DeLP como lenguaje de consulta: un enfoque preliminar al poder expresivo

Laura A. Cecchi[1] y Guillermo R. Simari[2]

[1] Grupo de Investigación en Lenguajes e Inteligencia Artificial
Facultad de Informática - Universidad Nacional del Comahue
[2] Laboratorio de Investigación y Desarrollo en Inteligencia Artificial
Depto. de Ciencias e Ingeniería de la Computación - Universidad Nacional del Sur

lcecchi@gmail.com grsimari@gmail.com

Resumen. La Programación en Lógica Rebatible (DeLP) es una extensión de la Programación en Lógica que permite representar conocimiento tentativo y razonar a través de argumentos a partir de él. Actualmente, existen muchas aplicaciones de DeLP a bases de datos y la web, por lo que resulta de importancia el análisis de la complejidad computacional y del poder expresivo del lenguaje. En este trabajo, se propone a DeLP como lenguaje de consulta para bases de datos ordenadas finitas. Asimismo, se relacionan resultados de la complejidad computacional ya alcanzados, con otros modelos y con la Complejidad Descriptiva. Finalmente, se avanza en el estudio del poder expresivo del lenguaje de consulta rebatible, analizando las consultas expresables en $FO(TC)$, que coinciden con los problemas resolubles en NL. Se demuestra que las consultas expresables en $FO(TC)$ son un subconjunto de las expresables en el lenguaje de consulta basado en DeLP.

Palabras Clave: Sistemas Argumentativos, Razonamiento Rebatible, Programación en Lógica, Bases de Datos, Complejidad, Poder Expresivo

1 Introducción

Una tarea fundamental en la Ciencia de la Computación teórica es clasificar a las problemas de acuerdo a la complejidad de los algoritmos que los resuelven. Históricamente, los problemas han sido clasificados de acuerdo a dos recursos relevantes: tiempo y espacio. Así, la complejidad computacional mide cuánto tiempo o espacio en memoria se necesita en función del tamaño de la entrada.

La Complejidad Descriptiva[16, 13] es un enfoque lógico a la computación en el que la complejidad computacional de un problema puede ser visto como la riqueza del lenguaje que se necesita para especificar el problema. La Complejidad Descriptiva caracteriza a las clases de complejidad por la categoría de la lógica que se requiere para expresar los lenguajes que pertenecen a la clase. Así, podemos clasificar los problemas, también en términos de la complejidad de los lenguajes formales que permiten expresarlos. Las Complejidades Computacional y Descriptiva tienen una relación muy cercana: las clases más importantes de complejidad computacional tienen una caracterización descriptiva.

La Programación en Lógica Rebatible (Defeasible Logic Programming), de ahora en más DeLP, es una herramienta basada en argumentación para representar conocimiento tentativo y razonar con él [15]. Su semántica operacional está basada en un análisis dialéctico donde los argumentos a favor y en contra de un literal interactúan con el objeto de determinar si ese literal es aceptado entre las creencias de un agente que razona[3]. Actualmente, existen desarrollos de DeLP en relación a bases de datos [10, 11] y la web [8, 9], por lo que resulta de importancia el análisis del sistema desde el punto de vista de la complejidad computacional como del poder expresivo, ya que la primera nos indica cuán difícil es responder una consulta, mientras que la segunda da una caracterización precisa de los conceptos que son posibles definir como consultas.

Algunos resultados sobre complejidad computacional han sido presentados sobre frameworks abstractos de argumentación, basados en las semánticas admisible y preferida (ver [12]). Sin embargo, estos resultados no se aplican directamente a DeLP. Por otra parte, no se han realizado estudios de sistemas argumentativos como lenguaje de consulta, ni tampoco se ha analizado el poder expresivo de los sistemas argumentativos.

En este trabajo, se propone un concepto nuevo introduciendo a DeLP como lenguaje de consulta rebatible para bases de datos ordenadas finitas, basado en sistemas argumentativos. Al caracterizar un lenguaje de consultas un aspecto importante es determinar su poder expresivo

Asimismo, se avanza en el estudio de la complejidad computacional y el poder expresivo del lenguaje. En cuanto a la complejidad computacional, se retoman algunos resultados alcanzados y se los relaciona con otros modelos computacionales, como la Máquina de Turing Alternante. Así, conociendo que el problema de decisión: "dado un árbol dialéctico $\mathcal{T}$, determinar si la raíz está etiquetada con U" es PSPACE-completo[4], se demuestra que es AP-completo. A través de la Complejidad Descriptiva relacionamos las clases de complejidad con el poder expresivo necesario para caracterizar el problema. Si bien la complejidad descriptiva del lenguaje, no coincide necesariamente con la complejidad del cómputo de una consulta, esta relación ayuda a determinar y circunscribir el poder expresivo del lenguaje.

Finalmente, en cuanto al poder expresivo, se analizan aquellas consultas que requieren de un operador de clausura transitiva y se prueba que el conjunto de las consultas expresables en el lenguaje de consulta rebatible está incluido en el conjunto de las consultas $FO(TC)$, las que coincide con las consultas de complejidad computacional NL.

El resto de este trabajo está estructurado como sigue. Primero introducimos brevemente los fundamentos de DeLP, y se presenta la definición del lenguaje de consulta rebatible, basado en DeLP. Luego, describimos sucintamente la teoría de prueba y la semántica declarativa basada en juegos de DeLP, a fin de especificar la semántica de consulta general y consulta booleana bajo este nuevo lenguaje. En la sección siguiente, se detallan algunos resultados alcanzados sobre la complejidad computacional y se los relaciona con el modelo de Máquinas de

[3] Ver `http://lidia.cs.uns.edu.ar/DeLP`

Turing Alternantes, mostrando la analogía con un juego, y con la Complejidad Descriptiva, especificando la clase de lenguaje que se requiere para expresar el problema de decisión. A continuación, se analizan dos clases de consultas que no son expresables en el lenguaje de bases de datos relacionales sin extender al lenguaje con un operador de clausura transitiva. En este sentido, se individualiza el problema, y se propone una forma de resolverlo en el lenguaje de consulta rebatible. Se demuestra que las consultas expresables bajo el nuevo lenguaje de consulta rebatible es un subconjunto de $FO(TC)$, que coincide con las consultas resolubles en NL. Finalmente, resumimos la contribución de este trabajo y presentamos futuras líneas de investigación.

2 Sintaxis del lenguaje de consulta rebatible

En esta sección introducimos y formalizamos la idea de considerar a DeLP como un lenguaje de consulta de bases de datos relacionales. Con este objetivo, presentamos los conceptos básicos de DeLP(ver [15] para más detalles). En el lenguaje de DeLP un literal L es un átomo A o un átomo negado $\sim A$, donde $\sim$ representa la negación fuerte en el sentido de la programación en lógica. El complemento de un literal L, que notamos $\overline{L}$, está definido como: $\overline{L} = \sim A$, si L es un átomo A, de otro modo, si L es un átomo negado $\sim A$, $\overline{L} = A$.

Definición 1. *Una* regla estricta *es un par ordenado, que notamos* Cabeza $\leftarrow$ Cuerpo *cuyo primer miembro* Cabeza *es un literal* L_0*, y cuyo segundo miembro* Cuerpo *es un conjunto finito de literales. Si el cuerpo es un conjunto vacío, luego podemos escribir* L_0*., y será llamado un* Hecho. *Una* regla rebatible *es un par ordenado, que notamos* Cabeza $-\prec$ Cuerpo *cuyo primer miembro* Cabeza *es un literal* L_0*, y cuyo segundo miembro* Cuerpo *es un conjunto finito no vacío de literales*$\{L_1, \ldots L_n, n > 0\}$.
Un programa lógico rebatible *(defeasible logic program)* $\mathcal{P}$*, que abreviaremos de.l.p., es un conjunto finito de reglas estrictas* Π_R *y hechos* Π_F*,* $\Pi = \Pi_F \cup \Pi_R$*, y un conjunto finito de reglas rebatibles* Δ.

Intuitivamente, mientras que Π es un conjunto de conocimiento certero y libre de excepciones, Δ es un conjunto de conocimiento rebatible, i.e., información tentativa que puede ser usada, siempre que nada pueda ser argumentado en su contra.

Una base de datos relacional es modelada como una estructura lógica finita. Un *esquema de base de datos* R es un conjunto finito de esquemas de relaciones $\{R_1, \ldots, R_n\}$[21]. Un *esquema de relación* R_i tiene un nombre N_i y una lista finita de atributos $A_1, \ldots, A_{l_i}$, siendo l_i la aridad de la relación. Asumimos que existe un conjunto finito de objetos $\mathcal{U}$, el universo dado sobre un dominio, que pueden ser utilizados en las relaciones. Dado un esquema de relación R_i, una instancia de relación es un conjunto de tuplas de la forma $(a_1, \ldots, a_{l_i})$, donde $a_j \in \mathcal{U}(1 \leq j \leq l_i)$. Una instancia de una base de datos es un conjunto de instancias de relaciones. El conjunto de objetos que ocurren en la base de datos, un subconjunto de $\mathcal{U}$, lo denominaremos *dominio activo*. Asumiremos que el

universo está libre de funciones y que tiene predefinido un orden total, de modo de poder utilizar resultados basados en bases de datos ordenadas.

La base de datos relacional de entrada se corresponde con Π_F ground en la definición de un *de.l.p.*, que denominaremos parte extensional. Considerando la definición de bases de datos deductivas, la parte intensional correspondiente es $\Pi_R \cup \Delta$, las reglas.

Definición 2. *Dado un de.l.p.* $\mathcal{P} = \langle \Pi_F, \Pi_R, \Delta \rangle$*, el universo de la base de datos* $\mathcal{U}$ *es el universo de Herbrand de de.l.p.* $\mathcal{H}(P)$*. Una* consulta general *es un conjunto finito de reglas estrictas y rebatibles junto con un átomo* A_{l_i} *de aridad* l_i*. Una* función de evaluación de una consulta general *es un mapeo desde una consulta en un conjunto de* l_i*-tuplas de* $\mathcal{U}$*,*

$$AQ : \Pi_R \times \Delta \times A_{l_i} \rightarrow 2^{\text{ATRIBUTOS}_{l_i}} \qquad \text{ATRIBUTOS}_{l_i} = \{\langle u_1 \dots u_{l_i} \rangle | u_j \in \mathcal{U}\}.$$

Una consulta boolena *es un conjunto finito de reglas estrictas y rebatibles junto con un literal ground* L*. Una* función de evaluación de una consulta booleana *es un mapeo desde una consulta booleana en* $\{SI, NO\}$

$$AQ_B : \Pi_R \times \Delta \times L \rightarrow \{SI, NO\}.$$

El significado intuitivo de una respuesta a una consulta general es el siguiente: deseamos computar todas las tuplas que cumplen la relación A_{l_i}, que pueden ser inferida bajo la semántica de DeLP. Nótese que en este caso deberíamos recorrer todos los elementos del dominio para hacer la consulta correspondiente con la relación.

Por otro lado, el significado intuitivo de la respuesta a una consulta booleana es el siguiente: queremos saber si L se infiere bajo la semántica de DeLP teniendo en cuenta la consulta y las instancias ground de la parte intensional, la base de datos de entrada.

Ejemplo 1. Consideremos el ejemplo de las aves Tweety, Clueca y Tina. Supongamos las relaciones unarias AVE, GALLINA y ASUSTADO que tienen solo un atributo: el nombre. El dominio del atributo es STRING y el universo incluye a $\{Tweety, Tina, Clueca\}$. La instancia de la base de datos es la siguiente:

AVE

Nombre
Tweety
Tina
Clueca

GALLINA

Nombre
Tina
Clueca

ASUSTADO

Nombre
Tina

La consulta general sobre qué objetos vuelan es:

$$\{ave(X) \leftarrow gallina(X) \qquad vuela(X) \multimap ave(X)$$

$$\neg vuela(X) \multimap gallina(X) \qquad vuela(X) \multimap gallina(X) \wedge asustado(X)\}$$

con el átomo vuela(X).

Por otra parte, la consulta booleana sobre si Tina vuela es:

$$\{ave(X) \leftarrow gallina(X) \qquad vuela(X) \multimap ave(X)$$

$$\neg vuela(X) \multimap gallina(X) \qquad vuela(X) \multimap gallina(X) \wedge asustado(X)\}$$

con el literal ground vuela(Tina).

3 Semántica de las consultas bajo el lenguaje de consulta rebatible

La semántica de DeLP está basada en los desarrollos de sistemas argumentativos rebatibles [18, 20]. Con el objeto de determinar si un literal L está soportado por un *de.l.p.* se construye un árbol dialéctico para L. La raíz del árbol dialéctico es un argumento para L y todo otro nodo en el árbol es un derrotador de su padre. En cada nivel, para un nodo dado, debemos considerar todos los argumentos en su contra. Así, cada nodo tiene un descendiente por cada derrotador. Existen ciertas restricciones al construir el árbol entre las que cuenta que un subargumento de un argumento ya introducido en una rama no puede ser reintroducido, evitando de este modo ciclos (ver [15] para más detalles).

Diremos que un literal L está *garantizado* si existe un argumento para L y en su árbol dialéctico cada derrotador de la raíz está derrotado. Recursivamente, esto lleva a un proceso de marcado del árbol que comienza considerando que las hojas del árbol son argumentos no derrotados, ya que no tienen derrotadores.

El razonamiento basado en argumentos tiene una analogía con los juegos. El árbol de dialéctica puede ser visto como un juego entre dos jugadores: el proponente y el oponente. Si el juego es ganado por el proponente entonces decimos que el literal está garantizado. La semántica declarativa basada en juegos $\mathcal{GS}$ de DeLP captura a la teoría de prueba de DeLP a través de un modelo minimal trivaluado (más detalles en [5, 6, 2, 3]).

Existen cuatro posibles respuestas para una consulta L en DeLP: SI si L está garantizado, esto es, existe un juego ganado por el proponente para L, NO si $\overline{L}$ está garantizado (i.e., existe un juego del complemento de L ganado por el proponente), INDECISO si ni L ni $\overline{L}$ están garantizados, esto es, no existe un juego ganado por el proponente ni para L ni para su complemento, y DESCONOCIDO si L no está en la signatura subyacente del programa.

Así, definimos el comportamiento de una respuesta a una consulta general a la base de datos, de acuerdo a la semántica basada en juegos. Las definiciones que siguen se restringirá a instancias de bases de datos y consultas, i.e., relaciones ground.

Definición 3. *Sean Π_F, Π_R y Δ, instanciados sobre el universo, y una consulta general $Q = \langle \Pi_R, \Delta, A_{l_i} \rangle$, la función de evaluación de la consulta general AQ se define como sigue:*

$$AQ(Q) = \{\langle u_1 \dots u_{l_i} \rangle | u_j \in \mathcal{U}, 1 \leq j \leq l_i \text{ y existe un juego ganado por el proponente que comienza con un argumento para } A_{l_i}(u_1, \dots, u_{l_i})\}$$

Equivalentemente, podemos decir que la respuesta a una consulta general es el conjunto de las tuplas, tales que $A_{l_i}(u_1, \ldots, u_{l_i})$ está garantizado.

Definición 4. *Sean Π_F, Π_R y Δ, instanciados sobre el universo, y una consulta booleana $Q_B = \langle \Pi_R, \Delta, L \rangle$, la función de evaluación de una consulta booleana AQ_B se define como sigue:*

$$AQ_B(Q) = \begin{cases} SI & \textit{si existe un juego ganado por el proponente que comienza} \\ & \textit{con un argumento para } L \ \ (L \textit{ está garantizado}) \\ NO & \textit{si existe un juego ganado por el proponente que comienza} \\ & \textit{con un argumento para } \overline{L} \ \ (\overline{L} \textit{ está garantizado}) \end{cases}$$

Ejemplo 2. Consideremos nuevamente el ejemplo de las aves. La consulta general que contiene al átomo $vuela(X)$ da como respuesta $\{\langle Tweety \rangle, \langle Tina \rangle\}$. La consulta booleana tiene como respuesta SI.

4 Relacionando la Complejidad Computacional de DeLP

Del estudio y análisis de DeLP y de su teoría de prueba y de su semántica declarativa basada en juegos se alcanzaron los siguientes resultados en complejidad:

- Determinar si $\langle \mathcal{A}, h \rangle$ es un argumento es P-completo [3].
- La existencia de un argumento está en la clase NP [3].
- La inexistencia de un argumento está en la clase co-NP (implícito en [3]).
- Dado un árbol dialéctico $\mathcal{T}$, determinar si la raíz está etiquetada con U es PSPACE-completo [4].

A partir de estos resultados es posible caracterizar los problemas en otras clases de complejidad a las que pertenecen, particularmente teniendo en cuenta la similitud entre la teoría de prueba y la interacción entre dos jugadores. Estas relaciones ayudan a determinar el poder expresivo del lenguaje.

4.1 Relación con las Máquinas de Turing Alternante

Las Máquinas de Turing Alternante(de ahora en más ATM)[7] son una generalización de las Máquinas de Turing no determinísticas que tienen dos clases de estados: los estados existenciales y los estados universales. Una ATM en una configuración C *acepta* la cadena si y solamente si:

- C está en un estado final de aceptación; o
- C está en un estado existencial, esto es, etiquetado con $\exists$, y existe una configuración C' siguiente a C que acepta la cadena; o
- C está en un estado universal, esto es, etiquetado con $\forall$, y existe al menos una configuración siguiente y todas las configuraciones siguientes terminan aceptando la cadena.

La clase de complejidad $ATIME[(t(n)]$ se define como el conjunto de lenguajes decididos por una ATM usando un máximo de $O(t(n))$ pasos de tiempo en cualquier computación sobre una entrada de longitud n[16, 1, 17]. En particular, aquellos lenguajes decididos en tiempo polinomial por una ATM se definen como la clase $AP = ATIME[n^k]$ que coincide con la clase de complejidad PSPACE[16, 1, 17].

Corolario 1. *Dado un árbol dialéctico $\mathcal{T}$, determinar si la raíz está etiquetada con U es AP-completo.*

Demostración. Membresía a AP: Hemos probado que dado un árbol dialéctico $\mathcal{T}$, determinar si la raíz está etiquetada con U es PSPACE-completo[4]. Dado que el problema pertenece a PSPACE, luego pertenece a AP.
Completitud: QSAT es AP-completo[17]. En [4] se presentó una reducción polinomial de QSAT al problema de determinar si la raíz de $\mathcal{T}$ está etiquetada con U. Por lo tanto, queda probada la completitud.

4.2 Relación con la Complejidad Descriptiva

La Complejidad Descriptiva permite comprender mejor la complejidad computacional de un problema de decisión, analizando cuán rico debe ser un lenguaje para expresar ese problema. Si bien la complejidad computacional no coincide con el poder expresivo de un lenguaje, determinar la clase de lenguaje requerido para resolver una consulta ayuda en la estimación del poder expresivo del lenguaje.

En este sentido, el problema de verificar si dada una estructura $\langle \mathcal{A}, h \rangle$ es un argumento que pertenece a P, puede ser expresado en lógica de primer orden extendida con el menor punto fijo $FO(LFP)$, ya que $FO(LFP) = P$[16]. Esto tiene su razón de ser, en que se requiere formalizar la noción de definición inductiva.

El problema de decisión de existencia de un argumento pertenece a NP y por lo tanto, puede ser expresado como una consulta booleana en lógica de segundo orden existencial $SO\exists$, ya que $SO\exists = NP$[14]. El problema de la inexistencia de un argumento para un literal pertenece a co-NP. Así, puede ser expresado como una consulta booleana en lógica de segundo orden universal, ya que $SO\forall = co - NP$[16].

Finalmente, el problema de decisión "dado un árbol dialéctico $\mathcal{T}$, determinar si la raíz está etiquetada con U", que está en PSPACE, equivalentemente en AP, puede ser expresado como una consulta booleana en lógica de primer orden extendida con el punto fijo parcial $FO(PFP)$, ya que $PSPACE = FO(PFP)$[16].

5 Análisis de consultas expresables con el operador de clausura transitiva

En las secciones anteriores hemos definido a DeLP como un lenguaje de consulta para el manejo de bases de datos relacionales y hemos relacionado su complejidad computacional con la complejidad descriptiva, con el objeto primordial de analizar su poder expresivo. En otras palabras, deseamos responder ¿cuáles relaciones pueden ser expresadas por el lenguaje y cuáles no?

Existen un conjunto de consultas representativas que son interesantes y que ayudan a determinar el poder expresivo de un lenguaje de consulta. En esta sección analizaremos las consultas que requieren de un operador de clausura transitiva para ser expresadas y su relación con el lenguaje de consulta basado en DeLP.

Clausura Transitiva (TC): Sea G un grafo dirigido finito y dos vértices v y v' en G. Consulta REACH: ¿existe un camino desde v a v'?

Esta consulta puede ser expresada en DATALOG, luego puede ser expresada en nuestro lenguaje de consultas rebatible. La siguiente consulta booleana expresa a REACH:

$$\{camino(X,Y) \leftarrow arco(X,Y) \quad camino(X,Y) \leftarrow camino(X,Z), camino(Z,Y)\}$$

y el átomo $camino(v, v')$. Nótese que Δ es vacío.

El problema REACH es NL-completo y $NL = FO(pos\, TC)$, siendo $FO(pos\, TC)$ la clausura de la Lógica de Primer Orden con ocurrencias de TC que nunca ocurran sobre una negación[16]. Luego el poder expresivo del lenguaje de consulta rebatible incluye a consultas expresables en $FO(pos\ TC)$, que coinciden con aquellas computables en NL[16].

Teorema 1. *Las consultas definibles a partir del lenguaje de consulta rebatible basado en DeLP son un subconjunto de las consultas expresables en $FO(TC)$.*
Demostración. Datalog puede expresar la clasusura transitiva de una relación [21]. Datalog puede ser caracterizado como un sublenguaje del lenguaje de consulta rebatible, donde $\Delta = \varnothing$ y las reglas tienen solamente átomos en cabeza y cuerpo. Así el conjunto de consultas expresables en Datalog, $FO(pos\ TC)$[19] también lo son en el lenguaje de consulta rebatible. Para bases de datos finitas ordenadas $FO(posTC) = FO(TC)$[16]. Luego queda probado el teorema.

Clausura Transitiva Determinística (DTC): Sea S un grafo dirigido finito. Consulta REACH_d: ¿existe un camino determinístico desde v a v'? Esto es, para cada vértice u, existe como máximo un vértice x para el que existe un arco (u, x).

DTC se define como sigue: sea $\varphi(\overline{x}, \overline{y})$ una relación de primer orden, donde $\overline{x}$ y $\overline{y}$ son tuplas de variables $\varphi_{dtc}(\overline{x}, \overline{y}) = \varphi(\overline{x}, \overline{y}) \wedge [(\forall \overline{z}) \neg\varphi(\overline{x}, \overline{z}) \vee \overline{y} = \overline{z}]$ Así, $\varphi_{dtc}(\overline{x}, \overline{y})$ es verdadera si $\overline{y}$ es el único descendiente de $\overline{x}$. Luego, $DTC\varphi \equiv TC\varphi_{dtc}$.

En este caso, la consulta puede hacerse teniendo en cuenta que deberemos complementar los todos las relaciones existentes en el vocabulario del lenguaje. Para cada esquema de relación R_i de aridad l_i deberemos agregar a la consulta la siguiente regla: $\neg R_i(X_1, \ldots, X_{l_i}) \multimap auxiliar(X_1, \ldots, X_{l_i})$. Por otra parte, si existe una relación de aridad n, ya sea en la parte intensional o extensional, entonces deberemos incorporar a Π_F una relación *auxiliar* de aridad n, instanciada con todos los objetos del universo. Puede probarse que una base de datos extendida con la relación auxiliar y la consulta incluyendo a la regla antes mencionada, la semántica basada en juegos permite caracterizar el comportamiento del completamiento de la base de datos.

Luego la consulta para resolver el caso REACH_d es:

$$\{caminod(X,Y) \leftarrow arcod(X,Y). \quad caminod(X,Y) \leftarrow caminod(X,Z), caminod(Z,Y)$$
$$arcod(X,Y) \leftarrow arco(X,Y) \wedge \neg otro\_arco(X,Y).$$
$$otro\_arco(X,Y) \leftarrow a(X,Z), Z \neq Y \quad \neg otro\_arco(X,Y) \prec auxiliar(X,Y)\}$$

y el átomo $caminod(v, v')$.

El conjunto de las consultas expresables en $FO(DTC)$ coincide con aquellas computables en L[16] y, dado que $L \subseteq NL$, luego están incluidas en el conjunto de consultas expresables en el lenguaje de consulta de rebatible.

6 Conclusiones y Trabajos Futuros

En este trabajo se ha propuesto a DeLP como lenguaje de consulta rebatible para bases de datos finitas ordenadas. Se definió su sintaxis adaptando conceptos heredados de las bases de datos relacionales. De igual modo, se introdujo la semántica de una consulta general y de una consulta booleana a la base de datos, basándonos en la semántica basada en juegos.

Asimismo, se retomó el estudio de la complejidad computacional relacionando algunos resultados alcanzados, con las Máquinas de Turing Alternantes, obteniendo como resultado principal que el problema de decisión: "dado un árbol dialéctico $\mathcal{T}$, determinar si la raíz está etiquetada con U" es AP-completo. A fin de orientar el estudio del poder expresivo, se relacionó a los resultados en complejidad temporal y espacial con la Complejidad Descriptiva.

Finalmente, en cuanto al poder expresivo, se individualizaron dos problemas que requieren de la clausura transitiva para ser expresados y se expresaron las consultas en el nuevo lenguaje de consulta rebatible. Una de las contribuciones principales es la demostración de que el conjunto de las consultas expresables en el lenguaje de consulta rebatible está incluido en el conjunto de las consultas $FO(TC)$, que coincide con las consultas de complejidad computacional NL.

Entre nuestros trabajos futuros, están analizar si las fórmulas de primer orden extendidas con el menor punto fijo y las fórmulas de segundo orden extendidas con el operador de clausura transitiva son expresables en el lenguaje de consulta rebatible. Asimismo, se pretende relacionar estos resultados con las complejidades de dato y combinada, concepto introducido en [3].

Referencias

1. Sanjeev Arora and Boaz Barak, editors. *Computational Complexity: a modern approach*. Cambridge University Press, New York, 2009.
2. Laura A. Cecchi, Pablo R. Fillottrani, and Guillermo R. Simari. An Analysis of the Computational Complexity of DeLP through Game Semantics. In *XI Congreso Argentino de Ciencias de la Computación*, pages 1170–1181, Argentina, Octubre 2005. Universidad Nacional de Entre Ríos.
3. Laura A. Cecchi, Pablo R. Fillottrani, and Guillermo R. Simari. On the complexity of DeLP through game semantics. In J. Dix and A. Hunter, editors, *XI International Workshops on Nonmonotonic Reasoning*, pages 386–394, Clausthal University, 2006.

4. Laura A. Cecchi and Guillermo Simari. *DeLP marking procedure for dialectical trees is PSPACE-complete*. EDULP - Editorial de la Universidad de La Plata, 2012. En Prensa.
5. Laura A. Cecchi and Guillermo R. Simari. Sobre la Relación entre la Definición Declarativa y Procedural de Argumento. In *VI CACiC*, pages 465–476, Ushuaia, 2000.
6. Laura A. Cecchi and Guillermo R. Simari. Sobre la relación entre la Semántica GS y el Razonamiento Rebatible. In *X CACiC - Universidad Nacional de La Matanza*, pages 1883–1894, San Justo - Pcia. de Buenos Aires, 2004.
7. Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.
8. C. Chesñevar and A. Maguitman. An Argumentative Approach to Assessing Natural Language Usage based on the Web Corpus. In *Proc. of the European Conference on Artificial Intelligence (ECAI) 2004*, pages 581–585, Valencia, Spain, August 2004.
9. C. Chesñevar and A. Maguitman. ARGUENET: An Argument-Based Recommender System for Solving Web Search Queries. In *Proc. of the 2nd IEEE Intl. IS-2004 Conference*, pages 282–287, Varna, Bulgaria, June 2004.
10. Cristhian A. D. Deagustini, Santiago E. Fulladoza Dalibon, Sebastian Gottifredi, Marcelo A. Falappa, Carlos I. Chesñevar, and Guillermo R. Simari. Supporting defeasible argumentation processes over relational databases. *Lecture Notes in Computer Science. Proceedings of The Ninth International Workshop on Argumentation in Multi-Agent Systems, ArgMAS'2012*, 2012.
11. Cristhian A. D. Deagustini, Santiago E. Fulladoza Dalibon, Sebastian Gottifredi, Marcelo A. Falappa, and Guillermo R. Simari. Consistent Query Answering Using Relational Databases Through Argumentation. *Lecture Notes in Computer Science.The Twenty-third International Conference on Database and Expert Systems Applications, DEXA'2012*, 7447, 2012.
12. Paul E. Dunne and Michael Wooldridge. *Complexity of Abstract Argumentation*, pages 85–103. Springer, 2009.
13. Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Springer-Verlag, 1995.
14. Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. Karp, editor, *Complexity of Computation. SIAM-AMS Proceedings*, volume 7, pages 43–73, 1974.
15. Alejandro J. García and Guillermo R. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.
16. Neil Immerman, editor. *Descriptive Complexity*. Springer-Verlag, New York, 1999.
17. Christos Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.
18. John Pollock. Defeasible Reasoning. *Cognitive Science*, 11:481–518, 1987.
19. Peter Schauble and Beat Wuthrich. On the Expressive Power of Query Languages. *ACM Transactions on Information Systems*, 12(1):69–91, 1994.
20. Guillermo R. Simari and Ronald P. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 53:125–157, 1992.
21. Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press, 1988.