Una Implementación de Disputación basada en Lógica Default

Claudio Vaucheret Gerardo Parra
cvauche@uncoma.edu.ar, gparra@uncoma.edu.ar
Departamento de Informática y Estadística
Fac. de Economía y Administración
Universidad Nacional del Comahue
Neuquén, NEUQÉN

Resumen

En [2] Gerhard Brewka realizó una reconstrucción de la Teoría de Disputación Formal de Nicholas Rescher [5]. En su trabajo, Brewka completó la formalización de la presentación de *Dialectics: A Controversy-Oriented Approach to the Theory of Knowledge.* El formalismo no-monótono que utilizó para ello es una Logica Default con Especificidad)[1], denominada SDL, que es una variante de la Lógica Default de Reiter[4] que hace preferencia de reglas default más específicas sobre otras más generales.

SDL genera un orden parcial entre las reglas default para modelar la especificidad, y lo utiliza para generar las extensiones de las teorías. Para una misma teoría, SDL genera menos extensiones que la Lógica Default de Reiter pues el orden parcial entre las reglas restringe el número de extensiones posibles solo a aquellas que no violan el orden de especificidad entre las reglas. En este trabajo se describen sucesivamente la Lógica Default con Especificidad y las modificaciones introducidas reemplazando algunas de las definiciones de Brewka. Brevemente se describe la implementación del sistema resultante. El formalismo de Rescher se supone conocido y no será presentado aquí.

Una Implementación de Disputación basada en Lógica Default

1 Introducción

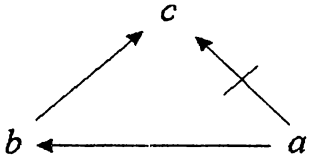
En [2] Gerhard Brewka realizó una reconstrucción de la Teoría de Disputación Formal de Nicholas Rescher [5]. En su trabajo, Brewka completó la formalización de la presentación de *Dialectics: A Controversy-Oriented Approach to the Theory of Knowledge.* El formalismo no-monótono que utilizó para ello es una Logica Default con Especificidad)[1], denominada SDL, que es una variante de la Lógica Default de Reiter[4] que hace preferencia de reglas default más específicas sobre otras más generales.

SDL genera un orden parcial "<" entre las reglas default para modelar la especificidad, y lo utiliza para generar las extensiones de las teorías. Para una misma teoría, SDL genera menos extensiones que la Lógica Default de Reiter pues el orden parcial entre las reglas restringe el número de extensiones posibles solo a aquellas que no violan el orden de especificidad entre las reglas.

Por ejemplo, si D es el conjunto de reglas defaults y C el conjunto de hechos contingentes. En el caso de

$$\begin{aligned} D &= \{a \rightarrow \neg c, b \rightarrow c, a \rightarrow b\} \\ C &= \{a\} \end{aligned}$$

que corresponde al siguiente diagrama:

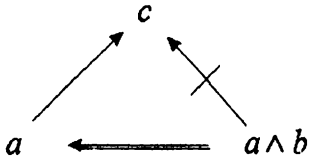


El orden parcial "<" de SDL establece que $a \rightarrow \neg c < b \rightarrow c$ y $a \rightarrow b < b \rightarrow c$. Por lo tanto $\neg c$ estará contenido en todas las extensiones de la teoría y será una conclusión escéptica de la misma. Lo que no ocurre en la Lógica Default ordinaria en donde algunas extensiones contendrán c y otras $\neg c$.

Otro caso donde SDL modela correctamente la especificidad es el siguiente:

$$\begin{aligned} D &= \{a \rightarrow c, a \wedge b \rightarrow \neg c\} \\ C &= \{a, b\} \end{aligned}$$

con diagrama

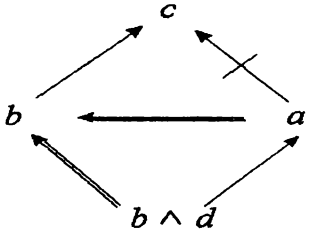


Aquí el orden parcial generado cumple $a \wedge b \rightarrow \neg c < a \rightarrow c$ y $\neg c$ pertenece a todas las extensiones de la teoría. Sin embargo SDL no logra capturar algunos casos de especificidad por la manera en que define el orden entre las reglas.

Consideremos el ejemplo:

$$\begin{aligned} D &= \{a \rightarrow \neg c, b \rightarrow c, b \wedge d \rightarrow a\} \\ C &= \{b, d\} \end{aligned}$$

que corresponde al siguiente diagrama



Resulta que únicamente $a \rightarrow \neg c < b \rightarrow c$. Por lo tanto $\neg c$ pertenece a dos extensiones, pero c pertenece a una extensión y $\neg c$ no es una conclusión escéptica de la teoría como se esperaría. La dificultad aquí es que la regla $b \wedge d \rightarrow a$ entra en conflicto con $b \rightarrow c$ y SDL no la define más específica de acuerdo a su ordenamiento.

Justamente el ejemplo presentado en [2] corresponde al esquema anterior, lo que motivó nuestra redefinición de algunos aspectos de SDL para contemplar estos casos. En la siguiente sección resumimos SDL, luego presentamos las modificaciones propuestas y por último comparamos los resultados de las dos aproximaciones en el ejemplo citado.

2 Lógica Default con especificidad

En la Lógica Default, introducida por Reiter[4], las teorías consisten de un conjunto de hechos W y un conjunto de defaults D. Cada default es de la forma $A{:}B_1, \ldots, B_n/C$. Una teoría default genera extensiones que son definidas como puntos fijos de un operador Γ.

Γ mapea un conjunto de fórmulas S al conjunto más pequeño cerrado deductivamente S' que contiene a W y satisface la condición: si $A{:}B_1, \ldots, B_n/C \in D, A \in S'$ y para todo $i(1 \leq i \leq n)$ $\neg B_i \notin S$ entoces $C \in S'$. La estensiones representan conjuntos de creencias aceptables que un razonador puede adoptar. Pueden ser usadas para definir un arelación de inferencia escéptica donde una fórmula es probable si y solo siestá contenidad en todas las extensiones de (D, W). Brewka utiliza reglas default de la forma $A : B/B$, denomindas *normales*, abreviandolas $A \rightarrow B$.

Para poder incluir la especificidad en la Lógica Default, Brewka en [1] primero desarrolló una versión priorizada de DL llamada PDL donde además de hechos y reglas default es especificado un orden parcial estricto entre las reglas. A continuación mostramos como se definen las extensiones en esta nueva lógica.

Definición 1 *Sea E un conjuntos de fórmulas, $\delta = a \rightarrow c$ un default. Decimos que δ está activo en E sssi (1) $a \in E$, (2) $c \notin E$, y (3) $\neg c \notin E$.*

Definición 2 *Sea $\Delta = (D, W, <)$ una teoría default (priorizada), $\ll$ un orden total conteniendo $<$. Decimos que E es la (PDL) extensión de Δ generada por $\ll$ sssi $E = \cup E_i$, donde $E_0 := Th(W)$, y*

$$E_{i+1} = \begin{cases} E_i & \text{si ningún default esta activo en } E_i \\ Th(E_i \cup \{c\}) & \text{en caso contrario, donde } c \text{ es el consecuente del} \\ & \text{default } \ll\text{-minimal que está activo en } E_i \end{cases}$$

Definición 3 *Sea $\Delta = (D, W, <)$ una teoría default (priorizada). E es una (PDL) extensión de Δ sssi existe un orden total estricto conteniendo a $<$ que genera E.*

Con el fin de definir el ordenamiento de prioridad representando la especificidad, Brewka distinguió dentro de W entre conocimiento básico T y hechos contingentes C. Para un conjunto de reglas default D dado usa $P \vdash_D q$ para expresar que q está contenida en el mas pequeño conjunto de fórmulas, cerrado para la deducción que contiene a P y es cerrado bajo D, donde D es interpretado como un conjunto de reglas de inferencia. Las SDL-extensiones se definen como sigue.

Definición 4 *Sea $\Delta = (D, T, C)$ una teoría default , $D' \subseteq D$ is p-conflictivo (en Δ) si y solo si para algún $\delta = pre \rightarrow cons \in D'$ tenemos*

$$T \cup \{pre\} \vdash_{D'} false.$$

Definición 5 *Sea $\Delta = (D, T, C)$ una teoría default. $<_\Delta$ es el ordenamiento de especificidad asociado con Δ sssi $<_\Delta$ es el ordenamiento parcial estricto mas pequeño sobre Δ que satisface la condición*

$$\delta_1 = pre_1 \rightarrow cons_1 <_\Delta \delta_2 = pre_2 \rightarrow cons_2$$

siempre que

1. *δ_1, δ_2 pertenecen a un conjunto minimal D' de defaults p-conflictivos en Δ y*
2. *$T \cup \{pre_1\} \vdash_{D'} false$, y $T \cup \{pre_2\} \nvdash_{D'} false$.*

3 Modificación de Lógica Default con especificidad

Retomando el ejemplo anterior:

$$\begin{aligned} D &= \{a \rightarrow \neg c, b \rightarrow c, b \wedge d \rightarrow a\} \\ C &= \{b, d\} \end{aligned}$$

Si bien D es p-conflictivo, no es minimal pues $D' = D - \{b \wedge d \rightarrow a\}$ también lo es. Por lo tanto no se establece la relación

$$b \wedge d \rightarrow a <_\Delta b \rightarrow c$$

Como $b \rightarrow c$ puede estar antes que $b \wedge d \rightarrow a$, el default $b \rightarrow c$ puede estar activo y c pertenecerá a una extensión.

Modificaremos la definición de SDL reemplazando la clasificación de p-conflictivo por la de *p-conflictivo por un default.* Es decir establecemos una correspondencia entre el conjunto de reglas y la regla que activa el conflicto.

Definición 4' *Sea $\Delta = (D, T, C)$ una teoría default , $D' \subseteq D$ y $\delta = pre \rightarrow cons \in D'$. D' es p-conflictivo* **por** *δ (en Δ) si y solo si*

$$T \cup \{pre\} \vdash_{D'} false.$$

Definición 5' *Sea $\Delta = (D, T, C)$ una teoría default. $<_\Delta$ es el ordenamiento de especificidad asociado con Δ sssi $<_\Delta$ es el ordenamiento parcial estricto mas pequeño sobre Δ que satisface la condición*

$$\delta_1 = pre_1 \rightarrow cons_1 <_\Delta \delta_2 = pre_2 \rightarrow cons_2$$

siempre que δ_2 pertenezca a un conjunto minimal D' de defaults p-conflictivo por δ_1 en Δ y $T \cup \{pre_2\} \nvdash_{D'} false$.

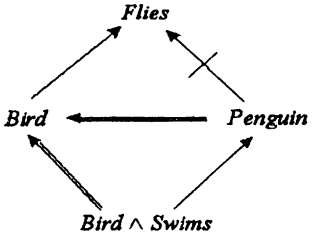
En la siguiente sección ilustraremos con el ejemplo utilizado por Brewka en [2] cómo estas modificaciones generan distintas extensiones que SDL.

4 Ejemplo

Primero veamos cuales son las extensiones que genera SDL. El ejemplo en [2] es el siguiente:

$$\begin{aligned} D &= \{Bird \rightarrow Flies, Penguin \rightarrow \neg Flies, Bird \wedge Swims \rightarrow Penguin\} \\ C &= \{Bird, Swims\} \\ T &= \{Peng \supset Bird\} \end{aligned}$$

que corresponde al siguiente diagrama:



Según la Definición 5 el ordenamiento $<_\Delta$ de este ejemplo consiste solo de $\{Penguin \rightarrow \neg Flies <_\Delta Bird \rightarrow Flies\}$ pues $D' = \{Bird \rightarrow Flies, Penguin \rightarrow \neg Flies\}$ es el único subconjunto de defaults p-conflictivo minimal (D es p-conflictivo pero no minimal). Y $T \cup \{Penguin\} \vdash_{D'} false$, y $T \cup \{Bird\} \nvdash_{D'} false$.

Las posibles (PDL) extensiones son las siguientes:

- Si $Penguin \rightarrow \neg Flies \ll Bird \wedge Swims \rightarrow Penguin \ll Bird \rightarrow Flies$ entonces

i	E_i	default $\ll$-minimal que está activo en E_i
0	$\{Bird, Swims\}$	$Bird \wedge Swims \rightarrow Penguin$
1	$\{Bird, Swims, Penguin\}$	$Penguin \rightarrow \neg Flies$
2	$\{Bird, Swims, Penguin, \neg Flies\}$	ninguno

$E = \{Bird, Swims, Penguin, \neg \mathbf{Flies}\}$

- Si $Bird \wedge Swims \rightarrow Penguin \ll Penguin \rightarrow \neg Flies \ll Bird \rightarrow Flies$ entonces

i	E_i	default $\ll$-minimal que está activo en E_i
0	$\{Bird, Swims\}$	$Bird \wedge Swims \rightarrow Penguin$
1	$\{Bird, Swims, Penguin\}$	$Penguin \rightarrow \neg Flies$
2	$\{Bird, Swims, Penguin, \neg Flies\}$	ninguno

$E = \{Bird, Swims, Penguin, \neg \mathbf{Flies}\}$

- Si $Penguin \rightarrow \neg Flies \ll Bird \rightarrow Flies \ll Bird \wedge Swims \rightarrow Penguin$ entonces

i	E_i	default $\ll$-minimal que está activo en E_i
0	$\{Bird, Swims\}$	$Bird \rightarrow Flies$
1	$\{Bird, Swims, Flies\}$	$Bird \wedge Swims \rightarrow Penguin$
2	$\{Bird, Swims, Flies, Penguin\}$	ninguno

$E = \{Bird, Swims, \mathbf{Flies}, Penguin\}$

A continuación mostramos como funciona SDL modificado.

Según la Definición 5' el ordenamiento $<_\Delta$ de este ejemplo consiste de $\{Penguin \rightarrow \neg Flies <_\Delta Bird \rightarrow Flies, Bird \wedge Swims \rightarrow Penguin <_\Delta Bird \rightarrow Flies\}$ pues $D' = \{Bird \rightarrow Flies, Penguin \rightarrow \neg Flies\}$ es p-conflictivo minimal por $Penguin \rightarrow \neg Flies$, y $T \cup \{Bird\} \nvdash_{D'} false$. Y D es p-conflictivo minimal por $Bird \wedge Swims \rightarrow Penguin$, y $T \cup \{Bird\} \nvdash_D false$.

Las posibles (PDL) extensiones son las siguientes:

- Si $Penguin \rightarrow \neg Flies \ll Bird \wedge Swims \rightarrow Penguin \ll Bird \rightarrow Flies$ entonces

i	E_i	default $\ll$-minimal que está activo en E_i
0	$\{Bird, Swims\}$	$Bird \wedge Swims \rightarrow Penguin$
1	$\{Bird, Swims, Penguin\}$	$Penguin \rightarrow \neg Flies$
2	$\{Bird, Swims, Penguin, \neg Flies\}$	ninguno

$E = \{Bird, Swims, Penguin, \neg \mathbf{Flies}\}$

- Si $Bird \wedge Swims \rightarrow Penguin \ll Penguin \rightarrow \neg Flies \ll Bird \rightarrow Flies$ entonces

i	E_i	default $\ll$-minimal que está activo en E_i
0	$\{Bird, Swims\}$	$Bird \wedge Swims \rightarrow Penguin$
1	$\{Bird, Swims, Penguin\}$	$Penguin \rightarrow \neg Flies$
2	$\{Bird, Swims, Penguin, \neg Flies\}$	ninguno

$E = \{Bird, Swims, Penguin, \neg \mathbf{Flies}\}$

5 Consideraciones sobre la Implementación

Para la implementación se utilizó Prolog reduciendo el lenguaje lógico del sistema a reglas de Horn y aumentandolo con las reglas default. De esta manera se pudo utilizar la máquina de inferencia propia del lenguaje. A continuación se describe suscintamente la operación del programa que permite simular la disputa.

Este permite ingresar un `Claim` e indicar las jugadas, tanto del proponente como del oponente. Cada vez que se completan las movidas de una jugada, el sistema verifica si el `Claim` es consecuencia de la teoría `Default` construída hasta el momento.

Al implementar SDL se utilizó la máquina de inferencia de Prolog para verificar inconsistencias en conjuntos de reglas y así obtener las (PDL) extensiones de las teorías Defaults. Traducimos las definiciones a predicados de Prolog que verifican si un `Claim` es inferido por las teorías Default con especificidad. El siguiente predicado muestra los distintos pasos para obtener el ordenamiento parcial de las reglas Default y para verificar si el `Claim` está en todas las extensiones.

```
consultarespec(Defaults,Claim,HechosContingentes):-
        conjuntodepartes(Defaults,SubconjDefault),
        pconflictivos(SubconjDefault,SubconjConflict),
        minimales(SubconjConflict,SubconjMinimales),
        crearorden(SubconjMinimales,Defaults,OrdenParcial),
        carg(HechosContingentes),
        bagof(X,permutar(Defaults,X),Permutaciones),
        contieneOrdenParcial(Permutaciones,PermOrden,OrdenParcial),
        probarTodasExt(PermOrden,Claim).
```

El primer predicado **conjuntodepartes** instancia la variable **SubconjDefault** con una lista de todos los subconjuntos de reglas defaults posibles.

El predicado **pconflictivos** selecciona entre los subconjuntos aquellos que son p-conflictivos, implementando la definición 4, luego **minimales** selecciona de estos últimos los subconjuntos que no contienen subconjuntos p-conflictivos.

Para generar el orden parcial entre las reglas default el predicado **crearorden** genera un conjunto de pares de reglas definiendo ese orden, utilizando las reglas Default y los subconjuntos minimales, implementando la definición 5.

El predicado **contieneOrdenParcial** selecciona entre todas las permutaciones de reglas default solo aquellas que contiene el orden parcial obtenido.

Por último **probarTodasExt** genera las extensiones de la teoría correspondientes a cada permutación y verifica que el Claim pertenezca a cada una de ellas, es decir si es una conclusión escéptica de la misma.

6 Conclusiones

Nuestra implementación de la reconstrucción de Brewka de la teoría de la disputación d Rescher permitió detectar algunos detalles que no permitían interpretar correctamente una cierta clase de ejemplos. Estos detalles fueron corregidos y las nuevas definiciones permiten que el nuevo sistema funcione correctamente. El uso de las características propias de Prolog permitió que el sistema pudiera ser implementado con claridad favoreciendo la prototipación de varias posibles soluciones a los problemas encontrados.

7 Referencias

[1] Brewka, Gerhard. *Adding Priorities and Specificity to Default Logic*, GMD technical report.

[2] Brewka, Gerhard. *A Reconstruction of Rescher's Theory of Formal Disputation Based on Default Logic*, GMD technical report.]

[3] Gordon, T. *The Pleadings Game: An Artificial Intelligence Model of Procedural Justice,* Ph. D. Dissertation, T. U. Darmstadt, 1993.

[4] Reiter, Raymond. *A Logic for Default Reasoning,* Artificial Intelligence **13** (1980) pp 81-132.

[5] Rescher, Nicholas. *Dialectics: A Controversy-Oriented Approach to the Theory of Knowledge*, State University of New York Press, Albany, 1977.