El marcado de un árbol dialéctico en DeLP es PSPACE-completo

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

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

lcecchi@uncoma.edu.ar 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. Su semántica operacional está basada en un análisis dialéctico donde argumentos a favor y en contra de un literal interactúan, construyendo un árbol dialéctico. En este trabajo, nos enfocamos al estudio de la complejidad del proceso de marcado del árbol dialéctico, con el cual se determina si su raíz fue derrotada. Este punto es de gran importancia en DeLP, ya que permite determinar si el literal soportado por el argumento raíz del árbol será considerado entre las creencias de un agente que razona. Hemos probado que el marcado del árbol dialéctico es PSPACE-completo.

Palabras Clave: Sistemas Argumentativos, Razonamiento Rebatible, Programación en Lógica, Complejidad

1 Introducción

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 [9]. 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].

La teoría de la complejidad se ha vuelto una herramienta muy importante para comparar diferentes formalismos y para ayudar a mejorar las implementaciones, siempre que sea posible. Por esta razón es importante analizar la complejidad computacional y el poder expresivo de DeLP, ya que la primera nos

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

indica cuán difícil es responder una consulta, mientras que la segunda da una caracterización precisa de los conceptos que son posibles definir como consultas.

Históricamente, las implementaciones de los sistemas de argumentación han sido limitados al área legal en la que no hay restricciones de respuesta en tiempo real [11, 14]. Sin embargo, en los últimos años, varias aplicaciones han sido desarrolladas e implementadas usando sistemas argumentativos. Por ejemplo, aplicaciones relacionadas con sistemas multiagentes y con búsqueda en la web [5–7]. La escalabilidad y la robustez de tales enfoques depende fuertemente de las propiedades computacionales de los algoritmos que los soportan. Por lo tanto, es necesario estudiar estas propiedades con el objeto de expandir los campos de aplicación de los sistemas argumentativos.

Algunos resultados sobre complejidad computacional han sido presentados sobre frameworks abstractos de argumentación, basados en las semánticas admisible y preferida (ver [8]). Sin embargo, estos resultados no se aplican directamente a DeLP.

En este trabajo, se presenta el estudio del marcado de un árbol de decisión en DeLP. Este proceso es fundamental en DeLP pues permite determinar si un literal pertenece a las creencias de un agente. Considerando la analogía de un árbol dialéctico con los juegos y con el problema de decisión QSAT se demuestra que el marcado de un árbol dialéctico es PSPACE-completo.

El resto de este trabajo está estructurado como sigue. Primero introducimos brevemente los fundamentos de DeLP, y describimos su semántica operacional. Luego, analizamos la complejidad computacional del marcado del árbol dialéctico. En este sentido, se explica la analogía existente con el problema de decisión QSAT, se presenta una reducción y se demuestra que el marcado es PSPACE-completo. Finalmente, resumimos la contribución de este trabajo y presentamos futuras líneas de investigación.

2 DeLP

En esta sección introducimos algunos conceptos básicos de DeLP (ver [9] 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 ground*[4] L_0*, y cuyo segundo miembro* Cuerpo *es un conjunto finito de literales ground* $\{L_1, \ldots L_n, n > 0\}$*. 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 ground* L_0*, y cuyo segundo miembro*

[4] Ground: que no contiene variables. Por no encontrar un vocablo en español que se ajuste correctamente, utilizaremos el término en inglés.

Cuerpo *es un conjunto finito no vacío de literales ground* $\{L_1, \ldots L_n, n > 0\}$. *Un* programa lógico rebatible *(defeasible logic program)* $\mathcal{P}$*, que abreviaremos de.l.p., es un conjunto finito de reglas estrictas y hechos* Π *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.

La semántica operacional de DeLP está basada en los desarrollos de sistemas argumentativos rebatibles [11, 13]. Un *argumento* para un literal L es un subconjunto minimal de Δ que junto con Π deriva consistentemente L. La noción de derivación corresponde a la derivación usual SLD usada en la programación en lógica, ejecutada en un proceso backward sobre las reglas estrictas y rebatibles, donde los átomos negados son tratados como nuevos átomos en la signatura subyacente. Así, un agente puede explicar un literal L, a través de su argumento.

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 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. Por otra parte, se exige que el conjunto de los argumentos que soportan a L (en una rama del árbol dialéctico, los argumentos en posición impar) sea consistente. Idénticamente, se exige que el conjunto de los argumentos en contra (posición par en la rama) también sea consistente. Finalmente, si dos argumentos se derrotan entre sí, se considera una derrota por bloqueo. Si un argumento A es derrotado por otro B por bloqueo, luego A tiene a B como hijo en el árbol dialéctico. En estos casos, se exige que los derrotadores de B que se agregan al árbol como hijos de B, no sean derrotadores por bloqueo, esto es, los nodos hijos de B no pueden ser derrotados por B.

Un criterio de comparación es necesario para determinar cuando un argumento derrota a otro. Aunque existen varias relaciones de preferencia, en este trabajo nos abstraemos de este tema.

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. La siguiente definición especifica el procedimiento que marca a los nodos de un árbol dialéctico con las etiquetas "U", no derrotado, o "D", derrotado.

Definición 2. *Sea* $\mathcal{T}$ *un árbol de dialéctica cuya raíz es el argumento* $\mathcal{A}$ *de* L. *Un árbol de dialéctica marcado, denotado* $\mathcal{T}^*$ *puede obtenerse marcando cada nodo en* $\mathcal{T}$ *de la siguiente forma:*

1. *Todas las hojas de* $\mathcal{T}$ *se marcan con "U" en* $\mathcal{T}^*$.

2. *Sea N un nodo interno de $\mathcal{T}$. El nodo N se marca con "U" si todo nodo hijo de N está marcado con "D", y N se marca con "D" si existe al menos un nodo hijo de N marcado con "U".*

Un agente creerá en un literal L, si L es un literal garantizado, i.e., si existe un árbol dialéctico cuya raíz es un argumento que soporta a L y el proceso de marcado finaliza con la raíz etiquetada con U.

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

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 (la raíz del árbol dialéctico está etiquetada con U) 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 [3, 4, 1, 2]).

3 Complejidad del marcado del árbol dialéctico

DeLP es un sistema de razonamiento donde cada consecuencia de un *de.l.p.* es analizada considerando todos los argumento a favor y en contra de él. De este modo se construye un árbol dialéctico a través del cual se analiza si un literal está garantizado. Este árbol dialéctico es análogo a la idea de un juego entre dos participantes: un proponente y un oponente. Así es posible definir declarativamente si existe una estrategia ganadora[4, 1, 2] para el proponente, esto es, si el proceso de marcado del árbol dialéctico termina etiquetando a la raíz con U. Un problema de decisión ampliamente estudiado que se relaciona con los juegos es QSAT.

El problema de decisión QSAT o *satisfacibilidad cuantificada* se define como sigue: dada una fórmula Booleana ϕ en forma normal conjuntiva con las variables Booleanas $x_1, x_2, \ldots x_n$, ¿es la fórmula en forma normal prenex(los cuantificadores aparecen todos a la izquierda de la fórmula) $\exists x_1 \forall x_2 \exists x_3 \ldots Q_n x_n \phi$ satisfacible? siendo Q_i uno de los dos cuantificadores $\forall$ o $\exists$ según i sea par o impar.

QSAT es PSPACE-complete[10] y es interesante ya que tiene una gran similitud con un juego de dos jugadores: $\exists$ y $\forall$. Los dos jugadores mueven en forma alternativa, siendo $\exists$ quien mueve primero. Un movimiento consiste en determinar el valor de verdad de la siguiente variable, siendo $\exists$ el que fija el valor para x_i si i es impar y $\forall$ el que fija el valor si i es par. Después de n movimientos, número de variables, uno de los dos jugadores gana el juego.

Así, para juegos generalizados tenemos el siguiente problema de decisión: dada una situación legal del juego y un jugador, ¿tiene ese jugador una estrategia ganadora? Por ejemplo, si tenemos los jugadores A y B y comienza A, luego la

pregunta puede ser expresada como:

$$\exists \text{ movimiento para } A \; \forall \text{ movimiento para } B \; \exists \ldots (A \text{ gana})$$

Nótese que esto presenta una semejanza con la forma en que se hace el marcado de un árbol dialéctico, en el que el jugador $\exists$ intenta hacer satisfacible a la fórmula sin ser derrotado y el jugador $\forall$ trata de hacer insatisfacible a la fórmula.

Con el objeto de probar que el marcado del árbol dialéctico es PSPACE-completo debemos encontrar una reducción de un problema PSPACE-completo a él. Así comenzaremos con la definición de la transformación del problema de decisión QSAT a éste.

Definición 3. *Definimos la siguiente transformación de una fórmula Booleana cuantificada en forma normal conjuntiva prenex* $\exists x_1 \forall x_2 \exists x_3 \ldots Q_n x_n \phi$ *a un árbol dialéctico:*

- *Los nodos del árbol dialéctico son:*
  n es impar $\{\phi, \neg\phi, x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n, C_1, \ldots C_m, \neg C_1, \ldots \neg C_m\}$
  n es par $\{\phi, \neg\phi, x_1, \ldots, x_n, x_{n+1}, \neg x_1, \ldots, \neg x_n, C_1, \ldots C_m, \neg C_1, \ldots \neg C_m\}$
  siendo ϕ *la fórmula booleana libre de cuantificadores,* x_i, $1 \leq i \leq n$ *las variables en* ϕ, x_{n+1} *un argumento auxiliar y* $C_1, \ldots C_m$ *todas las cláusulas que aparecen en* ϕ, *esto es,* $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$.
- *La relación de ataque y derrota entre los nodos se define del siguiente modo:*
  n es impar $\{(\neg\phi, \phi), (x_1, \neg\phi), (\neg x_1, \neg\phi), (x_2, x_1), (x_2, \neg x_1), (\neg x_2, x_1), (\neg x_2, \neg x_1), (x_3, x_2), (x_3, \neg x_2), (\neg x_3, x_2), (\neg x_3, \neg x_2), \ldots, (x_n, x_{n-1}), (x_n, \neg x_{n-1}), (\neg x_n, x_{n-1}), (\neg x_n, \neg x_{n-1})\}$

  n es par $\{(\neg\phi, \phi), (x_1, \neg\phi), (\neg x_1, \neg\phi), (x_2, x_1), (x_2, \neg x_1), (\neg x_2, x_1), (\neg x_2, \neg x_1), (x_3, x_2), (x_3, \neg x_2), (\neg x_3, x_2), (\neg x_3, \neg x_2), \ldots, (x_n, x_{n-1}), (x_n, \neg x_{n-1}), (\neg x_n, x_{n-1}), (\neg x_n, \neg x_{n-1})(x_{n+1}, x_n), (x_{n+1}, \neg x_n)\}$

  Finalmente, en ambos casos, cada cláusula C_i *ataca y derrota a* x_n *si* n *es impar y a* x_{n+1} *si* n *es par y si* C_i *es verdadera bajo la asignación de valores de verdad en esa rama, entonces* C_i *es derrotada por* $\neg C_i$.

Ejemplo 1. Consideremos la siguiente fórmula cuantificada

$$\phi \; = \; \exists x_1 \forall x_2 \exists x_3 (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_2 \vee \neg x_3)$$

El árbol dialéctico que se generaría es el que se muestra en las Figuras 1 y 2.

El nodo raíz es la fórmula booleana que es derrotada por su negación. A partir de este punto, cada variable derrota a la variable del nivel anterior. Cada nodo correspondiente a la variable es el equivalente a indicar si es verdadera (nodo x_i) o falsa (nodo $\neg x_i$).

Obsérvese que cada cláusula en la fórmula derrota a la última variable. Si la cláusula es verdadera en esa rama (según como se fue asignando los valores de verdad de las variables), entonces es derrotada por la negación de esa cláusula.

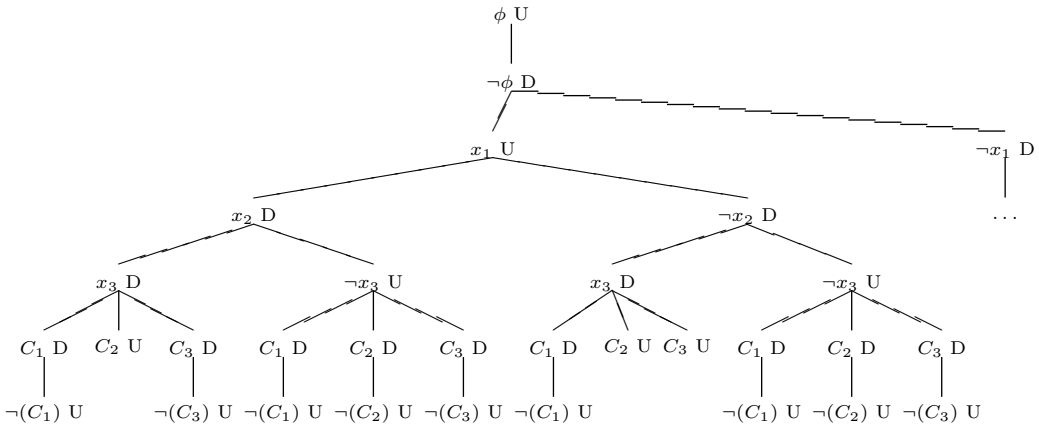

Fig. 1. Árbol dialéctico parcial etiquetado de la fórmula Booleana donde $C_1 = x_1 \vee x_2$, $C_2 = \neg x_1 \vee \neg x_3$ y $C_3 = x_2 \vee \neg x_3$.

El árbol muestra el marcado comenzando en las hojas con U. Nótese que aquella asignación que hace verdadera a todas las cláusulas termina asignando a la jugada de la variable existencial la etiqueta U y las asignaciones que no hacen a todas las cláusulas verdaderas asignan a la variable existencial D.

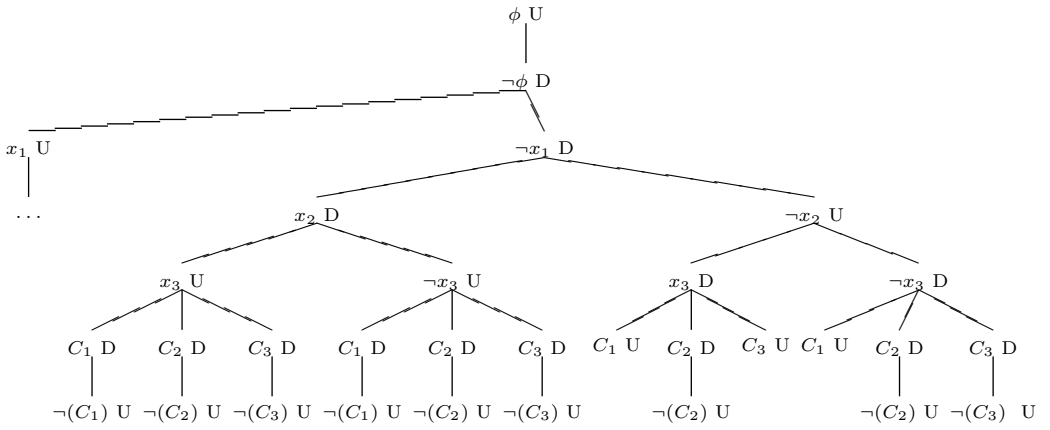

Fig. 2. Árbol dialéctico parcial etiquetado de la fórmula Booleana donde $C_1 = x_1 \vee x_2$, $C_2 = \neg x_1 \vee \neg x_3$ y $C_3 = x_2 \vee \neg x_3$.

En la Figura 1 se ve que existe un valor para $x_1 = true$, tal que para todo valor de x_2, existe un valor para x_3 (para $x_2 = true$, $x_3 = false$ y para $x_2 = false$, $x_3 = false$) que satisface a la fórmula. En la Figura 2 se observa que para $x_1 = false$ no se satisface la fórmula para todos los valores de x_2, en particular no se satisface para $x_2 = false$.

Recordemos que en el marcado del árbol dialéctico el nodo padre se marca con U cuando *todos sus hijos* están marcados con D. Intuitivamente, quisiéramos

poder utilizar esto cuando el nodo padre corresponde a una variable cuantificada universalmente, de este modo controlamos que *todos los hijos* satisfagan a la fórmula.

Por otro lado, un nodo padre se etiqueta con D cuando al menos un hijo está etiquetado con U. Éste es el comportamiento que deseamos de un padre que corresponde a una variable cuantificada existencialmente.

Intuitivamente, lo que buscamos es que el marcado de los niveles que juega $\exists$ quede todo etiquetado con D, para que el nivel superior, correspondiente al jugador $\forall$ quede etiquetado con U y así sucesivamente.

Ejemplo 2. Consideremos la siguiente fórmula cuantificada

$$\phi \ = \ \exists x \forall y \ (x \vee y) \wedge (\neg x \vee \neg y)$$

El árbol dialéctico que se generaría es el que se muestra en la Figura 3. La fórmula es insatisfacible y el árbol termina etiquetado con D.

En este caso la cantidad de variables es par, por lo que requerimos un nodo auxiliar para hacer que en el árbol dialéctico la primer variable analizada corresponda al jugador $\forall$.

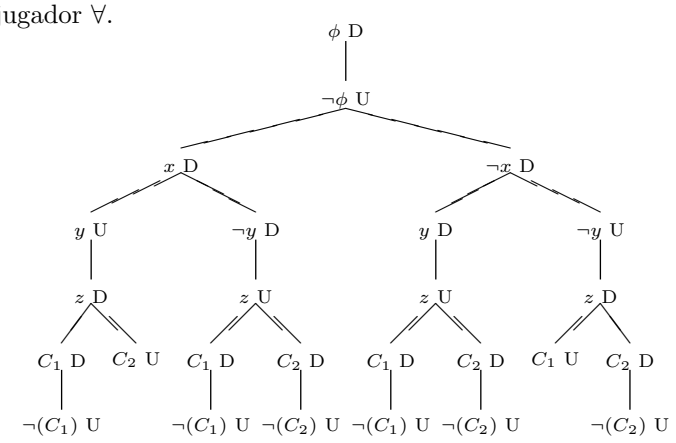

Fig. 3. Árbol dialéctico etiquetado de la fórmula Booleana $\phi \ = \ \exists x \forall y (x \vee y) \wedge (\neg x \vee \neg y)$, donde $C_1 = x \vee y$ y $C_2 = \neg x \vee \neg y$ y z es la variable auxiliar.

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

Demostración. **Pertenencia a** PSPACE**:** El árbol dialéctico es finito por definición, ya que el *de.l.p.* es finito y no puede haber ciclos, ya que no está permitido argumentar dos veces el mismo argumento. Recorrer un árbol en forma depth-first requiere espacio polinomial [12]. Así, recorrer el árbol dialéctico requiere espacio polinomial en el número de argumentos del *de.l.p.*.

Completitud: Sea ϕ una fórmula booleana instancia del problema de decisión QSAT. Debemos probar que ϕ es satisfacible si y solamente si el árbol dialéctico está etiquetado con U.

Para probar este resultado, utilizaremos la transformación definida anteriormente desde una fórmula Booleana instancia de QSAT a un árbol dialéctico. Dicha transformación puede ser realizada en espacio polinomial.

Si ϕ es satisfacible entonces en algún subárbol de la transformación de ϕ, los últimos dos niveles son de la forma presentada en la Figura 4. Esto es, todas las cláusulas $C_1, \ldots, C_m$ son verdaderas, luego el nodo A está etiquetado con U.

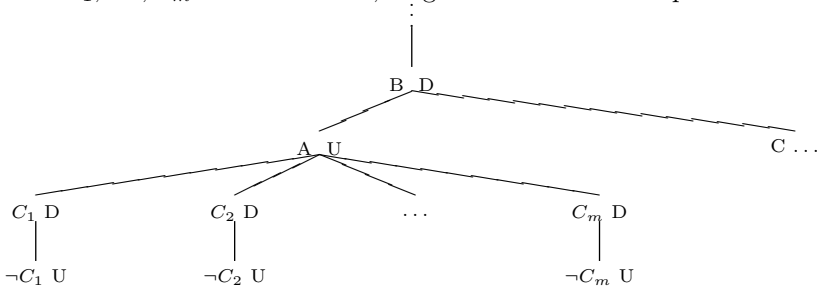

Fig. 4. Forma del árbol de decisión cuando las clásusulas son todas verdaderas.

- Si la fórmula contiene un número n impar de variables, luego el nodo A corresponde con la representación de algún valor de verdad de x_n e indica que existe un valor para x_n para el cual ϕ es verdadera. El nivel del padre de A, estaría etiquetado con D y la rama que lleva a C no deberíamos recorrerla pues estamos analizando que $\forall x_{n-1} \exists x_n$ y para el valor tomado por el nodo B existe un valor de x_n (valor en A).
  Para que la fórmula sea satisfacible este proceso debería repetirse en el hermano de B con el valor de verdad alternativo de x_{n-1}. Así ambos estarían etiquetados con D y su padre existencial con U. El proceso se repite hasta el primer nivel que representa un cuantificador universal, que siguiendo lo explicado anteriormente estará etiquetado con U y por lo tanto, $\neg\phi$ etiquetada con D y finalmente, ϕ con U.
- Si la fórmula contiene un número n par de variables entonces el nodo A es un auxiliar y el nodo B está etiquetado con D. La situación es la que se presenta en la Figura 5.
  El nivel de B es la representación de x_n con un valor de verdad y ese nivel de nodos representa a $\forall x_n$. Dado que la fórmula es satisfacible debe repetirse el proceso con el valor alternativo a B (nodo C), que quedará etiquetado con D. Así, $\forall x_n$ se satisface y el nodo E, que representa a $\exists x_{n-1}$, queda etiquetado con U. El proceso se repite como en el caso anterior y la raíz queda etiquetada con U.

Si la raíz del árbol dialéctico obtenido transformando a una fórmula Booleana instancia de QSAT está etiquetada con U, entonces el segundo nivel está marcado

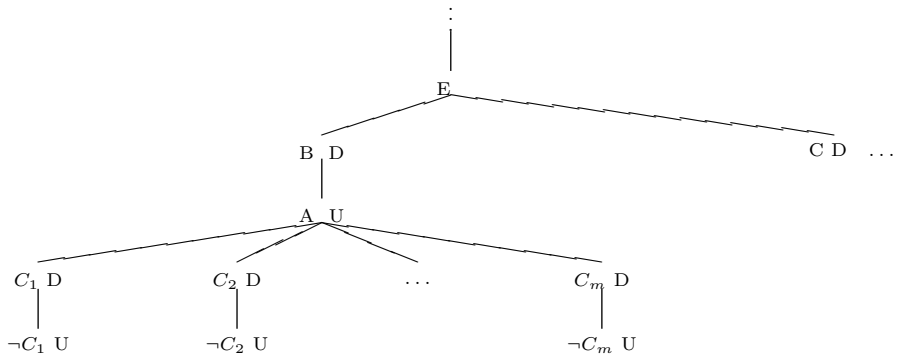

Fig. 5. Árbol dialéctico que representa a una fórmula boolena con un número n par de variables

con D (nodo que representa a $\neg\phi$). El siguiente nivel representa un cuantificador existencial, así alguno de los nodos en ese nivel debe estar etiquetado con U y los hijos de tal nodo deben estar todos etiquetados con D, correspondiendo con un cuantificador universal y así sucesivamente.

- Si la fórmula tiene una cantidad impar de variables, entonces algún nodo que represente $\exists x_n$ estará etiquetado con U y sus hijos (las cláusulas de la fórmula) con D. Así es que los nodos que representan a las cláusulas no pueden ser hojas y por lo tanto son verdaderas.
- Si la fórmula tiene una cantidad par de variables, entonces los nodos que representen $\forall x_n$ estarán etiquetados con D y su hijo auxiliar con U. Finalmente, las cláusulas de la fórmula con D. Así es que los nodos que representan a las cláusulas no pueden ser hojas y por lo tanto son verdaderas.

Por lo tanto, el marcado de un árbol dialéctico es PSPACE-completo.

4 Conclusiones y Trabajos Futuros

En este trabajo hemos analizado la complejidad del marcado del árbol dialéctico que realiza DeLP. Este punto es central en DeLP ya que permite determinar si un literal está garantizado y, por lo tanto, pertenecerá a las creencias de un agente que razona.

En este sentido, se mostró que existe una similitud entre la construcción del árbol dialéctico y un juego y se aprovechó la semejanza entre un juego y el problema de decisión QSAT. La principal contribución de este trabajo es la demostración de que el marcado de un árbol de dialéctica es PSPACE-completo.

En [1, 2] se introducen algunos problemas de decisión relevantes para DeLP y se presentan resultados en cuanto a la complejidad computacional. Asimismo, se estudia a DeLP como lenguaje de consulta, con el objeto de utilizar a DeLP sobre tecnologías de bases de datos.

Entre nuestros trabajos futuros, están determinar la complejidad combinada que se introdujo en[2] utilizando el resultado que hemos presentado en este trabajo. Asimismo, se desea estudiar la complejidad descriptiva de DeLP, esto es la clase de consultas expresables en DeLP.

Referencias

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. C. Chesñevar, A. Maguitman, and G. Simari. Argument-based critics and recommenders: A qualitative perspective on user support systems. *Data & Knowledge Engineering*, 2005.
8. Paul E. Dunne and Michael Wooldridge. *Complexity of Abstract Argumentation*, pages 85–103. Springer, 2009. Argumentation in Artificial Intelligence- Chapter 5.
9. Alejandro J. García and Guillermo R. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.
10. Christos Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.
11. John Pollock. Defeasible Reasoning. *Cognitive Science*, 11:481–518, 1987.
12. Stuart Russell and Peter Norvig. *Artificial Intelligence: A modern approach*. Prentice Hall, New Jersey, third edition, 2010.
13. Guillermo R. Simari and Ronald P. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 53:125–157, 1992.
14. B. Verheij. Argumed - a template-based argument mediation system for laweyers. In J.C. Hage, T. J. Bench-Capon, A.W. Koers, C.N.J. de Vey Mestdagh, and C.A. Grütters, editors, *Legal Knowledge Based Systems. JURIX: The Eleventh Conference*, pages 113–130, 1998.