Modelado de Decisiones Sesgadas en Answer Set Programming

Miguel Alonso Peinado Portillo

Licenciatura en Matemáticas
Universidad Autónoma de Ciudad Juárez
al120547@alumnos.uacj.mx

Abstract. Decision making is an issue whose heyday has grown since the twentieth century with the beginning of the study of artificial intelligence. Among the topics of interest of the called *decision making*, we have the modeling of preferences, which is the main topic of this work. We give the basics of the Answer Set Programming theory (ASP), followed by a characterization of the ODLP programs and possible extensions of language, to end with a validation process that shows its usefulness to model problems where interfere options with some kind of preference on them.

Resumen. La toma de decisiones es un tema cuyo auge ha crecido desde el siglo XX con el comienzo del estudio de la inteligencia artificial. Entre los temas de interés del llamado *decision making*, tenemos el modelado de preferencias, el cual es el tema principal de este trabajo. Se darán nociones básicas de la teoría de *Answer Set Programming* (ASP), seguido de una caracterización de los programas ODLP y sus posibles extensiones de lenguaje, para terminar con un proceso de validación que muestre su utilidad para modelar problemas donde interfieren opciones con algún tipo de preferencia sobre ellas.

Keywords. Answer Set Programming, Decisiones Sesgadas, Lógica, Preferencias.

1 Introducción

En la vida diaria existen un gran número de situaciones donde es necesario elegir entre distintas opciones que se nos presentan. Por lo general, estas opciones vienen acompañadas con cierta jerarquía entre ellas que son resultado de las preferencias de cada persona. Tomemos como ejemplo el menú de un restaurante. Aquí existen varias opciones de platillos que serán elegidos de acuerdo a los gustos de cada persona. Así, supongamos que en el menú existen estos platillos de comida: {carne, pollo, pescado}; además, tiene estas dos opciones como bebida: {vino tinto, vino blanco}.

Si un cliente llega y nos dice que prefiere el pescado a la carne y la carne al pollo, añadido a que en caso de carne blanca entonces prefiere vino blanco y que prefiere vino tinto en caso de comer carne. La dificultad de estas preferencias se compli-

ca entre más alternativas se tenga en el menú. Si bien una persona realiza el proceso de elección sin razonarlo de esta forma, al momento de automatizarlo resultaría muy complejo sin el marco teórico que se presenta en este trabajo.

Aquí surgen varias preguntas al respecto. ¿Cómo podemos modelar este tipo de situación? ¿Qué pasaría si al llegar al restaurante nos dicen que se les acabo el pescado? ¿Cuál menú es el que mejor satisface las preferencias del cliente en esta situación? Aunque este ejemplo puede parecer sencillo de resolver, existen problemas de asignación de horarios en escuelas o de empresas que utilizan estas mismas ideas para resolverse y que requieren ser automatizados debido al número de reglas y variables que pueden llegar a tener.

Como veremos en este trabajo, el modelado en Answer Sets resulta útil para representar este tipo de problemas, con el empleo de proposiciones lógicas con disyunción ordenada, que serán definidos en las secciones siguientes.

Por otro lado, cabe mencionar que para el cálculo automático de los Answer Sets se ha utilizado un solver llamado DLV en línea, el cual está disponible en http://logic-lab.sourceforge.net/dlv.html. Además, todas las traducciones de los artículos de la lengua inglesa fueron realizadas por mi persona.

2 Justificación

Es necesario caracterizar la forma de modelar problemas que ayuden a tomar decisiones y encontrar la manera más adecuada para modelar estas situaciones. La utilización de Answer Sets se ha incrementado en los últimos años con trabajos tan importantes y demandados sobre teoría de juegos (De Clercq, Bauters, Schockaert, De Cock, & Nowé, 2014) o en la asignación de horarios en una escuela (Banbara, Takehide, & Tamura, 2013), por lo que se ha sido la semántica elegida para representar sesgos y restricciones (Niemela, Syrjanen, & Brewka, 2002).

El conocer mejor y el caracterizar el marco teórico nos brindará beneficios importantes no sólo en la lógica clásica per se, sino también repercute en las aplicaciones prácticas que se derivan. Tal es el caso de los agentes inteligentes, diagnósticos y teoría de juegos, por mencionar sólo algunos.

3 Objetivo

Caracterizar inferencia bajo sesgos y restricciones en Answer Sets Programming para determinar sus alcances y limitaciones.

4 Modelado en Answer Sets:

En esta sección daremos una breve introducción sobre semántica de Answer Sets, ya que es la herramienta que utilizaremos para el modelado de problemas con preferencias. Para mayor detalle, puede consultar (Gelfond & Lifschitz, Classical Negation in Logic Programs and Disjunctive Database, 1991).

Los programas lógicos tradicionales trabajan bajo el principio de "la suposición del mundo cerrado" (Gelfond & Lifschitz, Classical Negation in Logic Programs and Disjunctive Database, 1991), el cual nos dice que cuando no hay información de que una proposición es verdadera, ésta es tomada como falsa. Esto se debe a que estos programas funcionan con la llamada negación por falla, lo que permite que representen situaciones donde hay información incompleta. La utilización de esta negación es con el fin de concluir si una proposición es verdadera o no lo es, pero en el caso de que no lo sea, eso no implica que la proposición sea falsa, sino que no hay suficiente evidencia para definir su veracidad.

Esta idea lleva a la necesidad de distinguir entre dos tipos de negaciones: la negación por falla (también llamada negación débil, denotada aquí por "¬") que nos dice que no hay evidencia de que un átomo sea verdadero; y la negación clásica, o negación fuerte, que denotaremos aquí por "∼".

Definamos ahora el lenguaje de Answer Set Programming, ya que es el que nos permite trabajar en con los programas lógicos de nuestro interés.

Definición 1 (Lenguaje ASP para programas lógicos, $\mathcal{L}_{ASP}$) En los siguiente, $\mathcal{L}_{ASP}$ es un lenguaje de lógica proposicional con símbolos proposicionales: $a_0,\ a_1, \ldots$; conectivos: "∧" (conjunción); disyunción, denotada por "∨"; "←" (derivación, también denotada como "→"); constantes proposicionales "⊤" (tautología); "⊥" (contradicción); "¬" (negación por falla o negación débil, también denotada por la palabra not; "∼" (negación fuerte, equivalentemente denotada como "−"); símbolos auxiliares "(", ")" (paréntesis). Los símbolos proposicionales son llamados átomos o proposiciones atómicas. Una literal es un átomo o un átomo negado fuertemente. Una regla es un par ordenado $Head(r) \leftarrow Body(r)$.

Ahora podemos definir lo que es programa lógico extendido, que son los que nos permiten definir la semántica en **Answer** Sets para representar problemas donde se utilice la negación fuerte.

Definición 2 Un programa lógico extendido es un conjunto de reglas de la forma

$$r: \quad L_0 \leftarrow L_1, \ldots, L_m, \neg L_{m+1}, \ldots, \neg L_n,$$

donde cada L_i con $1 \leq i \leq n$ es una literal.

Para poder definir un Answer Set, necesitamos primero un par de conceptos. Llamamos un programa lógico básico a un programa extendido cuyas reglas no contienen negación por falla. Formalmente, decimos que un conjunto de literales X de un programa básico Π es cerrado bajo Π si para cada $r \in \Pi$, si se cumple que $\{L_1, \ldots, L_m\} \subset X$, entonces $L_0 \in X$. Al conjunto mínimo de átomos que es cerrado bajo Π lo denotaremos por $Cn(\Pi)$ y constituye el Answer Set de Π.

Para el caso general en donde Π es un programa extendido, definamos Π^X como el conjunto que se obtiene al eliminar:

1. cada regla que tenga una fórmula de la forma ¬L en Body(r) para alguna $L \in X$.
2. Todas las fórmulas de la forma $\neg L$ de $Body(r)$ de las reglas restantes.

Como se puede observar, el programa Π^X que se obtiene es un programa básico, por lo que ya está definido su answer set. Por tanto, si ambos conjuntos coinciden, entonces decimos que X es un answer set de Π. Es decir, el answer set de Π se caracteriza por $Cn(\Pi^X) = X$. Diremos que un answer set es consistente si no contiene un átomo y su negación fuerte a la vez.

En palabras informales, podemos decir que un answer set de Π es un conjunto de literales tales que todos sus elementos son verdaderos el programa Π. Es decir, son todas esas literales que son conclusiones del programa. Veamos el siguiente ejemplo ilustrativo:

Ejemplo 1 Sea Π_1 el programa definido por las siguientes reglas:

$$p \leftarrow p$$
$$q \leftarrow \neg p$$

De acuerdo con la definición de Answer Set, concluimos que el único que existe para Π_1 es {q}.

5 Programas Lógicos con Disyunción Ordenada

En (Brewka, Benferhat, & Le Berre, Qualitative Choice Logic, 2002), se introduce un nuevo conectivo lógico llamado disyunción ordenada, denotada por "$\overrightarrow{\times}$". Intuitivamente, A $\overrightarrow{\times}$ B significa: si es posible A; si A no es posible, entonces al menos B. Este conectivo nos permite modelar problemas donde intervienen varias opciones a elegir, pero además existe una cierta preferencia entre las distintas opciones.

Con la introducción de este conectivo, se puede definir un nuevo tipo de programa lógico, que es una generalización de un programa extendido. Las siguientes ocho definiciones fueron obtenidas de (Brewka, Logic Programming with Ordered Disjunction, 2002).

Definición 3 Un programa lógico con disyunción ordenada (ODLP por sus siglas en inglés) es un conjunto de reglas de la forma

$$r: \quad C_1 \overrightarrow{\times} \dots \overrightarrow{\times} C_n \leftarrow A_1, \dots, A_m, \neg B_1, \dots, \neg B_k,$$

donde cada $C_i\ (1 \le i \le n), A_j\ (1 \le j \le m), B_l (1 \le l \le k)$, son literales. Note que un programa lógico extendido es un caso particular donde $n = 1$.

Ejemplo 2 Sea R el programa con disyunción ordenada que se compone de las siguientes reglas:

$$r_1: \quad A \overrightarrow{\times} B \leftarrow \neg C.$$
$$r_2: \quad B \overrightarrow{\times} C \leftarrow \neg D.$$

La definición de Answer Set para un ODLP es distinta a la ya definida para un programa extendido. Ésta se define por medio de programas separados.

Definición 4 Sea r: $C_1 \overrightarrow{\times} \dots \overrightarrow{\times} C_n \leftarrow$ Body una regla. Para $k \le n$, definamos la k-ésima opción de r como:

$$r^k: \quad C_k \leftarrow \text{Body} \cup \{\neg C_1, \dots, \neg C_{k-1}\}.$$

Definición 5 Sea P un ODLP. Decimos que P' es un programa separado de P si se obtiene al remplazar cada regla de P por una de sus opciones.

Para ilustrar mejor estas dos definiciones, continuemos con nuestro Ejemplo 2. Como cada regla de R sólo tiene dos opciones, entonces se obtienen cuatro programas separados.

$A \leftarrow \neg C$ $\qquad$ $A \leftarrow \neg C$

$B \leftarrow \neg D$ $\qquad$ $C \leftarrow \neg D, \neg B$

$B \leftarrow \neg C, \neg A$ $\qquad$ $B \leftarrow \neg C, \neg A$

$B \leftarrow \neg D$ $\qquad$ $C \leftarrow \neg D, \neg B$

Como se puede observar, los programas separados no contienen disyunción ordenada, por tanto, para estos programas sin disyunción ya hemos definido sus answer sets. Esto nos lleva a la siguiente definición.

Definición 6 Sea P un ODLP. Un conjunto de literales X es un answer set de P si es un answer set consistente de un programa separado de P.

Nuevamente del Ejemplo 2, tenemos tres answer sets de P resultantes de los programas separados: $\{A, B\}, \{B\}, \{C\}$. La primera pregunta que nos podemos hacer es sobre cuál de estas tres opciones es preferible en el programa. Por lo tanto, es necesario definir un criterio para establecer qué answer set satisface mejor cada regla del programa.

Definición 7 Sea S un answer set de un ODLP P. Decimos que S satisface la regla r (denotado por $\deg_s(r)$):

$$C_1 \overrightarrow{\times} \dots \overrightarrow{\times} C_n \leftarrow A_1, \dots, A_m, \neg B_n, \dots, \neg B_k.$$

Con grado 1 si $A_j \notin S$ para algún j o $B_i \in S$ para algún i.

Con grado j, $1 \leq j \leq n$, si todos los $A_l \in S$, ningún $B_i \in S$ y $j = \min\{r \mid C_r \in S\}$.

Con esta definición, podemos ver que si llamamos $S_1 = \{A, B\}, S_2 = \{B\}, S_3 = \{C\}$ a los answer set del Ejemplo 2, tenemos que:

$\deg_{S_1}(r_1) = 1$	$\deg_{S_2}(r_1) = 2$	$\deg_{S_3}(r_1) = 1$
$\deg_{S_1}(r_2) = 1$	$\deg_{S_2}(r_2) = 1$	$\deg_{S_3}(r_2) = 2$

A partir de estos grados de satisfacción, se pueden definir ciertos criterios para decidir cuál answer set es preferible sobre otro. En (Niemela, Syrjanen, & Brewka, 2002) se definen tres distintos criterios, los cuales se presentan a continuación.

Para las siguientes tres definiciones, consideremos P un ODLP y S_1, S_2 dos answer sets de P. Además, definamos $S^i(P)$ como el conjunto de reglas en P que se satisfacen por S con grado i. Así, continuando con el Ejemplo 2, tenemos que $S_1^1(R) = \{r_1, r_2\}$, $S_1^2(R) = \emptyset, S_2^1(R) = \{r_2\}, S_2^2(R) = \{r_1\}, S_3^1(R) = \{r_1\}, S_3^2(R) = \{r_2\}$.

Definición 8 Decimos que S_1 es preferido por cardinal a S_2 (denotado por $S_1 >_c S_2$) si y sólo si existe un grado i tal que $|S_1^i(P)| > |S_2^i(P)|$ y para toda j < i, $|S_1^j(P)| = |S_2^j(P)|$.

Es decir, este criterio compara la cantidad de reglas en los conjuntos de grados de satisfacción. Así, tenemos que en el Ejemplo 3, $S_1 >_c S_2$ y que $S_1 >_c S_3$ ya que el cardinal de $S_1^1(R)$ es mayor que el de cualquier otro conjunto de grado de satisfacción. Además, se cumple que S_2 y S_3 son igualmente preferidos por cardinal ya que $|S_2^k(R)| = |S_3^k(R)|$ para $k \in \{1,2\}$.

Definición 9 Decimos que S_1 es inclusión-preferido a S_2 (denotado por $S_1 >_i S_2$) si y sólo si existe una k tal que $S_2^k(P) \subset S_1^k(P)$ y para toda $j < k$ se cumple que $S_1^j(P) = S_2^j(P)$.

Este criterio no compara la cantidad de reglas de los conjuntos, sino compara la contención entre los conjuntos de grados de satisfacción. Nuevamente en el Ejemplo 3, tenemos que $S_2^1(R) \subset S_1^1(R)$ y que $S_3^1(R) \subset S_1^1(R)$ y por tanto $S_1 >_i S_2$ y $S_1 >_i S_3$. Tanto S_2 como S_3 son igualmente preferidos por inclusión ya que no es posible establecer la contención de conjuntos en esos casos.

Definición 10 Decimos que S_1 es Pareto-preferido a S_2 (denotado por $S_1 >_p S_2$) si y sólo si existe $r \in P$ tal que $\deg_{S_1}(r) < \deg_{S_2}(r)$ y para ninguna $r' \in P, \deg_{S_1}(r') > \deg_{S_2}(r')$.

Este criterio utiliza los grados de satisfacción de las reglas para encontrar una regla cuyo grado de satisfacción sea menor en uno de los answer set, pero que para ninguna otra regla el grado sea mayor. Finalmente, en el Ejemplo 3 se cumple que $S_1 >_p S_2$ y $S_1 >_p S_3$ ya que el grado de satisfacción de ambas reglas es 1 para S_1, mientras que en S_2 y S_3 alguna de las dos reglas tiene grado de satisfacción igual a 2.

La siguiente proposición nos habla sobre la relación que existe entre estos tres distintos criterios. Puede encontrarse en (Niemela, Syrjanen, & Brewka, 2002).

Proposición 1 Sean S_1 y S_2 answer sets de un ODLP P. Entonces $S_1 > p\ S_2$ implica $S_1 >_i S_2$ y además $S_1 >_i S_2$ implica $S_1 >_c S_2$.

Ejemplo 3 Supongamos que quiere adquirir un automóvil de agencia, el cual puede ser un carro, una camioneta o una pick-up. Además, supongamos que puede escoger entre transmisión automática y transmisión estándar. Aquí se presentan las preferencias entre las distintas opciones.

- r_1: carro $\overrightarrow{\times}$ camioneta $\overrightarrow{\times}$ pick-up.
- r_2: estándar $\overrightarrow{\times}$ automático.
- r_3: automático ← pick-up.

Donde la regla número tres nos dice que si nuestra elección es una pick-up, entonces la queremos automática.

Como r_1 tiene tres opciones y r_2 tiene dos, por tanto resultan seis programas separados.

carro estándar automático ← pick-up	camioneta ← ¬carro estándar automático ← pick-up	pick-up ← ¬carro, ¬camioneta estándar automático ← pick-up
carro automático ← ¬estandar automático ← pick-up	camioneta ← ¬carro automático ← ¬estandar automático ← pick-up	pick-up ← ¬carro, ¬camioneta automático ← ¬estandar automático ← pick-up

Utilizando un solver DLV, encontramos que existen seis distintos answer sets, que son los siguientes:

$S_1 = \{\text{carro, estándar}\}$; $S_2 = \{\text{camioneta, estándar}\}$

$S_3 = \{\text{carro, automático}\}$; $S_4 = \{\text{camioneta, automático}\}$

$S_5 = \{\text{pick-up, automático}\}$; $S_6 = \{\text{pick-up, estándar, automático}\}$

De acuerdo a la Definición 7, la siguiente tabla nos muestra el grado de satisfacción para cada regla según cada answer set.

	S_1	S_2	S_3	S_4	S_5	S_6
r_1	1	2	1	2	3	3
r_2	1	1	2	2	2	1
r_3	1	1	1	1	1	1

Ahora analicemos cuál answer set es el preferido, utilizando los tres criterios definidos anteriormente. Mostramos los conjuntos $S_j^i(P)$ con $1 \leq i, j \leq 5$.

- $S_1^1(P) = \{r_1, r_2, r_3\}$; $S_1^2(P) = \emptyset$; $S_1^3(P) = \emptyset$.
- $S_2^1(P) = \{r_2, r_3\}$; $S_2^2(P) = \{r_1\}$; $S_2^3(P) = \emptyset$.
- $S_3^1(P) = \{r_1, r_3\}$; $S_3^2(P) = \{r_2\}$; $S_3^3(P) = \emptyset$.
- $S_4^1(P) = \{r_3\}$; $S_4^2(P) = \{r_1, r_2\}$; $S_4^3(P) = \emptyset$.
- $S_5^1(P) = \{r_3\}$; $S_5^2(P) = \{r_2\}$; $S_5^3(P) = \{r_1\}$.
- $S_6^1(P) = \{r_2, r_3\}$; $S_6^2(P) = \emptyset$; $S_6^3(P) = \{r_1\}$.

Así, tenemos que $|S_1^1(P)| > |S_j^1(P)|$ para toda $2 \leq j \leq 5$. Esto nos indica que $S_1 >_c S_j$. Similarmente, tenemos que $S_j^1(P) \subset S_1^1(P)$, por lo tanto $S_1 >_i S_j$. Finalmente, tenemos que $\deg_{S_1}(r_1) < \deg_{S_j}(r_1)$ con $2 \leq j \leq 5$, y, además, se cumple que $\deg_{S_1}(r_k) \leq \deg_{S_j}(r_k)$ con $k = 2, 3$. Por lo tanto, se cumple el criterio de Pareto y así $S_1 >_p S_j$. De acuerdo a estos tres criterios, concluimos que S_1 es nuestro Answer Set preferente.

Ahora suponga que al llegar a la agencia, nos informan que no tienen disponibles carros estándar, esto se traduce a agregar la siguiente regla a nuestro programa:

$$r_4: \sim\text{estándar} \leftarrow \text{carro}.$$

Esto nos elimina S_1 y S_3 de la lista de answer sets y además nos agrega uno nuevo $S_7 = \{\text{carro, automático, } \sim\text{estándar}\}$. Al no ser S_1 un answer set del nuevo programa, S_2 y S_7 son los answer sets candidatos a ser preferentes, pero si analizamos los crite-

rios, ninguno de los tres aplica para distinguir cuál de los dos es preferible. La pregunta aquí es ¿Qué otro criterio puede utilizarse para elegir entre estos dos answer sets?

6 Resultados.

Como se puede observar en el Ejemplo 3, los tres criterios que presentamos en este trabajo resultan útiles en unos casos, pero existen otros en donde nos son suficientes, como al agregar la restricción r_4 al programa. Por tanto, es necesario buscar nuevos criterios que permitan aplicarse a situaciones donde intervienen preferencias y restricciones. A pesar de esto, hemos confirmado que la semántica en Answer Sets es una herramienta útil para modelar sesgos y restricciones.

7 Conclusión.

La solución de los problemas donde intervienen preferencias puede resultar difícil de encontrar sin las herramientas adecuadas. Como acabamos de ilustrar en el Ejemplo 3, el modelado en Answer Sets resulta una opción viable para la resolución de este tipo de problemas. Además, con esta misma teoría pueden abordarse problemas más complejos donde intervengan un número significativamente mayor de reglas y de opciones. Otra ventaja es el uso de solver como DLV o PSmodels, que permitan automatizar los cálculos necesarios. Por otro lado, es importante continuar la investigación para resolver las cuestiones donde los criterios de elección del answer set preferente lleguen a fallar.

Referencias:

Banbara, M., Takehide, S., & Tamura, N. (2013). Answer Set Programming as a Modeling Languaje for Course Timetabling. *Theory and Practice of Logic Programming*.

Brewka, G. (2002). Logic Programming with Ordered Disjunction. *American Association for Artificial Intelligent*, 6.

Brewka, G., Benferhat, S., & Le Berre, D. (2002). Qualitative Choice Logic. *Elsevier*, 203-237.

De Clercq, S., Bauters, K., Schockaert, S., De Cock, M., & Nowé, A. (2014). Using Answer Set Programming for Solving Boolean Games. *Association for the Advancement of Artificial Intelligence (AAAI)*.

Gelfond, M., & Lifschitz, V. (1988). The Stable Model Semantics for Logic Programming.

Gelfond, M., & Lifschitz, V. (1991). Classical Negation in Logic Programs and Disjunctive Database. *New Generation Computing*, 1-21.

Niemela, I., Syrjanen, T., & Brewka, G. (2002). Implementing Order Disjunction Using Answer Set Solver for Normal Programs. 13.

Schaub, T., Anger, C., & et.al. (s.f.). A Glimpse of Answer Set Programming.