Sobre la Relación entre la Definición Declarativa y Procedural de Argumento

Laura A. Cecchi

Depto. de Informática y Estadística - Fa.E.A.
Universidad Nacional del Comahue
e-mail:lcecchi@uncoma.edu.ar

Guillermo R. Simari

Depto. de Ciencias de la Computación
Universidad Nacional del Sur
e-mail:grs@cs.uns.edu.ar

Palabras Claves: Argumentos - Razonamiento Rebatible - Programación en Lógica

Resumen

Los argumentos son la unidad básica del razonamiento rebatible. Esta clase de razonamiento no monotónico está fundamentado en el análisis dialéctico y constituye la semántica operacional de la Programación en Lógica Rebatible.

Con el espíritu de definir una semántica declarativa adecuada para la Programación en Lógica Rebatible es que, en este trabajo, se presenta una caracterización declarativa de la noción procedural de argumento. Dicha caracterización está basada en el concepto de consecuencias de un programa lógico.

Asimismo, se demuestra que la relación existente entre la definición declarativa introducida y la definición procedural es de equivalencia. Como paso intermedio se define una transformación sintáctica de programas lógico rebatibles básicos en programas lógicos definidos, mostrando algunas propiedades que relacionan las dos clases de programas lógicos.

1 Introducción

La Programación en Lógica Rebatible (de ahora en más PLR) es una extensión de la Programación en Lógica (PL) con una nueva clase de reglas, las reglas rebatibles. Estas reglas permiten representar conocimiento tentativo, aumentando, de este modo, el poder expresivo de la PL.

Los argumentos son la unidad básica del razonamiento rebatible. Esta clase de razonamiento rebatible está fundamentado en el análisis dialéctico y constituye la semántica operacional de la PLR. Si bien esta semántica es suficiente para verificar si una consulta a un programa lógico rebatible está justificada o no, creemos que el beneficio de una semántica declarativa para la PLR es doble. Por un lado, es necesaria para ayudar al programador a especificar el conocimiento y razonar a partir de él sin preocuparse por la parte de control del sistema. Por otro lado, la definición de una semántica declarativa ayuda en el estudio de la PLR como sistema de razonamiento no monótono. Un conjunto de propiedades [2], han sido presentadas con el objeto de clasificar y caracterizar el comportamiento de las semánticas de los programas lógicos con negación. Analizando el conjunto de propiedades que se cumplen es posible comparar al sistema con otros, mostrando sus ventajas y desventajas.

En [3] se introdujo una semántica basada en juegos para la PLR. Si bien esta semántica modela el análisis dialéctico a través de un juego, la noción de argumento quedó indefinida. En otras palabras, se asume que el conjunto de argumentos para un literal es dado por algún oráculo. Esto motivó una caracterización declarativa de la definición procedural de argumento. El estudio presentado en este trabajo está basado en un concepto introducido, en primer instancia por Tarski para la lógica clásica y, luego adaptado por Lifschitz para programas lógicos básicos.

La organización de este trabajo es la siguiente. En la sección 2, se realiza una revisión de los principales conceptos sobre la clase de los programas lógicos rebatibles. La sección 3 describe la semántica procedural de la PLR, ilustrando el significado de argumento. La sección 4 introduce las *consecuencias estrictas* de un programa lógico rebatible y como resultado inmediato, se presenta la definición declarativa de argumento. Con el objeto de probar la equivalencia entre las nociones declarativa y procedural de argumento, se introduce una transformación sintáctica no inyectiva de programas lógicos rebatibles básicos en programas lógicos definidos. En la sección 5 se describen y prueban algunas propiedades de la aplicación de dicha transformación. Como consecuencia de estas últimas, en esta misma sección, se presenta el principal resultado de este trabajo: la demostración de la *equivalencia* entre las definiciones declarativa y procedural de argumento. Finalmente, en la sección 6, se presentan nuestras conclusiones y se mencionan algunos posibles trabajos de investigación futuros.

2 Programación en Lógica Rebatible

La PLR permite representar conocimiento tentativo y certero, y con el objeto de distinguirlos consiste de dos clases diferentes de reglas: reglas estrictas y reglas rebatibles. Los diferenciaremos sintácticamente utilizando los metasímbolos $\leftarrow$ y $-\prec$.

El lenguaje de la PLR está formado por todos los posibles literales ground[1] que se puedan obtener a partir de la signatura del programa y lo notamos Lit. En otras palabras, L pertenece a Lit si L es un átomo ground A o un átomo ground negado $\sim A$, siendo $\sim$ la representación de la negación fuerte. Dado un literal L, notamos con $\overline{L}$ al *complemento* de

[1] Utilizamos el vocablo en inglés por no encontrar una traducción adecuada al castellano.

L con respecto a la negación fuerte y lo definimos como sigue. Si L es el átomo A entonces $\overline{L} =\sim A$; si L es el átomo negado $\sim A$ entonces $\overline{L} = A$.

Definición 2.1. *Reglas estrictas y rebatibles*
Una *Regla Estricta* es un par ordenado $L \leftarrow L_1, \ldots, L_n$ donde $L \in Lit$ y $\{L_1, \ldots, L_n\}$ es un subconjunto de Lit.
Una *Regla Rebatible* es un par ordenado $L \prec\!\!- L_1, \ldots L_m$ donde $L \in Lit$ y $\{L_1, \ldots, L_m\}$ es un subconjunto de Lit.
En ambos casos, la parte izquierda del par, L, será llamada *consecuente o cabeza de la regla* y la parte derecha será llamada *antecedente o cuerpo de la regla*. Como es usual, si el cuerpo es vacío, i.e., $n = 0$, entonces la regla estricta se transforma en $L \leftarrow true$ o $L \leftarrow$ y es llamada *hecho* y la regla rebatible ($m = 0$) se transforma en $L \prec\!\!- true$ o $L \prec\!\!-$ y la llamaremos *presuposición*. ■

Las reglas rebatibles son reglas que pueden ser derrotadas en presencia de evidencia contraria. Estas reglas pueden interpretarse como *razones para creer en el antecedente Cuerpo proveen razones para creer en el consecuente Cabeza*[10] y sus consecuencias siempre serán sujetas a debate.

La PLR puede ser clasificada de acuerdo a como las reglas, estrictas y rebatibles, son construidas. La clasificación que presentamos sigue las definiciones dadas por J. W. Lloyd en [8], M. Gelfond y V. Lifschitz en [6] y J.J. Alferes y L. M. Pereira en [1].

Definición 2.2. Sean $\{ A, A_1, \ldots, A_n \} \subseteq Lit$ un conjunto finito de átomos y $\{L, L_1, \ldots, L_n\}$ un subconjunto finito de literales en Lit. Clasificamos del siguiente modo a las reglas:

- La regla estricta $A \leftarrow A_1, \ldots, A_n$ se denomina *Regla Estricta Definida*. Análogamente, la regla rebatible $A \prec\!\!- A_1, \ldots, A_n$ se denomina *Regla Rebatible Definida*
- La regla estricta $L \leftarrow L_1, \ldots, L_n$ se denomina *Regla Estricta Básica*. Análogamente, la regla rebatible $L \prec\!\!- L_1, \ldots, L_n$ se denomina *Regla Rebatible Básica*.

■

Definición 2.3. Un *programa lógico rebatible*[4] es un conjunto de reglas estrictas y rebatibles. Si $\mathcal{P}$ es un programa lógico rebatible, distinguiremos el subconjunto Π de reglas estrictas en $\mathcal{P}$, y el subconjunto Δ de reglas rebatibles en $\mathcal{P}$. Cuando se requiera denotaremos $\mathcal{P}$ como $\langle \Pi, \Delta \rangle$. ■

Del mismo modo en que hemos clasificado a las reglas estrictas y rebatibles, lo haremos con los programas lógicos rebatibles. Diremos que un programa lógico rebatible es *definido* si sus reglas estrictas y rebatibles son definidas y *básico* si sus reglas estrictas y rebatibles son básicas.

Una vez representado el conocimiento en un programa lógico rebatible, estaremos interesados en responder a consultas basados en tal conocimiento.

Definición 2.4. Una *consulta rebatible* , denotada por $\{Q_1, \ldots, Q_n\}$, es una conjunto de literales ground. La *meta rebatible* correspondiente a la consulta se define como $\prec\!\!- Q_1, \ldots, Q_n$.
Análogamente a una regla, una consulta rebatible y una meta rebatible pueden ser clasificadas en definida y básica dependiendo de si cada Q_i ($1 \leq i \leq n$) es un átomo o un literal. ■

Una consulta rebatible es la conjunción de los literales que la componen, de modo que una consulta será consecuencia de un programa, si lo es cada uno de sus miembros.

El lenguaje de un programa lógico rebatible no permite el uso de variables, y por lo tanto, un programa lógico rebatible no permitirá en la definición de sus reglas términos no ground. Es posible pensar a un programa lógico que contenga variables como un *esquema* del programa[7]. Así una regla R que contenga variables es un esquema de regla y puede ser vista como una notación abreviada de todas sus posibles instanciaciones ground. Este concepto puede ser extendido para los programas lógicos rebatibles. Mientras que un esquema de un programa lógico rebatible puede contener un conjunto finito de esquemas de reglas rebatibles y estrictas, el programa lógico rebatible obtenido a partir de aquél, puede consistir de infinitas reglas ground rebatibles y estrictas, con la sola presencia de un símbolo de función. Las consultas que contengan términos no ground, serán consideradas también esquemas de consultas.

Ejemplo 2.1. El siguiente es un esquema de programa lógico rebatible básico $\mathcal{P}$ que representa información sobre la nacionalidad de una persona.

$$\begin{aligned}\Pi = \{&ciudadanía(andrea, italiana), origen(andrea, argentina), optó(andrea, francesa)\}\\ \Delta = \{&nacionalidad(X,Y) \prec ciudadanía(X,Y)\\ &nacionalidad(X,Y) \prec origen(X,Y)\\ &nacionalidad(X,Y) \prec optó(X,Y)\\ &\neg nacionalidad(X,Y) \prec origen(X,Y) \wedge optó(X,Z) \wedge Z \neq Y\\ &doble\_nacionalidad(X) \prec nacionalidad(X,Y) \wedge nacionalidad(X,Z) \wedge Z \neq Y\}\end{aligned}$$

Aunque los dos últimos esquemas de reglas contengan las condiciones $Z \neq Y$, éstas pueden ser eliminadas al reemplazarlas por las instancias ground correspondientes, teniendo en cuenta la restricción. ■

Este trabajo estará circunscripto a programas lógicos rebatibles básicos, i.e., programas lógicos rebatibles con negación fuerte. Siempre que no sea ambiguo, nos referiremos a programas lógicos rebatibles básicos simplemente como programas lógicos rebatibles o programas.

3 Semántica Operacional

La semántica operacional de la Programación en Lógica Rebatible Básica (de ahora en más PLRB) se basa en los *sistemas de argumentación*, que constituyen una formalización del razonamiento no monótono. Con el objeto de explicar una creencia, un agente realiza un análisis dialéctico teniendo en cuenta las explicaciones tentativas que pueda construir para tal creencia. Las explicaciones que soportan a la creencia se denominan *argumentos* y son el fundamento del razonamiento rebatible.

Una creencia q es aceptada por un agente si existe un argumento $\mathcal{A}$ de q que lo justifique, i.e., no existen contraargumentos que derroten a $\mathcal{A}$. Ya que los derrotadores son también argumentos podrían existir derrotadores para estos últimos y así continuando.

Las siguientes definiciones, que son adaptaciones de las dadas por [4], formalizan la idea de derivación rebatible.

Definición 3.1. Sean $\prec Q_1, \ldots, Q_n$ una meta rebatible básica y R una regla estricta o rebatible con cabeza Q_i, $1 \leq i \leq n$ y cuerpo $L_1, \ldots, L_m$. Luego la nueva meta rebatible $\prec Q_1, \ldots, Q_{i-1}, L_1, \ldots, L_m, Q_{i+1}, \ldots, Q_n$ es *derivable rebatiblemente*. ■

Definición 3.2. Sean $\mathcal{P}$ un programa rebatible y Q una consulta rebatible básica. Una *derivación rebatible* consiste de una secuencia finita de metas rebatibles $\prec Q = Q_0, Q_1, \ldots, Q_n$, donde cada Q_{i+1} es derivado rebatiblemente a partir de $Q_i, 0 \leq i < n$ y Q_n es la meta vacía. Diremos que Q se deriva rebatiblemente a partir de $\mathcal{P}$. ■

Ya que la PLRB permite representar conocimiento contradictorio, diremos que un programa rebatible $\mathcal{P}$ es contradictorio si existe un literal L tal que $\mathcal{P}$ deriva rebatiblemente a L y a $\overline{L}$.

A partir de estas definiciones podremos formalizar la idea de argumento.

Definición 3.3. [5] Sea $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa lógico rebatible. Una *estructura de argumento* para un literal ground h en el contexto Π, que denotaremos $\langle \mathcal{A}, h \rangle_\Pi$ o simplemente $\langle \mathcal{A}, h \rangle$, si $\mathcal{A}$ es un subconjunto de Δ que cumple las siguientes condiciones:

1. h se deriva rebatiblemente a partir de $\Pi \cup \mathcal{A}$,
2. $\Pi \cup \mathcal{A}$ no es contradictorio, y
3. $\mathcal{A}$ es minimal con respecto a la inclusión de conjuntos, i.e., no existe $\mathcal{A}' \subset \mathcal{A}$ tal que $\mathcal{A}'$ satisface las dos primeras condiciones.

Un argumento $\langle \mathcal{B}, h' \rangle$ es un *subargumento* de $\langle \mathcal{A}, h \rangle$ si $\mathcal{B} \subseteq \mathcal{A}$. ■

Aunque cuando se referencia a un *argumento* para h se hace mención de $\mathcal{A}$, siempre que no sea ambiguo abusaremos del lenguaje denominando a la estructura de argumento $\langle \mathcal{A}, h \rangle$ simplemente argumento.

Definición 3.4. [5] Dos argumentos $\langle \mathcal{B}, h' \rangle$ y $\langle \mathcal{A}, h \rangle$ están en *desacuerdo* cuando $\Pi \cup \{h, h'\}$ sea contradictorio. ■

Una vez que se ha construido un argumento para h, debemos encontrar cada argumento $\langle \mathcal{B}', q \rangle$ tal que exista un subargumento $\langle \mathcal{A}', h' \rangle$ de $\langle \mathcal{A}, h \rangle$ y que $\langle \mathcal{A}', h' \rangle$ y $\langle \mathcal{B}', q \rangle$ estén en desacuerdo. $\langle \mathcal{B}', q \rangle$ se denomina *contraargumento* de $\langle \mathcal{A}, h \rangle$.

Ejemplo 3.1. Consideremos nuevamente el programa del ejemplo 2.1. Los argumentos para la consulta $doble\_nacionalidad(andrea)$ son:

$$\mathcal{A}_1 = \{ \; doble\_nac(andrea) \prec nac(andrea, argentina) \wedge nac(andrea, italiana);\; nac(andrea, argentina) \prec origen(andrea, argentina);\; nac(andrea, italiana) \prec ciudadanía(andrea, italiana) \; \}$$

$$\mathcal{A}_2 = \{ \; doble\_nac(andrea) \prec nac(andrea, argentina) \wedge nac(andrea, francesa);\; nac(andrea, argentina) \prec origen(andrea, argentina);\; nac(andrea, francesa) \prec optó(andrea, francesa) \; \}$$

$$\mathcal{A}_3 = \{ \; doble\_nac(andrea) \prec nac(andrea, italiana) \wedge nac(andrea, francesa);\; nac(andrea, italiana) \prec ciudadanía(andrea, italiana);\; nac(andrea, francesa) \prec optó(andrea, francesa) \; \}$$

Existen otros argumentos que se generan al instanciar en forma simétrica las nacionalidades en la regla $doble\_nacionalidad(andrea)$.

Note que el siguiente argumento:

$\mathcal{A}_4 = \{\neg nac(andrea, argentina) \multimap origen(andrea, argentina) \wedge optó(andrea, francesa)\}$

es un contraargumento del subargumento

$$\langle\{nac(andrea, argentina) \multimap origen(andrea, argentina)\}, nac(andrea, argentina)\rangle$$

de los argumentos $\mathcal{A}_1$ y $\mathcal{A}_2$. ■

De todos los contraargumentos que puedan existir para un argumento, nosotros estaremos interesados solamente en aquellos que lo derroten. Con el objeto de poder decidir entre contraargumentos, se define una relación de preferencia entre argumentos. Esta relación puede ser especificada de diferentes maneras, como por ejemplo, prioridad entre reglas [9] y especificidad[10].

El análisis dialéctico, que puede ser visto gráficamente como un árbol, se construye comenzando con un argumento para la consulta, considerando, luego, todos los contraargumentos que lo derroten. Ya que los contraargumentos son argumentos, se deberá considerar los contraargumentos de los contraargumentos y así sucesivamente.

La consulta es una consecuencia del sistema si en el árbol que representa su análisis dialéctico resulta que cada contraargumento que podría derrotar al argumento en la raíz es a su vez derrotado, teniendo en cuenta que las hojas del árbol son argumentos no derrotados.

Ejemplo 3.2. Continuando con el ejemplo de las nacionalidades y siendo que el análisis dialéctico está fuera del alcance de este trabajo, sólo daremos el resultado para la consulta, sin entrar en los detalles. Bajo la relación de preferencia de especificidad, el contraargumento $\mathcal{A}_4$ derrota a los argumentos $\mathcal{A}_1$ y $\mathcal{A}_2$. Así *andrea* tiene doble nacionalidad por ser italiana y francesa. ■

4 Argumentos: un estudio declarativo

Dado un programa estamos interesados en obtener el conjunto de sus consecuencias lógicas, i.e., su semántica. Para esto necesitamos dar una definición declarativa equivalente de argumento. Basamos esta definición en la de operación de consecuencia.

Una operación de consecuencia Cn_T en el sentido estándar deriva de Tarski y es una operación sobre conjuntos de sentencias. Dado un conjunto de sentencias Γ se define $Cn_T(\Gamma)$ como el conjunto de sentencias que puede ser derivado clásicamente a partir de Γ y de los axiomas lógicos usando las reglas de inferencia.

Nuestro interés es definir el conjunto de *literales ground* que son consecuencias de los programas lógicos. Así la operación de consecuencia se define sobre un programa lógico rebatible en Lit. La contraparte de las sentencias en los programas lógicos son los literales ground. Dado un programa lógico rebatible exigiremos que el conjunto de las consecuencias de un programa sea "cerrado"bajo las reglas estrictas y rebatibles, i.e., si una regla puede ser aplicada, entonces la consecuencia de esa regla debe pertenecer al conjunto de las consecuencias del programa.

Deberemos distinguir dos casos de acuerdo a si el programa es o no consistente. Diremos que un programa es consistente si no existe un literal L, tal que L y su complemento $\overline{L}$ pertenezcan a sus consecuencias. De otro modo, diremos que el programa es inconsistente.

Extenderemos las definiciones dadas en [7] para manejar programas lógicos rebatibles.

Definición 4.1. Sea X un conjunto de literales. X es *consistente* si no existe un literal L tal que $\{L, \overline{L}\} \subseteq X$. X es *lógicamente cerrado* si es consistente o es igual a Lit. Diremos que X es *cerrado estrictamente bajo un programa* $\mathcal{P}$, si para cada regla estricta y rebatible de la forma $Cabeza \twoheadleftarrow Cuerpo$ de $\mathcal{P}$, donde $\twoheadleftarrow$ indica $\leftarrow$ y $\prec\!\!\!-$ respectivamente, $Cabeza \in X$ siempre que $Cuerpo \subseteq X$. ■

La definición anterior no refleja el comportamiento de las reglas rebatibles, ya que es aplicada, sin considerar posibles elementos en el conocimiento que invaliden su uso, mostrando, finalmente, una conducta idéntica a la regla estricta.

Definición 4.2. Sea $\mathcal{P}$ un programa. El *conjunto de las consecuencias estrictas* de $\mathcal{P}$, que notaremos $Cn_E(\mathcal{P})$, es el menor conjunto de literales que es lógicamente cerrado y cerrado estrictamente bajo $\mathcal{P}$. ■

Si el programa es consistente, entonces un literal pertenecerá al conjunto de sus consecuencias si es la consecuencia de una regla estricta o rebatible aplicable. Si el programa es inconsistente entonces sus consecuencias coinciden con Lit, i.e., todo el lenguaje es consecuencia de un programa inconsistente. Siendo que está permitido en la PLRB expresar información contradictoria, será de poca ayuda dar el conjunto de todas las consecuencias de un programa, pues en la mayoría de los casos este conjunto será Lit. Sin embargo, la definición de argumento exige que el conjunto resultante de unir el subconjunto de reglas rebatibles que lo componen a Π sea consistente. Por esta razón, utilizaremos este concepto para dar la definición declarativa equivalente de argumento para un literal.

Definición 4.3. Sea $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa. Un argumento $\mathcal{A}$ para un literal ground q es un subconjunto de las reglas rebatibles de $\mathcal{P}$, $\mathcal{A} \subseteq \Delta$, tal que:

1. $q \in Cn_E(\Pi \cup \mathcal{A})$
2. $Cn_E(\Pi \cup \mathcal{A})$ es consistente, i.e., $Cn_E(\Pi \cup \mathcal{A}) \neq Lit$
3. $\mathcal{A}$ es minimal con respecto a la inclusión, i.e., no existe $\mathcal{A}' \subseteq \mathcal{A}$ que cumpla (1) y (2).

Un argumento $\langle \mathcal{B}, h' \rangle$ es un *subargumento* de $\langle \mathcal{A}, h \rangle$ si $\mathcal{B} \subseteq \mathcal{A}$. ■

Definición 4.4. Dos argumentos $\langle \mathcal{B}, h' \rangle$ y $\langle \mathcal{A}, h \rangle$ están en *desacuerdo* si $\Pi \cup \{h, h'\}$ es inconsistente o en forma equivalente cuando $Cn_E(\Pi \cup \{h, h'\}) = Lit$. ■

Ejemplo 4.1. Consideremos el argumento $\mathcal{A}_1$ del programa del ejemplo 2.1. El conjunto de $Cn_E(\Pi \cup \mathcal{A}_1)$ es

$$\big\{ciudadanía(andrea, italiana), origen(andrea, argentina), optó(andrea, francesa),\\ nac(andrea, argentina), nac(andrea, italiana), doble\_nac(andrea)\big\}$$

$\mathcal{A}_1$ cumple la defineión 4.3 de argumento, ya que $doble\_nac(andrea) \in Cn_E(\Pi \cup \mathcal{A}_1)$, $Cn_E(\Pi \cup \mathcal{A}_1)$ es consistente y, por último, $\mathcal{A}_1$ es minimal con respecto a la inclusión de conjuntos. ■

Con el objeto de demostrar que la caracterización declarativa de argumento es equivalente a la procedural, introducimos una transformación sintáctica de programas rebatibles básicos en programas definidos sin reglas rebatibles. Esta transformación es una extensión de la presentada en [7] y nos permitirá aprovechar todas las propiedades demostradas para los programas lógicos sin reglas rebatibles.

Definición 4.5. Sea $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa rebatible básico. Para cada átomo $A \in Lit$, seleccione un nuevo símbolo A', tal que $A' \notin Lit$. Definimos en forma recursiva $definido(\mathcal{P})$,cuyo lenguaje será $Lit_{d(P)} = \{A \in Lit | A \textit{es un átomo }\} \cup \{A' | A \textit{ es un átomo en } Lit \textit{ y } A' \textit{ es el átomo ground seleccionado para } A\}$

(i) $definido(A) = A$, si A es un átomo.

(ii) $definido(\neg A) = A'$, siendo A' el nuevo átomo ground asignado al átomo A.

(iii) $definido(Cabeza \leftarrow Cuerpo) = definido(Cabeza) \leftarrow definido(Cuerpo)$.

(iv) $definido(Cabeza \prec\!\!\!- Cuerpo) = definido(Cabeza) \leftarrow definido(Cuerpo)$.

(v) $definido(\langle \Pi, \Delta \rangle) = \{definido(R) | R \in \Pi \cup \Delta\}$.

Sea X un conjunto de literales llamaremos $definido(X) = \{definido(x) | x \in X\}$ ■

Ejemplo 4.2. El siguiente esquema de programa es el resultado de aplicar $definido$ al programa del ejemplo 2.1.

$\{ciudadanía(andrea, italiana)$
$origen(andrea, argentina)$
$optó(andrea, francesa)$
$nacionalidad(X, Y) \leftarrow ciudadanía(X, Y)$
$nacionalidad(X, Y) \leftarrow origen(X, Y)$
$nacionalidad(X, Y) \leftarrow optó(X, Y)$
$nacionalidad'(X, Y) \leftarrow origen(X, Y) \wedge optó(X, Z) \wedge Z \neq Y$
$doble\_nacionalidad(X) \leftarrow nacionalidad(X, Y) \wedge nacionalidad(X, Z) \wedge Z \neq Y\}$ ■

Esta transformación no es inyectiva, ya que existen diversos programas lógicos rebatibles cuyo programa transformado a través de $definido$ coincide.

En la siguiente sección demostraremos que las definiciones declarativa y procedural de argumento son equivalentes utilizando a los programas transformados a través de $definido$.

5 Equivalencia entre las definiciones declarativa y procedural de argumento

En esta sección se presenta el resultado más importante de este trabajo: la equivalencia entre las definiciones procedural y declarativa de argumento. Para alcanzar tal resultado probaremos primero algunas propiedades sobre la relación entre un programa rebatible y su transformación sintáctica a través de $definido$. Las demostraciones se muestran en el apéndice.

Lema 5.1. Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa y $\Lambda \subseteq \Delta$, tal que $\Pi \cup \Lambda$ es consistente. Entonces

$$Cn_E(definido(\Pi \cup \Lambda)) = definido(Cn_E(\Pi \cup \Lambda))$$

■

La igualdad no vale si $\Pi \cup \Lambda$ es inconsistente. Esto se debe a que en este caso $Cn_E(\Pi \cup \Lambda)$ es *Lit* y siendo que $definido(\Pi \cup \Lambda)$ nunca será inconsistente, las consecuencias estrictas no necesariamente son igual a $Lit_{d(P)}$.

Corolario 5.2. Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa y $\Lambda \subseteq \Delta$. Entonces

$$Cn_E(definido(\Pi \cup \Lambda)) \subseteq definido(Cn_E(\Pi \cup \Lambda))$$

■

Evidentemente $definido(\mathcal{P})$ es un programa lógico definido sin reglas rebatibles por lo que podremos aprovechar todas las propiedades demostradas para ellos.

Lema 5.3. [7] Sea $\Pi \cup \Delta$ un programa y $\Lambda \subseteq \Delta$. $Cn_E(definido(\Pi \cup \Lambda))$ es el menor modelo de $definido(\Pi \cup \Lambda)$. ■

Lema 5.4. [8] Sean $\mathcal{P}$ un programa lógico definido sin reglas rebatibles y q un átomo. q pertenece al modelo mínimo de $\mathcal{P}$ si y sólo si existe una refutación SLD de $\mathcal{P} \cup \{\leftarrow q\}$. ■

Note que la definición de derivación rebatible a partir de un programa lógico rebatible es equivalente a la refutación a partir de programas definidos no rebatibles. El siguiente lema lo formaliza.

Lema 5.5. Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa, $\Lambda \subseteq \Delta$, q un literal y $q' = definido(q)$. Entonces,

$$definido(\Pi \cup \Lambda) \vdash q' \textit{ si y sólo si } q \textit{ es derivable rebatiblemente a partir de } \Pi \cup \Lambda$$

■

Teorema 5.6. Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa y $\mathcal{A} \subseteq \Delta$. $\mathcal{A}$ es un argumento para un literal h según la definición declarativa 4.3 si y sólo si $\mathcal{A}$ es un argumento para el literal h según la definición procedural 3.3. ■

El teorema anterior nos habilita para utilizar cualquiera de las definiciones en forma indistinta. Note que si $\mathcal{A}$ es un argumento para h, al calcular $Cn_E(\Pi \cup \mathcal{A})$ tenemos a nuestro alcance todos los literales que fueron necesarios para derivar rebatiblemente h, potenciales puntos de ataques para otros argumentos.

6 Conclusión y Trabajo Futuro

Hemos caracterizado de un modo declarativo a la noción de argumento, basándonos en las consecuencias estrictas del conjunto de reglas estrictas unido a un subconjunto de reglas rebatibles, potencial argumento. Asimismo, se han presentado las definiciones correspondiente de subargumento y argumentos en desacuerdo. Hemos demostrado que la relación existente entre el concepto declarativo y el procedural es de equivalencia. Como paso intermedio se especificó una transformación sintáctica de programas lógico rebatibles básicos en programas lógicos definidos y se demostraron algunas propiedades que relacionan las dos clases de programas lógicos.

Creemos que estos resultados nos permitirán alcanzar uno de nuestros estudios futuros: una semántica declarativa adecuada para la PLRB.

Entre nuestros trabajos futuros se encuentra también, decidir si esta caracterización puede ayudar a dar significado a programas lógicos rebatibles que incluyan la negación por falla.

Referencias

[1] Julio José Alferes and Luis Moniz Pereira. *Reasoning with Logic Programming*. Springer-Verlag, Berlin, 1996.

[2] Gerhard Brewka, Jürgen Dix, and Kurt Konolige. *Nonmonotonic Reasoning. An Overview. Lecture Notes Number 73*. CSLI Publications, Stanford - United States, 1997.

[3] Laura A. Cecchi and Guillermo R. Simari. Una semántica declarativa basada en juegos para la programación en lógica rebatible básica. In *Proceedings of ICIE*, 2000.

[4] A. García and G. R. Simari. Strong and Default Negation in Defeasible Logic Programming. In *4th Dutch/German Workshop on Nonmonotonic Reasoning Techniques and their applications*, Amsterdam, 25 - 27, Marzo 1999.

[5] A. García, G. R. Simari, and Carlos Chesñevar. An Argumentative Framework for Reasoning with Inconsistent and Incomplete Information. In *ECAI 98, Proceedings de Workshop on Practical Reasoning and Rationality, 13th. European Conference on Artificial Intelligence*, Brighton, England, 1998.

[6] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In David Warren and Peter Szeredi, editors, *Logic Programming: Proceedings of the Seventh International Conference*, pages 579–597, 1990.

[7] Vladimir Lifschitz. Foundations of logic programming. In G. Brewka, editor, *Principles of Knowledge Representation*, pages 1–57. CSLI Publications, 1996.

[8] John W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, New York, second edition, 1987.

[9] H. Prakken and G. Sartor. A dialectical model of assessing conflicting arguments in legal reasoning. *Artificial Intelligence and Law*, 4:331–368, 1996.

[10] Guillermo R. Simari and R.P. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, pages 125–157, 1992.

A Apéndice

A.1 Prueba de los lemas

Lema A.7. Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa rebatible básico, $C \subseteq Lit$ y $C' = definido(C)$. Si C' es cerrado bajo $definido(\Pi \cup \Delta)$, entonces C es cerrado bajo $\Pi \cup \Delta$.

Demostración: Sea $A \leftarrow B_1, \ldots, B_n \in \Pi \cup \Delta$ y $\{B_1, \ldots, B_n\} \subseteq C$, debemos probar que $A \in C$.
Ya que $A \leftarrow B_1, \ldots, B_n \in \Pi \cup \Delta$ entonces $definido\big(A \leftarrow B_1, \ldots, B_n\big) \in definido\big(\Pi \cup \Delta\big)$. Además $\{B_1, \ldots, B_n\} \subseteq C$, luego $definido\big(\{B_1, \ldots, B_n\}\big) \subseteq C'$. Por hipótesis, C' es cerrado bajo $definido(\Pi \cup \Delta)$, así $definido(A) \in C'$. Siendo que la transformación $definido$ no genera nuevos elementos, entonces $A \in C$. ∎

Lema 5.1 *Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa y $\Lambda \subseteq \Delta$, tal que $\Pi \cup \Lambda$ es consistente. Entonces*

$$Cn_E(definido(\Pi \cup \Lambda)) = definido(Cn_E(\Pi \cup \Lambda))$$

Demostración: Probaremos por definición que el conjunto de las consecuencias estrictas de $(definido(\Pi \cup \Lambda))$ es $definido(Cn_E(\Pi \cup \Lambda))$.

i $definido(Cn_E(\Pi \cup \Lambda))$ es lógicamente cerrado. Ya que en cualquier transformación a través de $definido$ no existen literales complementarios, $definido(Cn_E(\Pi \cup \Lambda))$ es consistente.

ii $definido(Cn_E(\Pi \cup \Lambda))$ es cerrado bajo $definido(\Pi \cup \Lambda)$.
Sea $A \leftarrow B_1, \ldots, B_n \in definido(\Pi \cup \Lambda)$ y $\{B_1, \ldots, B_n\} \subseteq definido(Cn_E(\Pi \cup \Lambda))$. Debemos probar que $A \in definido(Cn_E(\Pi \cup \Lambda))$.
Supongamos que $A \notin definido(Cn_E(\Pi \cup \Lambda))$, con $definido(A') = A$. Sabiendo que

$$A \leftarrow B_1, \ldots, B_n \in definido(\Pi \cup \Lambda)$$

entonces existe

$$A' \leftarrow B'_1, \ldots, B'_n \in \Pi \text{ o bien } A' \prec\!\!\!- B'_1, \ldots, B'_n \in \Lambda$$

siendo $definido(B'_i) = B_i$, $1 \leq i \leq n$. Ya que $\{B_1, \ldots, B_n\} \subseteq definido(Cn_E(\Pi \cup \Lambda))$, entonces $\{B'_1, \ldots, B'_n\} \subseteq Cn_E(\Pi \cup \Lambda)$. Pero $A' \notin Cn_E(\Pi \cup \Lambda)$. Contradicción.

iii $definido(Cn_E(\Pi \cup \Lambda))$ es el conjunto minimal que cumple (i)-(ii).
Supongamos que existe $C \subset definido(Cn_E(\Pi \cup \Lambda))$, tal que cumple (i)-(ii), luego existe un átomo A tal que $A \notin C$ y $A \in definido(Cn_E(\Pi \cup \Lambda))$.
Sea C' el conjunto de literales, tal que $C = definido(C')$. Como C es cerrado bajo $definido(\Pi \cup \Lambda)$, entonces por el lema A.7 C' es cerrado bajo $\Pi \cup \Lambda$. Por otra parte, $\Pi \cup \Lambda$ es consistente, así C' es lógicamente cerrado.
Por lo tanto, C' es lógicamente cerrado y cerrado bajo $\Pi \cup \Lambda$ y $C' \subset Cn_E(\Pi \cup \Lambda)$. Contradicción.

$\therefore$ $Cn_E(definido(\Pi \cup \Lambda)) = definido(Cn_E(\Pi \cup \Lambda))$. ■

Corolario 5.2 *Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa y $\Lambda \subseteq \Delta$. Entonces*

$$Cn_E(definido(\Pi \cup \Lambda)) \subseteq definido(Cn_E(\Pi \cup \Lambda))$$

Demostración: Contemplaremos dos casos:

- Si $\Pi \cup \Lambda$ es consistente, entonces el corolario es directo por el lema 5.1.
- Si $\Pi \cup \Lambda$ es inconsistente, entonces $Cn_E(\Pi \cup \Lambda) = Lit$ y $definido(Lit)$ es igual al lenguaje de $definido(\Pi \cup \Delta)$, i.e., $Lit_{d(P)}$. Así por definición $Cn_E(definido(\Pi \cup \Lambda)) \subseteq Lit_{d(P)}$.

■

Lema 5.5 *Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa, $\Lambda \subseteq \Delta$, q un literal y $q' = definido(q)$. Entonces,*

$$definido(\Pi \cup \Lambda) \vdash q' \text{ si y sólo si } q \text{ es derivable rebatiblemente a partir de } \Pi \cup \Lambda$$

Demostración: Por hipótesis, $definido(\Pi \cup \Lambda) \vdash q'$, luego existe una refutación SLD de $definido(\Pi \cup \Lambda) \cup \{\leftarrow q'\}$. Sea tal refutación dada por la secuencia $\leftarrow q' = Q'_0, Q'_1, \ldots, Q'_n = \square$.
Probaremos por inducción sobre la longitud de la refutación SLD, que existe una derivación rebatible de q desde $\Pi \cup \Lambda$.

Caso Base: $n = 1$. En este caso la refutación SLD tiene la forma

$$\leftarrow q' = Q'_0, Q'_1 = \square$$

Así debe existir una regla $q' \leftarrow true \in definido(\Pi \cup \Lambda)$. Luego existe $q \leftarrow true \in \Pi$ o bien $q \leftarrow true \in \Lambda$. Por lo tanto, $Q_1 = \square$ se deriva rebatiblemente desde $\Pi \cup \Lambda$.

Paso inductivo: Supongamos que el lema vale para $k < n$. Debemos probar que vale para n.
Q'_1 fue obtenido a partir de Q'_0 utilizando una regla de la forma

$$q' \leftarrow A'_1 \ldots A'_m$$

siendo $Q'_1 = \leftarrow A'_1 \ldots A'_m$. Así existe

$$q \leftarrow A_1 \ldots A_m \in \Pi \text{ o bien } q \prec\!\!\!- A_1 \ldots A_m \in \Lambda \quad (1)$$

siendo $A'_i = definido(A_i),\ 1 \leq i \leq m$.
La refutación SLD $Q'_1 = \leftarrow A'_1 \ldots A'_m, Q'_2, \ldots, Q'_n = \square$ tiene menos de n pasos. Por hipótesis inductiva, existe una derivación rebatible $Q_1 = \mathrel{-\!\prec} A_1 \ldots A_m, Q_2, \ldots, Q_r = \square$.
$\mathrel{-\!\prec} A_1 \ldots A_m$ es derivable rebatiblemente desde $\mathrel{-\!\prec} q$ utilizando alguna de las reglas 1.
Por lo tanto,

$$Q_0 = \mathrel{-\!\prec} q,\ Q_1 = \mathrel{-\!\prec} A_1 \ldots A_m, Q_2, \ldots, Q_r = \square$$

es una derivación rebatible de q a partir de $\Pi \cup \Lambda$.

El recíproco se prueba en forma análoga. ■

A.2 Prueba del Teorema de Equivalencia de las Definiciones Procedural y Declarativa

Lema A.8. Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa y $\Lambda \subseteq \Delta$, tal que $\Pi \cup \Lambda$ es consistente. $h \in Cn_E(\Pi \cup \Lambda)$ si y sólo si h es derivable rebatiblemente a partir de $\Pi \cup \Lambda$.

Demostración: $h \in Cn_E(\Pi \cup \Lambda)$ si y sólo si
$h' = definido(h)$ y $h' \in definido(Cn_E(\Pi \cup \Lambda))$ si y sólo si
por ser $\Pi \cup \Lambda$ consistente, entonces aplicando el lema 5.1 $h' \in Cn_E(definido(\Pi \cup \Lambda))$ si y sólo si
por lema 5.3 $Cn_E(definido(\Pi \cup \Lambda))$ es el menor modelo de $definido(\Pi \cup \Lambda)$ si y sólo si
por lema 5.4 existe una refutación SLD de $definido(\Pi \cup \Lambda) \cup \{\leftarrow h'\}$ si y sólo si
h es derivable rebatiblemente a partir de $\Pi \cup \Lambda$. ■

Lema A.9. Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa y $\Lambda \subseteq \Delta$. $\Pi \cup \Lambda$ es consistente si y sólo si $\Pi \cup \Lambda$ no es contradictorio.

Demostración: Supongamos que $\Pi \cup \Lambda$ es contradictorio. Luego existe un literal L tal que $\Pi \cup \Lambda$ deriva rebatiblemente a L y a $\overline{L}$. Por lema 5.5 $definido(\Pi \cup \Lambda) \vdash L'$ y $definido(\Pi \cup \Lambda) \vdash \overline{L}'$ siendo $L' = definido(L)$ y $\overline{L}' = definido(\overline{L})$.
Por el lema 5.4 $\{L', \overline{L}'\}$ está incluido en el modelo minimal y aplicando el lema 5.3 tenemos que $\{L', \overline{L}'\} \subseteq Cn_E(definido(\Pi \cup \Lambda))$.
Por corolario 5.2 $\{L', \overline{L}'\} \subseteq definido(Cn_E(\Pi \cup \Lambda))$. Por lo tanto, $\{L, \overline{L}\} \subseteq Cn_E(\Pi \cup \Lambda)$, i.e., $\Pi \cup \Lambda$ es inconsistente. Esta contradicción provino de suponer que $\Pi \cup \Lambda$ es contradictorio.

Recíprocamente, si $\Pi \cup \Lambda$ es inconsistente, entonces existe al menos un literal L que genera tal inconsistencia y que hace que $Cn_E(\Pi \cup \Lambda) = Lit$. Para tal literal L debe ser que $\{L'\overline{L}'\} \subseteq Cn_E(definido(\Pi \cup \Lambda))$, siendo $L' = definido(L)$ y $\overline{L}' = definido(\overline{L})$. Por los lemas 5.3 y 5.4 $\{L'\overline{L}'\}$ está incluido en el modelo mínimo de $definido(\Pi \cup \Lambda)$ y, por lo tanto, $definido(\Pi \cup \Lambda) \vdash L'$ y $definido(\Pi \cup \Lambda) \vdash \overline{L}'$. Luego tanto L como $\overline{L}$ son derivables rebatiblemente a partir de $\Pi \cup \Lambda$ por el lema 5.5, i.e., $\Pi \cup \Lambda$ es contradictorio. ■

Teorema 5.6 *Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa y $\mathcal{A} \subseteq \Delta$. $\mathcal{A}$ es un argumento para un literal h según la definición declarativa 4.3 si y sólo si $\mathcal{A}$ es un argumento para el literal h según la definición procedural 3.3.*

Demostración: Probaremos que las tres condiciones de las definiciones son equivalentes.

1. $Cn_E(\Pi \cup \mathcal{A})$ es consistente si y sólo si $\Pi \cup \mathcal{A}$ no es contradictorio. Lema A.9.
2. En el punto anterior, se mostró que $\Pi \cup \mathcal{A}$ es consistente, por lo tanto, $h \in Cn_E(\Pi \cup \mathcal{A})$ si y sólo si h es derivable rebatiblemente a partir de $\Pi \cup \mathcal{A}$, por lema A.3.
3. Sea $\mathcal{A}$ un argumento para h bajo la definición declarativa 4.3. Debemos probar que $\mathcal{A}$ es el conjunto minimal que cumple las condiciones (1) y (2) de la definición procedural 3.3.
   Supongamos que existe $\mathcal{A}' \subseteq \mathcal{A}$ que cumple las condiciones (1) y (2) de la definición procedural 3.3.
   Ya que $\Pi \cup \mathcal{A}'$ deriva rebatiblemente a h y que $\Pi \cup \mathcal{A}'$ no es contradictotio, por lema $h \in Cn_E(\Pi \cup \mathcal{A}')$.
   Además $Cn_E(\Pi \cup \mathcal{A}')$ es consistente. Por lo tanto, $\mathcal{A}$ no es un argumento para h según la definición declarativa, pues no es minimal. Contradicción. Dicha contradicción provino de suponer que $\mathcal{A}$ no es el conjunto minimal que cumple (1) y (2) de la definición procedural de argumento.
   El recírproco se demuestra en forma análoga.

Luego ambas definiciones son equivalentes. ■