Argumentación Temporal utilizando ℓ-DeLP

M. C. D. Budán[1,2,3] **M. J. Gómez Lucero**[1,2] **A. J. García**[1,2]
G. R. Simari[2]

[1] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
[2] Laboratorio de Investigación y Desarrollo de Inteligencia Artificial (LIDIA),
Dep. de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur,
Bahía Blanca, CP 8000, Argentina
[3] Dep. de Informática, Universidad Nacional de Santiago del Estero,
Santiago del Estero, CP 4200, Argentina
{mcdb, mjg, ajg, grs}@cs.uns.edu.ar

Resumen En este trabajo se presenta una forma de argumentación en la que se considera la validez temporal de los argumentos involucrados. La herramienta utilizada es una extensión existente de *Defeasible Logic Programming* denominada *Labeled-DeLP*. Esta extensión permite considerar intervalos de tiempo en los que las cláusulas que componen un determinado argumento están "activas", permitiendo describir por extensión la vigencia temporal de estas cláusulas. Asi se determinan los intervalos de tiempo en los que el argumento esta activo. Al efectuar una consulta, el sistema deberá tener en cuenta el intervalo de tiempo para el que se está efectuando la misma, analizando cuales son los argumentos que están activos en dicho intervalo.

Palabras Clave: Argumentación Temporal, Argumentación Abstracta, Razonamiento Rebatible.

1. Introducción

Argumentar es un proceso que consiste en defender una afirmación dando razones para su aceptación; tanto la afirmación original como el soporte ofrecido para ella están abiertos a la duda manifestada por afirmaciones contrapuestas. Es usual denominar *argumento* a la estructura compuesta por una afirmación o *conclusión*, junto con el soporte ofrecido o *premisas* y la descripción de la forma en como las premisas soportan a la afirmación. Este último componente se denomina comúnmente *inferencia*. El proceso de argumentación puede presentarse como un juego entre dos participantes: un *proponente* cuyo rol es apoyar al argumento original y un *oponente* cuya actividad tiende a interponerse con el argumento original . Los argumentos del proponente se denominan argumentos *pro* y los del oponente se denominan *op*. Un argumento ofrecido para socabar otro argumento se denomina *contraargumentos*. Así, el proceso comienza con la introducción de un argumento *pro* por parte del proponente, luego el oponente ofrece sus contraargumentos. En este momento el proponente se transforma en oponente de su contrincante y ofrece contraargumentos para los contraargumentos introducidos. El proceso continúa de esta manera hasta agotarse.

La intuición que permite basar el razonamiento en un proceso argumentativo es que una afirmación es creíble si se puede argumentar a su favor con éxito, *i.e.*, en el caso de que existan contra-argumentos, estos pueden ser derrotados. En otras palabras, un agente racional creerá o no en una sentencia dependiendo de que los argumentos que soporten dicha declaración puedan defender exitosamente de todos los posibles ataques producidos por otros argumentos. En base a esto, existen los sistemas argumentativos que constituyen una formalización del razonamiento rebatible, en el cual las conclusiones obtenidas pueden ser rechazadas ante la aparición de una nueva evidencia.

Inicialmente propuesto por Dung [3], el modelo de argumentación abstracta se basa en considerar un conjunto de argumentos y una relación binaria definida sobre el conjunto de todos los posibles argumentos que representa el ataque de un argumento hacia otro. Así, el formalismo considera una dupla $AF = (AR, ATTACKS)$, donde AR es el conjunto de argumentos considerados y $ATTACKS$ representa la relación de ataque definida sobre AR. Este modelo gira en torno de dos nociones básicas: la de "aceptabilidad" y la de "admisibilidad"de un conjunto de argumentos.

Existen muchas extensiones que surgen a partir de este formalismo abstracto, entre ellas se encuentra el timed abstract framework [1,2]. Este formalismo aporta la posibilidad de encontrar el intervalo en que un argumento es aceptable. En los argumentos temporales (*Timed Abstract Framework*) cada argumento tiene asociado un conjunto de intervalos de tiempo en los que este argumento está activo. De esta manera, al formalismo abstracto se le agrega un componente Av que define los intervalos de tiempo en los cuales los argumentos se encuentran activos, obteniendo una 3-tupla $(AR, ATTACKS, Av)$. El tiempo cobra así importancia en la determinación de la aceptabilidad de un argumento.

Una forma de argumentación en la que se considera la forma en que los argumentos están construidos, es la Programación en Lógica Rebatible (Defeasible Logic Programming o DeLP); este es un formalismo que combina la programación lógica y la argumentación rebatible. DeLP incorpora el formalismo de argumentación para tratar los conocimientos contradictorios. Este formalismo, permite la identificación de las piezas de conocimiento que están en contradicción y a través de un proceso dialéctico, decidir cuál de ellos prevalece.

El objetivo de este trabajo es instanciar el Timed Abstract Frameworks a través de DeLP definiendo los componentes que no aparecen en argumentation abstracta, *i.e.*, el lenguaje de representación, la construcción de un argumento en ese lenguaje, y las nociones de ataque y derrota. Extenderemos DeLP para establecer, para cada elemento del programa los intervalos de tiempo en el que está activo, y en función de esta información determinar cuando un argumento esta activo. Usaremos una extensión de DeLP conocida como ℓ-DeLP que utiliza el álgebra de etiquetas, y una función de propagación de las mismas, para manejar la meta-información asociada a cada uno de los argumentos de un árbol dialéctico.

2. Programación en Lógica Rebatible - DeLP

La programación en lógica rebatible (*Defeasible Logic Programming - DeLP*)[4], combina la programación lógica y la argumentación rebatible. DeLP extiende la programación en lógica convencional empleando la argumentación rebatible para capturar aspectos del razonamiento de sentido común difíciles de expresar en la programación en lógica convencional. Los programas lógicos rebatibles permiten expresar información potencialmente inconsistente o incompleta, pudiendo decidir entre metas contradictorias mediante un determinado criterio de preferencia.

Un programa lógico rebatible consiste de: (1) un conjunto Δ de reglas rebatibles de la forma $L \prec P_1, \ldots, P_n$, con $n \geq 0$, que representa información incompleta o tentativa, donde L y cada P_i son literales, y (2) un conjunto Π de reglas de la forma $L \leftarrow P_1, \ldots, P_n$, con $n \geq 0$, que representa información estricta, donde L y cada P_i son literales. En el caso particular cuando $n = 0$, se denotará como $L \leftarrow True$, y en esta oportunidad L se denomina hecho. Es importante tener en cuenta que el conjunto Π debe ser consistente ya que representa información concreta (indisputable). Por el contrario, el conjunto Δ puede ser o no consistente, esto se debe a que este conjunto representa información tentativa.

Definición 1 *Sea h un literal y $\mathcal{P} = (\Pi, \Delta)$ un programa lógico rebatible. Un* argumento *para h es un par $\langle A, h\rangle$, donde A es un conjunto de reglas rebatibles de Δ, tal que:*

1. *Existe una derivación rebatible para h a partir de $\Pi \cup A$.*
2. *$\Pi \cup A$ es no contradictorio, y*
3. *A es minimal, es decir, no existe un subconjunto propio A', $A' \subsetneq A$ que satisface (1) y (2).*

Un argumento $\langle B, q\rangle$ es un sub-argumento *de $\langle A, h\rangle$ si y sólo si, $B \subseteq A$.*

Este formalismo, permite la identificación de las piezas de conocimiento que están en contradicción y a través de un proceso dialéctico, decidir cuál de ellos es el que prevalece. Los argumentos pueden interactuar unos con otros de diferentes maneras a través de las relaciones de desacuerdo, contra-argumento y derrota. Estas nociones se definen a continuación:

Definición 2 *Sea $\mathcal{P} = (\Pi, \Delta)$ un programa. Dos literales h y h' están en* desacuerdo*, si y sólo si el conjunto $\Pi \cup \{h, h'\}$ es contradictorio.*

El ejemplo más simple de literales en desacuerdo son dos literales complementarios como "p" y "$\sim p$", ya que $\Pi \cup \{p, \sim p\}$ es contradictorio, cualquiera sea el conjunto Π. Sin embargo, dos literales que no sean complementarios, como "p" y "q", también pueden estar en desacuerdo, *e.g.*, si $\Pi = \{h \leftarrow p, \sim h \leftarrow q\}$.

Definición 3 *Sean $\langle A_1, h_1\rangle$, $\langle A_2, h_2\rangle$ dos argumentos en $\mathcal{P} = (\Pi, \Delta)$. Decimos que $\langle A_1, h_1\rangle$ contra-argumenta a $\langle A_2, h_2\rangle$ en el literal h si y sólo si existe un sub-argumento $\langle A, h\rangle$ de $\langle A_2, h_2\rangle$ tal que h y h_1 están en* desacuerdo*. El argumento $\langle A, h\rangle$ se llama sub-argumento de desacuerdo, y el literal h será el punto de contra-argumentación. Si $\langle A_1, h_1\rangle$ contra-argumenta a $\langle A_2, h_2\rangle$, entonces tam-*

bién se dirá que $\langle A_1, h_1\rangle$ *ataca a* $\langle A_2, h_2\rangle$*, o que* $\langle A_1, h_1\rangle$ *es un contra-argumento para* $\langle A_2, h_2\rangle$*.*

Dos argumentos en conflicto pueden compararse mediante el criterio de especificidad, que establece un orden de preferencia entre los mismos. Especificidad favorece argumentos que contengan mayor información o sustenten su conclusión en forma más directa. Seguidamente se enuncia su definición formal:

Definición 4 *Sean* $\langle A_1, h_1\rangle$, $\langle A_2, h_2\rangle$ *dos argumentos en* $\mathcal{P}$*, y* L *el lenguaje generado por* $\mathcal{P}$*. Se dice que* $\langle A_1, h_1\rangle$ *es* estrictamente más específico *que* $\langle A_1, h_1\rangle$*, denotado como* $\langle A_1, h_1\rangle \succ \langle A_2, h_2\rangle$*, si y sólo si:*

1. *Para todo* $e \in Sent(L)$ *que* e *activa no trivialmente* A_1*, i.e.,* $\Pi \cup \{e\} \cup A_1 \nvdash h_1$ *y* $\Pi \cup \{e\} \nvdash h_1$ *se verifica que* e *activa* A_2*, i.e.,* $\Pi \cup \{e\} \cup A_2 \nvdash h_2$*.*
2. *Existe* $e \in Sent(L)$ *que* e *no activa* A_1*, i.e.,* $\Pi \cup \{e\} \cup A_1 \nvdash h_1$*, y* e *activa no trivialmente* A_2*, i.e.,* $\Pi \cup \{e\} \cup A_2 \nvdash h_2$*, y* $\Pi \cup \{e\} \nvdash h_2$*.*

Definición 5 *Sean* $\langle A_1, h_1\rangle$, $\langle A_2, h_2\rangle$ *dos argumentos en* $\mathcal{P}$*. Se dice que* $\langle A_2, h_2\rangle$ *derrota a* $\langle A_1, h_1\rangle$ *en el literal* h *si y sólo si existe un sub-argumento* $\langle A, h\rangle$ *de* $\langle A_1, h_1\rangle$ *tal que:* $\langle A_2, h_2\rangle$ *contra-argumenta a* $\langle A_1, h_1\rangle$ *en el literal* h *y se cumple una de estas dos opciones:*

1. $\langle A_2, h_2\rangle$ *es estrictamente más específico que* $\langle A, h\rangle$ *(derrotador propio), o*
2. $\langle A_2, h_2\rangle$ *es incomparable con* $\langle A, h\rangle$ *(derrotador de bloqueo)*

La presencia de múltiples derrotadores para un argumento produce una ramificación en líneas de argumentación, dando origen a un árbol de derrotadores que se denominará árbol de dialéctica.

Definición 6 *Sea* $\langle A, h\rangle$ *un argumento en* $\mathcal{P} = (\Pi, \Delta)$*. Un* árbol de dialéctica *para* $\langle A, h\rangle$ *a partir de* $\mathcal{P}$*, denotado* $\mathcal{T}(\langle A, h\rangle)$*, se construye de la siguiente forma:*

1. *La raíz del árbol es etiquetada con* $\langle A, h\rangle$*.*
2. *Sea* N *un nodo del árbol etiquetado* $\langle A_n, h_n\rangle$*, y* $[\langle A, h\rangle, \langle A_1, h_1\rangle, \langle A_2, h_2\rangle, \ldots, \langle A_n, h_n\rangle]$ *la secuencia de etiquetas desde la raíz hasta* N*.*
   Sean $\langle B_1, q_1\rangle, \langle B_2, q_2\rangle, \ldots, \langle B_k, q_k\rangle$ *todos los derrotadores de* $\langle A_n, h_n\rangle$*. Para cada derrotador* $\langle B_i, q_i\rangle$ *con* $(1 \leq i \leq k)$*, tal que la línea de argumentación* $[\langle A, h\rangle, \langle A_1, h_1\rangle, \langle A_2, h_2\rangle, \ldots, \langle A_n, h_n\rangle, \langle B_i, q_i\rangle]$ *es aceptable, existe un nodo* N_i *hijo de* N *etiquetado con* $\langle B_i, q_i\rangle$*. Si no existe ningún derrotador* $\langle B_i, q_i\rangle$ *en tales condiciones, entonces el nodo* N *es una hoja.*

En este árbol, cada camino desde la raíz hasta una hoja corresponde a una línea de argumentación. Una vez construido el árbol de dialéctica se realiza es proceso de marcado del árbol para determinar la aceptabilidad o garantía de un determinado literal h.

Definición 7 *Sea* $Tree\langle A, h\rangle$ *el árbol de dialéctica para* $\langle A, h\rangle$*. El* árbol de dialéctica marcado*, denotado* $\mathcal{T}^*(\langle A, h\rangle)$ *se obtiene marcando cada nodo en* $\mathcal{T}(\langle A, h\rangle)$ *de la siguiente forma:*

1. *Todas las hojas de* $\mathcal{T}(\langle A, h\rangle)$ *se marcan con "*$\mathcal{U}$*" en* $\mathcal{T}^*(\langle A, h\rangle)$*.*

2. *Sea N un nodo interno de $\mathcal{T}(\langle A, h\rangle)$. El nodo N se marca con "U"si todos sus nodo hijos están marcados con "D", y N se marca con "D"si tiene al menos un nodo hijo marcado con "U".*

Definición 8 *Sea $\mathcal{P} = (\Pi, \Delta)$ un programa lógico rebatible y h un literal. Sea $\langle A, h\rangle$ un argumento para h, y $\mathcal{T}^*(\langle A, h\rangle)$ el árbol de dialéctica marcado asociado con $\langle A, h\rangle$. El literal h está* garantizado *si la raíz de $\mathcal{T}^*(\langle A, h\rangle)$ está marcada con "U".*

3. DeLP con Etiquetas (ℓ-DeLP)

Es posible añadir a los elementos del programa meta-información con algún propósito específico. Esta meta-información podría ser: probabilidad, grado de certeza, confiabilidad de la fuente, intervalos de tiempo, *etc.* Para ello, dado un dominio de etiquetas $\mathcal{L}$ se extiende el lenguaje de DeLP de manera tal de asociar un elemento de $\mathcal{L}$ a cada uno de los elementos de un programa $\mathcal{P}$ y por extensión a los argumentos que se pueden obtener del mismo. ℓ-DeLP [5] utiliza etiquetas para mantener la meta-información extendiendo DeLP de la siguiente manera:

1. Se agrega una etiqueta de $\mathcal{L}$ a cada elemento del programa.
2. Se introduce una función *ALS* (*Argument Label Synthesis*) que permite determinar la etiqueta asociada a un argumento en base a las etiquetas de los elementos que lo componen.

Dado un argumento $\langle A, a\rangle$, o bien a es un hecho en el programa y por lo tanto $A = \emptyset$, o a se derivó utilizando una regla r con cabeza a y cuerpo $b_1, b_2, \ldots, b_n$; en este último caso, A resulta de unir $B_1, B_2, \ldots, B_n$; donde $\langle B_i, b_i\rangle$ es sub-argumento de $\langle A, a\rangle$ para $1 \leq i \leq n$, además de la regla r si es que r es una regla rebatible. Así, la etiqueta asociada a $\langle A, a\rangle$ por *ALS* se define como, $\ell_{\langle A, a\rangle} = ALS([\,], \ell_a)$, donde ℓ_a es la etiqueta asociada con a si es un hecho, y $\ell_{\langle A, a\rangle} = ALS([\ell_{\langle B_1, b_1\rangle}, \ell_{\langle B_2, b_2\rangle}, \ldots, \ell_{\langle B_n, b_n\rangle}], \ell_r)$ en el caso general, donde ℓ_r y ℓ_{b_i}, $1 \leq i \leq n$, son las etiquetas asociadas con r y b_i, $1 \leq i \leq n$. *ALS* permite sintetizar la etiqueta asociada con un argumento a partir de las etiquetas de los elementos que lo componen.

4. Argumentación Abstracta

Phan Minh Dung[3] desarrolló una abstracción de la argumentación rebatible mediante los *frameworks de argumentación* (Argumentation Framework - *AF*). En un *AF* la estructura de los argumentos permanece sin especificar y el análisis se centra en la interacción existente entre los mismos. Un *framework de argumentación* se define como un par compuesto por un conjunto de argumentos y una relación binaria que representa los ataques entre argumentos. Un argumento es una entidad abstracta que cumple un determinado rol al estar relacionado con otros argumentos. Formalmente, un framework de argumentación

está definido como un par $AF = (AR, ATTACKS)$, en donde AR es un conjunto de argumentos, y $ATTACKS$ es una relación binaria en AR, es decir, $ATTACKS \subseteq AR \times AR$.

En primer lugar se presenta la noción de aceptabilidad. Un argumento A es aceptable para un agente racional G, si G puede defender a A de todos los ataques posibles, *i.e.*, el conjunto de los argumentos aceptados por G debe ser capaz de defenderse a sí mismo. Esta intuiciones se formalizan en las siguientes definiciones tomadas de las presentaciones originales [3].

Definición 9 *Sea $AF = (AR, ATTACKS)$ un framework de argumentación.*

- *Un conjunto $S \subseteq AR$ se denomina* libre de conflictos *si no existen dos argumentos $A, B \in S$ tal que A ataca a B, i.e., $(A, B) \notin ATTACKS$.*
- *Un argumento $A \in AR$ es* aceptable con respecto *a $S \subseteq AR$ si y sólo si para cada $B \in AR$ se cumple que si B ataca a A entonces B es atacado por S.*
- *Un conjunto libre de conflictos $S \subseteq AR$ es* admisible *si y sólo si cada argumento en S es aceptable con respecto a S.*
- *Sea $E \subseteq AR$ admisible, E es una* extensión completa *de AF si y sólo si E contiene cada argumento que es aceptable con respecto a E.*
- *Sea $E \subseteq AR$ admisible, E es una* extensión grounded *de AF si y solo si E es minimal con respecto a la inclusión de conjuntos, tal que es admisible y completa.*

A continuación se presentará un ejemplo de framework de argumentación:

$AF = (AR, ATTACKS)$
$AR = A, B, C, D, E, F, G$
$ATTACKS =$
$\{(B, A), (C, B), (E, A),$
$(G, E), (F, G), (G, D)\}$

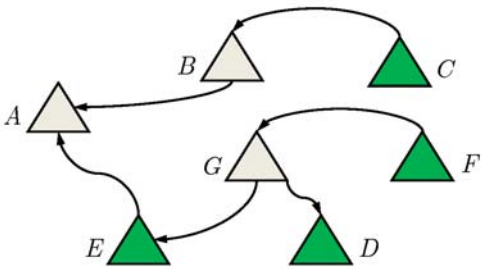

Figura 1: Framework de Argumentación

El conjunto $S = \{C, D, E, F\}$ es admisible, contiene todos los argumentos defensores de A, luego S es una extensión completa que además cumple con la propiedades necesarias para ser una extensión grounded.

5. Argumentación Abstracta Temporal

La argumentación abstracta temporal (Timed Abstract Frameworks - TAF) permite argumentar teniendo en cuenta el intervalo de tiempo específico en que los argumentos están activos. En TAF cada argumento tiene asociado un conjunto de intervalos de tiempo ampliando la Definición 9, agregando un tercer componente Av que obtiene los intervalos de tiempo correspondientes a un argumento, representándose como $TAF = (AR, ATTACKS, Av)$. Comenzaremos

definiendo la estructura temporal, asociando la línea de tiempo con la recta real y donde un intervalo de tiempo será un intervalo real.

Definición 10 *Un* intervalo de tiempo *I representa un período de tiempo continuo, que se identifica por un par ordenado de puntos temporales. El punto inicial es llamado* Punto de Inicio *de I, y el punto final es llamado* Punto de Fin *de I. El intervalo puede ser:*

1. Cerrado*: define un período de tiempo en el cual están incluidos los puntos que definen el intervalo (*Punto de Inicio *y* Punto de Fin*), denotados como $[a, b]$.*
2. Abierto*: define un período de tiempo en el cual no están incluidos los puntos que definen el intervalo (Punto de Inicio y Fin), denotados como (a, b).*
3. Semi-Abierto *o* Semi-Cerrado*: define un período de tiempo el cual incluye uno de los puntos que define el intervalo pero no ambos. Dependiendo cual es el que es incluido, son denotados de la siguiente forma $(a, b]$(si está incluido el punto de Fin) o $[a, b)$ (si está incluido el punto de Inicio).*

Definición 11 *Sean I_1, I_2 dos intervalos, su* intersección *es definida como: $I_1 \Cap I_2 = [x, y]$ donde $x, y \in I_1$ y $x, y \in I_2$, tal que, no hay $w, z \in I_1$ y $w, z \in I_2$ con $w < x$ o $y < z$.*

Definición 12 *Sean S_1, S_2 dos conjuntos de intervalos, su* intersección*, denotada $S_1 \Cap S_2$, se define como $S_1 \Cap S_2 = \{I : I = I_1 \Cap I_2 \neq [\ ]; \forall I_1 \in S_1; I_2 \in S_2\}$.*

Podemos ahora completar la definición del TAF.

Definición 13 *[***Framework de Argumentación con Tiempo - TAF***] es una 3-tupla $TAF = (AR, ATTACKS, Av)$ donde AR es un conjunto de argumentos, $ATTACKS$ es una relación binaria de ataque y Av es una función definida como $Av : AR \longrightarrow I(AR)$ que determina los intervalos de tiempo en el cual los argumentos se encuentran activos.*

En la Definición 13, la función Av define un conjunto de intervalos de tiempo para los cual un argumento se encuentra activo, a continuación se establece cómo se definirá cuáles son los tipos de intervalos de tiempo que es posible obtener y qué operaciones sobre los mismos son necesarias.

$AF = (AR, ATTACKS, Av)$
$AR = \{A, B, C, D, E, F, G\}$
$ATTACKS =$
$\{(B, A), (C, B), (E, A),$
$(G, E), (F, G), (G, D)\}$
$Av = \{A - [10, 50][80, 120]$
$B - [55, 75]$
$C - [40, 65]$
$D - [10, 30]$
$E - [20, 90]$
$F - [5, 30]$
$G - [10, 40]\}$

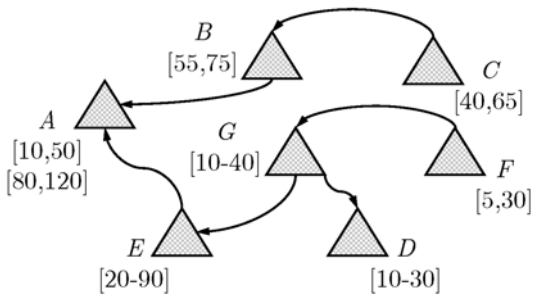

Figura 2: Argumentación Temporal

Ahora supongamos que se efectúa una consulta en la que se pregunte por el argumento A para determinar la aceptabilidad del mismo, en un intervalo $[100, 120]$. Analizando únicamente la relación $ATTACKS$ podríamos decir que

el argumento A no se encuentra aceptado, ya que F derrota a G, por lo tanto E se encuentra vivo y es quien derrota A. Ahora analizaremos el mismo ejemplo pero añadiendo al análisis el nuevo componente Av que son los intervalos de tiempo en los cuales los argumentos están activos.

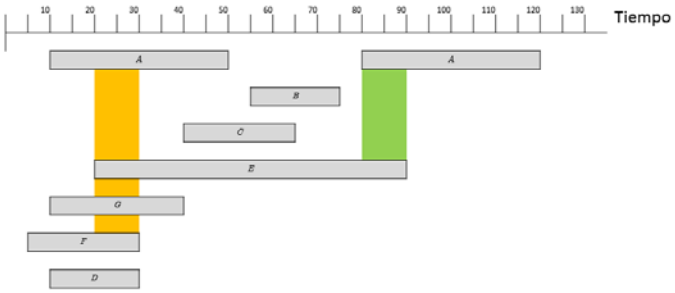

Figura 3: Representación de los argumentos en una línea de tiempo

La Figura 3 muestra que en el intervalo $[100, 120]$, el argumento A se encuentra activo y es aceptado en dicho intervalo. El ataque de E está activo en $[20, 90]$, y A en $[10, 50]$ y $[80, 120]$. Los intervalos para los cuales el argumento A está activo pero no es aceptado son los intervalos de tiempo $[20, 30]$ y $[80, 90]$ por los ataques del argumento E hacia A.

6. Tiempo con ℓ-DeLP

En base a lo presentado en los apartados anteriores, a través de ℓ-DeLP es posible manejar meta-información que puede estar asociada tanto a las cláusulas que componen un argumento como al mismo argumento. Para la aplicación de ℓ-DeLP es necesario definir 2 componentes: (1) el dominio de las etiquetas y (2) la función ALS (*Argument Label Sythesis*).

Definición 14 *Las cláusulas del programa tendrán asociado un conjunto de intervalos de tiempo. Los mismos se corresponderán a la Definición 10. Por otro lado, restringiremos a que los componentes de un intervalo de tiempo X, Y, deberán ser con $X, Y \geq 0$ y $X, Y \neq \infty$.*

Definición 15 *la función ALS será definida como $ALS([\ell b_1, \ell b_2, ..., \ell b_n], \ell_c) = INTER(\ell b_1, \ell b_2, ..., \ell b_n, \ell_c)$. Donde $INTER$ corresponde a la función de Intersección de la Definición 11 y Definición 12.*

De acuerdo a las definiciones introducidas con anterioridad sobre *Framework de Argumentación con Tiempo* y ℓ-DeLP la función Av puede definirse como: $Av(\langle A, h\rangle) = \ell\langle A, h\rangle$. En otras palabras, la función Av quedará definida como la intersección de los intervalos de tiempo asociados a las cláusulas que componen un argumento (ya sea rebatible o estricta) determinando en qué intervalo de tiempo dicho argumento es válido. Por ejemplo, en base al ejemplo presentado en la Figura 2 podemos contar con el siguiente programa DeLP que permite derivar los argumentos del conjunto $ARGS = \{A, B, C, D, E, F, G\}$.

Ejemplo 1 *Se presenta un ejemplo de un programa ℓ-DeLP con Etiquetas de Tiempo:*

$$\mathcal{P} =$$

$$\left\{\begin{array}{llll} a \prec s, c : [10,60]; [80,150] & n \prec c, l : [55,100] & \sim n \leftarrow m, t : [40,65] & m : [0,150] \\ s \prec j, l : [10,130] & \sim s \prec j, r : [0,90] & \sim f \leftarrow j, p : [5,30] & t : [0,150] \\ c \prec j, k : [0,120] & r \leftarrow l, p : [20,130] & r \leftarrow p, t : [10,30] & p : [0,150] \\ k \prec m, l : [10,50]; [70,140] & \sim r \leftarrow j, f : [10,60] & j : [0,150] & c : [0,150] \\ \sim k \prec j, n : [30,75] & f \leftarrow p, m : [0,40] & l : [0,150] & \end{array}\right\}$$

A partir de este programa en ℓ-DeLP, es posible obtener el siguiente framework de argumentación, donde estudiaremos el argumento A, por lo que se detallará la estructura de A a través de su árbol de derivación:

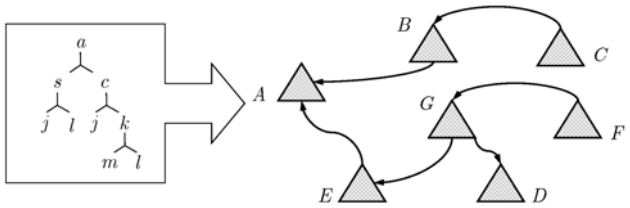

Figura 4: Árbol de Derivación - Framework de Argumentación

Una vez definido el árbol de derivación del argumento A, procederemos con la aplicación de la función $ALS([\ell b_1, \ell b_2, ..., \ell b_n], \ell_c) = INTER(\ell b_1, \ell b_2, ..., \ell b_n, \ell_c)$.

1. $\ell_k = ALS([\ell_m, \ell_l], \ell_{\ k \prec m, l}) = INTER(\ell_m, \ell_l, \ell_{\ k \prec m, l}) =$
   $INTER([0,150], [0,150], \{[10,50]; [70,140]\}) = \{[10,50]; [70,140]\}$
2. $\ell_c = ALS([\ell_j, \ell_k], \ell_{\ c \prec j, k}) = INTER(\ell_j, \ell_k, \ell_{\ c \prec j, k}) =$
   $INTER([0,150], \{[10,50]; [70,140]\}, [0,120]) = \{[10,50]; [70,120]\}$
3. $\ell_s = ALS([\ell_j, \ell_l], \ell_{\ s \prec j, l}) = INTER(\ell_j, \ell_l, \ell_{\ s \prec j, l}) =$
   $INTER([0,150], [0,150], [10,130]) = \{[10,130]\}$
4. $\ell_a = ALS([\ell_s, \ell_c], \ell_{\ a \prec s, c}) = INTER(\ell_s, \ell_c, \ell_{\ a \prec s, c}) =$
   $INTER([10,130], \{[10,50]; [70,120]\}, \{[10,60]; [80,150]\}) = \{[10,50]; [80,120]\}$

En base al programa en ℓ-DeLP presentando anteriormente y al árbol de derivación para el argumento A, luego de aplicar la función ALS sobre la derivación de A, podemos concluir que A se encuentra activo en los intervalos de tiempo $\{[10,50]; [80,120]\}$. Efectuando los mismos pasos para los demás árboles de derivación, podemos obtener el grafo con etiquetas de tiempo de la Figura 2

Ahora supongamos que se efectúa una consulta q en la que se pregunte si el literal a es aceptado en un intervalo de tiempo $[100-120]$. En la Figura 3 se puede ver que en el intervalo [100-120] el argumento A se encuentra activo y es aceptado, ya que no existe ningún contra-argumento en el intervalo a analizar. Consideremos ahora el intervalo $[80-90]$, el argumento A se encuentra activo, pero en el mismo intervalo también se encuentra activo el argumento E, y además existen literales en contradicción, entre ellos $(s, \sim s)$.

En los árboles de derivación para los argumentos A y E se observa que existe un conflicto en s y $\sim s$. Por ende, estos argumentos se encuentran en desacuerdo. En base al criterio de comparación que utiliza DeLP, como se observó en la Definición 8, el argumento preferido es E debido a que $\{(s \prec j, r); (r \prec l, p), \sim s\} \succeq$

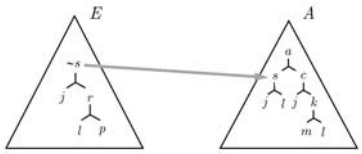

Figura 5: Árboles de Derivación del Argumento A y Argumento E

$\{(s -\!\prec j, l), s\}$. Luego, el literal a no es aceptado en el intervalo de tiempo definido para la consulta.

7. Conclusión

En este trabajo se asociaron intervalos de tiempo a los argumentos, construyendo un framework de argumentación temporal. De esta manera, es posible obtener una representación más adecuada de la realidad, debido a que en el entorno las aceptaciones, pre-suposiciones y percepciones pueden variar en el tiempo. A través de la extensión existente de DeLP denominada ℓ-DeLP, se definieron etiquetas de intervalos de tiempo para los componentes de un programa, y a través de una función denominada ALS se determinaron los intervalos de tiempo para los cual el argumento esta activo. De esta manera, al efectuar una consulta se analiza bajo qué intervalo de tiempo se está efectuando la misma, se determina cuáles son los argumentos que están activos en el intervalo de tiempo definido para la consulta, determinando los contra-argumentos activos en el intervalo y finalmente la respuesta a la consulta.

Referencias

1. M.L. Cobo, D.C. Martinez, and G.R. Simari. An approach to timed abstract argumentation. In *Proc. of Int. Workshop of Non-monotonic Reasoning 2010*, 2010.
2. M.L. Cobo, D.C. Martinez, and G.R. Simari. On admissibility in timed abstract argumentation frameworks. In Helder Coelho, Rudi Studer, and Michael Wooldridge, editors, *ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 1007–1008. IOS Press, 2010.
3. Phan M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.
4. A. J. García and G.R. Simari. Defeasible logic programming: An argumentative approach. *Theory Practice of Logic Programming*, 4(1):95–138, 2004.
5. M. Gómez Lucero, C. Chesñevar, G. Simari, and A. García. Extensión de la argumentación rebatible para considerar etiquetas. *VIII Workshop de Investigadores en Ciencias de la Computación*, pages 189–193, 2006.