El criterio de especificidad en la programación en lógica rebatible [1]

Alejandro J. García[2] Guillermo R. Simari

Grupo de Investigación en Inteligencia Artificial (GIIA)
Instituto de Ciencias e Ingenieria de la Computación
Departamento de Ciencias de la Computación
Universidad Nacional del Sur
Av.Alem 1253 - (8000) Bahía Blanca, ARGENTINA
FAX: (54) (91) 563401
e-mail: ccgarcia@criba.edu.ar grs@criba.edu.ar

Resumen

La *programación en lógica rebatible* es una extensión de la programación en lógica que captura aspectos del razonamiento del sentido común que son difíciles de expresar en la programación en lógica tradicional. Los *programas lógicos rebatibles* (PLR) permiten representar información incompleta y potencialmente inconsistente, y utilizan los conceptos de la *argumentación rebatible* a fin de poder decidir entre metas contradictorias. Para esto se utiliza un criterio de preferencia entre derivaciones rebatibles. Tanto la argumentación rebatible, como la programación en lógica rebatible, pueden definirse independientemente del criterio de preferencia entre argumentos. Sin embargo, uno de los criterios más utilizados es la *especificidad*. La programación en lógica rebatible incorpora además conceptos que no estaban presentes en la argumentación rebatible: la negación por falla y las presuposiciones.

El objetivo de este trabajo es definir el criterio de especificidad para los PLR, de tal manera que pueda comparar correctamente argumentos que contengan presuposiciones y el operador de negación por falla. Además, se presenta una definición refinada que permite calcular en forma eficiente cuando un argumento es más específico que otro.

[1]Financiado parcialmente por la Secretaría de Ciencia y Técnica, Universidad Nacional del Sur.

[2]Becario del Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), República Argentina

1. Introducción

La *programación en lógica rebatible* [15, 4] es una extensión de la programación en lógica que captura aspectos del razonamiento del sentido común que son difíciles de expresar en la programación en lógica tradicional. Los *programas lógicos rebatibles* (PLR) permiten trabajar con información incompleta y potencialmente inconsistente, y utilizan los conceptos de la *argumentación rebatible* [6, 13, 14, 15], a fin de poder decidir entre metas contradictorias, para lo cuál necesitan de un criterio de preferencia entre derivaciones rebatibles. La programación en lógica rebatible incorpora además conceptos que no estaban presentes en la argumentación rebatible: la negación por falla y las presuposiciones.

El objetivo de este trabajo es definir el criterio de especificidad para los PLR, de tal manera que pueda comparar correctamente argumentos que contengan presuposiciones y el operador de negación por falla. Además, se presenta una definición refinada que permite calcular en forma eficiente cuando un argumento es más específico que otro.

En el lenguaje de la programación en lógica rebatible, (siguiendo la definición de Lloyd [9]) un *literal* "l" es un átomo "a" o un átomo negado "$\neg a$". El símbolo "$\neg$" representa la negación clásica, y el símbolo "*not*" la negación por falla finita. El símbolo "$\sim$" en cambio, se utiliza para indicar el complemento de un literal con respecto a la negación clásica, *i.e.*, $\sim l = \neg l$, y $\sim\neg l = l$.

Un *programa lógico rebatible* (PLR), es un conjunto finito de *cláusulas de programa extendido* (CPE) y *cláusulas de programa rebatible* (CPR). Una CPE es una cláusula de la forma "$l \leftarrow p_1, \ldots, p_n$", $(n \geq 0)$ donde l es un literal, y cada p_i es un literal o un literal precedido por el símbolo *not* de la negación por falla. Si n=0, entonces se denotará "$l \leftarrow$ TRUE", y se dirá que l es un *hecho* (donde TRUE tiene la interpretación usual). Una CPR en cambio, es una cláusula de la forma "$l \dashrightarrow p_1, \ldots, p_n$", con las mismas consideraciones para los p_i que en las CPE. El símbolo "$\dashrightarrow$" se utiliza para distinguir una CPR de una CPE, porque una CPR se utilizará para representar conocimiento rebatible, *i.e.*, información tentativa que puede ser usada en la medida que no sea contradecida. La cláusula "$l \dashrightarrow A$" debe leerse como: "razones para creer en el antecedente A son buenas razones para creer en el consecuente l". En una CPR, si n=0 se denotará "$l \dashrightarrow$ TRUE", y se dira que l es una *presuposición.*

En lo que sigue se denotará con $\mathcal{S}$, al conjunto de CPE de un PLR, y con $\mathcal{D}$, al conjunto de CPR. Una *meta definida* es simplemente un literal m. Una *prueba rebatible* para una meta definida m (ver apéndice, definición 6.1) es el conjunto de cláusulas de programa instanciadas (CPE y CPR), que permiten derivar m. Un conjunto de cláusulas es *consistente* (resp. *inconsistente*) si no es posible (resp. es posible) probar rebatiblemente un par de literales complementarios. Dado un PLR, el conjunto $\mathcal{S}$ de CPE debe ser consistente, mientras que el conjunto $\mathcal{D}$ de CPR y el propio PLR pueden ser inconsistentes. Al final de este trabajo se presenta un apéndice donde se amplían estos conceptos (más detalles pueden encontrarse en [15] y [4]).

El objetivo de incluir la negación clásica es poder representar información potencialmente inconsistente, ya que tanto una meta de la forma "$\neg a$", como una de la forma "a" pueden ser derivadas rebatiblemente de un PLR. Esto permite escribir cláusulas como "¬peligroso(X) $-\prec$ niño(X)" o "peligroso(X) $-\prec$ ¬conocido(X)". La negación por falla, en cambio, se incluye para poder trabajar con información incompleta. Es por ello que la utilización de la negación por falla se restringe sólo al cuerpo de una cláusula, por ejemplo, "peligroso(X) $-\prec$ *not* peligroso(X)".

La consulta de una meta m tendrá éxito, si m pertenece al *conjunto de respuestas positivas* del lenguaje, esto es, existe un argumento que es una justificación de m. El proceso de obtención de una justificación para m, involucra la construcción de un *argumento aceptable* para m. Un *argumento* para una meta m es el subconjunto de CPR instanciadas utilizadas en la generación de una prueba rebatible para m (ver apéndice definición 6.2). Para decidir sobre la aceptabilidad de un argumento, se construyen a partir del PLR, *contraargumentos* que son posibles *derrotadores* (ver apéndice, definiciones 6.3 y 6.4). De igual forma, se verifica la aceptabilidad de los posibles derrotadores. Aquellos derrotadores aceptables son comparados con el argumento original usando un *criterio de preferencia*. Un argumento es una justificación, si es mejor que todos sus derrotadores aceptados. La semántica de un PLR queda caracterizada por cuatro conjuntos de repuestas: *positivas, negativas, indecisas, y desconocidas* (ver apéndice definición 6.8).

2. La criterio de especificidad en los PLR

Cuando dos metas complementarias pueden ser derivadas rebatiblemente de un PLR, resulta necesario definir un criterio de inferencia, a fin de que sólo una de las metas tenga éxito. Para ello se utiliza un criterio de comparación entre los argumentos que sustentan dichas metas. La comparación de argumentos es un problema abierto, y existen diferentes propuestas que intentan solucionarlo. Una de ellas es el *criterio de especificidad*, el cuál fue descripto por Poole en [12], y separadamente por Loui en [10]. Algunos sistemas de razonamiento rebatible han utilizado este criterio para comparar sólo reglas rebatibles (ver Nute [11]), y otros para comparar argumentos enteros (ver Simari [14]).

La definición 2.1, introduce la relación de especificidad entre argumentos del sistema de argumentación rebatible definido en [14]. En dicho sistema, $\Delta^{\downarrow}$ representa el conjunto de reglas rebatibles instanciadas, y $\mathcal{K}$ al conjunto de reglas no rebatibles ($\mathcal{K}= \mathcal{K}_G \cup \mathcal{K}_P$), distinguiendo con $\mathcal{K}_P$ al conocimiento particular (subconjunto de hechos instanciados) y con $\mathcal{K}_G$ al concimiento general (reglas sin instanciar). El símbolo "$\mid\sim$ " se utiliza para denotar la derivación rebatible.

Definición 2.1 Especificidad [14]
Sea $\mathcal{L} = \{l : l$ es un literal instanciado y $\mathcal{K} \cup \Delta^{\downarrow} \mid\sim l \}$. El argumento $\mathcal{A}$ para h_1 es *estrictamente más específico* que el argumento $\mathcal{B}$ para h_2, si y solo si

1. para todo conjunto $C \subseteq \mathcal{L}$
   si $\mathcal{K}_G \cup C \cup \mathcal{A} \mid\sim h_1$ (C activa $\mathcal{A}$) y $\mathcal{K}_G \cup C \not\mid\sim h_1$, (activación no trivial)
   entonces $\mathcal{K}_G \cup C \cup \mathcal{B} \mid\sim h_2$. ($C$ activa $\mathcal{B}$)
2. existe un conjunto $C' \subseteq \mathcal{L}$ tal que:
   $\mathcal{K}_G \cup C' \cup \mathcal{B} \mid\sim h_2$, ($C'$ activa $\mathcal{B}$), $\mathcal{K}_G \cup C' \not\mid\sim h_2$ (activación no trivial)
   y $\mathcal{K}_G \cup C' \cup \mathcal{A} \not\mid\sim h_1$. ($C'$ no activa $\mathcal{A}$)

Si un argumento $\mathcal{A}$ es estrictamente más específico que $\mathcal{B}$, se denotará $\mathcal{A} \succ \mathcal{B}$. En el caso que $\mathcal{A} \not\succ \mathcal{B}$ y que $\mathcal{B} \not\succ \mathcal{A}$, se dirá que $\mathcal{A}$ y $\mathcal{B}$ son incomparables, y se notará $\mathcal{A} \not\approx \mathcal{B}$. □

Ejemplo 2.1 Considérense los argumentos $\mathcal{A}_1 = \{ \text{a}_1 \prec \text{b,c} \}$, $\mathcal{A}_2 = \{ \text{a}_2 \prec \text{b} \}$, $\mathcal{A}_3 = \{ a_3 \prec \text{d} \}$. $\mathcal{A}_4 = \{ \text{a}_4 \prec \text{c} ; \text{c} \prec \text{d} \}$. Utilizando el criterio de especificidad, resulta: $\mathcal{A}_1 \succ \mathcal{A}_2$, $\mathcal{A}_3 \succ \mathcal{A}_4$, $\mathcal{A}_1 \not\approx \mathcal{A}_3$, $\mathcal{A}_2 \not\approx \mathcal{A}_3$, $\mathcal{A}_1 \succ \mathcal{A}_4$, y $\mathcal{A}_2 \not\approx \mathcal{A}_4$. □

Observación 1 : Dado el PLR formado por $\mathcal{S} = \{a \leftarrow h ; h \leftarrow \text{TRUE} \}$ y $\mathcal{D} = \{ b \prec h \}$, a partir de él se puede obtener el argumento $\mathcal{A} = \{\}$ para el literal a, y el argumento $\mathcal{B} = \{$ b $\prec$ h$\}$ para el literal b. Notese que un argumento vacío, corresponde a una derivación estricta, esto es, no utiliza ninguna CPR, y por lo tanto no hay nada en el que pueda ser refutado. Contrario a las espectativas de que el argumento $\mathcal{A}$ sea más específico que $\mathcal{B}$, siguiendo la definición 2.1 ambos argumentos son incomparables. A fin de que la especificidad se comporte adecuadamente con argumentos vacíos, un argumento vacío será preferido a uno no vacío, y en el caso que ambos sean vacíos se considerarán incomparables. Cabe destacar, que si en particular se comparan dos argumentos para literales complementarios, este problema no aparece, ya que la condición de consistencia de la definición de argumento impedirá la formación del argumento no vacío. Por ejemplo en el PLR $\mathcal{P} = \{a \leftarrow h ; h \leftarrow \text{TRUE} ; \neg a \prec h \}$ es imposible construir un argumento para $\neg a$. □

El costo computacional de la definición 2.1 es muy alto, ya que necesita considerar todos los subconjuntos de $\mathcal{L}$. El conjunto $\mathcal{L}$ contiene a todos los literales que se pueden derivar rebatiblemente del programa, y si tiene n elementos, entonce hay 2^n subconjuntos para considerar. El principal problema es que al aplicar la definición, se están considerando gran cantidad de conjuntos de literales que no están relacionados con los argumentos en comparación.

Una definición equivalente a la 2.1, pero con un menor costo computacional, puede obtenerse si se focaliza únicamente sobre los literales de los argumentos. A continuación se introducirán algunos conceptos a fin de obtener la nueva definición de especificidad. En lo que resta de la sección, se asumirá que las cláusulas de programa no poseen presuposiciones ni submetas con el operador "*not*". La incorporación de estos dos nuevos elementos se analizará en las dos secciones siguientes. Para adaptar la definición de especificidad a los PLR, se debe definir al símbolo "$\mid\sim$ " como una abreviatura de "existe una prueba rebatible", y reemplazar a $\mathcal{K}$ por el conjunto de CPE $\mathcal{S}$, y a $\Delta^{\downarrow}$ por el conjunto de CPR instanciadas.

Definición 2.2 *Argumento completado.* Sea $\mathcal{A}$ un argumento para el literal instanciado h, el argumento completado de $\mathcal{A}$, denotado $\mathcal{A}^c$, es un conjunto de CPE y CPR formado por las CPR de $\mathcal{A}$, más el subconjunto de CPE de $\mathcal{S}$ que no son hechos, y que fueron utilizadas para obtener la prueba de h a partir de $\mathcal{S} \cup \mathcal{A}$. □

Ejemplo 2.2 : En el PLR formado por $\mathcal{S}$={ h ← a ; b ← d ; d ← TRUE ; c ← TRUE}, y $\mathcal{D}${ a $-\prec$ b,c }, $\mathcal{A}$= {a $-\prec$ b,c} es un argumento para el literal h. El argumento completado para $\mathcal{A}$ es $\mathcal{A}^c$={ h ← a ; a $-\prec$ b,c ; b ← d }. □

Observación 2 Dado un argumento $\mathcal{A}$ para h, no existe un único argumento completado $\mathcal{A}^c$, ya que puede haber diferentes reglas en $\mathcal{S}$ con las cuales construir $\mathcal{A}$. No obstante, la diferencia entre dos argumentos completados $\mathcal{A}^c$ y $\mathcal{A}^{c\prime}$, son sólo CPE. □

Definición 2.3 *Conjunto de literales básicos de $\mathcal{A}^c$.* Sea $\mathcal{A}^c$ un argumento completado, el conjunto de literales básicos de $\mathcal{A}^c$ denotado $Lit(\mathcal{A}^c)$, es el conjunto de literales que aparecen en los antecedentes y consecuentes de toda cláusula de $\mathcal{A}^c$. □

Definición 2.4 *Conjunto de activación.* Sea $\mathcal{A}^c$ un argumento completado, y sea $Lit(\mathcal{A}^c)$ el conjunto de literales correspondiente. Un subconjunto $U \subseteq Lit(\mathcal{A}^c)$ es un *conjunto de activación* de $\mathcal{A}^c$, si U junto con $\mathcal{A}^c$ derivan rebatiblemente a h (*i.e.*, $U \cup \mathcal{A}^c \mid\sim h$), y además U es minimal con respecto a la inclusión de conjuntos (*i.e.*, no existe $U' \subseteq U$ tal que $U' \cup \mathcal{A}^c \mid\sim h$). Se denotará con Act-sets($\mathcal{A}^c$), al conjunto de todos los conjuntos de activación de $\mathcal{A}^c$. □

Ejemplo 2.3 La diferencia entre considerar sólo los conjuntos de activación en lugar de todos los subconjuntos de $\mathcal{L}$, es apreciable en el siguiente ejemplo. Sea $\mathcal{P}$ el PLR: $\mathcal{S}$= { a ← b,c ; c ← f,g ; z ← h ; x ← h ; y ← h ; h ← TRUE ; i ← TRUE ; j ← TRUE ; k ← TRUE ; g ← TRUE } y, $\mathcal{D}$= { b $-\prec$ d,e ; d $-\prec$ h,i ; f $-\prec$ j,k ; t $-\prec$ x ;u $-\prec$ y }.
Sea $\mathcal{A}^c$ el argumento completado: { a ← b,c ; b $-\prec$ d,e ; d $-\prec$ h,i ; c ← f,g ; f $-\prec$ j,k }

En este caso $\mathcal{L}$= {a,b,c,d,e,f,g,h,i,j,k,u,t,x,y,z }, por lo tanto hay 2^{16} subconjuntos para considerar, además $Lit(\mathcal{A}^c)$= {a,b,c,d,e,f,g,h,i,j,k}, lo cuál sólo bajaria el número de subconjuntos a 2.048. Sin embargo, solamente hay 10 conjuntos de activación para considerar: {a}, {b,c}, {d,e,c}, {h,i,e,c}, {h,i,e,f,g}, {h,i,e,j,k,g}, {d,e,f,g}, {d,e,j,k,g}, {b,f,g}, {b,j,k,g}. □

En el ejemplo 2.3 { b,c }, {a}, y {b,f,g} son conjuntos de activación que trivialmente derivan la conclusión h. Esta clase de conjuntos de activación no son tenidos en cuenta por la definición de especificidad, ya que la condición "$U \cup \mathcal{K}_G \not\vdash h$", lo prohibe.

Definición 2.5 : *Conjunto de activación no trivial.* Dado un argumento completado $\mathcal{A}^c$ para la meta h, se dirá que U es un *conjunto de activación no trivial* de $\mathcal{A}^c$, si U es un conjunto de activación de $\mathcal{A}^c$, y $U \cup \mathcal{K}_G \not\vdash h$. Se denotará con NTAct-sets($\mathcal{A}^c$) al conjunto de todos los conjuntos de activación no triviales de $\mathcal{A}^c$. □

Algoritmo 1: *Conjuntos de activación no triviales*
Entrada: El argumento completado $\mathcal{A}^c$ para h.
Salida: NTAct-sets($\mathcal{A}^c$) y Act-sets($\mathcal{A}^c$).

1. Se inicializa una pila P con el par ({ h },trivial).
2. Act-sets($\mathcal{A}^c$) y NTAct-sets($\mathcal{A}^c$) se inicializan en vacío.
3. REPETIR HASTA QUE P esté vacía
   (a) Seleccionar el par (n,*tipo*) del tope de P,
   (b) Para cada literal $l_i \in n$ que sea consecuente de una cláusula r de $\mathcal{A}^c$, reemplazar en n a l_i por los literales del antecedente de r, con lo cuál se obtiene un nuevo conjunto de activación n_i.
   El *tipo* de cada n_i será *trivial* sólo si el *tipo* de n es *trivial* y r es un CPE. En caso contrario el *tipo* de de n_i será *no trivial.*
   (c) Todos los nuevos conjuntos de activación n_i que no hallan sido previamenente expandidos, son agregados al tope de P.
   (d) El conjunto n se agrega a Act-sets($\mathcal{A}^c$).
   Si el tipo de n es *no trivial* entonces n se agrega a NTAct-sets($\mathcal{A}^c$).
4. RETORNAR NTAct-sets($\mathcal{A}^c$) y Act-sets($\mathcal{A}^c$).

El algoritmo 1 indica como calcular todos los conjuntos de activación no triviales de un argumento completado. Para determinar si un conjunto de activación es trivial o no, se observa si fue usada o no una CPR. Nótese que dado un argumento $\mathcal{A}$ para una meta h, el conjunto $\{h\}$ es un conjunto de activación trivial de $\mathcal{A}$.

Como puede verse, el algoritmo 1 calcula todos los conjuntos de activación de una manera eficiente, simplemente recorriendo el argumento completado. Aplicando el algoritmo al PLR del ejemplo 2.3, se obtiene NTAct-sets($\mathcal{A}^c$)= { {d,e,c}, {h,i,e,c}, {h,i,e,f,g}, {h,i,e,j,k,g}, {d,e,f,g}, {d,e,j,k,g}, {b,j,k,g} }. El siguiente paso será entonces definir el criterio de especificidad para que sólo tenga que considerar los conjuntos de activación no triviales.

Definición 2.6 *Especificidad*
Dado el argumento no vacío $\mathcal{A}$ para la meta h_1, y el argumento no vacío $\mathcal{B}$ para h_2, y $\mathcal{A}^c$, $\mathcal{B}^c$ sus respectivos argumentos completados. Se dirá que $\mathcal{A}$ es *estrictamente más específico* que $\mathcal{B}$ si y sólo si,
(1) para todo conjunto $U \in$ NTAct-sets($\mathcal{A}^c$), se cumple que $U \cup \mathcal{K}_G \cup \mathcal{B} \mid\sim h_2$, y
(2) existe un conjunto $U' \in$ NTAct-sets($\mathcal{B}^c$), se cumple que $U' \cup \mathcal{A}^c \not\mid\sim h_1$.
En el caso que alguno de los argumentos sea vacío se procederá como indica la observación 1 □

La definición 2.6 será equivalente a la 2.1 sólo cuando halla un único completamiento para cada argumento (ver observación 2). Sin embargo se puede construir una definición equivalente de la siguiente forma: supóngase que en lugar de un único $\mathcal{A}^c$, existe un conjunto $\{\mathcal{A}^c{}_i\}$ de argumentos completados para $\mathcal{A}$. Se puede definir los

conjuntos Act-sets($\mathcal{A}$)=$\bigcup_{i=1}^{n}$ Act-sets($\mathcal{A}^c{}_i$) y NTAct-sets($\mathcal{A}$)=$\bigcup_{i=1}^{n}$ NTAct-sets($\mathcal{A}^c{}_i$), y entonces reformular la nueva definición de especificidad, reemplazando NTAct-sets($\mathcal{A}^c$) por NTAct-sets($\mathcal{A}$), y NTAct-sets($\mathcal{B}^c$) por NTAct-sets($\mathcal{B}$).

Proposición 2.1 *La definición 2.6 en términos de NTAct-sets($\mathcal{A}$), y NTAct-sets($\mathcal{B}$) es equivalente a la definición 2.1.* La demostración puede encontrarse en [3]. □

3. La especificidad y la negación por falla

La relación de especificidad no fue definida para manejar situaciones donde esté presente la negación por falla. Al incluir el operador "*not*" en el lenguaje, hay que definir un criterio para que puedan compararse por especificidad argumentos que contengan cláusulas con "*not*". En esta sección se analizarán diferentes alternativas y se desarrollará una extensión de la relación de especificidad. Considérese el siguiente ejemplo:

Ejemplo 3.1 Dado el PLR:

peligroso(X) $-\!\prec$ *not* perro-bueno(X)
¬peligroso(X) $-\!\prec$ *not* perro-bueno(X), *not* mordió-alguien(X)
¬peligroso(X) $-\!\prec$ ¬perro-callejero(X)
¬perro-callejero(negro) $\leftarrow$ TRUE

se puede construir el argumento $\mathcal{A}$ = { peligroso(negro) $-\!\prec$ *not* perro-bueno(negro) } para la meta "peligroso(negro)", y los argumentos $\mathcal{B}$ = { ¬peligroso(negro) $-\!\prec$ *not* perro-bueno(negro), *not* mordio-alguien(negro) }, y $\mathcal{C}$ = { ¬peligroso(negro) $-\!\prec$ ¬perro-callejero(negro) } para la meta "¬peligroso(negro)". □

En el ejemplo anterior, uno esperaría que el argumento $\mathcal{B}$ sea más específico que $\mathcal{A}$ ($\mathcal{B} \succ \mathcal{A}$), ya que $\mathcal{B}$ está utilizando mayor información para llegar a la conclusión. En el caso de $\mathcal{C}$ y $\mathcal{A}$ ambos utilizan diferente información, por lo que se esperaría que ninguno fuera más específico que el otro ($\mathcal{A} \not\succ \mathcal{C}$). Como se verá a continuación existen diferentes formas de considerar submetas de la forma "*not p*" al momento de comparar por especificidad.

El criterio más simple y directo, sería ignorar todas las submetas "*not p*", esto es, no incluirlas en los conjuntos de activación. Siguiendo esta propuesta, los argumentos $\mathcal{A}$ y $\mathcal{B}$ del ejemplo 3.1 no tendrían conjuntos de activación no triviales, y por lo tanto resultarían incomparables ($\mathcal{A} \not\succ \mathcal{B}$). El argumento $\mathcal{C}$ tendría un solo conjunto de activación: { ¬perro-callejero(negro) }, y por lo tanto $\mathcal{C}$ sería estrictamente más específico que $\mathcal{A}$. Como puede verse, los resultados de utilizar esta propuesta no son los esperados.

Como una submeta "*not p*" tiene éxito cuando no existe un justificación para p, las apariciones de una submeta "*not p*" pueden considerarse como información muy débil. Otro criterio sería entonces preferir aquellos argumentos que tengan menos submetas con "*not*". Sin embargo, utilizando este criterio, $\mathcal{C} \succ \mathcal{A}$ y $\mathcal{B} \succ \mathcal{A}$, lo cuál tampoco es el resultado esperado.

De lo analizado hasta el momento puede verse, que resulta importante que las submetas con el operador "*not*" sean consideradas de igual manera que un literal probado rebatiblemente. Por lo tanto, se debe considerar a las submetas "*not p*" de un argumento como un literal más dentro del conjunto de activación. Para implementar este criterio, únicamente hay que modificar la definición 2.3:

Definición 3.1 *Conjunto de literales básicos de $\mathcal{A}^c$.* Sea $\mathcal{A}^c$ un argumento completado, el conjunto de literales básicos de $\mathcal{A}^c$ denotado $Lit(\mathcal{A}^c)$, es el conjunto de literales que aparecen en los antecedentes y consecuentes de toda cláusula de $\mathcal{A}^c$, más las submetas de la forma "*not p*" (*p* un literal instanciado), que aparecen en las cláusulas de $\mathcal{A}^c$. □

De esta forma, en el ejemplo 3.1, el conjunto de activación no trivial de $\mathcal{A}$ es { *not* perro-bueno(negro) }, el de $\mathcal{B}$ { *not* perro-bueno(negro), *not* mordio-alguien(negro) }, y el de $\mathcal{C}$ es { ¬perro-callejero(negro) }. Utilizando ahora la definición de especificidad con estos conjuntos de activación, $\mathcal{B} \succ \mathcal{A}$, y $\mathcal{C} \not\succ \mathcal{A}$.

Esta alternativa es la que mejor responde a las espectativas propuestas, ya que prefiere aquellos argumentos que están basados en información más específica. Por lo tanto este será el criterio elegido para extender la definición de especificidad. En la tabla 1 se muestran diferentes ejemplos, con el resultado de aplicar el criterio elegido. En todos los casos se asume que están presente los siguientes hechos: "h ← TRUE" y "a ← TRUE".

a)	$\mathcal{A}_a$ ={	$c_a \prec$ a }	b)	$\mathcal{A}_b$ ={	$c_b \prec$ a, *not* p }
	$\mathcal{B}_a$ ={	$\neg c_a \prec$ *not* p }		$\mathcal{B}_b$ ={	$\neg c_b \prec$ *not* p }
		$\mathcal{A}_a \not\succ \mathcal{B}_a$			$\mathcal{A}_b \succ \mathcal{B}_b$
c)	$\mathcal{A}_c$ ={	$c_c \prec$ a, *not* p }	d)	$\mathcal{A}_d$ ={	$c_d \prec$ *not* p, *not* q }
	$\mathcal{B}_c$ ={	$\neg c_c \prec$ a }		$\mathcal{B}_d$ ={	$\neg c_d \prec$ *not* q }
		$\mathcal{A}_c \succ \mathcal{B}_c$			$\mathcal{A}_d \succ \mathcal{B}_d$
e)	$\mathcal{A}_e$ ={	$c_e \prec$ *not* p, *not* q}	f)	$\mathcal{A}_f$ ={	$c_f \prec$ *not* p }
	$\mathcal{B}_e$ ={	$\neg c_e \prec$ a }		$\mathcal{B}_f$ ={	$\neg c_f \prec$ TRUE}
		$\mathcal{A}_e \not\succ \mathcal{B}_e$			$\mathcal{A}_f \succ \mathcal{B}_f$
g)	$\mathcal{A}_g$ ={	$c_g \prec$ a,h, *not* p }	h)	$\mathcal{A}_h$ ={	$c_h \prec \neg$p }
	$\mathcal{B}_g$ ={	$\neg c_g \prec$ a }		$\mathcal{B}_h$ ={	$\neg c_h \prec$ *not* p }
		$\mathcal{A}_g \succ \mathcal{B}_g$			$\mathcal{A}_h \not\succ \mathcal{B}_h$

Tabla 1: Ejemplos con el operador "*not*".

Observación 3 : Dentro de un argumento, los literales que tienen negación por falla sólo están como hojas del árbol de derivación rebatible. Esto permite eliminar del análisis la comparación por directitud [10], ya que no es posible una cadena de reglas con "not". □

4. Comparación de argumentos que continen presuposiciones

Una *presuposición* es una CPR con antecedente vacío, la cuál se denota "$p \prec$ TRUE", mientras que un *hecho* es una CPE denotada "$h \leftarrow$ TRUE". Los hechos representan información altamente segura ya que son cláusulas no rebatibles incondicionales, mientras que las presuposiciones representan información tentativa, introducida por el programador, para ser usada por el PLR ante la ausencia de mejor información. La definición de especificidad estaba desarrollada para argumentos donde no había presuposiciones. Por lo tanto, al comparar dos argumentos hay que tener en cuenta esta nueva situación.

En la tabla 2 se muestran diferentes casos donde se comparan argumentos que poseen presuposiciones. En todos los casos se asume que están presentes los siguientes hechos: "h $\leftarrow$ TRUE" y "a $\leftarrow$ TRUE". A continuación se analizarán diferentes alternativas para extender el criterio de especificidad, a fin de que considere argumentos que tienen presuposiciones.

$\mathcal{A}_a = \{$ $c_a \prec$ h $\}$ $\mathcal{B}_a = \{$ $\neg c_a \prec$ TRUE $\}$	$\mathcal{A}_b = \{$ $c_b \prec$ a,h,s s $\prec$ TRUE $\}$ $\mathcal{B}_b = \{$ $\neg c_b \prec$ a$\}$	$\mathcal{A}_c = \{$ $c_c \prec$ s,h s $\prec$ TRUE $\}$ $\mathcal{B}_c = \{$ $\neg c_c \prec$ h $\}$
$\mathcal{A}_d = \{$ $c_d \prec$ s,h s $\prec$ TRUE $\}$ $\mathcal{B}_d = \{$ $\neg c_d \prec$ s s $\prec$ TRUE $\}$	$\mathcal{A}_e = \{$ $c_e \prec$ s,r s $\prec$ TRUE r $\prec$ TRUE $\}$ $\mathcal{B}_e = \{$ $\neg c_e \prec$ r r $\prec$ TRUE $\}$	$\mathcal{A}_f = \{$ $c_f \prec$ h $\}$ $\mathcal{B}_f = \{$ $\neg c_f \prec$ s,r s $\prec$ TRUE r $\prec$ TRUE $\}$
$\mathcal{A}_g = \{$ $c_g \prec$ TRUE $\}$ $\mathcal{B}_g = \{$ $\neg c_g \prec$ s s $\prec$ TRUE $\}$	$\mathcal{A}_h = \{$ $c_h \prec$ s s $\prec$ TRUE $\}$ $\mathcal{B}_h = \{$ $\neg c_h \prec$ q q $\prec$ s s $\prec$ TRUE$\}$	$\mathcal{A}_i = \{$ $c_i \prec$ p p $\prec$ h $\}$ $\mathcal{B}_i = \{$ $\neg c_i \prec$ TRUE $\}$
$\mathcal{A}_j = \{$ $c_j \prec$ h$\}$ $\mathcal{B}_j = \{$ $\neg c_j \prec$ s s $\prec$ TRUE $\}$	$\mathcal{A}_k = \{$ $c_k \prec$ p p $\prec$ h $\}$ $\mathcal{B}_k = \{$ $\neg c_k \prec$ s s $\prec$ TRUE$\}$	$\mathcal{A}_l = \{$ $c_l \prec$ TRUE $\}$ $\mathcal{B}_l = \{$ $\neg c_l \prec$ TRUE $\}$
$\mathcal{A}_m = \{$ $c_m \prec$ h$\}$ $\mathcal{B}_m = \{$ $\neg c_m \prec$ p,q p $\prec$ s ; q $\prec$ r s $\prec$ TRUE r $\prec$ TRUE $\}$	$\mathcal{A}_n = \{$ $c_n \prec$ h$\}$ $\mathcal{B}_n = \{$ $\neg c_n \prec$ p p $\prec$ s s $\prec$ TRUE $\}$	$\mathcal{A}_o = \{$ $c_o \prec$ TRUE $\}$ $\mathcal{A}_o = \{$ $\neg c_o \prec$ s,r s $\prec$ TRUE r $\prec$ TRUE $\}$

Tabla 2: Ejemplos con presuposiciones.

En la tabla 3 figuran los resultados de comparar los argumentos $\mathcal{A}$ y $\mathcal{B}$ de todos los casos, con cada una de las alternativas planteadas. Por ejemplo, si en la columna (b) fila (1) figura $\succ$, significa que $\mathcal{A}_b \succ \mathcal{B}_b$.

Resultado	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
esperado	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\not\succ$	$\succ$	$\succ$	$\succ$
(1)	$\succ$	$\succ$	$\not\succ$	$\succ$	$\not\succ$	$\succ$	$\not\succ$	$\not\succ$	$\succ$	$\succ$	$\succ$	$\not\succ$	$\succ$	$\succ$	$\not\succ$
(2)	$\prec$	$\succ$	$\succ$	$\succ$	$\succ$	$\not\succ$	$\not\succ$	$\succ$	$\prec$	$\not\succ$	$\not\succ$	$\not\succ$	$\not\succ$	$\not\succ$	$\not\succ$
(3)	$\prec$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\prec$	$\succ$	$\succ$	$\not\succ$	$\succ$	$\succ$	$\succ$
(4)	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$	$\not\succ$	$\succ$	$\succ$	$\succ$

Tabla 3: Resultados de aplicar las diferentes alternativas.

Definición 4.1 Un argumento completado estará *basado en hechos*, si fue utilizado al menos un hecho en su derivación rebatible, o posee una submeta "*not p*". De lo contrario, *i.e.*, se utilizaron sólo presuposiciones, se dirá que no esta basado en hechos, o que está *basado en presuposiciones.* □

Alternativa 1: usar la definición de especificidad tal cuál está planteada. En este caso, como las presuposiciones forman parte del argumento completado, estas se activarán con cualquier conjunto de activación. Como puede verse en los resultados expuestos en la tabla 3, en los casos e, g, y h, donde los dos argumentos están basados sólo en presuposiciones, la relación de especificidad no se comporta de acuerdo a lo esperado. Lo mismo pasa en casos (como el c) donde la diferencia entre dos argumentos es un conjunto de presuposiciones. Este comportamiento es debido a que las presuposiciones están incluídas en el argumento, y los conjuntos de activación no tienen el efecto deseado.

Alternativa 2: sacar las presuposiciones del argumento completado y usar la definición de especificidad tal cuál está planteada. En esta propuesta, las presuposiciones no pueden activarse con cualquier conjunto de activación, ya que no están en el argumento completado. De esta manera, los casos e, g y h dan el resultado esperado. Lamentablemente, en casos como f, j, k, m, n y o, donde antes se lograba el resultado esperado, ahora no es así, debido a que desde el punto de vista de los conjuntos de activación las presuposiciones se convierten en hechos. Sin embargo, es posible remediar fácilmente esta situación, si en esos casos se prefieren los argumentos que están basados en hechos sobre los que están basados en presuposiciones.

Alternativa 3: utilizar la alternativa 2, y en los casos que sean incomparables preferir aquellos argumentos que estén basados en hechos por sobre aquellos que estén basados en presuposiciones (los resultados pueden verse en la tabla 3). Esto mejora muchos de los casos de la alternativa 2, salvo los casos como a o i donde uno de los argumentos es vacio pero basado en presuposiciones. En estos casos el criterio establecido en la observación 1 indicaba preferir el argumento vacío, pero en ese punto todavía no habían sido consideradas las presuposiciones.

Alternativa 4: utilizar la alternativa 3, pero con el siguiente criterio para los argumentos vacíos: (a) los argumentos vacíos basados en hechos, son preferidos a los no vacíos, y a los vacíos basados en presuposiciones; y (b) los argumentos vacíos basados en presuposiciones son preferidos sólo a los no vacíos basados en presuposiciones. Esta alternativa

corresponde a un refinamiento de las alternativas 2 y 3, y da los resultados esperados. Por lo tanto será la elegida para comparar argumentos en los programas lógicos rebatibles.

5. Conclusiones

La especificidad ha demostrado ser un buen criterio de comparación entre argumentos. En este trabajo se muestra como puede definirse el criterio de especificidad basado en conjuntos de activación, y de esta forma poder realizar una implementación eficiente. Como la programación en lógica rebatible incorpora conceptos que no estaban presentes en los sistemas de argumentación rebatible, se desarrolló una extensión de la definición de especificidad para poder considerar argumentos que contengan presuposiciones y el operador de negación por falla.

6. Apéndice: programación en lógica rebatible

Las siguientes definiciones fueron extraidas de [15, 4, 13], y corresponden a los principales conceptos de la programación en lógica rebatible, y de la argumentación rebatible (por más detalles referirse a [14, 13, 6]).

Definición 6.1 Una *prueba rebatible* para una meta definida m a partir de un PLR $\mathcal{P}$, es un conjunto finito de CPE y CPR definido recursivamente de la siguiente forma:

1. Si existe un hecho "$c \leftarrow$ TRUE" (presuposición "$c \dashrightarrow$ TRUE") en $\mathcal{P}$, tal que m unifica con c, con unificador mas general σ), el conjunto $\{c\sigma \leftarrow \text{TRUE}\}$ ($\{c\sigma \dashrightarrow \text{TRUE}\}$) es una prueba rebatible para m.
2. Si existe una CPE "$c \leftarrow L$" (CPR "$c \dashrightarrow L$") tal que c unifica con m (con umg σ), y existe un prueba rebatible $\mathcal{F}_i$ para cada uno de los elementos de $L\sigma$, entonces, $\{c\sigma \leftarrow L\sigma\} \cup (\bigcup_i \mathcal{F}_i)$ ($\{c\sigma \dashrightarrow L\sigma\} \cup (\bigcup_i \mathcal{F}_i)$) es una prueba rebatible para m. A fin de que no se produzcan ciclos, la cláusula "$c\sigma \leftarrow L\sigma$" ("$c\sigma \dashrightarrow L\sigma$") no debe aparecer en ninguno de los conjuntos $\mathcal{F}_i$.
3. Si m es una submeta que tiene el operador de negación por falla *not* (*i.e.*, m es *not* l), y no existe una *justificación* para el literal l, entonces el conjunto vacio es una prueba rebatible para m.
4. Si no se da ninguno de los casos anteriores, entonces no existe una prueba rebatible para m.

Definición 6.2 Dado un PLR, formado por el conjunto $\mathcal{S}$ de CPE, y el conjunto $\mathcal{D}$ de CPR, un *argumento* $\mathcal{A}$ para un literal h, es un subconjunto de CPR instanciadas, tal que: (1) Existe una prueba rebatible de m a partir de $\mathcal{S} \cup \mathcal{A}$ (denotado $\mathcal{S} \cup \mathcal{A} \mid\sim m$), (2) $\mathcal{S} \cup \mathcal{A}$ es consistente, y (3) $\mathcal{A}$ es el menor subconjunto (con respecto a la inclusión de conjuntos) que cumple las dos condiciones anteriores.
Si $\mathcal{A}$ es un argumento para h, también se dirá que $\langle A, h\rangle$ es una *estructura de argumento*. Además $\langle B, q\rangle$ es un *subargumento* de $\langle A, h\rangle$ si y sólo si, $B \subseteq A$.

Definición 6.3 $\langle A_1, h_1\rangle$ *contraargumenta a* $\langle A_2, h_2\rangle$ en un literal h, sssi, (1) existe un subargumento $\langle A, h\rangle$ de $\langle A_2, h_2\rangle$ tal que $\mathcal{S} \cup \{h, h_1\}$ es inconsistente, y (2) para todo subargumento propio $\langle B, q\rangle$ de $\langle A_1, h_1\rangle$, no es el caso que $\mathcal{S} \cup \{h_2, q\}$ sea inconsistente.

Definición 6.4 : $\langle A_1, h_1\rangle$ *derrota a* $\langle A_2, h_2\rangle$ en un literal h, sssi, existe un subargumento $\langle A, h\rangle$ de $\langle A_2, h_2\rangle$ tal que: $\langle A_1, h_1\rangle$ contraargumenta a $\langle A_2, h_2\rangle$ en el literal h y
(1) $\langle A_1, h_1\rangle$ es estrictamente más específico que $\langle A, h\rangle$ (derrotador propio), o
(2) $\langle A_1, h_1\rangle$ no puede compararse con $\langle A, h\rangle$ (derrotador de bloqueo).

Definición 6.5 : Un *árbol de dialéctica* para $\langle A, h\rangle$, denotado $\mathcal{T}_{\langle A, h\rangle}$, se define recursivamente como sigue:

1. Un nodo que contiene una estructura de argumento $\langle A, h\rangle$ sin derrotadores (propios o de bloqueo), es un árbol de dialéctica para $\langle A, h\rangle$, y es también la raíz del árbol.
2. Supóngase que $\langle A, h\rangle$ es una estructura de argumento con derrotadores (propios o de bloqueo) $\langle A_1, h_1\rangle$, $\langle A_2, h_2\rangle$, ..., $\langle A_n, h_n\rangle$. El árbol de dialéctica $\mathcal{T}_{\langle A, h\rangle}$, para $\langle A, h\rangle$ se construye poniendo a $\langle A, h\rangle$ como nodo raíz, y haciendo que este nodo sea el padre de las raices de los árboles de dialéctica de $\langle A_1, h_1\rangle$, $\langle A_2, h_2\rangle$, ..., $\langle A_n, h_n\rangle$, *i.e.*, $\mathcal{T}_{\langle A_1, h_1\rangle}$, $\mathcal{T}_{\langle A_2, h_2\rangle}$, ..., $\mathcal{T}_{\langle A_n, h_n\rangle}$.

Definición 6.6 Los nodos de un árbol de dialéctica $\mathcal{T}_{\langle A, h\rangle}$ se etiquetan recursivamente como *nodos no-derrotados* (nodos-U) y *nodos derrotados* (nodos-D) como sigue:

1. Las hojas de $\mathcal{T}_{\langle A, h\rangle}$ son *nodos-U*.
2. Sea $\langle B, q\rangle$ un nodo interno de $\mathcal{T}_{\langle A, h\rangle}$. $\langle B, q\rangle$ será un *nodo-U* sssi todo hijo de $\langle B, q\rangle$ es un *nodo-D*. $\langle B, q\rangle$ será un *nodo-D* sssi tiene al menos un hijo que es *nodo-U*.

Definición 6.7 $\langle A, h\rangle$ es una *justifición* para h sssi la raíz de $\mathcal{T}_{\langle A, h\rangle}$ es un nodo-U.

Definición 6.8 *Conjuntos de respuestas de un* PLR $\mathcal{P}$

1. El *conjunto de respuestas positivas* de $\mathcal{P}$ es un conjunto finito de literales L tal que para todo $h \in L$, existe una justificación para h.
2. El *conjunto de respuestas negativas* de $\mathcal{P}$ es un conjunto finito de literales L tal que para todo $h \in L$, se cumple: (a) o bien para cada argumento $\mathcal{A}$ de h, existe un derrotador propio de $\mathcal{A}$; o bien (b) no existe un argumento para h, pero existe un argumento $\mathcal{B}$ que es una justificación para $\sim h$
3. El *conjunto de respuestas indecisas* de $\mathcal{P}$ es un conjunto finito de literales L tal que para todo $h \in L$, se cumple que para todo argumento $\mathcal{A}$ de h, $\mathcal{A}$ no tiene derrotadores propios, pero si tiene al menos un derrotador de bloqueo.
4. El *conjunto de respuestas desconocidas* de $\mathcal{P}$ es un conjunto posiblemente infinito de literales que no pertenecen a los conjuntos de respuestas anteriores.

Definición 6.9 : Dado un PLR $\mathcal{P}$ y una meta definida m, un interprete de programas lógicos rebatibles, responderá:

- SI, en el caso que m pertenezca al conjunto de respuestas positivas.
- NO, en el caso que m pertenezca al conjunto de respuestas negativas.
- INDECISO, en el caso que m pertenezca al conjunto de respuestas indecisas.
- DESCONOCIDO, en el caso que m pertenezca al conjunto de respuestas desconocidas.

7. Referencias

[1] Alferes José J. Pereira Luis M. *Contradiction: when avoidance equals removal (part I and II).* Proc. of Extensions of Logic Programming, 4th International Workshop ELP'93 St. Andrews U.K. March 1993.

[2] Dung Phan M. *On the Acceptability of Argumets and its Fudamental Role in Nonmonotonic Reasoning and Logic Programming.* Proc. of Int. Join Conf. on Artificial Intelligence (IJCAI) 1993.

[3] García A. J. *Computing Specificity.* Reporte Técnico GIIA-UNS, Grupo de Investigación en Inteligencia Artificial, 1996, Universidad Nacional del Sur.

[4] García A. J. *Una aproximación a la programación en lógica rebatible.* 2do. Workshop en Aspectos Teoricos de la Inteligencia Artificial (ATIA'95). Bahía Blanca, Octubre de 1995.

[5] García A. J. y Simari G. R. *Compilación de programas lógicos que utilizan la negación por falla. Una extensión de la máquina abstracta de Warren* 1er. Congreso Argentino de Ciencias de la Computación. Bahía Blanca, Noviembre de 1995.

[6] García A. J., Chesñevar C. I., and Simari G. R. *Making Argument Systems Computationally Attractive.* Proc. XIII Int. Conf. of the Chilean Society for Computer Science, October 1993.

[7] García A. J., Chesñevar C. I., and Simari G. R. *Bases de Argumentos: su mantenimiento y revisión.* XIX Conferencia Latinoamericana de Informática. Buenos Aires, Agosto 1993.

[8] Gelfond M. and Lifschitz V. *Logic Programs with Classical Negation.* Proc. of 7th. Int. Conf. on Logic Programming (ICLP) 1990

[9] Lloyd J. W. *Foundations of Logical Programming.* 2nd. edition, Springer-Verlag 1987

[10] Loui R. P. *Defeats among arguments: a system of defeasible inference* Comp. Intell. 3, 100-106, 1987

[11] Nute Donald, *Basic defeasible logic*, in *Intensional Logics for Programming*, Ed by Luis Fariñas del Cerro, Claredon Press – Oxford (c) 1992.

[12] Poole D. *On the Comparison of Theories: Preferring the most Specific Explanation* Proc. of the 9th Int. Joint Conf. on Artificial Intelligence.

[13] Simari G. R. , Chesñevar C. I., and García A. J. *The Role of Dialectics in Defeasible Argumentation.* XIV Int. Conf. of Chilean Computer Science Society, November 1994.

[14] Simari G. R. and Loui R. P. *A Mathematical Treatment of Defeasible Reasoning and its Implementation.* Artificial Intelligence, 53: 125–157,1992.

[15] Simari G. R. y García A. J. *A Knowledge Representation Language for Defeasible Argumentation.* XXI Conferencia Latinoamericana de Informática. Canela, Brasil, Agosto 1995.

[16] Vreeswijk G. *Studies in Defeasible Argumentation (Ph.D. Thesis).* Vrije University, Amsterdam, Holanda. March 1993.