Reformulación del modelo Linda para compartir conocimiento en Sistemas Multi-agente

Luciano Héctor Tamargo **Alejandro Javier García** **Marcelo Alejandro Falappa**
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Laboratorio de Investigación y Desarrollo de Inteligencia Artificial (LIDIA)
Departamento de Ciencias e Ingeniería de la Computación - Universidad Nacional del Sur
{lt, ajg, mfalappa}@cs.uns.edu.ar

Abstract

The aim of this work is to analyze the knowledge interchange on multi-agent systems using shared knowledge areas. The project involves the analysis and reformulation of some aspects of the Linda model, adding new model level features. This work proposes the idea of distributed tuples spaces, offering the possibility of coexistence among different tuples spaces with distinct features, being stored in remote sites. This distributed model requires the definition of new operations whose domain and range are tuples spaces, and the formulation of access constraints for the agents.

Key words: Multi-Agents Systems, Agents, Linda, Shared knowledge.

1. INTRODUCCIÓN

En un sistema multi-agente los agentes deben poder intercambiar información, ya sea en un entorno colaborativo o basado en negociación. Este intercambio puede hacerse por medio de pasaje de mensajes o utilizando áreas de conocimiento compartidas. Este trabajo tiene como objetivo analizar el intercambio de conocimiento en sistemas multi-agente por medio de áreas de conocimiento compartidas. La importancia del mismo radica en que, disponer de un modelo para compartir conocimiento, es un aspecto a resolver en sistemas multi-agente con agentes deliberativos. Por lo tanto, sería interesante contar con un modelo para el mantenimiento de bases de conocimiento que permita abstraerse de los aspectos de implementación.

El modelo Linda [2, 3, 4, 5], es un modelo de memoria compartida originalmente definido para proveer comunicación y sincronización en programas con paralelismo. Utiliza un espacio de memoria compartida llamado *"espacio de tuplas"* (ET), sobre el cual propone seis operaciones. Agregando estas operaciones del ET a un lenguaje base se genera un dialecto de programación paralela. Los ETs son una abstracción a partir de la cual los procesos cooperan y se comunican. Estos pueden verse como áreas de memoria compartidas asociativas referenciadas que no requieren hardware subyacente para el área de memoria física donde residen. Es importante notar que los ETs difieren de las relaciones que son propuestas en el modelo de datos relacional [6], debido a que las relaciones sólo aceptan tuplas que tienen formato (las mismas deben tener la misma aridad, y sus atributos deben ser tipados y mantener un orden); sin embargo, los espacios de tuplas aceptan tuplas sin formato.

Linda provee cuatro operaciones básicas de espacio de tuplas, out, in , rd y $eval$, y dos variantes adicionales inp y rdp. La operación **out(t)** causa que la tupla t sea agregada al ET; **in(t)** causa que alguna tupla s que concuerde con la tupla-molde t sea retirada del ET (lectura destructiva), y si no se encuentra una tupla *s* que coincida cuando *in(t)* se ejecuta, la ejecución del proceso se suspende

Financiado parcialmente por CONICET (PIP 5050), Universidad Nacional del Sur (PGI 24/ZN11) y Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 Nro 13096).

hasta que una tupla que coincida esté disponible (es una primitiva bloqueante); **rd(t)** es igual a *in(t)*, excepto que la tupla que coincide no es retirada y permanece en el ET. Las versiones predicativas de *in* y *rd*, **inp** y **rdp** respectivamente, intentan localizar una tupla que coincida, y retornan 0 si la búsqueda falla, en caso contrario retornan 1. La operación **eval(t)** es igual a *out(t)*, excepto que t es evaluado después (en paralelo) en vez de antes de ser ingresado en el ET.

Luego de las primeras definiciones del modelo Linda, surgieron varias extensiones, por ejemplo [8, 1, 10] (ver sección 5), en las cuales se hace referencia a la utilización de múltiples espacios de tuplas para obtener mejoras sobre aspectos que conciernen a los ambientes distribuidos y paralelos. A diferencia de éstas, este trabajo propone reformular algunos aspectos del modelo Linda con el objetivo de lograr adaptarlo a un entorno distribuido en un sistema multi-agente con agentes con conocimiento. Se usa como punto de partida este modelo por que es un formalismo simple y eficiente, que puede brindar comunicación y sincronización en un sistema multi-agente a través de memoria compartida.

En este trabajo se propone la idea de espacios de tuplas distribuidos, brindando la posibilidad de coexistir diferentes espacios de tuplas con características distintas, residiendo en áreas de memorias remotas. Para que un modelo satisfaga esta propuesta, se plantea reformular las definiciones originales de Linda en los siguientes aspectos:

1. Cantidad de espacios de tuplas.
2. Restricción de acceso al espacio.
3. Distribución del ET.

El artículo está organizado como sigue: en la sección 2 se plantea el aspecto relacionado a la cantidad de ETs, en el cual se analiza las ventajas y desventajas de tener un único ET o varios, y también, se introducen nuevas operaciones que tienen como dominio y rango ETs; en la sección 3 se describe el aspecto de restricción de acceso a los ETs; en la sección 4 se analizan distintas alternativas para asociar ETs con áreas de memoria; y en la sección 5 se plantean los trabajos relacionados, la conclusión y el trabajo a futuro.

2. OPERACIONES SOBRE MULTIPLES ESPACIOS DE TUPLAS

Para compartir conocimiento se podría utilizar **sólo un espacio de tuplas** o varios espacios de tuplas. En caso que un sistema multi-agente utilice uno solo y el intercambio de conocimiento entre los agentes del sistema se realizara sólo por medio del ET, la comunicación entre los agentes que intervienen en el sistema fluirá a través de este único ET.

Esta posibilidad tiene como *ventaja* que explota las virtudes del modelo Linda, en cuanto a simplicidad en la comunicación y forma de compartir conocimiento. Algunos autores argumentan que Linda es simple, poderoso y elegante [2]. Sin embargo, al aplicarlo para compartir conocimiento utilizando un único ET, tiene como *desventajas* que sugiere un esquema centralizado (un SMA es inherentemente distribuido) que expone la falta de robustez. Además, el hecho de que todos los agentes deban comunicarse a través del mismo ET puede generar una sobrecarga que perjudicaría la eficiencia del sistema multi-agente. En adición a esto, las comunicaciones no son seguras, debido a que todos los agentes acceden al mismo ET y, por consiguiente, tienen acceso a la misma información. Esto se puede solucionar incorporando un mecanismo de seguridad, como por ejemplo, agregando permisos de accesos.

La otra posibilidad, la cual es propuesta en trabajos como [8, 1, 10], consiste en permitir la existencia de **varios espacios de tuplas**. Esto brinda la posibilidad de que un sistema multi-agente pueda trabajar con más de un área de conocimiento permitiendo descentralizar el sistema, lo que provoca que el conocimiento sea compartido de manera distribuida. Además, se obtiene mayor eficiencia en el

flujo de comunicación en el sistema, debido a la desconcentración de tráfico de información. Esto es, se pueden repartir grupos de agentes en diferentes ETs para que compartan conocimiento y, de esta manera, descongestionar los ETs. Se puede objetar que la posibilidad de permitir varios espacios de tuplas puede ser cubierta haciendo correr en paralelo varios servidores Linda que provean un único ET. Sin embargo, esta alternativa provee abstracción debido a que los usuarios no necesitan hacer correr un nuevo servidor Linda para generar un nuevo ET.

Como se puede observar, sería deseable proveer más de un ET y, de esta manera, mejorar la comunicación proveyendo diferentes ETs con características diversas. A partir de la posibilidad de que puedan coexistir varios ETs en un mismo sistema multi-agente, surgen naturalmente, nuevas operaciones sobre espacios de tuplas. Esto es, operaciones que tengan como dominio y rango ETs, lo cual hace al modelo aún más interesante y poderoso. Pero se debe notar que, el hecho de agregar más operaciones que se aplican sobre ETs hace que el modelo pierda *ortogonalidad* [11], obligando al usuario a pensar en más operaciones. Estas nuevas operaciones se definen en la siguiente sección.

2.1. Operaciones que tienen como dominio y rango espacios de tuplas

Ante la posibilidad de que en un sistema multi-agente puedan coexistir varios espacios de tuplas, resulta interesante disponer de operaciones que tengan como dominio y rango ETs, como se ha visto en los trabajos [1, 9] donde se sugieren las operaciones *collect* y *copy-collect*, respectivamente. A diferencia de éstas dos, en esta sección se definirán y ejemplificarán operaciones para unir dos ETs, obtener los elementos comunes y calcular la diferencia, entre otras, que en algún sentido recuerdan a las operaciones del álgebra relacional propuestas en [6]. Sin embargo, difieren de estas ya que en el álgebra relacional las relaciones están formateadas y, como se mencionó con anterioridad, los espacios de tuplas no. Es importante destacar que los espacios de tuplas pueden contener tuplas repetidas. Si bien se podría objetar que esto genera redundancia, las tuplas repetidas pueden ser útiles cuando las mismas se usan como "tokens de información", como se muestra en el ejemplo 1. Por lo tanto, las operaciones que se sugerirán tendrán variantes que admiten tuplas replicadas, las cuales se asemejan a las operaciones multi-set del álgebra relacional extendida propuestas en [7]. En el mismo sentido que las operaciones del álgebra relacional, estas difieren de las propuestas en este trabajo, ya que las relaciones sólo admiten tuplas con formato.

Ejemplo 1 *Considere un sistema multi-agente donde participan cuatro agentes (Ag_1, Ag_2, Ag_3 y Ag_4) que están vinculados a un espacio de tuplas. El agente Ag_3 tiene acceso a una impresora ($print_A$) y Ag_4 tiene acceso a otra impresora ($print_B$). Los agentes Ag_1 y Ag_2 solicitan imprimir documentos por medio de* solicitudes de impresión *que ponen en el espacio en forma de tuplas, para que alguno de los agentes que tienen acceso a las impresoras (Ag_3 y Ag_4) retiren la solicitud y envíen a imprimir los trabajos. Bajo estas circunstancias, Ag_1 y Ag_2 pueden solicitar imprimir el mismo documento poniendo en el ET la misma tupla, quedando de esta manera una "tupla replicada". Esto es de utilidad, pues Ag_3 puede retirar una de las solicitudes y la otra puede ser retirada por Ag_4 para luego enviar a imprimir ambas solicitudes en paralelo a sendas impresoras.*

Dentro de un ET, el orden de las tuplas no será considerado relevante. Por lo tanto, en las operaciones que se describen a continuación, cuando hay varias tuplas repetidas, y se debe decidir cual de ellas será parte de la solución, la elección se realizará de manera no determinística.

2.1.1. Operaciones para unión de espacios de tuplas: Estas operaciones permiten, por ejemplo, que un agente que está compartiendo conocimiento en diferentes ETs con distintos agentes, logre juntar en un único ET todo su conocimiento, permitiéndole de esta manera, poder trabajar con su conocimiento global sin alterar el conocimiento de los demás agentes.

Definición 1 : **(Unión total)** Dados dos espacios de tuplas ET_1 y ET_2 la union total $ET_1 \uplus ET_2$ es un espacio de tuplas que contiene todas las tuplas que pertenecen a ET_1 más todas las tuplas que pertenecen a ET_2.

Definición 2 : **(Unión sin réplicas)** Dados dos espacios de tuplas ET_1 y ET_2 la union sin réplicas $ET_1 \bigcup ET_2$ es un espacio de tuplas que contiene las tuplas que pertenecen a ET_1 más las tuplas que pertenecen a ET_2, sin considerar repeticiones de tuplas.

A continuación, se muestra un ejemplo en el cual se podrá notar con claridad la diferencia que existe entre ambas operaciones. Por cuestiones de simplicidad, las tuplas de los ejemplos serán representadas por letras proposicionales *a, b, c* y *d*, debido a que por el momento sólo interesa el comportamiento de las operaciones y no como unifican las tuplas durante el proceso de la operación. En este trabajo, un espacio de tuplas ET_1 que contiene las tuplas *a*, *a* y *c* se denotará ET_1 = *[a,a,c]*. Para la implementación de éstas operaciones se debe proveer la definición de igualdad entre tuplas. Como en este trabajo en los ejemplos se utilizarán letras proposicionales para representar tuplas, dos tuplas son iguales si son representadas por la misma letra proposicional.

Ejemplo 2 *Considere los siguientes espacios de tuplas:* ET_1 *= [a,b,b,a,d] y* ET_2 *= [c,a,b,c,b].*

$ET_1 \uplus ET_2$ *= [a,b,b,a,d,c,a,b,c,b]* $\quad$ $ET_1 \bigcup ET_2$ *= [a,b,d,c]*
$ET_1 \uplus ET_1$ *= [a,b,b,a,d,a,b,b,a,d]* $\quad$ $ET_1 \bigcup ET_1$ *= [a,b,d]*

Aquí se puede observar que el resultado de la operación $ET_1 \uplus ET_2$ *mantiene las repeticiones de los dos espacios de tuplas, mientras que la operación* $ET_1 \bigcup ET_2$ *elimina toda repetición. También se puede obserbar que la operación "unión sin réplica" brinda la posibilidad de retornar el contenido sin réplicas de un ET realizando* $ET_1 \bigcup ET_1$*. Mientras que* $ET_1 \uplus ET_1$ *duplica el contenido del ET.*

Ejemplo 3 *Extendiendo el ejemplo 1, supongamos que existe un agente que tiene como rol recolectar solicitudes de impresión que luego envía a imprimir, y que tiene acceso a una impresora y a dos espacios de tuplas* ET_1 *y* ET_2 *en los cuales se ponen solicitudes de impresión en forma de tuplas. En caso que el agente desee recoger todas las solicitudes de impresión en un único ET, con el objetivo de realizar una administración generalizada de las solicitudes, deberá realizar una "unión total"* $ET_1 \uplus ET_2$*. Si sólo desea obtener una única versión por solicitud, deberá realizar una "unión sin réplicas"* $ET_1 \bigcup ET_2$*.*

2.1.2. Operaciones para intersección de espacios de tuplas: Estas operaciones permiten, por ejemplo, que un agente que está compartiendo conocimiento en diferentes ETs, con distintos agentes, logre obtener en un único ET el conocimiento que comparte en común con los diferentes grupos de agentes.

Definición 3 : **(Intersección total)** Dados dos espacios de tuplas ET_1 y ET_2 la intersección total $ET_1 \biguplus ET_2$ es un espacio de tuplas que contiene todas las tuplas que pertenecen tanto a ET_1 como a ET_2.

Definición 4 : **(Intersección sin réplicas)** Dados dos espacios de tuplas ET_1 y ET_2 la intersección sin réplicas $ET_1 \bigcap ET_2$ es un espacio de tuplas que contiene tuplas que pertenecen tanto a ET_1 como a ET_2, sin considerar repeticiones de tuplas.

Observación: Notar que $ET_1 \bigcap ET_2 \equiv (ET_1 \biguplus ET_2) \bigcup (ET_1 \biguplus ET_2)$, es decir, la intersección sin réplicas es equivalente a la unión sin réplicas de las intersecciones totales de los ETs.

Ejemplo 4 *Considere los siguientes espacios de tuplas:* ET_1 *= [a,b,b,a,d] y* ET_2 *= [c,a,b,c,b].*

$ET_1 \biguplus ET_2$ *= [a,b,b]* $\quad$ $ET_1 \bigcap ET_2$ *= [a,b]*
$ET_1 \biguplus ET_1$ *= [a,b,b,a,d]* $\quad$ $ET_1 \bigcap ET_1$ *= [a,b,d]*

Aquí se puede observar que el resultado de la operación $ET_1 \pitchfork ET_2$ admite tuplas repetidas en el caso de que existan en la intersección, mientras que la operación $ET_1 \bigcap ET_2$ elimina toda repetición existente en la intersección. Además se puede observar que al igual que en la "unión sin réplicas", la "intersección sin réplicas" brinda la posibilidad de retornar el contenido sin réplicas de un ET realizando $ET_1 \bigcap ET_1$. Esto es, $ET_1 \bigcap ET_1 \equiv ET_1 \bigcup ET_1$.

Ejemplo 5 *Supongamos que se cuenta con un agente servidor Ag_s el cual tiene como rol administrar los accesos que producen dos grupos de agentes a una base de datos remota. Cada grupo está vinculado a un ET diferente ET_1 y ET_2, en el cual ponen los accesos a la base de datos en forma de tuplas. El Ag_s, con el objetivo de realizar los accesos de forma eficiente y agilizada, intentará responder primero a aquellas solicitudes de accesos que tienen en común ambos grupos, es decir, las que aparecen en ambos ETs. Para ello es necesario que realice una intersección de los espacios. Si los accesos son de lectura, es conveniente que la intersección sea sin réplicas $ET_1 \bigcap ET_2$, ya que con una sola versión del acceso, al Ag_s le alcanza para proveer el dato a los diferentes agentes que se encuentran en ambos grupos que solicitaron tal acceso. Pero si los accesos son de escritura, es deseable que la intersección sea total $ET_1 \pitchfork ET_2$, debido a que los datos actualizados en la base deben quedar con las versiones correctas.*

2.1.3. Diferencia: Esta operación, por ejemplo, permite que se extraiga conocimiento de un ET que no es común a otro. Esto es, permite que se elimine conocimiento de un ET que ya es tratado en otro, evitando de esta manera trabajar sobre la misma información en grupos de agentes diferentes. Como resultado para el sistema multi-agente, genera que no se solape el trabajo.

Definición 5 : **(Diferencia)** Dados dos espacios de tuplas ET_1 y ET_2 la diferencia $ET_1 - ET_2$ es un espacio de tuplas que contiene todas las tuplas que pertenecen a ET_1 y no pertenecen a ET_2.

Ejemplo 6 *Considere los siguientes espacios de tuplas: ET_1 = [a,b,b,a,d] y ET_2 = [c,a,b,c,b].*

$$ET_1 - ET_2 = [a, d] \qquad ET_1 - ET_1 = [\ ]$$

El caso $ET_1 - ET_1$ permite notar que si se aplica la "diferencia" a un espacio de tuplas con si mismo se obtiene un espacio de tuplas vacío.

Ejemplo 7 *Supongamos que se cuenta con un agente de impresión, como en el ejemplo 3, y que cuando el agente toma una solicitud de impresión de un espacio de tuplas ET_1 y la envía a imprimir, pone dicha solicitud en un nuevo espacio de tuplas ET_2 por cuestiones administrativas. Si el trabajo del agente se suspende, por causas ajenas al sistema, cuando se reanuda el mismo, el agente deberá enviar a imprimir las solicitudes restantes. Para ello el agente debe realizar una "diferencia" $ET_1 - ET_2$ para obtener aquellas solicitudes que no fueron impresas.*

2.1.4. Diferencia simétrica: Esta operación permite, por ejemplo, obtener en un espacio de tuplas el conocimiento que no tienen en común los espacios de tuplas que son operandos de la misma.

Definición 6 : **(Diferencia simétrica)** Dados dos espacios de tuplas ET_1 y ET_2 la diferencia simétrica $ET_1 \sim ET_2$ es un espacio de tuplas que contiene todas las tuplas que pertenecen a ET_1 y no pertenecen a ET_2, más todas las tuplas que pertenecen a ET_2 y no pertenecen a ET_1.

Observación: Notar que $ET_1 \sim ET_2 \equiv (ET_1 - ET_2) \biguplus (ET_2 - ET_1) \equiv (ET_1 - (ET_1 \pitchfork ET_2)) \biguplus (ET_2 - (ET_1 \pitchfork ET_2))$.

Ejemplo 8 *Considere los siguientes espacios de tuplas: ET_1 = [a,b,b,a,d] y ET_2 = [c,a,b,c,b].*

$$ET_1 \sim ET_2 = [a, d, c, c] \qquad ET_1 \sim ET_1 = [\ ]$$

El caso $ET_1 \sim ET_1$ permite notar que si se aplica la "diferencia simétrica" a un espacio de tuplas con si mismo se obtiene un espacio de tuplas vacío.

Ejemplo 9 *Supongamos que un agente servidor de accesos a una base de datos remota (como el mencionado en el ejemplo 5), ya atendió las solicitudes que tienen en común dos ETs (luego de hacer una intersección) y solo resta atender aquellas solicitudes de accesos que no tienen en común los ETs. Para ello realiza una "diferencia simétrica".*

2.1.5. Variantes: Como se ha visto, el resultado de las operaciones de unión, intersección y diferencia es retornado en un nuevo ET. Una variante a esto, consiste en almacenar el resultado de la ejecución de la operación en uno de los operandos (o en ambos) alterando de esta manera el contenido del mismo. A continuación se muestra una tabla que consta de 3 columnas, en la primera figuran las operaciones que se han definido anteriormente (bajo el nombre de operaciones originales), y en las otras dos se muestran las variantes a estas. En la segunda columna, se encuentran aquellas operaciones que varían sólo en el hecho de que el resultado es almacenado en el primer operando. Por ejemplo, en el caso de la unión total ($ET_1 \uplus ET_2$) su variante es la unión total aumentativa ($ET_1 \overleftarrow{\uplus} ET_2$), la cual se comporta de manera similar, con la diferencia que el resultado de su ejecución es almacenado en ET_1, alterando de esta manera el contenido del mismo. En la tercer columna, se encuentran aquellas operaciones que varían en el hecho de que el resultado es almacenado en ambos operandos. Por ejemplo, en el caso de la unión total ($ET_1 \uplus ET_2$) su variante es la unión total aumentativa simétrica ($ET_1 \overleftrightarrow{\uplus} ET_2$), la cual se comporta de manera similar, con la diferencia que el resultado de su ejecución es almacenado en ET_1 y en ET_2, alterando de esta manera el contenido de los mismos.

Operación original	**Altera el primer operando**	**Altera ambos operandos**
Unión total $(ET_1 \uplus ET_2)$	Unión total aumentativa $(ET_1 \overleftarrow{\uplus} ET_2)$	Unión total aumentativa simétrica $(ET_1 \overleftrightarrow{\uplus} ET_2)$
Unión sin réplicas $(ET_1 \bigcup ET_2)$	Unión sin réplicas aumentativa $(ET_1 \overleftarrow{\bigcup} ET_2)$	Unión sin réplicas aumentativa simétrica $(ET_1 \overleftrightarrow{\bigcup} ET_2)$
Intersección total $(ET_1 \pitchfork ET_2)$	Intersección total destructiva $(ET_1 \overleftarrow{\pitchfork} ET_2)$	Intersección total destructiva simétrica $(ET_1 \overleftrightarrow{\pitchfork} ET_2)$
Intersección sin réplicas $(ET_1 \bigcap ET_2)$	Intersección sin réplicas destructiva $(ET_1 \overleftarrow{\bigcap} ET_2)$	Intersección sin réplicas destructiva sim. $(ET_1 \overleftrightarrow{\bigcap} ET_2)$
Diferencia $(ET_1 - ET_2)$	Diferencia destructiva $(ET_1 \overleftarrow{-} ET_2)$	- No definida -
Diferencia simétrica $(ET_1 \sim ET_2)$	Diferencia simétrica destructiva $(ET_1 \overleftarrow{\sim} ET_2)$	- No definida -

Tabla 1: Operaciones que alteran el contenido de sus operandos

Como se puede observar en las operaciones diferencia y diferencia simétrica no hay variantes definidas que almacenen su resultado en los dos operandos. En todas las variantes mencionadas, se debe tener en cuenta el tipo de acceso a los espacios que tiene el agente que ejecuta la operación (acceso de lectura o escritura). Los tipos de accesos serán tenidos en cuenta en la siguiente sección.

Otras variantes a las uniones e intersecciones que se han mostrado, consisten en realizar *uniones múltiples* e *intersecciones múltiples*. Esto es, operaciones que acepten más de dos ETs como operandos.

2.1.6. Copiar espacio: Toma como entrada un ET y devuelve uno nuevo con las mismas tuplas. Esta operación es interesante cuando se necesita que otro grupo de agentes realice un procesamiento paralelo del conocimiento, además resulta de utilidad cuando es necesario hacer resguardos del

conocimiento. Esta operación es equivalente a hacer $ET_{vacio} \overleftarrow{\uplus} ET_1$, donde ET_{vacio} no tiene tuplas. De esta manera en ET_{vacio} quedarán todas las tuplas que están en ET_1.

2.1.7. Blanquear espacio: Toma como entrada un ET y elimina todas las tuplas que contiene. Resulta equivalente hacer $ET_1 \overleftarrow{-} ET_1$, es decir, si se hace la "diferencia destructiva" de un ET sobre sí mismo, se obtiene el mismo ET pero vacío.

Observación: Se podría objetar que todas las operaciones vistas en la sección 2.1 se pueden implementar con las primitivas provistas por el modelo Linda. Sin embargo, proveer estas operaciones brinda un nivel más alto de abstracción, lo cual facilita la tarea de implementación de agentes.

En el caso de que en un sistema multi-agente se permita la existencia de varios espacios de tuplas, se podría analizar diferentes alternativas para asociar agentes con ETs. Esto es, Linda sólo sugiere un único ET que es público a todos los agentes, pero ante la posibilidad de proveer varios ETs se podría discriminar el acceso a los mismos, ya que todos los ETs que figuren en el sistema podrían ser accedidos por todos los agentes.

3. VINCULACIÓN DE AGENTES A ESPACIOS DE TUPLAS

Ante la posibilidad de que puedan coexistir en un mismo sistema multi-agente varios ETs, se debe analizar los accesos a los mismos por parte de los agentes. Se pueden vislumbrar dos posibilidades, esto es, los espacios de tuplas pueden ser de acceso público o de acceso restringido.

Los **espacios de tuplas "públicos"** son aquellos en los cuales el acceso por parte de los agentes a los mismos, no es discriminado. Esto es, todo agente que intervenga en el sistema multi-agente tendrá acceso irrestricto al ET. Este tipo de acceso es el propuesto por el modelo Linda, ya que no sugiere ningún tipo de restricción de acceso a los ETs.

Este tipo de espacios podrían ser considerados como espacios de tuplas por *defecto*. Esto significa que los agentes que deseen acceder al espacio, no necesitan vincularse previamente al mismo. Por lo tanto, si algún agente provoca la ejecución de alguna primitiva, ocasionará un acceso al espacio, sin necesidad de vincularse al mismo con anterioridad.

Los espacios de tuplas públicos, tiene como *ventaja* que se desentiende de la problemática que genera administrar los accesos y vinculaciones de los agentes a los ETs, haciendo que el modelo sea simple. Además permite la existencia de ETs por defecto. Pero como *desventaja* puede observarse que el acceso público no brinda seguridad en las comunicaciones. Esto se debe, a que todos los agentes pueden acceder a toda la información que se encuentra en el área de memoria compartida. Ante esta situación, si un agente desea comunicarse con otro, y deja una tupla en el espacio, nada impide que un agente no deseado, retire la tupla ejecutando una operación *in*. De todas formas, esto puede solucionarse incorporando algún mecanismo de seguridad, como por ejemplo, agregando permisos de accesos. Además, si existen varios ETs públicos, todos los agentes del sistema tendrían acceso a todos los ETs, lo cual resulta redundante. Esto es, toda ejecución de una operación (efectuada por cualquier agente) afectará a todos los ETs, quedando el estado de los mismos con igual conocimiento.

Como se mencionó en el inicio de esta sección, otra opción es proveer **espacios de tuplas "restringidos"**, los cuales son deseables en el caso de que puedan existir varios ETs. Con esta característica, sólo tendrán acceso a los espacios aquellos agentes que estén asociados a los mismos. Para lograr este comportamiento en los ETs, se necesita administrar de alguna manera las vinculaciones de los agentes a los mismos. Esto es, los agentes deben vincularse a un ET para accederlo. Esta vinculación puede ser otorgada o no, dependiendo de la característica del ET.

Esta propuesta permite obtener ETs "privados", en el caso de que se restrinja el acceso sólo a un agente. De esta manera, se brinda la posibilidad de mantener un ET por agente. Esto resulta útil en

aquellas situaciones donde se requiere que los agentes puedan disponer de una base de conocimiento propia y exclusiva. En este caso, si los agentes desean comunicarse deberán hacerlo por medio de otro espacio de tuplas.

También pueden definirse ETs ”grupales”. Esto es, se podría restringir el acceso al ET a un grupo de agentes que estén relacionados de alguna manera. Por ejemplo, la relación puede estar determinada de la siguiente forma: todos los agentes del grupo trabajan en la misma tarea. En base a esta relación, los agentes que trabajan sobre el mismo objetivo comparten conocimiento, permitiendo así que la solución de un problema pueda ser dividida, para que cada agente ataque una parte y pueda contribuir a la solución global poniendo sus resultados en el espacio de tuplas.

Los ETs restringidos permiten compartir conocimiento de manera segura. Por ejemplo, si se desea que pares de agentes compartan conocimiento de manera segura, evitando el problema que agentes ajenos a la comunicación saquen tuplas, se crean ETs con acceso restringido a dos agentes. Como se puede observar, los diferentes tipos de espacios que se pueden generar a partir de esta propuesta, provoca que se puedan obtener esquemas con diferentes características para lograr cubrir un espectro aún mayor de necesidades a la hora de desarrollar un sistema multi-agente.

El costo de proveer ETs restringidos radica en que se deben administrar los accesos y vinculaciones de los agentes a los ETs, haciendo que el modelo sea más complejo. Además, las primitivas propuestas por el modelo Linda, ante cada ejecución deberán chequear si el agente que hizo correr la operación está vinculado al ET.

Por lo tanto, para que un agente pueda acceder a un espacio de tuplas con esta característica debe estar vinculado al mismo. El acceso obtenido por el agente, una vez vinculado al espacio puede ser de diferentes tipos. Los tipos de accesos serán analizados en la siguiente subsección.

3.1. Tipos de accesos

Ante la posibilidad de proveer espacios con acceso restringido, se debe evaluar el tipo de acceso que pueden obtener los agentes sobre los ETs. Los accesos pueden ser de “sólo lectura”, “sólo escritura” o de “lectura-escritura”. A continuación se muestran dos ejemplos, en el primero se muestran los tipos de accesos que deben adquirir los agentes cuando desean utilizar las operaciones propuestas por Linda. En el segundo se muestran los tipos de accesos que deben adquirir los agentes cuando desean utilizar las operaciones propuestas en este trabajo en la sección 2.1.

Ejemplo 10 ***Tipos de accesos para utilizar operaciones Linda****. Aquellos agentes que deseen utilizar primitivas del modelo Linda que alteran el contenido de los espacios, tales como* out, in, eval *e* inp*, deberán adquirir un acceso al ET, al menos de “sólo escritura”. Si desean utilizar alguna de las primitivas restantes (*rd *o* rdp*), deberán adquirir un acceso al ET, al menos de “sólo lectura”.*

Ejemplo 11 ***Tipos de accesos para utilizar operaciones que tienen como dominio y rango ETs****. Aquellos agentes que deseen utilizar las operaciones que alteran el contenido de los ETs, deberán adquirir un acceso al ET, al menos de “sólo escritura”. Estas operaciones son, blanquear espacio, y las destructivas y aumentativas propuestas como variantes de la unión, intersección, diferencia y diferencia simétrica. Si desean utilizar alguna de las operaciones restantes, deberán adquirir un acceso al ET, al menos de “sólo lectura”.*

En la sección siguiente se realizará el análisis del último aspecto del modelo Linda que este trabajo sugiere reformular (distribución del espacio de tuplas). Aquí se analizarán diferentes alternativas para asociar espacios de tuplas con áreas de memoria.

4. ALTERNATIVAS PARA ASOCIAR ETS CON ÁREAS DE MEMORIA

De acuerdo a como lo define el modelo Linda, un ET es un concepto abstracto (abstracción de memoria compartida); por lo tanto, el mismo puede residir en diferentes áreas de memoria o sólo en una. Asimismo, un área de memoria puede albergar varios ETs diferentes. Por lo tanto, aunque no haya una definición explícita en cuanto a la implementación del ET, sería deseable permitir en un sistema multi-agente la existencia de varios ETs con diferentes características en cuanto a su asignación en áreas de memorias. Las diferentes alternativas que se sugieren son:

- Alternativa 1: Un ET residiendo en un único área de memoria.
- Alternativa 2: Un ET residiendo en más de un área de memoria.
  - Alternativa 2.1: Con réplicas.
  - Alternativa 2.2: Fragmentado.
  - Alternativa 2.3: Fragmentado con réplicas.
- Alternativa 3: Varios ETs residiendo en un único área de memoria.

Observación: Se podría objetar que hace falta una alternativa que haga referencia a la posibilidad de que existan varios espacios de tuplas en más de un área de memoria. Sin embargo, esta alternativa resulta de una combinación de tener varios espacios (como se sugiere en [8] y [7]) con las características de las alternativas 1 y 2 enunciadas arriba.

En la figura 1 se muestra una representación gráfica de cada alternativa (aparecen de izquierda a derecha en el orden en que fueron enunciadas). Aquí un rectángulo punteado representa a un espacio de tuplas, un círculo representa a un agente y una línea que conecta un círculo con un rectángulo punteado representa un vínculo entre un agente y un ET haciendo referencia a que dicho agente puede acceder a dicho espacio. Un rectángulo de línea continua representa un área de memoria residiendo físicamente en un nodo. Notar que cuando un rectángulo punteado (ET) encierra a un rectángulo de línea llena (área de memoria), o viceversa, significa que el ET está residiendo en esa área de memoria.

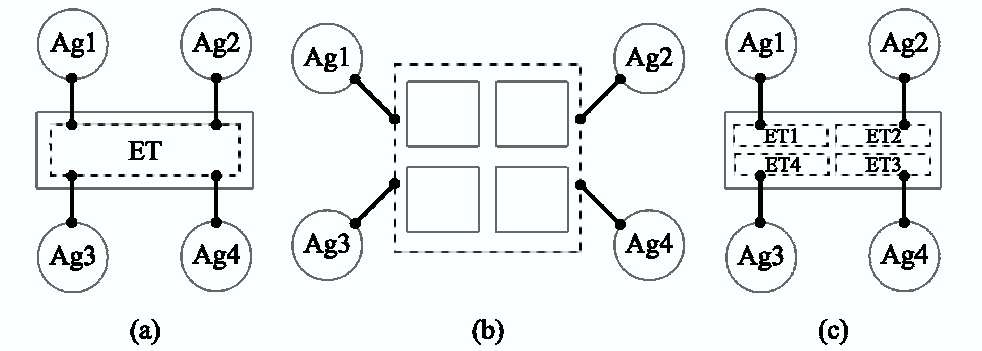

Figura 1: Alternativas para asociar espacios de tuplas con áreas de memoria

En cualquiera de estas alternativas, los agentes que están vinculados al ET tienen acceso a la misma información, es decir, “todos ven lo mismo”. La diferencia que existe entre estas alternativas radica en la distribución del ET. En resumen, la visión lógica de los agentes es la misma más allá de la distribución del espacio.

Los agentes que ejecuten cada operación Linda, sobre espacios de tuplas de cualquier alternativa, perciben el mismo comportamiento en todas ellas, es decir, que la implementación de las operaciones es transparente para los agentes que las utilizan. De esta manera, se logra que los agentes compartan conocimiento desentendiéndose de cómo está implementado el ET. A continuación se describen cada una de las alternativas antes mencionadas.

4.1. Alternativa 1: Un ET residiendo en un único área de memoria

La primer alternativa, la cual se muestra en la figura 1(a), consiste de disponer de ETs que sólo pueden residir en un único área de memoria. Esta propuesta proporciona simpleza y eficiencia de, en cuanto a la semántica operacional de sus primitivas, y además brinda la posibilidad de que todos los agentes que comparten un ET, se comuniquen sólo por medio del ET (memoria compartida), sin necesidad de recurrir a otra técnica, como por ejemplo, pasaje de mensajes.

Como desventaja se puede observar que esta alternativa sugiere un esquema centralizado (un SMA es inherentemente distribuido) que expone la falta de robustez. Además, un espacio de tuplas con estas características representa un problema en cuanto a espacio, si el sistema multi-agente es muy grande y muchos agentes lo acceden.

4.2. Alternativa 2: Un ET residiendo en más de un área de memoria

Ante las desventajas enunciadas en la alternativa 1, surge una segunda alternativa que es mostrada en la figura 1(b), que propone que un ET resida (en forma remotamente distribuida) en más de un área de memoria. En esta figura, cada rectángulo de línea llena representa las distintas áreas de memorias ubicadas físicamente en diferentes nodos, en las cuales reside el ET que está representado por el recuadro punteado que los contiene. Los ETs con estas características pueden tener diferentes filosofías de almacenamiento de tuplas, las cuales se asemejan a las técnicas de almacenamiento de datos vistas en base de datos distribuidas [12]. Esto es:

► **Alternativa 2.1 "Con réplicas":** En este caso el espacio de tuplas está residiendo en áreas de memorias remotamente distribuidas, con una filosofía de replicación de tuplas en cada nodo donde reside el espacio. Para lograr esto, hay que mantener un administrador de bloqueos y réplicas. Este administrador se encargará de bloquear los accesos al espacio de tuplas mientras se está actualizando el mismo. Además deberá encargarse de actualizar de inmediato todas las réplicas de la tupla modificada. Este administrador deberá mantener comunicación por medio de otro ET con aquellos nodos en los que reside el ET.

Ante una actualización sobre un ET de estas características, las acciones a tomar por la misma, deben ejecutarse en todos los nodos donde está residiendo el espacio de tuplas en cuestión. Esto es, si un agente ejecuta la primitiva *out(t)*, y el mismo está vinculado a un espacio como el que se ve en la figura 1(b), la tupla *t* debe ser agregada en todos los nodos.

Esta alternativa es deseada, cuando se requiere un sistema robusto. Esto se debe a que proporciona "*disponibilidad*", ya que si uno de los nodos que contiene una tupla *t* falla, entonces se la puede buscar en otro nodo. Un ET con estas características "*incrementa el paralelismo*", ya que en el caso en que la mayor parte de los accesos a la tupla *t* resulten sólo en la lectura de la tupla, varios nodos podrán procesar consultas que involucren a *t* en simultáneo. Pero tiene como desventaja el "*incremento del overhead en actualizaciones*", ya que hay que mantener un administrador que asegure que todas las réplicas de la tupla *t* sean consistentes. Cada actualización sobre la tupla *t* debe ser propagada a los nodos conteniendo las respectivas réplicas.

► **Alternativa 2.2 "Fragmentado":** Una variante del caso anterior consiste de un espacio de tuplas que reside en varias áreas de memorias remotamente distribuidas y, a diferencia del caso anterior, aquí no se replican los datos en los distintos nodos, sino que en cada uno de ellos reside una región del espacio de tuplas a representar. Esto es, en cada nodo se podrá encontrar información que es local al mismo, y que es un subconjunto de la información perteneciente al espacio de tuplas en general. El total de la información contenida en el espacio de tuplas resulta de la "*unión total*" de toda la información contenida en los diferentes nodos donde se alberga dicho espacio.

Se debe notar que es necesario que exista un administrador para cada ET con esta característica.

Este administrador se encarga de localizar las tuplas en los distintos nodos donde reside el espacio. Puede haber un único administrador, el cual expone un esquema centralizado, o varios administradores. Este último aspecto no será analizado en este trabajo ya que escapa a los objetivos del mismo. Este administrador deberá mantener comunicación por medio de otro ET con aquellos nodos en los que reside el ET.

Ante una actualización sobre un ET de estas características, las acciones a tomar por la misma, deben ejecutarse en uno de los nodos donde está residiendo el espacio de tuplas en cuestión. La elección del nodo en el cual tomará acción dicha operación, será fijada por alguna política preestablecida. Por ejemplo, que todos los nodos donde reside el ET preserven paridad en cuanto a la cantidad de tuplas, evitando de esta manera la sobrecarga de algún nodo, lo que provoca que se mejoren las búsquedas. Otro ejemplo de una política a seguir, consiste en que la tupla sea agregada en el nodo donde está corriendo el agente, si es que este está corriendo en un nodo donde reside el ET.

Un esquema con esta característica resulta de utilidad cuando el tamaño del ET es muy grande y se necesitan varias áreas de memoria para albergar al mismo. Pero tiene como desventaja que por cada ET con esta característica, se debe mantener uno o más administradores de tuplas.

► **Alternativa 2.3 "Fragmentado con réplicas":** Esta es una combinación de las alternativas 2.1 y 2.2 donde, un espacio de tuplas es particionado en varios fragmentos y el sistema mantiene varias copias de estos. Aquí se puede observar que se mantiene las ventajas y desventajas de ambas alternativas, proporcionando mayor flexibilidad al sistema.

4.3. Alternativa 3: Varios ETs residiendo en un único área de memoria

La última alternativa puede verse representada gráficamente en la figura 1(c). Ésta consiste de varios ETs residiendo en un mismo área de memoria. Un esquema de estas características mantiene varios ETs centralizados en un mismo nodo. Esto es, brinda la posibilidad de centralizar los ETs en un mismo área de memoria permitiendo así coexistir en un mismo nodo varios ETs consumiendo una menor cantidad de recursos. Por lo tanto, resulta útil en el caso de que un sistema multi-agente requiera, para su ejecución, varios ETs y no se cuente con áreas de memorias remotas como para albergar cada ET (falta de recursos). Sin embargo, se puede observar que si el nodo donde se centralizan los ETs se cae, se pierde toda la información, lo cual no sucede si dichos espacios se encuentran distribuidos. Esto evidencia lo poco robusta que resulta esta alternativa.

5. TRABAJO RELACIONADO, CONCLUSIONES Y TRABAJO A FUTURO

Existen en la literatura varias extensiones al modelo Linda que utilizan multiples espacios de tuplas [8, 1, 9, 10]. En dichos trabajos estos multiples espacios se utilizan para obtener mejoras sobre aspectos particulares que conciernen a la programación distribuida y paralela. A diferencia de ellos, el objetivo de este trabajo consiste en adaptar el modelo de espacios de tuplas a un sistema multi-agente en donde participan agentes con conocimiento. Esto permite la definición de operaciones que apuntan a la manipulación masiva del conocimiento de los agentes. Estas operaciones se asemejan a las del álgebra realcional [6] (en las versiones que no admiten tuplas replicadas) y a las operaciones multi-set del álgebra relacional extendida [7] (en las versiones que admiten tuplas replicadas). Sin embargo, difieren en el hecho de que los ETs admiten tuplas sin formato, y las relaciones del álgebra relacional sólo admiten tuplas con formato, es decir, deben tener la misma aridad y los atributos de las mismas deben mantener un orden y ser tipados.

En este trabajo se analizó el intercambio de conocimiento en sistemas multi-agente (SMA) utilizando áreas de conocimiento compartidas. Este análisis se basó principalmente en los fundamentos

sugeridos por el modelo Linda, debido a que este es un formalismo simple y eficiente, que puede brindar comunicación y sincronización en un sistema multi-agente a través de memoria compartida. En base a Linda se planteó la idea de espacios de tuplas (ETs) distribuidos, brindando la posibilidad de coexistir diferentes ETs con características distintas, residiendo en áreas de memorias remotas. Esto involucró reformular algunos aspectos del modelo Linda con el objetivo de lograr adaptarlo a un entorno distribuido en un SMA con agentes con conocimiento. Los aspectos que se reformularon fueron tres. El primer aspecto propone la existencia de más de un ET en un SMA (como se sugiere en [8, 10]). A partir de este aspecto se definieron nuevas operaciones sobre ETs que tienen como dominio y rango ETs, lo que provocó que el modelo sea aún más interesante y poderoso. El segundo aspecto que se consideró propone que los accesos a los ETs, por parte de los agentes, sean restringidos (Linda sugiere ETs públicos). El tercer aspecto que se trató, consiste en analizar diferentes alternativas para asociar ETs a áreas de memorias. Aquí se proponen, entre otras alternativas, ETs que residen en áreas de memorias remotamente distribuidas. Como trabajo a futuro se planea definir diferentes propiedades sobre las operaciones propuestas en la sección 2.1. Además, se realizará un análisis sobre el impacto que provocan las diferentes alternativas de asociación de ETs con áreas de memoria, en las diferentes operaciones definidas por Linda y las propuestas en este trabajo.

REFERENCIAS

[1] P. Butcher, A. Wood, and M. Atkins. Global synchronisation in Linda. *Concurrency: Practice and Experience*, 6(6):505–516, 1994.

[2] N. Carriero and D. Gelernter. Linda in context. *Comm. of the ACM*, 32(4):444–458, 1989.

[3] N. Carriero and D. Gelernter. Capitulo 3: Linda. In *How to write parallel programs*, 1992.

[4] Paolo Ciancarini. Coordination Languages as Software Integrators of Multiagent architectures. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 21(4):314–335, 1995.

[5] Paolo Ciancarini. Coordinating Multiagent Applications on the WWW: A Reference Architecture. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 24(5):362–374, May 1998.

[6] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[7] Paul W. P. J. Grefen and Rolf A. de By. A multi-set extended relational algebra - a formal approach to a practical issue. In *ICDE*, pages 80–88, 1994.

[8] K.K. Jensen. Toward a multiple tuple spaces. In *PhD thesis, Aalbrog University, Departament of Mathematics and Computer Science*, 1993.

[9] A. Rowstron, A. Douglas, and A. Wood. Copycollect: A new primitive for the linda model, 1996.

[10] A. Rowstron and A. Wood. Bonita: a set of tuple space primitives for distributed coordination. In *Proc. HICSS30, Sw Track*, pages 379–388, Hawaii, 1997. IEEE Computer Society Press.

[11] Robert W. Sebesta. Chapter 1: Preliminaries. In *Concepts of Programaing Languages*, 1996.

[12] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. Chapter 18: Distributed database. In *Database System Concepts*, 1999.