Hacia un Framework de Argumentación con Accrual de Argumentos

Mauro J. Gómez Lucero Carlos I. Chesñevar Guillermo R. Simari

Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Laboratorio de Investigación y Desarrollo de Inteligencia Artificial
Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur
Av. Alem 1253, (B8000CPB) Bahía Blanca, Argentina
Tel: (0291) 459-5135 / Fax: (0291) 459-5136
Email: {mjg, cic, grs}@cs.uns.edu.ar

Resumen

Dentro de la Inteligencia Artificial surgieron a lo largo del tiempo varios formalismos para modelar el *razonamiento de sentido común*, muchos de ellos basados en Argumentación. La *Programación en Lógica Rebatible* es uno de ellos, combinando resultados de la Programación en Lógica y la Argumentación Rebatible. No obstante, aunque la acumulación (accrual) de argumentos soportando una misma conclusión es un componente natural de un proceso argumentativo, la mayoría de los formalismos argumentativos existentes no lo modelan.

Este artículo presenta la propuesta de un framework para Argumentación al estilo de la Programación en Lógica Rebatible que modela la noción de acumulación de argumentos.

1. Introducción

La Argumentación es un mecanismo que los seres humanos generalmente empleamos para debatir acerca de alguna cuestión, ya sea con otros seres humanos o internamente con nosotros mismos. En un sentido general, es un proceso (de razonamiento) en el cual se consideran *argumentos* soportando conclusiones. Un argumento para una conclusión puede ser cuestionado o *atacado* por otro argumento, ya sea porque este segundo soporta una conclusión contraria a la del primero o porque lo contradice en algún otro punto (de ahí el término *Argumentación Rebatible*). De esta forma, durante el proceso de argumentación una conclusión originalmente justificada por un argumento puede dejar de estarlo al considerar nuevos argumentos. El propósito final de la argumentación es determinar las conclusiones *justificadas*.

La Inteligencia Artificial (IA) se ha enfrentado desde hace tiempo con el desafío de modelar el *razonamiento de sentido común*, que casi siempre ocurre a la luz de información incompleta y potencialmente inconsistente [1]. Dentro de la IA, varios formalismos surgieron para abordar este desafío. Muchos de ellos se basan en la idea de argumentación, modelando diversos aspectos del proceso argumentativo previamente descripto.

En [2] se listan cinco elementos principales presentes (en algunos casos implícitamente) en cualquier sistema de argumentación rebatible:

1. un lenguaje lógico subyacente (que constituye el medio para expresar la información acerca del dominio en que se basará la argumentación), junto a una noción asociada de consecuencia lógica, pilar para la definición de la noción de argumento,

Financiado parcialmente por CONICET (PIP 5050), Universidad Nacional del Sur (PGI 24/ZN11) y Agencia Nacional de Promoción Científica y Tecnológica.

2. una definicion de argumento,
3. una definición de conflicto o ataque entre argumentos,
4. evaluación de argumentos para determinar si el ataque tiene éxito (noción de derrota), y finalmente,
5. una definición del status de argumentos que contempla la interacción entre todos los argumentos disponibles y puede usarse para definir una noción de consecuencia lógica rebatible.

Algunos formalismos argumentativos (los más abstractos) especifican parcialmente, o directamente no especifican, la lógica subyacente. Estos pueden ser instanciados con varias lógicas alternativas dando lugar a distintos sistemas argumentativos concretos. Un ejemplo de este tipo de formalismos es el propuesto por Dung [3], donde la estructura interna de los argumentos se encuentra sin especificar. Dung trata la noción de argumento como primitiva y asume que los ataques entre argumentos "vienen dados" (es decir, no los computa), lo que le permite concentrarse en el quinto elemento: la interacción entre argumentos y el status final de los mismos.

En el otro extremo existen formalismos argumentativos concretos, basados en una lógica subyacente específica, la cual se emplea para expresar el conocimiento acerca dominio del que se dispone. En estos formalismos, ninguno de los restantes cuatro elementos se asume que vienen dados. Por el contrario, se brindan procedimientos para calcular (o al menos definiciones precisas a partir de las cuales se pueden obtener) los argumentos, ataques, derrotas y status de argumentos, a partir de la representación del conocimiento en la lógica subyacente. Un ejemplo de este tipo de formalismos es la *Programación en Lógica Rebatible* [4], donde la lógica subyacente es una extensión de la Programación en Lógica para incluir negación fuerte y representar información rebatible (además de estricta). Para un resumen completo del área ver [1] y [2].

Aunque todos los formalismos argumentativos propuestos consideran implícita o explícitamente los cinco elementos listados anteriormente, algunos capturan otros elementos o características del proceso argumentativo, que a diferencia de estos cinco, no son generalmente aceptados. Una de estas características, central a la línea de investigación que se presentará en este artículo, es la noción de *accrual* (acumulación) de argumentos, propuesta inicialmente por Pollock (1991) y desarrollada posteriormente por Verheij [5]. Básicamente, la noción de *accrual* de argumentos se basa en la idea de que varios argumentos para una misma conclusión tienen más fuerza, en conjunto, que cada uno de los argumentos individuales por separado. En otras palabras, argumentos para una misma conclusión pueden acumularse para conformar un argumento más "fuerte". Así, por ejemplo, dos argumentos que por separado eran derrotados, al acumularse podrían permanecer no derrotados. De forma análoga, dos argumentos tal que cada uno ataca a un tercero pero no logra derrotarlo, al acumularse podrían tener éxito.

Verheij presenta en [5] un formalismo basado en una noción de derrota compuesta (o múltiple) que captura adecuadamente el *accrual* de argumentos. Dicho formalismo abstrae, al igual que el propuesto por Dung, algunos de los cinco componentes. Aunque define concretamente la estructura interna de un argumento (2do elemento), no define una lógica subyacente (1er elemento). Más aún, los ataques y derrotas (3er y 4to elementos) no se calculan o definen a partir de la estructura de los argumentos, sino que deben especificarse explícitamente como parte del sistema.

2. DeLP: un formalismo argumentativo basado en una extensión de la programación en lógica

La *Programación en Lógica Rebatible* (DeLP) es un formalismo de representación de conocimiento y razonamiento que combina resultados de la Programación en Lógica y la Argumentación Rebati-

ble. A continuación se hará una breve explicación de DeLP, organizada en base a los cinco elementos fundamentales de un sistema argumentativo listados en la introducción. Para una presentación completa de DeLP ver [4].

Lógica subyacente

El lenguaje lógico en que se basa DeLP permite la representación tanto de conocimiento estricto como rebatible. Cuenta con reglas estrictas, denotadas $L_0 \leftarrow L_1, L_2, ..., L_n$ y hechos, denotados $L_0 \leftarrow$, para representar conocimiento estricto, además de reglas rebatibles, denotadas $L_0 -\!\prec L_1, L_2, ..., L_n$, para representar conocimiento débil o rebatible, donde los L_i son literales, es decir átomos fijos o átomos fijos negados (de la forma $\sim Atomo$). Se define la noción de *Programa Lógico Rebatible* como un par (Π, Δ), donde Π es un conjunto (no contradictorio) de reglas estrictas y hechos y Δ es un conjunto de reglas rebatibles.

Como noción de consecuencia lógica se emplea la noción de *Derivación Rebatible*, una especie de derivación SLD (Programación en Lógica) extendida apropiadamente para manejar literales negados de la forma $\sim p$ como nuevos predicados no_p.

Considere el programa DeLP $\mathcal{P}_1 = (\Pi_1, \Delta_1)$ donde:

$$\Pi_1 = \left\{ \begin{array}{l} ave(X) \leftarrow gallina(X) \\ gallina(tina) \\ gallina(little) \\ asustada(tina) \end{array} \right\} \quad \Delta_1 = \left\{ \begin{array}{l} vuela(X) -\!\prec ave(X) \\ vuela(X) -\!\prec gallina(X), asustada(X) \\ \sim vuela(X) -\!\prec gallina(X) \end{array} \right\}$$

Este programa tiene tres reglas rebatibles representando información tentativa sobre la habilidad de volar de las aves en general, de las gallinas y las gallinas asustadas. También tiene una regla estricta expresando que toda gallina es un ave y tres hechos estableciendo que 'tina' y 'little' son gallinas y que 'tina' está asustada.

Argumentos

Dado un programa DeLP (Π, Δ), un argumento $\mathcal{A}$ para un literal H (conclusión), notado $\langle \mathcal{A}, H \rangle$, es un conjunto no contradictorio y minimal de reglas de Δ que permite derivar H posiblemente usando reglas de Π. Los siguientes argumentos pueden obtenerse a partir de $\mathcal{P}_1$:

$\langle \mathcal{A}_1, vuela(tina) \rangle$, donde $\mathcal{A}_1 = \{vuela(tina) -\!\prec ave(tina)\}$
$\langle \mathcal{A}_2, \sim vuela(tina) \rangle$, donde $\mathcal{A}_2 = \{\sim vuela(tina) -\!\prec gallina(tina)\}$
$\langle \mathcal{A}_3, vuela(tina) \rangle$, donde $\mathcal{A}_3 = \{vuela(tina) -\!\prec gallina(tina), asustada(tina)\}$

Conflictos entre argumentos

En DeLP los conflictos entre argumentos surgen por contradicción lógica de sus conclusiones (o conclusiones intermedias). Concretamente, se dice que un argumento ataca a otro si la conclusión del primero contradice la conclusión del segundo o alguna de sus conclusiones intermedias. Por ejemplo, $\langle \mathcal{A}_2, \sim vuela(tina) \rangle$ ataca a $\langle \mathcal{A}_1, vuela(tina) \rangle$ y a $\langle \mathcal{A}_3, vuela(tina) \rangle$.

Evaluación de argumentos y noción de derrota

En DeLP el criterio de comparación entre argumentos es un parámetro del sistema. De esta manera, al aplicar DeLP a un dominio concreto, puede emplearse un criterio que resulte adecuado para dicho dominio. No obstante, al presentarse DeLP en [4] se asocia un criterio de comparación denominado *Especificidad Generalizada* como criterio por defecto. La especificidad generalizada, definida originalmente en [6], es un criterio sintáctico que prefiere los argumentos más directos (con menos

reglas) o basados en más información. Finalmente, establecido el criterio de comparación, se dice que un ataque de un argumento $\langle \mathcal{A}, h\rangle$ a un argumento $\langle \mathcal{B}, k\rangle$ constituye una derrota (o que $\langle \mathcal{A}, h\rangle$ derrota a $\langle \mathcal{B}, k\rangle$) si $\langle \mathcal{B}, k\rangle$ no es mejor que $\langle \mathcal{A}, h\rangle$ de acuerdo al criterio de comparación.

Considerando como criterio de comparación especificidad generalizada, $\langle \mathcal{A}_2, \sim vuela(tina)\rangle$ derrota a $\langle \mathcal{A}_1, vuela(tina)\rangle$ y $\langle \mathcal{A}_3, vuela(tina)\rangle$ derrota a $\langle \mathcal{A}_2, \sim vuela(tina)\rangle$, y estas son las únicas derrotas para el programa $\mathcal{P}_1$.

Status de argumentos y noción de consecuencia lógica rebatible

Para determinar el *status* (derrotado/no derrotado) de un argumento $\langle \mathcal{A}, h\rangle$ se efectía un análisis dialéctico donde se consideran todos sus derrotadores, los derrotadores de estos, y así siguiendo. Concretamente, se construye un árbol de argumentos denominado *Arbol Dialéctico*, con raíz $\langle \mathcal{A}, h\rangle$ y donde cada nodo tiene como hijos a sus derrotadores. Luego se analiza el árbol para determinar el *status* del argumento en la raíz. Finalmente, se dice que un literal h está *garantizado* si existe un argumento para h cuyo *status* es no derrotado.

En la figura 1 se muestra el árbol dialéctico para el argumento $\langle \mathcal{A}_1, vuela(tina)\rangle$. Observar que el argumento $\langle \mathcal{A}_2, \sim vuela(tina)\rangle$ "interfiere" con la justificación de 'vuela(tina)' y el argumento $\langle \mathcal{A}_3, vuela(tina)\rangle$ la restablece derrotando $\langle \mathcal{A}_2, \sim vuela(tina)\rangle$. Por lo tanto $vuela(tina)$ queda garantizado.

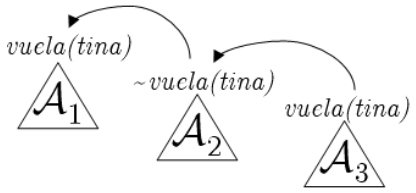

Figura 1: Arbol dialéctico para $vuela(tina)$.

3. Propuesta de un Framework para Argumentación con Accrual

Luego de un análisis abarcador del trabajo existente acerca de *accrual* de argumentos, se comenzó a trabajar en la definición de un framework para argumentación al estilo DeLP, pero que modele la noción de acumulación de argumentos. Al igual que DeLP, empleará como lógica subyacente una extensión de la Programación en Lógica para incluir negación fuerte y representar información rebatible (además de estricta). Contará con una definición de argumento similar a la de DeLP y con la noción de *argumento acumulado* para modelar la acumulación de argumentos. Los conflictos (ataques) entre argumentos acumulados surgirán, al igual que en DeLP, directamente de la contradicción lógica de sus conclusiones (o sub-conclusiones).

El abordaje de la evaluación de argumentos merece especial atención. A lo largo del tiempo se ha comprobado que no existe un criterio de comparación entre argumentos suficientemente general para adecuarse a cualquier dominio y a la vez suficientemente "contundente" para desambiguar la mayoría de los conflictos. Más aún, un buen criterio de comparación podría requerir información *acerca de* las reglas del programa (meta-información), como confiabilidad de la fuente, probabilidad, grado de certeza, prioridad, etc, para tomar una decisión. En este sentido, resultan interesantes las ideas desarrolladas por Gabbay al presentar los Sistemas Deductivos Etiquetados [7], donde las fórmulas tienen etiquetas asociadas. Siguiendo estas ideas, nuestro framework permitirá codificar información acerca de las reglas del programa en etiquetas asociadas a las mismas y permitirá especificar un mecanismo para calcular etiquetas asociadas a argumentos a partir de las etiquetas de las reglas que lo componen. Finalmente, brindará la posibilidad de definir un criterio basado en esta información.

El manejo de las etiquetas y la evaluación de argumentos se formalizará a través de la noción de álgebra de etiquetas. Este álgebra de etiquetas será un parámetro del sistema, y por lo tanto deberá definirse al aplicarlo a un dominio específico. Concretamente, el álgebra de etiquetas contará con un dominio de etiquetas, operadores para calcular la etiqueta asociada a un argumento a partir de las etiquetas asociadas a las reglas que lo componen, un operador de *acumulación* para calcular la etiqueta asociada a una acumulación de argumentos a partir de las etiquetas asociadas a los argumentos individuales y finalmente una relación de preferencia entre etiquetas de argumentos acumulados que se empleará para la evaluación de argumentos en conflicto.

Otra característica destacada de nuestro framework tiene que ver con la noción de derrota, que debe extenderse para contemplar la acumulación de argumentos. En DeLP, cuando un argumento ataca a otro, ya sea a su conclusión o a una conclusión intermedia, está amenazando a "todo" el argumento. Como consecuencia, si el ataque tiene éxito, es decir, el argumento atacado no es mejor que el que ataca, entonces este primero queda derrotado completamente (por supuesto, estamos concentrándonos en *un* solo ataque). Al considerar argumentos acumulados, la situación se torna más compleja. Un argumento acumulado representa un conjunto de argumentos simples, todos soportando una misma conclusión. Sin embargo, las conclusiones intermedias no necesariamente son compartidas por todos ellos. Una cierta conclusión intermedia I podría pertenecer solo a algunos de los argumentos simples. Por esta razón, un ataque a un argumento acumulado "dirigido" hacia una de sus conclusiones intermedias I, no necesariamente amenaza a todo el argumento acumulado, sino solo a la "parte" correspondiente a aquellos argumentos individuales que contienen a I. Si finalmente, a la luz de la evaluación el ataque resulta exitoso, solo la parte amenazada quedará derrotada. Denominamos *derrota parcial* a esta noción extendida de derrota.

Por último se definirá un procedimiento para calcular los argumentos aceptados y las conclusiones justificadas.

Referencias

[1] Carlos Chesñevar, Ana Maguitman, and Ronald Loui. Logical models of argument. *ACM Computing Surveys*, 32(4):337–383, 2000.

[2] H. Prakken and G. Vreeswijk. Logical systems for defeasible argumentation. In D. Gabbay and F.Guenther, editors, *Handbook of Phil. Logic*, pages 219–318, Kluwer, 2002.

[3] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–357, 1995.

[4] A. García and G. Simari. Defeasible logic programming: An argumentative approach. *Theory Practice of Logic Programming*, 4(1):95–138, 2004.

[5] Bart Verheij. Accrual of arguments in defeasible argumentation, 1995.

[6] Guillermo R. Simari and Ronald P. Loui. A Mathematical Treatment of Defeasible Reasoning and its Implementation. *Artificial Intelligence*, 53(1–2):125–157, 1992.

[7] D. M. Gabbay. Labelled deductive systems (vol. 1). *Oxford University Press (Volume 33 of Oxford Logic Guides)*, 1996.