ALGORITMO PARA LA REDUCCIÓN DE BASES DE DATOS ORIENTADAS A GRAFOS

Roxana Laurel Rodríguez
rlaurelr@doc.emi.edu.bo
ESCUELA MILITAR DE INGENIERIA
Manuel Vargas Guzmán
mvargasg@univalle.edu
UNIVERSIDAD PRIVADA DEL VALLE

La Paz, Bolivia

ALGORITHM FOR THE REDUCTION OF GRAPH-ORIENTED DATABASES

Abstract. - En Internet, las consultas o queries se las realizan en bases de datos orientadas a grafos de gran tamaño. La optimización de consultas en este tipo de bases de datos es un campo importante de investigación, ya que estas pueden llegar a tardar mucho dependiendo del tamaño del grafo. Una de las soluciones es realizar la consulta en un grafo equivalente, pero de menor tamaño. En este artículo planteamos una extensión a dicha solución que consiste en implementar algoritmos de equivalencias de procesos para poder "reducir" el grafo adicionando la semántica de la equivalencia débil, la cual nos permite comparar grafos con un cierto grado de equivalencia. Esta extensión a los algoritmos tradicionales nos permite hacer una comparación más "flexible" entre dos grafos, el resultado es que podemos eliminar ciertos parámetros que el usuario no usara´ comúnmente en sus consultas. Implementando este tipo de comparaciones entre grafos podemos reducir aún más las bases de datos que utilizando los algoritmos tradicionales de equivalencias.

Abstract.- On the Internet, queries or queries are made in databases oriented to large graphs. The optimization of queries in this type of databases is an important field of research, since these can take a long time depending on the size of the graph. One of the solutions is to perform the query in an equivalent graph, but smaller in size. In this article we propose an extension to this solution that consists in implementing algorithms of process equivalences in order to "reduce" the graph by adding the semantics of weak equivalence, which allows us to compare graphs with a certain degree of equivalence. This extension to

the traditional algorithms allows us to make a more “flexible” comparison between two graphs, the result is that we can eliminate certain parameters that the user will not commonly use in their queries. By implementing this type of comparisons between graphs we can reduce the databases even more than using the traditional equivalence algorithms.

Palabras Claves—flexible, bimulacion, reducción, trazos.

Keywords - flexible, bimulation, reduction, strokes.

I. INTRODUCCIÓN

En la actualidad, el Internet es la fuente de información más importante que existe, pero al mismo tiempo es muy difícil filtrar dicha información debido a la gran cantidad de datos que hay. Esta información está organizada en una estructura de grafos [1], de tal forma que los datos puedan crecer y conectarse en una manera más flexible. Sin embargo, consultar información en una base de datos orientada a grafos puede ser complicada y sobre todo muy lenta debido al gran tamaño que pueden llegar a tener estas. Es posible reducir el tiempo y el espacio utilizados en consultar a una base de datos, si ejecutamos la consulta en otra base de datos “equivalente” pero más pequeña (es decir, un grafo equivalente más pequeño). Esto significa que una de las soluciones a este problema es la reducción del grafo utilizando equivalencias, en [2] los autores describen dos algoritmos para reducir bases de datos orientadas a grafos utilizando equivalencias de procesos: la equivalencia de trazos y la bisimulaci´on.

Los algoritmos usados para comparar dos grafos nos devuelven dos resultados posibles: verdad o falso, es decir, o bien el grafo es equivalente o no. En este artículo presentamos una extensión de estos algoritmos tradicionales para la reducción de bases de datos orientadas a grafos, para esto utilizamos una comparación de grafos tomando en cuenta el “grado de similitud" entre estos. Un ejemplo de este enfoque, realizado por [3], compara procesos utilizando equivalencias, pero la respuesta no solo es verdad o falso, también puede ser algo entre medio. Esta idea surge porque cuando nosotros realizamos consultas en Internet a una base de datos en particular, utilizamos solamente algunos parámetros. Por ejemplo, si estamos buscando una determinada película, utilizaremos más frecuentemente parámetros como título, año, director o actor principal, pero menos frecuente lugar de filmación, editor de efectos especiales

u otros. Por lo tanto, no necesitamos toda la información de los grafos, solamente la \más relevante" (según algún criterio) y podemos reducir aún más los grafos si utilizamos una escala de lógica difusa en vez de booleana para la comparación de estos.

El contenido de este art culo está dividido en las siguientes partes: a continuación, introducimos la notación y conceptos importantes utilizados en el documento: equivalencias y web semántica. Luego presentamos nuestra solución y el algoritmo para la reducción de base de datos orientadas a grafos. Finalmente hablamos de las conclusiones y el trabajo que queda pendiente para el futuro.

II. PRELIMINARES

a. Equivalencias

En matemáticas una relación de equivalencia es una relación binaria que es reflexiva, simétrica y transitiva. Por ejemplo, la relación “es igual a", para cualquier objeto α, β y γ: α = α es reflexiva; si α = β entonces β = α es simétrica; y si α = β y β = γ entonces = es transitiva.

En teoría de procesos, un proceso es el comportamiento de un sistema (un sistema puede ser una máquina, una red, una base de datos, etc.). La equivalencia de procesos es una actividad que consiste en comparar el comportamiento de dos procesos según un criterio de nido, por ejemplo, es muy utilizado en el área de “verificación de sistemas" (comparar el comportamiento actual de un sistema con el comportamiento correcto del sistema). También se usa para reducir el tamaño de un sistema, comparando que un sistema es equivalente a otro más pequeño, etc.

El criterio de nido constituye la semántica de la equivalencia aplicada a teoría de procesos, es decir, que aspectos del comportamiento del sistema se tomaran en cuenta. van Gabbeek [4] nos presenta una clasificación de las equivalencias en un orden jerárquico de inclusión, donde la semántica de la equivalencia más amplia es la equivalencia de trazos y la más específica es la bisimulacion (que incluye a todas las otras semánticas).

i. Equivalencia de trazos

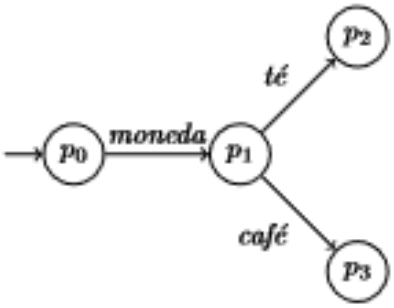
Notación Denotamos estados o procesos con letras p, q, …, acciones con letras a, b, c, … y un conjunto de acciones con letras mayúsculas A, B, … Si A = {a, b} es un conjunto de acciones, denotamos A*= {ϵ, a, b, aa, ab, bb, aaa, …} como el conjunto de todos los “trazos" (secuencias de acciones) que se pueden formar a

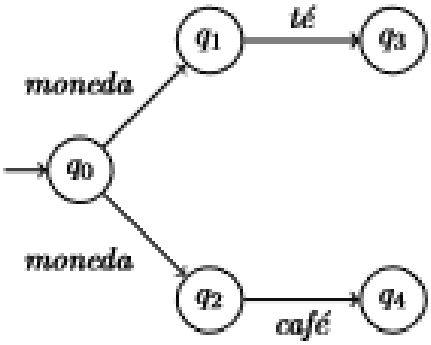
partir de A, donde ϵ representa una acción vacía.

Para modelar un sistema utilizaremos un sistema de transición de estados. Un sistema de transición de estados es una tupla (S, A, Δ), donde S es un conjunto de estados o procesos, A es un conjunto de acciones y Δ ⊆ (S ×A×S) es un conjunto de transiciones; para (p, a, q) ∈ Δ escribimos $p \xrightarrow{a} q$.

Un trazo finito t ∈ A* de un proceso p_0 es una secuencia finita de acciones a_1, ..., a_n tal que existen procesos p_0, ..., p_n con $p_{i-1} \xrightarrow{a_i} p_i$ para todos los i = 1, ..., n.
Escribimos $p_0 \xrightarrow{t} p_n$ si existe un trazo t de p0 que termina en pn.

Ejemplo 1 Un ejemplo simple es una maquina dispensadora de té y café que acepta una moneda y sirve una de esas bebidas.





Tenemos dos procesos p_0 y q_0 de los cuales podemos ejecutar las secuencias:
(moneda-te) o (moneda-café).

Definición 1 Sea A un conjunto de acciones

$$T(p) := \{t \in A^* \mid \exists p'.p \xrightarrow{t} p'\}$$

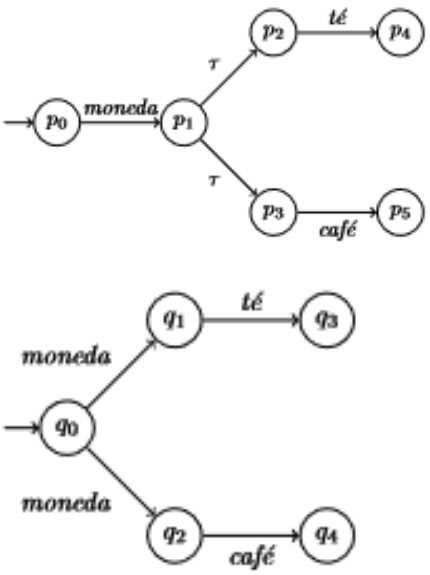
el conjunto de todos los trazos que puede ejecutar el proceso p. Dos procesos p y q son equivalentes en trazos nitos, si T (p) = T (q).
La definición anterior nos dice que, si dos procesos tienen el mismo conjunto de trazos o secuencias, entonces ambos procesos son equivalentes en la semántica de trazos nitos, del ejemplo anterior tenemos las acciones Acciones= {moneda, te, café} y el conjunto de trazos nitos de ambos procesos son:
T(p0) = T(q0) = {ϵ, moneda, moneda-té, moneda-café}.

2.1.2 Equivalencia débil de trazos

La equivalencia “débil" de trazos [8] es una variación de la equivalencia finita de trazos que se usa para

comparar sistemas que tienen comportamientos internos, dichos comportamientos son representados por acciones escondidas que denotamos con la letra ϒ.

Ejemplo 2 La máquina dispensadora de té y café que acepta una moneda y sirve una de esas bebidas con acciones escondidas.



Consideraremos un trazo nito débil entre dos procesos a un trazo que puede contener cualquier cantidad de acciones ϒ entre medio. Escribimos ϒ* a una secuencia que contiene cero o más acciones ϒ.

Definición 2 Un trazo finito débil w ∈ (A−{τ}) ∗ de un proceso p0 es una secuencia finita de acciones a1, ..., an tal que existen procesos p0, ..., pn con para todos los i = 1,...,n.

Escribimos $p_0 \xrightarrow{w} p_n$ si existe un trazo débil w de pυ que termina en pn.

Definición 3 Sea A un conjunto de acciones y W(p): = {w ∈ (A−{τ})∗ | ∃p'. $p \xrightarrow{w} p'$} el conjunto de todos los trazos débiles que puede ejecutar el pro-ceso p. Dos procesos p y q son débilmente equivalentes en trazos finito, si W(p) = W(q).

De manera similar a la equivalencia de trazos, si dos procesos tienen el mismo conjunto de trazos débiles, entonces son débilmente equivalentes en trazos finito, en el último ejemplo tenemos las acciones Acciones = {τ, moneda, té, café } y el conjunto de trazos débiles finito de ambos procesos son: W(p0) = W(q0) = {, moneda, moneda-té, moneda-café}.

2.2 Bases de datos orientadas a grafos

La World Wide Web ha cambiado la forma en que las personas nos comunicamos entre s y la forma en que llevamos a cabo muchas de las actividades cotidianas. En la actualidad, su uso predominante es el procesamiento de la información y las aplicaciones t picas son los sistemas de bases de datos, el procesamiento de texto y los juegos. La forma en la que la información se encuentra almacenada en la WWW se la puede

modelar utilizando una estructura de grafos. Un claro ejemplo son las especificaciones RDF [5] utilizadas en la web semántica [6].

2.2.1. La web semántica

La WWW ha convertido el conocimiento en algo colectivo, todo puede ser distribuido por todo el mundo sin mucho esfuerzo, lo que muestra muchas ventajas, pero también presenta algunos inconvenientes, como ser la recuperación, extracción y/o combinación de la información. Aunque el usuario tenga éxito en dicha acción, es la persona quien se debe encargar de examinar la información que está buscando. En resumen, el trabajo duro de realizar una búsqueda queda en manos de las personas y no así de las maquinas. Es por esto que se vio la necesidad de agregar semántica a la web utilizando ontologías para especificar datos con significado. Una ontología define las condiciones básicas y relaciones entre objetos de un dominio en particular. En la web semántica las ontologías proveen un vocabulario de términos, donde el significado (semántica) de tales términos esta formalmente especificado, lo que permite a la maquina "entender" y satisfacer las peticiones de las personas.

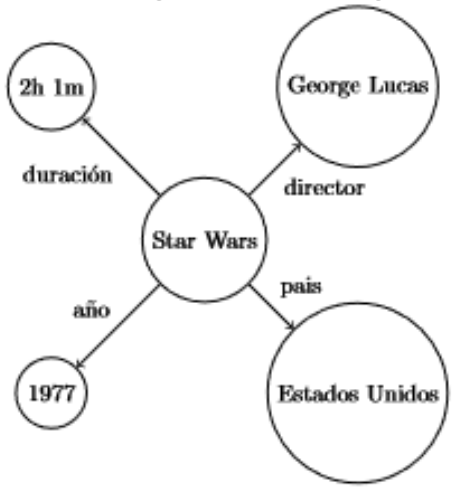
2.2.2. RDF

XML es un metalenguaje universal para definir datos estructurados, entre sus principales aplicaciones está el intercambio de contenidos de bases de datos, pero la desventaja es que no provee semántica. En este sentido se crea el marco de descripción de ecursos (Resource Description Framework - RDF), que es un lenguaje ontológico que combina XML + semántica para representar metadatos y describir formalmente el "significado" de la información de tal forma que la maquina pueda procesarla.

Ejemplo 3 Si queremos representar la descripción de una película en RDF el código se vería como sigue:

```
<rdf:Descripción
rdf:about="http://www.peliculas.com/dvd/Star Wars">
<dvd:director>George Lucas</dvd:director>
<dvd:pais>Estados Unidos</dvd:pais>
<dvd:duración>2h 1m</dvd:duración>
<dvd:año>1977</dvd:año>
</rdf:Descripción>
```

Esta información se la puede representar también utilizando una estructura de grafos como sigue:



Donde los nodos representan los objetos o entidades que deseamos almacenar, en RDF son los llamados recursos. y los arcos son las relaciones entre los recursos. Para nuestro análisis representamos esta información en forma de tuplas, es decir:

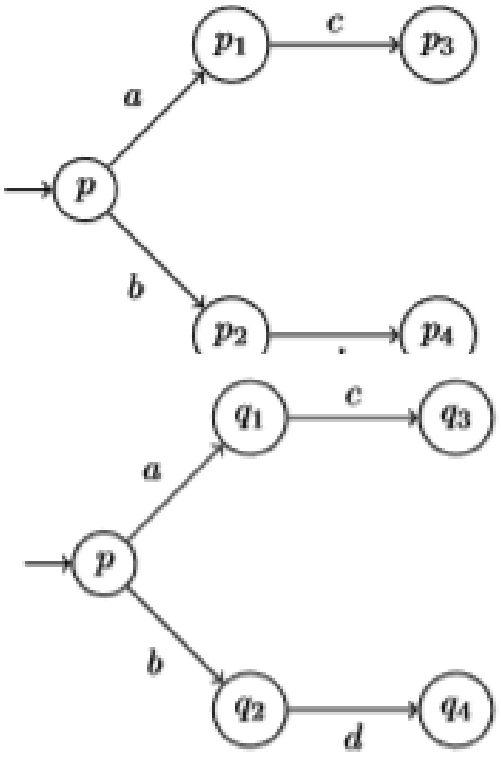
(“Star Wars”, “director”, “George Lucas”)
(“Star Wars”, “pais”, “Estados Unidos”)
(“Star Wars”, “duración”, “2h 1m”)
(“Star Wars”, “año”, “1977”)

Para simplificar, en las siguientes secciones utilizaremos las letras p, q, ... para representar las entidades o recursos y las letras a, b, c, ... para denotar las relaciones.

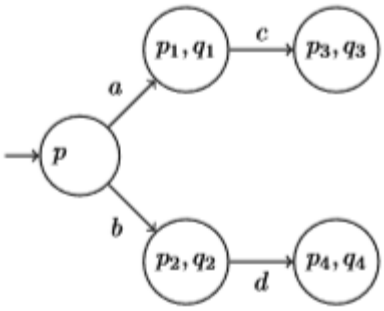
III. COMPARACIÓN DE GRAFOS POR GRADO DE EQUIVALENCIA

Cuando un usuario realiza una consulta a una base de datos, si bien necesita una vista con mucha o toda la información, el usuario no utiliza todos estos datos para realizar la consulta en sí. Por ejemplo, una película puede tener cientos de atributos, desde el título hasta los premios ganados; sin embargo, los parámetros que un usuario “estándar” utiliza para buscar una determinada película suelen ser pocos (título, actor principal, género, etc.). Es decir, raras veces se tendrá una consulta a la base de datos utilizando parámetros no comunes (por ejemplo, buscar por la duración de la película). Las consultas en una base de datos orientada a grafos son más rápidas si se ejecutan en grafos equivalentes más pequeños, ya que se reduce el espacio de búsqueda. En esta sección utilizaremos el concepto de la equivalencia débil de trazos para reducir una base de datos tomando en cuenta la importancia o relevancia de la información a ser comparada. De esta manera podemos omitir ciertos parámetros, por ejemplo, los menos utilizados al realizar consultas.

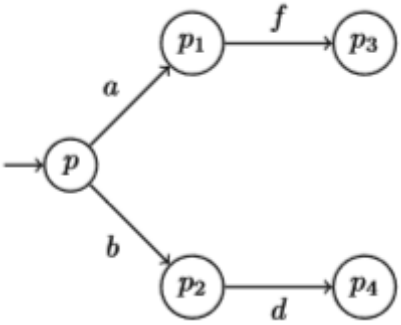
Primeramente, veamos un ejemplo de reducción utilizando el concepto de equivalencia de trazos, se tiene la siguiente base de datos:

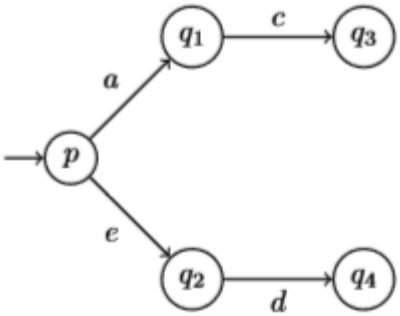


Aplicando la equivalencia de trazos podemos reducir el grafo a uno de menor tamaño. Para analizar la reducción de la base de datos observamos que nodo se repite, es decir p, el cual está unido a los nodos p_1 y q_1 con la relación a; y también está unido a los nodos p_2 y q_2 con la relación b. Luego podemos ver que los nodos p_1 y q_1 tienen una relación c hacia los nodos p_3 y q_3, respectivamente. Y finalmente, el nodo p_2 esta enlazado con el nodo p_4 con la relación d, que también conecta los nodos q_2 con q_4. El conjunto de trazos para el nodo que se repite es T(p) = {ϵ, a, b, ac, bd} y el grafo reducido es

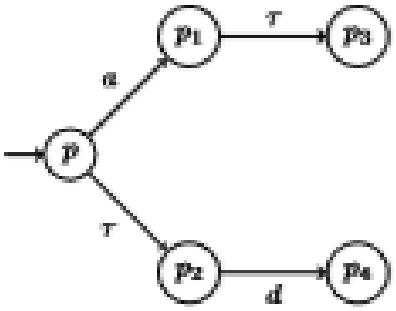


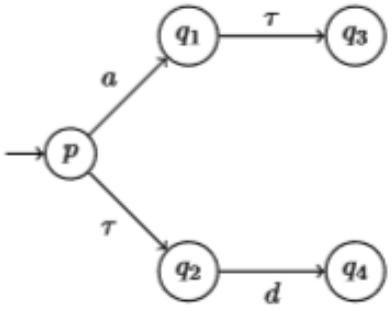
Ahora analizamos la base de datos con algunos cambios en la información y la modelamos de la siguiente forma:





El algoritmo de equivalencia de trazos ya no podrá reducirla debido a que la base de datos tiene las relaciones distintas f ≠c y b ≠ e y por consiguiente los trazos serán también distintos. Pero si determinamos que las relaciones b, c, f y e no son relevantes para una consulta, podemos cambiar la base de datos y reemplazar dichas relaciones por la relación τ como sigue a continuación:





Después del cambio los trazos débiles que p puede realizar son W (p) = {ϵ, a, d} y podemos ver que los grafos vuelven a ser equivalentes, aunque en este caso podemos utilizar una escala de lógica difusa y decir por ejemplo que los grafos son “muy" equivalentes o que los grafos son equivalentes en un cierto grado de similitud.

a. Algoritmo para la reducción de grafos

En esta sección presentamos el algoritmo para poder reducir una base de datos orientada a grafos que tiene como input dos grafos que comparten un nodo en común a partir del cual se simplificar la base de datos.

A continuación, se describe brevemente el algoritmo: en las líneas 2 y 3 se inicializa las dos variables que conformaran el grafo reducido, el conjunto de nodos S y el conjunto de transiciones Δ. Posteriormente entramos en la iteración principal del algoritmo que consiste en encontrar todos los trazos débiles w_1 y w_2, de ambos grafos, tal que los trazos sean equivalentes. Si son iguales, entonces entramos en otra iteración (línea 7) donde construimos el grafo reducido utilizando cualquiera de los trazos w_1 o w_2. Finalmente, observamos que en la línea 14 devolvemos el nuevo grafo reducido que consiste en los conjuntos S y Δ, que hemos ido construyendo en las líneas 9 y 10, y el conjunto de acciones A_1 o A_2 denotado como A.

1: **procedure** REDUCIR(G_1, G_2) ▷ donde $G_i = (S_i, A_i, \Delta_i)$ para $i \in \{1,2\}$
2: $S := \{\}$;
3: $\Delta := \{\}$;
4: **for all** $w_1, w_2 \in W(p_1), W(q_1)$ **do** ▷ donde $p_1 \in S_1, q_1 \in S_2$ y $p_1 = q_1$
5: **if** $w_1 = w_2$ **then** ▷ si son equivalentes creamos un nuevo grafo
6: $w := (w_1 \vee w_2)$
7: **while** w **is not empty do**
8: $a := first(w)$ ▷ toma el primer elemento de w y devuelve el resto
9: $\Delta \leftarrow \{(p_1, q_1) \xrightarrow{a} (p_2, q_2)\}$; ▷ donde $(p_i, a, q_i) \in \Delta_i$ para $i \in \{1,2\}$
10: $S \leftarrow \{(p_1, q_1), (p_2, q_2)\}$;
11: **end while**
12: **end if**
13: **end for**
14: **return** (S, A, Δ); ▷ devuelve el grafo reducido con $A = A_1 \vee A_2$
15: **end procedure**

IV. CONCLUSIONES Y TRABAJO FUTURO

Los algoritmos más importantes para la reducción de grafos se basan en "equivalencia de procesos" [4, 2], hemos propuesto un algoritmo utilizando la semántica de equivalencia débil de trazos, para poder resumir una base de datos sin tomar en cuenta los parámetros poco relevantes en consultas y de esta manera poder reducir el grafo un poco más. Cabe recalcar que esta extensión se aplica de la misma manera a otros algoritmos de equivalencia usados también para la reducción de grafos, como por ejemplo la bisimulación [4] se extiende a la bisimulación débil [7] de procesos. Con relación al trabajo futuro, se deberá realizar un análisis exhaustivo de la complejidad del algoritmo. Si bien el problema de buscar todas los "trazos" en un grafo esta en P-SPACE [8], en un trabajo futuro se debería considerar además la posibilidad de reducir de alguna manera el espacio y/o tiempo de ejecución utilizando técnicas para la reducción de algoritmos.

Bibliografía

[1] Marc Gyssens, Jan Paredaens, and Dirk Van Gucht. A graph-oriented object model for database end-user interfaces. SIGMOD Rec., 19(2):24–33, May 1990.

[2] Ala' Hawash, Anton Deik, Bilal Farraj, and Mustafa Jarrar. Towards query optimization for the data web: Disk-based algorithms: Trace equivalence and bisimilarity. In Proceedings of the 1st International Conference on Intelligent Semantic Web-Services and Applications, ISWSA '10, pages 17:1– 17:7, New York, NY, USA, 2010. ACM.

[3] W. M. P. van der Aalst, A. K. Alves de Medeiros, and A. J. M. M. Weijters. Process equivalence: Comparing two process models based on observed behavior. In Schahram Dustdar, Jos´e Luiz Fiadeiro, and Amit P. Sheth, editors, Business Process Management, pages 129–144, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[4] R. J. van Glabbeek. The linear time-branching time spectrum (extended abstract). In CONCUR, pages 278–297, 1990.

[5] A. G´omez-P´erez and O. Corcho. Ontology specification languages for the semantic web. IEEE Intelligent Systems, 17:54–60, 01 2002.

[6] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. Scientific American, 284(5):34–43, 2001.

[7] Anna Philippou, Insup Lee, and Oleg Sokolsky. Weak bisimulation for probabilistic systems. In Catuscia Palamidessi, editor, CONCUR 2000 — Concurrency Theory, pages 334–349, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[8] Alexander Rabinovich. Complexity of equivalence problems for concurrent systems of finite agents, 1995.

RECEPCION: 15/07/2019
APROBACION: 21/08//2019