubseteq_{ST} \subseteq Q \times Q$ es un preorden.*

Lema 15 *Sean G un STA y $s, r, s', r' \in Q$ estados de G tales que $s' \equiv_{ST} s \sqsubseteq_{ST} r \equiv_{ST} r'$, entonces $s' \sqsubseteq_{ST} r'$.*

Lema 16 *Sean G un STA y $s, r \in Q$ dos estados de G tales que $s \sqsubseteq_{ST} r$ entonces $s \underline{\leftrightarrow} r$.*

4 Teorema general de precongruencia

En esta sección presentaremos un teorema que nos habla de como interactúan la relación de ST-prebisimulación recién definida y los combinadores de procesos usuales. El resultado buscado en general es que la relación definida nos permite razonar modularmente sobre los procesos construidos por los combinadores. Esto quiere decir que deseamos mostrar que en general, dado un operador *op*, el hecho que $p \sqsubseteq_{ST} q$ nos basta para asegurar que $op(p) \sqsubseteq_{ST} op(q)$.

El teorema no se plantea para un lenguaje en particular. El resultado que presentamos aquí se cumple para una gran familia de operadores, todos aquellos que se encuentran en lenguajes que pueden describirse con la ayuda de la técnica SOS utilizando un cierto formato particular llamado "de base" en la literatura. En realidad este formato es suficientemente amplio para definir las álgebras de procesos más conocidas, como CCS [Mil80, Mil89]. Y es el formato más libre para el que se conoce una forma canónica de construir un STA a partir de una descripción SOS [BD92, EHP95].

Teorema 17 (Precongruencia) *La ST-prebisimulación es una precongruencia para todos los operadores definidos en una especificación SOS básica irreflexiva.*

En otras palabras, sea G_P el STA asociado a una especificación irreflexiva P y $\mathcal{R}_P$ la mayor ST-prebisimulación en G_P. Entonces, para todo $t_i, u_i \in T_\Sigma$ con $i = 1, \ldots, n$ y $f \in \Sigma^n$ se cumple que

$$\forall i = 1, \ldots, n. \mathcal{R}_P : t_i \sqsubseteq_{ST} u_i \Rightarrow \mathcal{R}_P : f(t_1, \ldots, t_n) \sqsubseteq_{ST} f(u_1, \ldots, u_n)$$

5 Comparación con otras propuestas

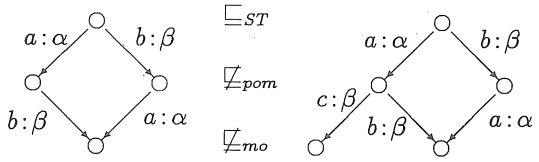
En esta sección comparamos el preorden $\sqsubseteq_{ST}$ definido en este trabajo con otros preórdenes que, buscando formalizar la noción de "más paralelo que", aparecen en la literatura.

A nuestro conocimiento existen tres propuestas relacionadas con la que aquí se presenta, el preorden presentado por Aceto en [Ace89] (inspirado en la *pomset bisimulation* [BC87, BC88], que aquí llamaremos $\sqsubseteq_{pom}$), el preorden $\sqsubseteq_{mo}$ introducido por Yankelevich en [Yan93] (inspirado en la *mixed order equivalence* [DDM89]) y el preorden $\sqsubseteq$ propuesto por el autor en [Fia95] (para STA deterministas).

El resultado es que $\sqsubseteq_{ST}$ es incomparable con $\sqsubseteq_{pom}$, mientras que es más gruesa que $\sqsubseteq_{mo}$ (inclusive considerando esta para el caso en que no aparece la acción sigilosa τ, que es el que aquí consideramos). En el caso de los STA deterministas, la propuesta presentada aquí coincide con $\sqsubseteq$.

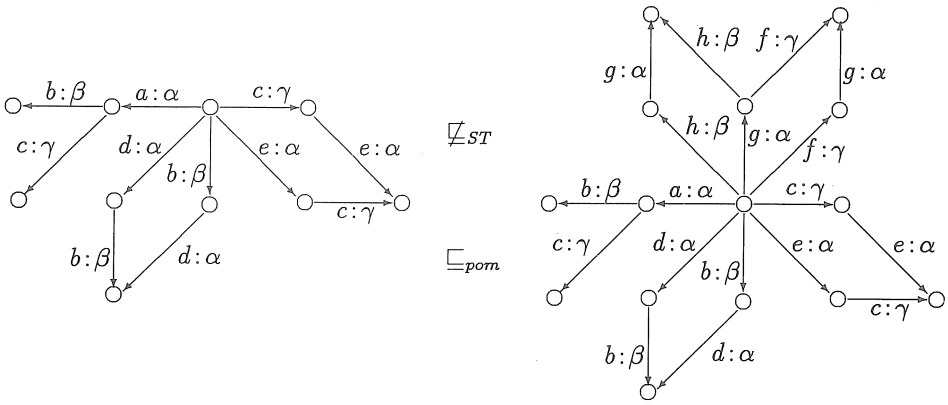
Los siguientes ejemplos nos permiten demostrar la relación que guarda nuestro trabajo con el trabajo de Aceto y parte de la relación que este guarda con el de Yankelevich. Del ejemplo

Ejemplo 18 *Mostramos aquí que $\sqsubseteq_{ST} \not\subseteq \sqsubseteq_{pom}$ y que $\sqsubseteq_{ST} \not\subseteq \sqsubseteq_{mo}$. En este ejemplo aIb.*



podemos concluir que $\sqsubseteq_{ST}$ no esta incluida ni en $\sqsubseteq_{pom}$ ni en $\sqsubseteq_{mo}$. Del ejemplo

Ejemplo 19 *Mostramos aquí que* $\sqsubseteq_{pom} \not\subseteq \sqsubseteq_{ST}$. *En este ejemplo* $I = \{(d,b),(e,c),(g,f),(g,h)$



podemos concluir que $\sqsubseteq_{pom}$ no esta incluido en $\sqsubseteq_{ST}$.

Puede demostrarse que $\sqsubseteq_{mo}$ es estríctamente más fino que $\sqsubseteq_{ST}$, y que en el caso determinista $\sqsubseteq_{ST}$ coincide con $\sqsubseteq$. Desafortunadamente para presentar las pruebas de estas propiedades es necesario agregar una buena cantidad de nuevas definiciones, por lo que hemos preferido simplemente enunciarlas. No es un trabajo complejo, pero si extenso.

6 Conclusiones

En este trabajo formalizamos una noción de "un programa es una implementación paralela de otro" en el marco de los Sistemas de Transiciones Asíncronos de manera independiente de cualquier lenguaje de programación, especificación o arquitectura.

Hemos definido para ello el preorden $\sqsubseteq_{ST}$ en los STA. Cuando $G_1 \sqsubseteq_{ST} G_2$ ambos hacen las mismas cosas (son equivalentes en todas las equivalencias semánticas de la literatura que no toman en cuenta el paralelismo) y a la vez G_2 lo hace de manera más paralela (los eventos que pueden ocurrir en paralelo en G_1 pueden hacerlo en G_2). Mostramos que si dos STA están relacionados por $\sqsubseteq_{ST}$ entonces son bisimilares, lo que nos asegura que tienen el mismo comportamiento. Probamos además que $\sqsubseteq_{ST}$ es compatible con la ST-bisimulación, en el sentido que preserva las clases de equivalencia. Por otro lado, al estar definido como una relación entre ST-estados, si el STA es finito $\sqsubseteq_{ST}$ es trivialmente decidible.

Nuestro preorden es estrictamente más grueso que el propuesto por Yankelevich en [Yan93], e incomparable con el preorden presentado por Aceto en [Ace89]. Presentamos ejemplos que evidencian estos resultados. Por otro lado, si reducimos nuestra definición al caso determinista, $\sqsubseteq_{ST}$ coincide con el presentado por el autor en [Fia95].

Por último, demostramos que, para una amplia familia de operadores de procesos, más precisamente, todos aquellos que puedan definirse a través de una semántica operacional estructurada básica [BIM88], $\sqsubseteq_{ST}$ resulta ser una precongruencia.

Existen varias líneas para continuar esta investigación. Una es ampliar nuestra definición a los STA con la acción invisible τ. Otra, sobre la cual estamos formando un grupo de trabajo, es seguir la línea de Aceto y buscar una axiomatización para el preorden.

Agradecimientos

Agradecemos a Pedro Ruben D'Argenio, Juan José Cabezas, Guillermo Calderón, Victoria Galatro, Martina Marré, Alfredo Olivero, Cecilia Pertino, Luis Sierra, Jorge Vidart, Daniel Yankelevich y Sergio Yovine que nos han hecho valiosos comentarios.

Este trabajo ha sido parcialmente financiado por el *Programa de Desarrollo de Ciencias Básicas*, (PEDECIBA), Uruguay.

Referencias

[Ace89] L. Aceto. On relating concurrency and nondeterminism. Report 6/89, Univ. Sussex, Brighton, GB, October 1989.

[BC87] G. Boudol and I. Castellani. On the semantics of concurrency: Partial orders and transition systems. In *Proc. CAAP'87, Pisa, LNCS 249*, pages 123–137. Springer-Verlag, March 1987.

[BC88] G. Boudol and I. Castellani. Concurrency and atomicity. *Theoretical Computer Science*, 59(1):25–84, 1988.

[BD92] E. Badouel and Ph. Darondeau. Structural operational specifications and trace automata. In *Proc. CONCUR'92, Stony Brook, NY, LNCS 630*, pages 302–316. Springer-Verlag, August 1992.

[Bed87] M. A. Bednarczyk. *Categories of Asynchronous Systems.* PhD thesis, Univ. Sussex, October 1987. Available as CS R 1/88.

[BIM88] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced: preliminary report. In *Proc. 15th ACM Symp. Principles of Programming Languages, San Diego, CA*, pages 229–239, January 1988.

[DDM89] P. Degano, R. De Nicola, and U. Montanari. Partial orderings descriptions and observations of nondeterministic concurrent processes. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 438–466. Springer-Verlag, 1989.

[EHP95] J. Echagüe, Z. Habbas, and S. Pinchinat. Structural Operational Semantics Specifications for True Concurrency., 1995.

[Fia95] G. Fialco. Cuándo un programa es más paralelo que otro? Pasantía, Escuela Superior Latinoamericana de Informática, 1995.

[GV87] R. J. van Glabbeek and F. Vaandrager. Petri net models for algebraic theories of concurrency. In *Proc. PARLE'87, vol. II: Parallel Languages, Eindhoven, LNCS 259*, pages 224–242. Springer-Verlag, June 1987.

[Kel76] R. M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, July 1976.

[Lee90] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science, vol. B.* Elsevier Science Publishers, 1990.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 23(3):267–310, 1983.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall Int., 1989.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *Proc. 5th GI Conf. on Th. Comp. Sci., LNCS 104*, pages 167–183. Springer-Verlag, March 1981.

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Lect. Notes, Aarhus University, Aarhus, DK, 1981.

[Pnu85] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *Proc. 12th ICALP, Nafplion, LNCS 194*, pages 15–32. Springer-Verlag, July 1985.

[Pra86] V. R. Pratt. Modeling concurrency with partial orders. *Int. J. Parallel Programming*, 15(1):33–71, 1986.

[Shi85] M. W. Shields. Concurrent machines. *The Computer Journal*, 28(5):449–465, 1985.

[Yan93] D. N. Yankelevich. *Parametric Views of Process Description Languages.* PhD thesis, Univ. Pisa, 1993.