**OEG Publication**

Ramos JA, Fernández-López M, Gómez-Pérez A

# *Algoritmo de agregación de* **mappings** *basado en reglas de selección*

XIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA-TTIA 09)
November 11th-13th, 2009.
Pages: 10
Sevilla, Spain.
ISBN: 978-84-692-6424-9.

# Algoritmo de agregación de *mappings* basado en reglas de selección

J. A. Ramos[1], M. Fernández-López[2], A. Gómez-Pérez[1]

[1] Ontology Engineering Group – Universidad Politécnica de Madrid
Avda. Montepríncipe s/n – 28660 – Boadilla del Monte, Madrid, Spain
{jarg, asun}@fi.upm.es

[2] Escuela Politécnica Superior. Universidad San Pablo CEU
Ctra. de Boadilla del Monte km 5,300
28668 - Boadilla del Monte, Madrid, Spain
mfernandez.eps@ceu.es

**Resumen.** Los sistemas de integración de información resuelven las diferencias entre las fuentes, en la mayoría de los casos, mediante la creación de *mappings*, puentes semánticos entre los elementos de las fuentes. Hasta ahora se han propuesto técnicas para generar un conjunto de *mappings* para cada par de elementos de las fuentes a integrar, sin embargo, hasta la publicación del presente trabajo, no se disponía de un algoritmo de agregación público y general para distintos tipos de fuentes que permitiera sintetizar este conjunto de *mappings* en uno solo. El enfoque aquí presentado está basado en reglas y ha sido aplicado en la integración de catálogos de mobiliario y en la integración de recursos en el dominio geoespacial.

**Palabras clave:** Gestión del conocimiento, Representación del conocimiento, Web semántica, *Mappings*.

## 1 Introducción

El clásico problema de la heterogeneidad se ha visto parcialmente resuelto desde hace tiempo con un progresivo aumento de metodologías y herramientas de integración de información, tanto a nivel de esquemas como a nivel de datos [1; 2; 3]. Estos sistemas en su mayoría representan las semejanzas entre las fuentes de información mediante el establecimiento de *mappings*, puentes semánticos entre los elementos de las fuentes. Uno de los retos actuales es el descubrimiento de estos *mappings*, un proceso del que dependerá la calidad de la integración total del sistema.

Dentro del proceso de descubrimiento de *mappings* [4], la fase de agregación de *mappings* consiste en el resumen de los conjuntos de *mappings* resultantes de la fase de comparación. Aparecerán problemas como duplicidades de funciones o conflictos entre relaciones de los *mappings* generados por unas técnicas y por otras. Hasta ahora, aunque se han abordado problemas relacionados (p. ej. [5; 6; 7; 8; 9; 10; 11; 12]), entre ellos la agregación de similitudes y la extracción de *mappings* como procesos separados, no se había proporcionado un algoritmo de agregación de *mappings*.

En el epígrafe 2 se proponen una serie de definiciones básicas sobre el dominio de los *mappings* que, al contrario de las proporcionadas por otros autores, son generales para cualquier tipo de fuente a integrar. En los epígrafes 3 y 4 se describe al algoritmo, el primero público que realiza esta tarea y que se ha aplicado en la integración de catálogos de mobiliario y en la integración de fuentes de información geoespacial. En la sección 5 se demuestra la completud del algoritmo y su complejidad polinomial. Por último, se presentan las conclusiones y las líneas futuras.

## 2 Definiciones básicas

Para ilustrar el descubrimiento de *mappings* anterior a la agregación y los componentes involucrados en éste, se puede tomar como referencia la Figura 1.

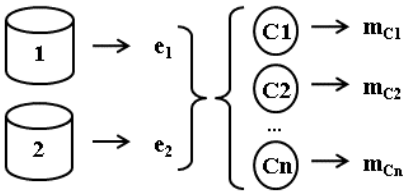


Figura 1. Descubrimiento previo a la agregación.

Como se puede apreciar, se parte de dos fuentes de información: *1* y *2*. Las fuentes de información constan de elementos (p. ej. $e_1$ y $e_2$). Los elementos $e_1$ y $e_2$ son la entrada de los comparadores $c_1$ a $c_n$. Para cada par de elementos cada uno de estos comparadores va a proporcionar, si procede, un *mapping*. En la figura, $m_{c1}$ es el *mapping* obtenido por el comparador $c_1$.

Dado que entre los diferentes autores no se alcanza un consenso en la terminología sobre *mappings*, en los siguientes párrafos se proporciona una serie de definiciones que están acordes con el uso que se hace habitualmente de estos términos y que son suficientemente abstractas como para contemplar la integración de tipos de fuentes de información diversas. En consecuencia, sobre todo por su generalidad, las definiciones presentadas a continuación son un aporte del presente trabajo.

*Definición 1*. Una *fuente* es un recurso que puede ser procesado por un computador. Formalmente, *fuente* se considerará un término básico.

*Ejemplo 1*. Son fuentes: una ontología, una base de datos, un catálogo, un diccionario, un programa, un algoritmo (procesado por un computador abstracto), etc.

*Definición 2*. Un *elemento* es una parte de una fuente a ser asociada con otra parte de otra fuente. Formalmente, *elemento* también será un término básico.

*Ejemplo 2*. Son elementos: el concepto *río*, el atributo *nombre* (de una ontología de accidentes geográficos), el término *"río"* (de un diccionario), la tabla *ríos* (de una base de datos), un método de una clase, una función utilizada en un algoritmo, etc.

*Definición 3*. Un *recurso externo* es aquello que es usado, además de las fuentes de partida, para realizar una determinada tarea. Este término también se considera básico.

*Ejemplo 3*. Muestras de recursos externos son: fuentes que no son aquellas sobre las que se realiza la tarea (p. ej. diccionarios auxiliares), información proporcionada por un experto, etc.

*Definición 4*. Dados dos conjuntos de elementos $E_1$ y $E_2$ de dos fuentes $f_1$ y $f_2$ respectivamente, se dice que una *relación* (binaria) R entre ambos conjuntos es un subconjunto del producto cartesiano $E_1 \times E_2$.

Ejemplo 4. Algunos casos de relaciones son: equivale a (≡), es subclase de (<), es superclase de (>), es disjunto con (⊥), se solapa con (*over*), es parte de, etc.

*Definición 5* (basada en [13]). Dadas dos fuentes $f_1$ y $f_2$, con conjuntos de elementos $E_1$ y $E_2$ respectivamente, se dice que un *mapping* es una tupla $m_{e1,e2} = <id, e_1, e_2, R, c>$, donde $e_1$ y $e_2$ son elementos de dos conjuntos $E_1$ y $E_2$ de las fuentes $f_1$ y $f_2$ respectivamente; *id* (identificador) es un objeto, por ejemplo un número, que permite distinguir la tupla de otras; *R* es una relación entre $E_1$ y $E_2$; y *c* (certeza) es un número real en el intervalo [0, 1].

*Ejemplo 5*. Un caso de *mapping* es *m = <1, río principal, río, <, 1>*, que significa que, de acuerdo con el *mapping* identificado como 1, es seguro (*certeza* igual a 1) que *río principal* es subclase de (<) *río*.

*Definición 6*. Dadas dos fuentes $f_1$ y $f_2$, con conjuntos de elementos $E_1$ y $E_2$ respectivamente, se dice que un *comparador* es una función $comp_{f1,f2}$: $\mathsf{P}(R_e) \rightarrow M \cup \{no\ hay\ mapping\}$, donde $R_e$ es el conjunto de posibles recursos externos utilizados en la obtención de *mappings* y M es un conjunto de *mappings*. Es decir, un comparador, para cada par de elementos $(e_1, e_2) \in E_1 \times E_2$, proporciona, si ha lugar, un *mapping*.

*Ejemplo 6*. Una muestra sencilla de comparador es el que establece: (1) si el nombre de $e_2$ es una subcadena del nombre de $e_1$, entonces se obtiene el *mapping* $<id, e_1, e_2, <, 0{,}7>$, es decir, $e_1$ es una subclase de $e_2$ con una certeza de 0,7; (2) si el nombre de $e_1$ es una subcadena del nombre de $e_2$, entonces se obtiene el *mapping* $<id, e_1, e_2, >, 0{,}7>$; y (3) se obtiene *no hay mapping* en otro caso. Así, con una certeza de 0,7, *río principal* es subclase de *río*, pues "*río*" es una subcadena de "*río principal*".

*Definición 7*. Dadas dos fuentes $f_1$ y $f_2$, se dice que un *generador de mappings* (o simplemente generador) es una función $gen_{f1,f2}$: $\mathsf{P}(R_e) \rightarrow \mathsf{P}(M \times C)$, donde $M \times C$ es el conjunto de pares formados por los *mappings* obtenidos y el comparador utilizado. Un generador invoca los comparadores y anota cada *mapping* con el comparador que lo ha descubierto, reuniendo los *mappings* de todos los comparadores en un solo conjunto.

*Ejemplo 7*. A partir de dos fuentes $f_1$ y $f_2$ con elementos $e_{11}, e_{12}, \ldots, e_{1n}$ y $e_{21}, e_{22}, \ldots, e_{2n}$, un generador podría obtener el conjunto de pares *mapping*/comparador: $R_{M \times C}$ = {(<1, $e_{11}$, $e_{23}$, <, 0,4>, $c_1$), (<2, $e_{11}$, $e_{23}$, <, 0,2>, $c_2$), (<3, $e_{11}$, $e_{23}$, ≡, 0,3>, $c_1$), (<4, $e_{12}$, $e_{24}$, >, 0,6>, $c_1$), (<5, $e_{12}$, $e_{24}$, ≡, 0,2>, $c_1$), (<6, $e_{13}$, $e_{25}$, parte de, 0,7>, $c_2$)}.

*Definición 8*. Dadas dos fuentes $f_1$ y $f_2$, se dice que un *agregador de mappings* (o simplemente agregador) es una función $agr_{f1,f2}$: $\mathsf{P}(R_e) \times \mathsf{P}(M \times C) \rightarrow \mathsf{P}(M)$. Un agregador de *mappings*, a partir de un conjunto de *mappings* entre dos fuentes (anotados con los comparadores que los han generado) y un conjunto de recursos externos, da lugar a otro conjunto que resume el conjunto de *mappings* de entrada.

*Ejemplo 8*. Un agregador, a partir del conjunto de *mappings* anotados del ejemplo 7, podría obtener: M = {<1, $e_{11}$, $e_{23}$, <, 0,4>, <4, $e_{12}$, $e_{24}$, >, 0,6>, <6, $e_{13}$, $e_{25}$, *parte de*, 0,7>}, dando lugar a un solo *mapping* para el par ($e_{11}$, $e_{23}$), otro para ($e_{12}$, $e_{24}$) y otro para ($e_{13}$, $e_{25}$).

Precisamente el propósito principal de este artículo es presentar un algoritmo de agregación de *mappings* completo para cualquier tipo de fuente.

## 3 Enfoque y algoritmo de agregación

Para lograr el objetivo de la agregación y conseguir a partir de M su agregado M**'**, se propone actuar separadamente sobre el conjunto de *mappings* de cada par de elementos y agregarlo mediante la aplicación de unas reglas de selección. Además de la información contenida dentro de los propios *mappings*, se va a hacer uso de información adicional determinada a priori por un experto en el dominio de las fuentes. Esta información consiste en un valor de prioridad o fiabilidad de los comparadores. Así, para cada comparador, el sistema tendrá un valor numérico comprendido entre 1 y 5, entendiéndose como 1 la menor prioridad y 5 la mayor. Este valor de prioridad servirá para dilucidar entre los resultados de dos comparadores para un mismo par de elementos. Así, se tiene:

*Definición 9*. Dado un conjunto de comparadores C, se llama *prioridad en C* a una función $pr_C$: $C \rightarrow [1, 5]$ que asigna un número entre 1 y 5 a cada comparador. Teniendo en cuenta que la prioridad la establece una persona, se asumirá que se trata de un recurso externo.

El algoritmo de agregación es el que se muestra en la Tabla 1. Consiste en una mera adición de los conjuntos resultantes de la selección de los subconjuntos de *mappings* asociados a cada par. Las reglas de SELECCIONAR se presentan en la siguiente sección. PRIMER_FILTRAR obtiene las parejas *mapping*/comparador de aquellos entre $e_1$ y $e_2$:

*Definición 10*. Un *filtrado de pareja mapping/comparador* con respecto a dos elementos es una función *fpar*: $E \times E \times \mathsf{P}(M \times C) \rightarrow \mathsf{P}(M \times C)$ que obtiene las parejas entre los elementos dados.

*Ejemplo 9*. Sea el siguiente conjunto de parejas *mapping*/comparador $R_{M \times C}$ del ejemplo 7, entonces *fpar*($e_{11}$, $e_{23}$, $R_{M \times C}$) = {(<1, $e_{11}$, $e_{23}$, <, 0,4>, $c_1$), (<2, $e_{11}$, $e_{23}$, <, 0,2>, $c_2$), (<3, $e_{11}$, $e_{23}$, ≡, 0,3>, $c_1$)}.

Tabla 1. Algoritmo de agregación de *mappings*.

ALGORITMO AGREGAR (permite evaluar la función

$$agr_{f1,f2}: \mathsf{P}(R_e) \times \mathsf{P}(M \times C) \rightarrow \mathsf{P}(M))$$

Entradas:

- los conjuntos de elementos $E_1$ y $E_2$ de las fuentes $f_1$ y $f_2 \in F \times F$;
- un conjunto de recursos externos $R_e$ con un único elemento: la función de prioridad $pr_c$ sobre los comparadores; y
- el conjunto de elementos del generador: $E \subseteq M \times C$, es decir, un subconjunto del producto cartesiano de los *mappings* posibles entre $f_1$ y $f_2$ (M) y el conjunto de comparadores utilizados en la generación (C)

Procedimiento AGREGAR($f_1$, $f_2$, $pr_c$, {($m_1$, $c_1$), ($m_2$, $c_2$), …, ($m_n$, $c_n$)})

$M_{salida} = \varnothing$

Para cada $e_1 \in E_1$

Para cada $e_2 \in E_2$

$M_{salida} = M_{salida} \cup$ SELECCIONAR($e_1$,$e_2$, $pr_c$,PRIMER_FILTRAR($e_1$,$e_2$,$\cup_{i=1, \ldots,n}\{(m_i, c_i)\}$))

/* PRIMER_FILTRAR evalúa la función *fpar*: $E \times E \times \mathsf{P}(M \times C) \rightarrow \mathsf{P}(M \times C)$ */

DEVOLVER $M_{salida}$

## 4 Reglas de selección

Dado el conjunto de *mappings* obtenido mediante un generador, en este trabajo se propone llevar a cabo los siguientes pasos para cada par de elementos $e_1$ y $e_2$. En primer lugar, se separa el conjunto de *mappings* en dos: *mappings* con relaciones de combinación predefinida y con relaciones de combinación no predefinida.

En caso de que el conjunto de *mappings* con relaciones de combinación no predefinida no sea vacío, se aplica una reducción, mediante combinación de las certezas, a un conjunto con un *mapping* por cada relación diferente.

En caso de que el conjunto de *mappings* con relaciones de combinación predefinida no esté vacío, se realiza una criba dejando sólo los *mappings* entre $e_1$ y $e_2$ obtenidos con los generadores de máxima prioridad. Si el conjunto obtenido de esta criba tiene un solo elemento, entonces se devuelve este conjunto; si tiene más de un elemento pero todos los *mappings* se refieren a la misma relación (r), se devuelve un *mapping* de la forma <*generarId*(), $e_1$, $e_2$, *r*, $c_{max}$>, donde *generarId*() da lugar a un nuevo identificador de *mapping*. Sin embargo, si la relación no es la misma, hay que realizar otra criba. Se hace otra criba sobre el resultado de la anterior dejando los *mappings* de máxima certeza (a esta máxima certeza se le llamará $c_{max}$). Si el conjunto obtenido en esta segunda criba tiene un solo elemento, entonces se devuelve tal conjunto. En caso contrario, componer una relación a partir de las del conjunto. Así, por ejemplo, si sólo aparecen las relaciones < y ≡, se devolverá un conjunto con un único elemento: el *mapping* <*generarId*(), $e_1$, $e_2$, $\leq$, $c_{max}$). Es decir, a partir de la relación *subclase de* y *es equivalente a*, se sintetiza la relación *es subclase de o equivalente a*. ¿Cómo se componen las relaciones? Hay una serie de ellas para las que se ha definido explícitamente la forma de componerlas (como es el caso de <, ≡, >, etc.). Para los demás grupos de relaciones, el ingeniero puede configurarse su propia

composición, o puede pedirle al usuario que proponga tal composición. Este último caso se etiquetará como INDETERMINADO.

En los siguientes párrafos se proporciona una serie de definiciones para formalizar la idea expuesta anteriormente. Después, se expondrán las reglas que permiten evaluar SELECCIONAR.

*Definición 11*. Se dice que $C_R$ *es un conjunto de relaciones de combinación predefinida* (*comb*) siempre y cuando sea subconjunto de {≡, <, >, ⊥, over}. Es decir: $comb(C_R) = C_R \subseteq$ {≡, <, >, ⊥, over}

*Definición 12*. Sea M el conjunto de posibles *mappings* entre dos fuentes $f_1$ y $f_2$. Se llama *separación de las relaciones de combinación predefinida* a una función $f_{comb}$: $\mathsf{P}(M) \rightarrow \mathsf{P}(M)$ que filtra el conjunto origen de *mappings* obteniendo sólo aquellos con relación predefinida.

*Definición 13*. Sea M el conjunto de posibles *mappings* entre dos fuentes $f_1$ y $f_2$. Se llama *separación de las relaciones de combinación no predefinida* a una función $f_{rel}$: $\mathsf{P}(M) \rightarrow \mathsf{P}(M)$ que filtra el conjunto origen de *mappings* obteniendo sólo aquellos con relación de combinación no predefinida.

*Ejemplo 10*. Suponiendo que M es el conjunto de *mappings* del ejemplo 7, $f_{comb}(M)$ = {<1, $e_{11}$, $e_{23}$, <, 0,4>, <2, $e_{11}$, $e_{23}$, <, 0,2>, <3, $e_{11}$, $e_{23}$, ≡, 0,3>, <4, $e_{12}$, $e_{24}$, >, 0,6>, <5, $e_{11}$, $e_{24}$, ≡, 0,2>}, mientras que $f_{rel}(M)$ = {<6, $e_{13}$, $e_{25}$, *parte de*, 0,7>}, que contiene el único *mapping* con relación que es de combinación no predefinida.

*Definición 14*. Se llama *función de combinación de certezas* a cualquier función *z*: $\mathsf{P}([0,1]) \rightarrow [0,1]$ que obtiene un valor de certeza a partir de los valores de certeza de entrada.

*Ejemplo 11*. z({0,3, 0,7, 0,8})=MAX({0,3, 0,7, 0,8})=0,8

*Definición 15*. Se llama *conjunto de certezas de una relación r* ($cer_r$) al conjunto de los valores de certeza de todos los *mappings* de un conjunto *M* cuya relación es *r*. Es decir: $cer_M(r)=\{c_i \mid m_i \in M \wedge m_i = <id_i, e_1, e_2, r, c_i>\}$.

*Definición 16*. Sea una función de combinación de certezas *z* y M el conjunto de posibles *mappings* entre dos elementos $e_1$ y $e_2$. Se llama *reducción de mappings de relación de combinación no predefinida* a una función $red_{e1e2z}$:$\mathsf{P}(M) \rightarrow \mathsf{P}(M)$ que, para cada conjunto origen A, obtiene un conjunto formado por un *mapping* para cada una de las relaciones que aparecen en $f_{rel}(A)$ y con un valor de certeza que sintetiza los valores de certeza de esa relación a través de z. Es decir: $red_{e1e2z}(\varnothing) = \varnothing$, y $red_{e1e2z}(\{\cup_{i=1\ldots n}\cup_{j=1\ldots mi}\{<id_{ij}, e_1, e_2, r_i, c_{ij}>\}) = \cup_{k=1\ldots q}\{<generarId(), e_1, e_2, r_k, z(\cup_{j=1\ldots mk}\{c_{kj}\})>\}$ donde cada $r_s \neq r_t$ si $s \neq t$, y las $r_k$ son las relaciones de los *mappings* obtenidos de aplicar $f_{rel}$.

*Ejemplo 12*. Dado el conjunto A = {<1, $e_{11}$, $e_{23}$, <, 0,4>, <2, $e_{11}$, $e_{23}$, <, 0,2>, <1, $e_{11}$, $e_{23}$, *tiene*, 0,4>, <3, $e_{11}$, $e_{23}$, *tiene*, 0,3>, <4, $e_{11}$, $e_{23}$, *va con*, 0,8>, <5, $e_{11}$, $e_{23}$, *comparten*, 0,7>, <8, $e_{11}$, $e_{23}$, *va con*, 0,3>}, $red_{e1123z}(A)$ = {<$m_1$, $e_{11}$, $e_{23}$, *tiene*, *z*({0,4, 0,3})>, <$m_4$, $e_{11}$, $e_{23}$, *va con*, *z*({0,8, 0,3})>, <$m_5$, $e_{11}$, $e_{23}$, *comparten*, 0,7>}.

*Definición 17*. Dado el conjunto de comparadores C, se llama *filtro de máxima prioridad* a una función $maxp_c$: $E_1 \times E_2 \times \mathsf{P}(M \times C) \times F_p \rightarrow \mathsf{P}(M)$, tal que obtiene el conjunto de *mappings* generados a partir de los comparadores con más prioridad. $E_1$ y $E_2$ son conjuntos de elementos de dos fuentes $f_1$ y $f_2$ respectivamente, $(M \times C)$ es el producto cartesiano de los *mappings* posibles entre $f_1$ y $f_2$ (M) y el conjunto de comparadores utilizados en la generación (C), y $F_p$ el conjunto de posibles funciones de prioridad. Es decir:

$$maxp_c(e_1, e_2, \cup_{i=1,\ldots,n}\{(m_i, c_i)\}, pr_c) = \{m_{e1,e2} \mid \exists\ comp_{f1,f2} \in C \text{ tq}$$
$$(m_{e1,e2}, comp_{f1,f2}) \in \cup_{i=1,\ldots,n}(m_i, c_i)\} \wedge pr_c(comp'_{f1,f2}) \leq pr_c(comp_{f1,f2})$$
$$\forall\ (m_{e1,e2},, comp'_{f1,f2}) \in \{\cup_{i=1,\ldots,n}(m_i, c_i)\}\ \}$$

*Ejemplo 13*. Supóngase el conjunto de parejas *mapping*/comparador del ejemplo 9, y las prioridades las siguientes $pr_C(c_1) = 5$, $pr_C(c_2) = 3$. En tal caso, $maxp_c(e_{11}, e_{23}, R_{M \times C}, pr_C) = \{<1, e_{11}, e_{23}, <, 0{,}4>, <3, e_{11}, e_{23}, \equiv, 0{,}3>\}$, pues los *mappings* del conjunto resultante son los obtenidos con $c_1$, el comparador de máxima prioridad.

*Definición 17*. Dado un conjunto de *mappings M*, se dice que *M* es de *relación única unirel(M)* si y si todos los *mappings* de *M* tienen la misma relación. Es decir:

$$unirel(M) = \exists r,, m_i = <id_i, e_1, e_2, r, c_i> \ \forall m_i \in M$$

*Definición 18*. Dado un conjunto de comparadores C, se llama *filtro de máxima certeza con máxima prioridad* a una función $maxc_c$: $E_1 \times E_2 \times \mathsf{P}(M \times C) \times F_p \rightarrow \mathsf{P}(M)$, tal que obtiene el conjunto de *mappings* entre $e_1$ y $e_2$ con mayor certeza de entre los generados por los comparadores con más prioridad. Es decir:

$$maxc_c(e_1, e_2, \cup_{i=1,\ldots,n}\{(m_i, c_i)\}, pr_c) = \{m_{e1,e2} \mid m_{e1,e2} \in maxp_c(e_1, e_2, \cup_{i=1,\ldots,n}\{(m_i, c_i)\},$$
$$pr_c)$$
$$\wedge\ m_{e1,e2} = <id, e_1, e_2, r, c> \wedge c' \leq c\ \ \forall m'_{e1,e2} \in maxp_c(e_1, e_2, \cup_{i=1,\ldots,n}\{(m_i, c_i)\}, pr_c) \wedge$$
$$m'_{e1,e2} = <id', e_1, e_2, r', c'>\}$$

*Definición 19*. La certeza que es mayor o igual que la de cualquier otro *mapping* del conjunto devuelto por la función $maxc_c$ se llama *certeza máxima* ($c_{max}$).

*Ejemplo 14*. En el ejemplo 12, $maxc_c(e_{11}, e_{23}, R_{M \times C}, pr_C) = \{<1, e_{11}, e_{23}, <, 0{,}4>\}$, pues, entre los elementos del conjunto obtenido con $maxc_c$, $<1, e_{11}, e_{23}, <, 0{,}4>$ tiene más credibilidad que $<3, e_{11}, e_{23}, \equiv, 0{,}3>$. En este caso, $c_{max} = 0{,}4$.

*Definición 20*. Se llama *función de relaciones de máxima certeza con máxima prioridad* a *r_max*: $E_1 \times E_2 \times \mathsf{P}(M \times C) \times F_p \rightarrow R$, que obtiene el conjunto de relaciones involucradas en los *mappings* resultado del *filtro de máxima certeza con máxima prioridad*, es decir:

$$r\_max(e_1, e_2, \cup_{i=1,\ldots,n}\{(m_i, c_i)\}, pr_c) =$$
$$\{r \mid <id, e_1, e_2, r, c> \in maxc_c(e_1, e_2, \cup_{i=1,\ldots,n}\{(m_i, c_i)\}, pr_c)\}$$

*Ejemplo 15*. En el ejemplo 12, $r\_max(e_{11}, e_{23}, R_{M \times C}, pr_C) = \{<\}$, pues la única relación con prioridad 0,4 (la máxima en este caso) es <.

*Definición 21*. La *función predefinida de combinación* g: $\mathsf{P}(\{\equiv, <, >, \perp, \text{over}\}) \rightarrow \{\equiv, <, >, \perp, \leq, \geq, \text{over}, \text{INDETERMINADA}\}$ es la mostrada en la Tabla 2.

Tabla 2. Función $g(r_{MAX})$.

| $r_{MAX}$ | $g(r_{MAX})$ | $r_{MAX}$ | $g(r_{MAX})$ | $r_{MAX}$ | $g(r_{MAX})$ |
|---|---|---|---|---|---|
| <, ≡ | ≤ | ⊥, ≡, < | ⊥ | over, < | < |
| <, > | INDETERMINADA | ⊥, ≡, > | ⊥ | over, > | > |
| <, >, ≡ | over | ⊥, ≡, <, > | ⊥ | over, ≡, < | ≤ |
| >, ≡ | ≥ | ⊥, > | ⊥ | over, ≡, > | ≥ |
| ⊥, ≡ | ⊥ | ⊥, over | INDETERMINADA | over, <, > | INDETERMINADA |
| ⊥, < | ⊥ | over, ≡ | ≡ | over, ≡, <, > | over |

La Tabla 3 presenta el conocimiento utilizado en SELECCIONAR.

Tabla 3. Cómo SELECCIONAR *mappings*.

<table>
<tr><td colspan="4">CONOCIMIENTO PARA SELECCIONAR</td></tr>
<tr><td colspan="4">Hechos:<br>– $e_1 \in E_1$ y $e_2 \in E_2$, siendo $E_1$ y $E_2$ los elementos de las fuentes $f_1$ y $f_2$ respectivamente<br>– la función de prioridad utilizada es $pr_c$<br>– la relación entre mappings y los comparadores que han servido para generarlos es: S = {($m_1$, $c_1$), ($m_2$, $c_2$), …, ($m_n$, $c_n$)}<br>– $S_{comb} = f_{comb} = \{(m_1, c_1), \ldots, (m_q, c_q)\}$ y $S_{rel} = f_{rel} = \{(m_1, c_1), \ldots, (m_r, c_r)\}$</td></tr>
<tr><td colspan="4">Abreviaturas (por orden alfabético):<br>– generarId() devuelve un nuevo id para un mapping.<br>– MaxC = $maxc_c(e_1, e_2, S_{comb}, pr_c)$, MaxP = $maxp_c(e_1, e_2, S_{comb}, pr_c)$ y RMax = r_max ($e_1$, $e_2$, $S_{comb}$, $pr_c$)</td></tr>
<tr><td colspan="4">Reglas</td></tr>
<tr><td colspan="3">$|S| = 0$</td><td>∅</td></tr>
<tr><td colspan="3">$|S_{rel}| = 0$</td><td>∅</td></tr>
<tr><td colspan="3">$|S_{rel}| > 0$</td><td>$red_{e1e2z}(S_{rel})$</td></tr>
<tr><td colspan="3">$|S_{comb}| = 0$</td><td>∅</td></tr>
<tr><td colspan="3">$|S_{comb}| = 1$</td><td>$S_{comb}$</td></tr>
<tr><td rowspan="3">$|S_{comb}| > 1$</td><td colspan="2">|MaxP| = 1</td><td>MaxP</td></tr>
<tr><td rowspan="2">|MaxP| > 1</td><td>unirel($S_{comb}$)</td><td>{<generarId(), $e_1$, $e_2$, r, $z(cer_{MaxP}(r))$>}</td></tr>
<tr><td>¬unirel($S_{comb}$)</td><td>|MaxC| = 1<br>MaxC<br>|MaxC| > 1<br>{<generarId(), $e_1$, $e_2$, g(RMax), $c_{max}$>}</td></tr>
</table>

## 5 Propiedades formales

De acuerdo con lo expuesto en los apartados anteriores, se pueden obtener las siguientes propiedades formales del algoritmo propuesto para la agregación:

*Teorema 1*. El algoritmo AGREGAR es *completo*.

*Esquema de la demostración*. La Tabla 1 muestra que se consideran todas las clases de entradas posibles y, por tanto, el algoritmo termina y da lugar a un resultado para todos los casos posibles.

*Teorema 2*. Sean dos fuentes $f_1$ y $f_2$, con conjuntos de elementos $E_1$ y $E_2$ (tales que $|E_1| = k_1 * n$ y $|E_2| = k_2 * n$), entre los que se ha establecido un conjunto de *mappings* M (tal que $|M| = k_m * n$), y sea $t$: $\{\text{AGREGAR}(f_1, f_2, pr_c, \cup_{i=1,\ldots,n}\{(m_i, c_i)\}) \mid f_1 \in F \wedge f_2 \in F \wedge pr_c \in F_p \wedge \cup_{i=1,\ldots,n}\{(m_i, c_i)\} \in \mathsf{P}(M \times C)\} \rightarrow |N$ la función que obtiene el número de pasos en la ejecución de AGREGAR para cada entrada concreta, entonces $t(.) = O(n^3)$.

*Demostración*. Por cada elemento de $E_1$ hay que recorrer cada elemento de $E_2$ (de ahí $|E_1| * |E_2|$). Por cada una de las parejas, hay que evaluar PRIMER_FILTRAR, que recorre todo el conjunto de mappings (de ahí |M|). Ahora bien, el cálculo de MaxP, MaxC, RMax, *comb*(RMax) y *g*(RMax) lleva un número de pasos lineal sobre |M|, pues el conjunto de mappings a manipular en cada una de estas computaciones nunca será mayor que el propio M. Es decir, estos cálculos no aportan al límite superior de *t*. En consecuencia se llega a $O(|E_1| * |E_2| * |M|) = O(k_1 * k_2 * k_m * n^3) = O(n^3)$.

Es importante observar que este resultado sobre la complejidad de AGREGAR es sólo cierto si la computación exigida por el usuario para resolver las indeterminaciones requiere un número de pasos lineal con respecto al tamaño de M.

## 6 Conclusiones y líneas futuras

A pesar de que el trabajo en el uso de *mappings* en la integración de fuentes heterogéneas ha sido intenso, hasta ahora nadie había publicado un algoritmo para la agregación de los *mappings* generados por una serie de comparadores. Precisamente por esta razón, en este artículo se presenta un algoritmo completo y de complejidad polinomial que permite llevar a cabo la agregación de *mappings* entre cualquier tipo de fuente. Como soporte a tal algoritmo, se ha proporcionado una terminología sobre el dominio de los *mappings* que es suficientemente abstracta como para ser utilizable en cualquier contexto sea cual sea la fuente.

El algoritmo propuesto se ha aplicado en la integración de catálogos de mobiliario, y se está aplicando en la integración de recursos (ontologías, catálogos, bases de datos, etc.) en el dominio geoespacial. Se encuentra funcionando como parte de la aplicación de descubrimiento de *mappings* OEGMapDiscover. Actualmente tiene implementada como función z el máximo del conjunto de certezas y como función de combinación de relaciones de combinación predefinida, la expresada en la **Tabla 2**.

Queda como línea futura el estudio de la combinación de otras relaciones aparte de las ya consideradas. También está previsto realizar experimentos controlados utilizando el algoritmo como base con el fin de evaluar comparadores estudiando los resultados obtenidos con distintas prioridades.

## 8 Agradecimientos

Este trabajo ha sido financiado parcialmente por el proyecto nacional “GeoBuddies: Anotación semántica colaborativa con dispositivos móviles en el camino de Santiago” (TSI 2007-65677 C02).

## 9 Referencias

[1] Batinti C, Lenzerini M, Navathe SB (1986) A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys, Vol. 18, No. 4, December 1986.

[2] Lenzerini M (2002) Data integration: a theoretical perspective. /PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems/, pp. 233-246, New York, NY, USA. ACM Press

[3] Beneventano D, Bergamaschi S (2004) The MOMIS Methodology for Integrating Hetrogeneous Data Sources. IFIP Congress Topical Sessions, 2004, pp. 19-24.

[4] Euzenat J, Shvaiko P (2007) Ontology Matching. Springer Verlag, 2007.

[5] Van Hage WR, Katrenko S, Schreiber G (2005) A Method to Combine Linguistic Ontology-Mapping Techniques. ISWC 2005. Lecture Notes in Computer Science – LNCS 3729. Springer-Verlag Berlin Heidelberg, 2005. Pp: 732-744.

[6] Svab O, Svatek V (2006) Combining Ontology Mapping Methods Using Bayesian Networks. International Workshop on Ontology Matching collocated with the 5th International Semantic Web Conference (ISWC-2006), November 2006.

[7] Besana (2006) A Framework for Combining Ontology and Schema Matchers with Dempster-Shafer. International Workshop on Ontology Matching collocated with the 5th International Semantic Web Conference (ISWC-2006), November 2006.

[8] Huza M, Harzallah M, Trichet F (2006) OntoMas: a Tutoring System dedicated to Ontology Matching. International Workshop on Ontology Matching collocated with the 5th International Semantic Web Conference (ISWC-2006), November 2006.

[9] Valarakos AG, Spiliopoulos V, Kotis K, Vouros G (2007) AUTOMS-F: A Java Franework for Synthesizing Ontology Mapping Methods. I-KNOW 2007 Special Track on Knowledge Organization and Semantic Technologies 2007, September 5, 2007, Graz

[10] Duchateau F, Bellahsene Z, Coletta R (2008) A Flexible Approach for Planning Schema Matching Algorithms Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Monterrey, Mexico. Pages: 249 – 264.

[11] Mochol M, Jentzsch A (2008) Towards a Rule-Based Matcher Selection. EKAW 2008. Lecture Notes in Artificial Intelligence – LNAI 5268. Springer-Verlag Berlin Heildelberg, 2008. Pp: 109-119.

[12] Euzenat J (2008) Algebras of ontology alignmet relations. International Semantic Web Conference (ISWC 2008). Lecture notes in computer science 5318. Pp: 387-402. Karlsruhe, Denmark, 2008

[13] Ramos JA, Gómez-Pérez A (2008) Mappings for the Semantic Web. IEEE Intelligent Systems – Trends & Controversies. Vol. 23-6, pp: 79-81. November, 2008.