Modelado de un problema de iluminación inteligente de una oficina usando un enfoque de actualización basado en Answer Set Programming

Mayra Cordero, Claudia Zepeda, José Luis Carballido

Benemérita Universidad Atónoma de Puebla,
Facultad de Ciencias de la Computación,
Puebla, Puebla, México

{maycorderor,czepedac,jlcarballido7}@gmail.com

Resumen. La actualización es una acción que nos permite renovar información, es decir, consiste en convertir información retrasada en información actual. Por otra parte, Answer Set Programming es un lenguaje de programación lógico y declarativo basado en la lógica. El propósito de este artículo es utilizar el enfoque de actualización de secuencias de programas lógicos basado en Answer Set Programming para modelar y resolver el problema de iluminación inteligente de una oficina. Este problema consiste en controlar la iluminación de una oficina donde el encendido y apagado de cuatro lámparas dependa de la ubicación de una persona.

Palabras clave: Actualización, programación lógica, Answer Set Programming.

1. Introducción

La programación lógica (LP) ofrece de manera natural representar conocimiento declarativo utilizando el lenguaje de la lógica matemática. La finalidad de un programa lógico es buscar todos los valores que hacen verdadero a un programa y como resultado de las combinaciones de dichos valores obtener una respuesta [9]. Para que un programa lógico tenga un significado y pueda ser interpretado por programas computacionales se le debe asignar una semántica, dicha semántica permite determinar el tipo de conclusiones que se pueden establecer a partir de un conjunto de reglas [9].

Debido a esto surgen lenguajes que permiten escribir programas lógicos tales como Answer Set Programming (ASP) [2,10]. ASP es un lenguaje de programación lógico y declarativo para la representación del conocimiento que nos permite manejar problemas con conocimiento por defecto y razonamiento no-monotóno mediante el concepto de negación como falla [14]. El éxito de ASP se debe en gran medida a su capacidad para resolver problemas [6]. Dentro de la gama de problemas que pueden enfrentarse con ASP se encuentran problemas de planeación, modelado de agentes lógicos, actualización y aplicaciones de inteligencia artificial en general [7,3,8,13].

La contribución de este trabajo es modelar y resolver el problema de iluminación inteligente de una oficina utilizando el enfoque de actualización basado en secuencias de programas lógicos y ASP [14,12]. Nos interesamos en este problema ya que una de las causas principales del excesivo consumo energetico es el tiempo que permanecen encendidas las lámparas sin ningún beneficio [4].

Existen diferentes tipos de oficinas, en esta ocasión tomaremos como modelo una oficina conformada por cuatro lámparas.

El Modelo del sistema consiste en controlar el estado de encendido y apagado de las cuatro lámparas dentro de la oficina usando cuatro sensores de presencia. Para este modelo proponemos que a dicha oficina solo tenga acceso una sola persona. Como las actividades de la persona dentro de la oficina no son estáticas el estado de encendido y apagado de las lámparas debe mantenerse actualizando para cambiar de un estado a otro según los detectores de presencia.

En la sección 2 llamada Marco teórico describimos los conceptos que se consideran importantes para resolver el problema de iluminación inteligente de oficinas. La sección 3 corresponde a la descripción de problema, el modelado del problema, la aplicación del enfoque de actualización, discusión de resultados y finalmente la sección 4 corresponde a las conclusiones.

2. Marco teórico

En esta sección presentamos un panorama general de las ideas y conceptos principales tratados para el desarrollo del problema de iluminación inteligente de una oficina. Incluimos los conceptos principales como Programas Lógicos y ASP, y Enfoque de Actualización.

2.1. Programas lógicos y ASP

Entendemos a los programas lógicos como teorías proposicionales. Una teoría proposicional es sólo un conjunto finito de fórmulas bien formadas. Se le puede llamar teoría o programa en el que no se presenta ninguna ambigüedad. Utilizamos el lenguaje de la lógica proposicional de la forma habitual para el modelado del problema de iluminación inteligente de una oficina. Consideramos símbolos proposicionales: $m, n, p, q, r, s, t, ...,$ conectivos proposicionales: $\wedge, \vee, \leftarrow, \bot, -, \neg$ y símbolos auxiliares: '(',')','.'. También consideramos los dos tipos de negación: negación fuerte o clásica escrita como $(-)$ y negación débil o fracaso escrita como $(\neg)$. Asumimos que para cualquier fórmula proposicional f, $\neg f$ es una abreviatura de $f \leftarrow \bot$. Un átomo es un símbolo proposicional. Un literal es un átomo a o la negación de un átomo $-a$. Dado un conjunto de átomos A, definimos $-A = \{-l \,|\, l \in A\}, \neg A = \{\neg l \,|\, l \in A\}$, y $Lit_A = A \cup -A$. La signatura de un programa lógico P denotada como $\mathcal{L}_P$, es el conjunto de átomos que se producen en P. Por lo general, la sintaxis de fórmulas bien formadas dentro de los programas lógicos se ha limitado a la forma $f \leftarrow g$ que es sólo otra manera de escribir $g \rightarrow f$. Las fórmulas bien formadas $f \leftarrow g$ se llaman reglas donde f

se llama la cabeza y g el cuerpo de la regla. Una regla con un cuerpo vacío, $f \leftarrow$, se llama un hecho, y una regla sin cabeza $\bot \leftarrow g$ se llama restricción. Hechos y restricciones también se denominan f y $\leftarrow g$ respectivamente. Queremos hacer hincapié en el hecho de que en el enfoque de actualización, un programa lógico se interpreta como una teoría proposicional. Como es habitual en ASP, utilizaremos programas proposicionales. En este trabajo utilizamos la semántica de answer sets definida en términos de la llamada reducción Gelfond-Lifschitz [5]. Vale la pena mencionar que en el enfoque de actualización un átomo y su contraparte negada nunca pueden ocurrir en el mismo answer set. Por último, se dice que B es un subconjunto de A (escrito como $B \subseteq A$) si y sólo si todos los miembros de B son miembros de A, y decimos que B es un subconjunto propio de A (escrito como $B \subset A$) si y sólo si $B \subseteq A$ y $B \neq A$.

2.2. Mínimos generalizados asnwer sets

En esta sección recordamos la sintaxis y la semántica de los programas lógicos abductivos como se presenta en [11].

Definición 2.21 (Programa Abductivo Lógico) *[1] Un* programa abductivo lógico *es un par $\langle P, A\rangle$ donde P es un programa arbitrario y A es un conjunto de literales, llamados* abductivos.

En [1], la noción de minimo generalizdo answer set se utiliza para definir la semántica de un programa lógico abductivo.

Definición 2.22 (Generalizados Answer Sets) *[1] $\langle M, \Delta\rangle$ es un generalizado answer set de un programa abductivo $\langle P, A\rangle$ si y sólo si $\Delta \subseteq A$ y M es un answer set de $P \cup \Delta$.*

En [1] también se presenta un orden entre generalizados answer sets a fin de obtener los mínimos generalizados answer sets de un programa lgico abductivo. Veremos que un mínimo generalizado answer set es un par $\langle M, \Delta\rangle$, nosotros sólo estamos interesados en M .

Definición 2.23 (Orden Abductivo de Inclusión) *[1] Podemos establecer un orden entre generalizados answer sets de la siguiente manera: Sea $\langle M_1, \Delta_1\rangle$ y $\langle M_2, \Delta_2\rangle$ generalizados answer sets de $\langle P, A\rangle$, definimos $\langle M_1, \Delta_1\rangle < \langle M_2, \Delta_2\rangle$ si $\Delta_1 \subset \Delta_2$.*

Definición 2.24 (Mínimos Generalizados Answer Sets) *[1] $\langle M, \Delta\rangle$ es un mínimo generalizado answer set de $\langle P, A\rangle$, si y sólo si $\langle M, \Delta\rangle$ es un generalizado answer set de $\langle P, A\rangle$ y es el mínimo abductivo por orden de inclusión.*

2.3. Mínimos extendidos generalizados Answer Sets

La semántica para actualizaciones que consiste en una secuencia de programas se basa en una extensión de la definición de mínimos generalizados answer

sets. La extensión se llama mínimos extendidos generalizados answer sets (MEG) [11]. También el enfoque de actualización utiliza la definición de un programa extendido abductivo lógico (EAL). Esta definición es similar a la definición 1, pero se agrega una función sobreyectiva. De un programa EAL obtenemos sus extendidos explícitos generalizados answer sets (EEG). Utilizamos la función sobreyectiva de un programa EAL para definir un orden entre sus EEG answer sets y obtener sus MEG answer sets [11].

Definición 2.31 (Programa Extendido Abductivo Lógico (EAL)) *[11] Un* programa extendido abductivo lógico (EAL) *es un triple $\langle P, A, f\rangle$ tal que P es un programa arbitrario, A es un conjunto de átomos, y f es alguna función sobreyectiva con dominio A y codominio $\{1, \ldots, N\}$, $N > 0$.*

***Ejemplo* 1** *Podemos definir un programa EAL $\langle P, A, f\rangle$ tal que P es el siguiente programa:*

$$\begin{aligned} a &\leftarrow \neg x_1^1. \\ b &\leftarrow a, \neg x_2^1. \\ -a &\leftarrow \neg x_1^2. \\ c &\leftarrow . \end{aligned}$$

A es el siguiente conjunto de átomos: $\{x_1^1, x_2^1, x_1^2\}$; y $f : A \to \{1, 2\}$ es la función tal que $f(x_j^i) = i$.

Definición 2.32 (Extendidos Generalizados answer sets (EG)) *[11] Sea $\langle P, A, f\rangle$ un programa EAL, M un conjunto de literales, y $\Delta \subseteq A$. Un* extendido generalizado (EG) *answer set de $\langle P, A, f\rangle$ es un par $\langle M, \Delta\rangle$ si M es un answer set de $P \cup \Delta$.*

***Ejemplo* 2** *Vamos a considerar el programa EAL $\langle P, A, f\rangle$ del Ejemplo 1. El cuadro 1 muestra los diferentes EG answer sets de $\langle P, A, f\rangle$ con su respectivo $\Delta \subseteq A$ (los subconjuntos que no aparecen en la tabla no tienen EG answer sets).*

Tabla 1. Los EG answer sets del programa $\langle P, A, f\rangle$ del Ejemplo 1.

Δ	$\langle M, \Delta\rangle$
$\{x_1^1, x_2^1\}$	$\langle\{x_1^1, x_2^1, -a, -b, c\}, \{x_1^1, x_2^1\}\rangle$
$\{x_1^1, x_2^1, x_1^2\}$	$\langle\{x_1^1, x_2^1, x_1^2, -a, c\}, \{x_1^1, x_2^1, x_1^2\}\rangle$
$\{x_1^1, x_2^1, x_2^2\}$	$\langle\{x_1^1, x_2^1, x_2^2, -b, c\}, \{x_1^1, x_2^1, x_2^2\}\rangle$
$\{x_1^1, x_2^1, x_1^2, x_2^2\}$	$\langle\{x_1^1, x_2^1, x_1^2, x_2^2, c\}, \{x_1^1, x_2^1, x_1^2, x_2^2\}\rangle$
$\{x_1^1, x_1^2, x_2^2\}$	$\langle\{x_1^1, x_1^2, x_2^2, c\}, \{x_1^1, x_1^2, x_2^2\}\rangle$
$\{x_2^1, x_1^2, x_2^2\}$	$\langle\{x_2^1, x_1^2, x_2^2, a, c\}, \{x_2^1, x_1^2, x_2^2\}\rangle$
$\{x_1^2, x_2^2\}$	$\langle\{x_1^2, x_2^2, a, b, c\}, \{x_1^2, x_2^2\}\rangle$

Ahora presentamos un orden entre los EG answer sets de un programa de EAL.

Definición 2.33 (Orden de inclusión entre los EG answer sets) *[11] Sea $\langle P, A, f\rangle$ un programa EAL donde el codominio de f es el conjunto $\{1, \ldots, N\}$, $N > 0$. Sea $A_i = \{a \in A | f(a) = i\}$. Establecemos una orden de inclusión entre EG answer sets de $\langle P, A, f\rangle$ como sigue: Sea $\langle M_1, \Delta_1\rangle$ y $\langle M_2, \Delta_2\rangle$ EG answer sets de $\langle P, A, f\rangle$. Definimos $\langle M_1, \Delta_1\rangle \leq_{inclu} \langle M_2, \Delta_2\rangle$ si y sólo si hay k, $1 \leq k \leq N$ tal que $(\Delta_1 \cap A_k) \subset (\Delta_2 \cap A_k)$, y para todo j, $k < j \leq N$, $(\Delta_1 \cap A_j) = (\Delta_2 \cap A_j)$.*

***Ejemplo* 3** *Vamos a considerar el programa EAL $\langle P, A, f\rangle$ del ejemplo 1 y sus EG answer sets del cuadro 1. Recordamos que $N = 2$ y $f : A \to \{1, 2\}$ es la función en la que $f(x^i_j) = i$. Podemos comprobar que $A_1 = \{x^1_1, x^1_2\}$ y $A_2 = \{x^2_1\}$. Así, de acuerdo al cuadro 1 y definición 2.33, podemos comprobar que*
$\langle\{x^1_2, x^2_1, x^2_2, a, c\}, \{x^1_2, x^2_1, x^2_2\}\rangle \leq_{inclu} \langle\{x^1_1, x^1_2, x^2_1, x^2_2, c\}, \{x^1_1, x^1_2, x^2_1, x^2_2\}\rangle$; ya que $k = 1$ tal que $(\{x^1_2, x^2_1, x^2_2\} \cap A_1) \subset (\{x^1_1, x^1_2, x^2_1, x^2_2\} \cap A_1)$, i.e, $\{x^1_2\} \subset \{x^1_1, x^1_2\}$, y para todo j, $1 < j \leq 2$ tenemos que
$(\{x^1_2, x^2_1, x^2_2\} \cap A_2) = (\{x^1_1, x^1_2, x^2_1, x^2_2\} \cap A_2)$, i.e, $\{x^2_1, x^2_2\} = \{x^2_1, x^2_2\}$.
De manera similar , podemos comprobar que
$\langle\{x^1_1, x^1_2, -a, -b, c\}, \{x^1_1, x^1_2\}\rangle \leq_{inclu} \langle\{x^1_1, x^1_2, x^2_1, -a, c\}, \{x^1_1, x^1_2, x^2_1\}\rangle$, ya que $K = 2$ de tal manera que $(\{x^1_1, x^1_2\} \cap A_2) \subset (\{x^1_1, x^1_2, x^2_1\} \cap A_2)$, i.e., $\emptyset \subset \{x^2_1\}$, y no hay j, $2 < j \leq 2$.

Notemos que también podemos definir un orden de cardinalidad entre los EG answer sets de un programa EAL $\langle P, A, f\rangle$, denotado por $\leq_{card}$. Esta definición puede obtenerse a partir de la Definición 2.33 reemplazando el conjunto de criterio de inclusión por conjunto de criterios de cardinalidad. En el resto de este trabajo se utilizará sólo orden de inclusión entre los EG answer sets y vamos a escribir $\leq$ para denotar este orden. También vale la pena mencionar que en Definicón 2.33 el programa de P se utiliza para obtener los EG answer sets, si bien no es utilizado para definir el orden entre los EG answer sets. El orden entre EG answer sets se define en términos de los subconjuntos de A (Δ_i and A_j). Por otra parte, este orden se puede utilizar para obtener los mínimos extendidos generalizados answer sets de un programa EAL $\langle P, A, f\rangle$. Veremos que un mínimo extendido generalizado answer set es un par $\langle M, \Delta\rangle$, por esta ocasión, sólo estamos interesados en M.

Definición 2.34 (Mínimos Extendidos Generalizados answer sets (MEG)) *[11] Sea $\langle P, A, f\rangle$ un programa EAL. $\langle M, \Delta\rangle$ es un* mínimo extendido generalizado answer set (MEG) *de $\langle P, A, f\rangle$ si $\langle M, \Delta\rangle$ es un EG answer set de P y no hay EG answer set $\langle M', \Delta'\rangle$ de $\langle P, A, f\rangle$ tal que $\langle M', \Delta'\rangle \leq \langle M, \Delta\rangle$.*

***Ejemplo* 4** *Vamos a considerar el programa EAL $\langle P, A, f\rangle$ del Ejemplo 1. De acuerdo con el cuadro 1 y la definición 2.34, podemos comprobar que*
$\langle\{x^1_1, x^1_2, -a, -b, c\}, \{x^1_1, x^1_2\}\rangle$ es el único MEG answer set de $\langle P, A, f\rangle$ ya que no hay EG answer set $\langle M', \Delta'\rangle$ de $\langle P, A, f\rangle$ tal que
$\langle M', \Delta'\rangle \leq \langle\{x^1_1, x^1_2, -a, -b, c\}, \{x^1_1, x^1_2\}\rangle$.

2.4. Enfoque de actualización utilizando MEG answer sets

Consideramos las secuencias de programas de actualización [11] . Formalmente una secuencia de programas de actualización, entendemos una secuencia $(P_1, \ldots, P_n)$ de programas lógicos donde N_{P_i} es el número de reglas de cada programa lógico. Decimos que $\mathbf{P}$ is una secuencia de actualización de programas sobre $\mathcal{L}_{\mathbf{P}}$ si y solo si $\mathcal{L}_{\mathbf{P}}$ representa un conjunto de átomos que aparecen en $\bigcup_{1 \leq i \leq n} P_i$ [9].

Definición 2.41 (Programa de actualización de una secuencia de (LP)) *[11] Dada una secuencia de actualización de programas $\boldsymbol{P} = (P_1, \ldots, P_n)$ sobre $\mathcal{L}_{\boldsymbol{P}}$, definimos una* secuencia de programas de actualización *$\boldsymbol{P}_{\oslash} = P_1 \oslash \cdots \oslash P_n$ sobre $\mathcal{L}^*_{\boldsymbol{P}}$ (que se extiende $\mathcal{L}_{\boldsymbol{P}}$ por nuevos átomos abducibles) constituidos por los siguientes elementos:*

1. *Todas las restricciones en $P_1, \ldots, P_{n-1}$,*
2. *Para cada $r_j \in P_i, 1 \leq i \leq n-1$, $1 \leq j \leq N_{P_i}$ añadimos la regla $r_j \leftarrow \neg b^i_j$, donde b^i_j es un nuevo átomo abducible,*
3. *Todas las reglas y restricciones $r \in P_n$.*

Definimos el programa EAL de $\mathbf{P}$ como triple $\langle \mathbf{P}_{\oslash}, B, f \rangle$ donde B es el conjunto de nuevos átomos abducibles de $\mathbf{P}_{\oslash}$, i.e., $B = \{b^i_j \mid b^i_j \in \mathbf{P}_{\oslash}, 1 \leq i \leq n-1, 1 \leq j \leq N_{P_i}\}$; y $f : B \to \{1, \ldots, n-1\}$ es la función sobreyectiva donde $f(b^i_j = i)$.

La semántica de nuestro operador de actualización se basa en los MEG answer sets de un programa EAL de una secuencia de actualización determinada.

Definición 2.42 *Sea $\langle \boldsymbol{P}_{\oslash}, B, f \rangle$ el programa EAL de una secuencia de actualización $\boldsymbol{P}$ sobre $\mathcal{L}_{\boldsymbol{P}}$. M es un $\oslash$-actualización answer set de $\boldsymbol{P}$, si M' es un MEG answer set de $\langle \boldsymbol{P}_{\oslash}, B, f \rangle$ y $M = M' \cap \mathcal{L}_{\boldsymbol{P}}$.*

***Ejemplo* 5** *Sea $\boldsymbol{P} = (P_1, P_2, P_3)$ una secuencia de actualización a través de $\mathcal{L}_{\boldsymbol{P}} = \{a, b, c\}$ donde: P_1 es el siguiente programa lógico,*

$$a \leftarrow .$$
$$b \leftarrow a.$$

P_2 es el siguiente programa lógico,

$$-a \leftarrow .$$
$$-b \leftarrow .$$

and P_3 es el siguiente programa lógico,

$$c \leftarrow .$$

Por lo tanto el programa de actualización $\boldsymbol{P}_{\oslash} = P_1 \oslash P_2 \oslash P_3$ sobre $\mathcal{L}^*_{\boldsymbol{P}}$ (que se extiende de $\mathcal{L}_{\boldsymbol{P}}$ por nuevos átomos abducibles $\{x^1_1, x^1_2, x^2_1, x^2_2\}$). Así el programa lógico de actualización $\boldsymbol{P}_{\oslash} = P_1 \oslash P_2 \oslash P_3$ es el siguiente programa lógico:

$$
\begin{aligned}
a &\leftarrow \neg x_1^1.\\
b &\leftarrow a, \neg x_2^1.\\
-a &\leftarrow \neg x_1^2.\\
-b &\leftarrow \neg x_2^2.\\
c &\leftarrow .
\end{aligned}
$$

El programa EAL de $\mathbf{P}$ es el triple $\langle P_{\oslash}, B, f\rangle$, donde $f : B \rightarrow \{1, 2\}$ es la función $f(b_j^i = i)$. Por lo tanto sabemos que su único MEG answer set es $\{x_1^1, -a, -b, c\}$. Por lo tanto, $\{-a, -b, c\}$ es el único $\oslash$-actualización answer set de $\mathbf{P}$.

3. Descripción del problema

Como ya mencionamos el problema consiste en modelar y resolver el problema de iluminación inteligente de una oficina. El problema está enfocado en una oficina conformada por cuatro lamparas l1, l2, l3 y l4 que se controlan por cuatro sensores de presencia s1, s2, s3 y s4. Los sensores s1,s2,s3 y s4 son los encargados de mantener el estado de encendido y apagado de las lamparas l1,l2,l3 y l4 respectivamente.

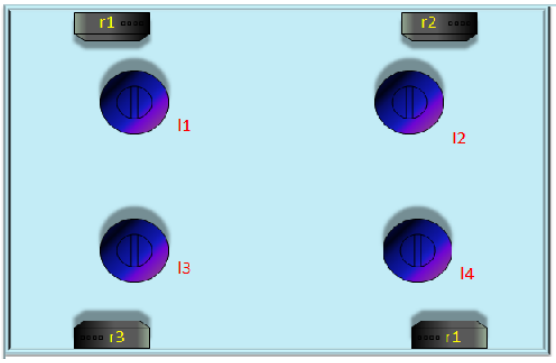

Fig. 1. Sistema de iluminación

Como se muestra en la figura 1 Los sensores s1 y s2 detectan presencia cuando una persona se encuentre en la oficina enfrente de lado derecho o izquierdo respectivamente y los sensores s3 y s4 detectan presencia cuando una persona se encuentre en la oficina atrás de lado derecho o izquierdo respectivamente. Los sensores son los encargados de actualizar el estado de las lámparas a encendido o apagado dependiendo si detectan personas o no. Si s1 detecta presencia el estado de l1 debe ser encendido, si s2 detecta presencia el estado de l2 debe ser encendido, si s3 detecta presencia el estado de l3 debe ser encendido y finalmente si s4 detecta presencia el estado de l4 debe ser encendido. Por otra parte si s1 no

detecta presencia el estado de l1 debe ser apagado, si s2 detecta no presencia el estado de l2 debe ser apagado, si s3 no detecta presencia el estado de l3 debe ser apagado y finalmente si s4 no detecta presencia el estado de l4 debe ser apagado.

3.1. Modelado del problema

Una ves que establecimos como pueden ser los estados de encendido y apagado de lámparas proponemos el siguiente escenario para ver el funcionamiento del sistema. El escenario consiste en suponer que inicialmente en un tiempo t1 no hay ninguna persona dentro de la oficina esto significa que ningún sensor ha detectado presencia y por lo tanto las cuatro lámparas se encuentran en un estado de apagado, posteriormente en un tiempo t2 ingresa una persona, obviamente esta persona se posiciona dentro de la oficina y en este caso suponemos que se posiciona atrás de lado izquierdo, por lo tanto el sensor encargado de esa posición es s3 y en un tiempo t3 eperamo que el sistema actualice el estado de los sensores y de las lámparas para poder encender l3. De esta manera el estado final que esperamos de nuestro sistema es que l3 se encuentre en un estado de encendido y l1,l2 y l4 se encuentren en une estado de apagado. A continuación presentamos tres diagramas que representan graficamente el escenario:

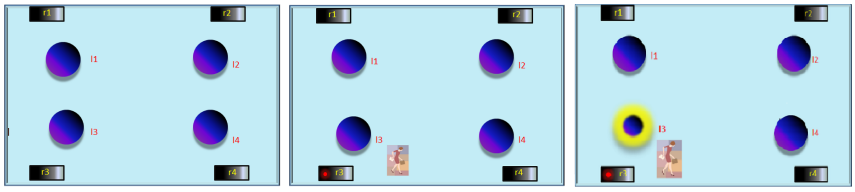

Fig. 2. Escenario en lo diferentes tiempos

Para modelar nuestro problema representamos la información por medio de programas lógicos, nosotros para este problema utilizamos tres programas lógicos P_1, P_2 y P_3 pero el enfoque de actualización permite utilizar los n-programas lógicos que se necesiten.

En esto tres programas lógicos representamos el comportamiento y las acciones que debe realizar nuestro sistema. Para representar el estado de encendido de las lámparas utilizamos el valor 1, de igual forma para representar si un sensor detecta presencia utilizamos también el valor 1. El programa P_1 lo utilizamos para representar la base de conocimiento inicial y corresponde al momento en el cúal no se encuentra la persona en la oficina y por lo tanto el estado de las lámparas es apagado, para las reglas de este programa utilizamos el concepto de negación clásica.

El programa P_2 lo utilizamos para representar el estado de las lámparas según los sensores de presencia. En las reglas de este programa utilizamos la negación como falla que nos permite representar que si no tenemos evidencia de que un sensor detecta presencia el estado de las lámparas debe ser apagado.

El programa $P3$ lo utilizamos para representar la información que indica que sucede cuando un sensor detecta presencia que lámpara es la que debe cambiar su estado a encendido. Finalmente representamos la situaciónes que se le pueden presentar al sistema, en este caso vamos a asumir que s3 detectó presencia. A continuación motramo los programas con sus reglas correspondientes:

P_1 :

$$r1 : -l1(1).$$
$$r2 : -l2(1).$$
$$r3 : -l3(1).$$
$$r4 : -l4(1).$$

P_2 :

$$r1 : -l1(1) \leftarrow \neg s1(1).$$
$$r2 : -l2(1) \leftarrow \neg s2(1).$$
$$r3 : -l3(1) \leftarrow \neg s3(1).$$
$$r4 : -l4(1) \leftarrow \neg s4(1).$$

P_3 :

$$r1 : l1(1) \leftarrow s1(1).$$
$$r4 : l2(1) \leftarrow s2(1).$$
$$r6 : l3(1) \leftarrow s3(1).$$
$$r8 : l4(1) \leftarrow s4(1).$$
$$r9 : s3(1).$$

3.2. Aplicación del enfoque de actualización

De manera formal decimos:
Sea $\mathbf{P} = P_1, P_2, P_3$ una secuencia de programas, donde P_1, P_2, P_3. Y sea B el conjunto de nuevos átomos abductivos $\{x_1^1, x_2^1, x_3^1, x_4^1, x_1^2, x_2^2, x_3^2, x_4^2\}$. El nuevo programa lógico de actualización $P_{\oslash}= P_1 \oslash P_2 \oslash P_3$ el siguiente programa:

$$r1 : -l1(1) \leftarrow \neg x_1^1.$$
$$r2 : -l2(1) \leftarrow \neg x_2^1.$$
$$r3 : -l3(1) \leftarrow \neg x_3^1.$$
$$r4 : -l4(1) \leftarrow \neg x_4^1.$$
$$r1 : -l1(1) \leftarrow \neg s1(1), \neg x_1^2.$$
$$r2 : -l2(1) \leftarrow \neg s2(1), \neg x_2^2.$$
$$r3 : -l3(1) \leftarrow \neg s3(1), \neg x_3^2.$$
$$r4 : -l4(1) \leftarrow \neg s4(1), \neg x_4^2.$$
$$r1 : l1(1) \leftarrow s1(1).$$
$$r4 : l2(1) \leftarrow s2(1).$$
$$r6 : l3(1) \leftarrow s3(1).$$
$$r8 : l4(1) \leftarrow s4(1).$$
$$r9 : s3(1).$$

Una vez obtenido el nuevo programa actualizado, obtuvimos el conjunto de EEG answer sets. De todo el conjunto de EEG answer sets obtenemos el MEG answer set por medio del orden de inclusion. Por lo tanto el programa EAL de $\mathbf{P}$

es un triple $\langle P_{\oslash}, B, f\rangle$ donde $f{:}\rightarrow \{1,2\}$es la función $f(b^i_j = i)$. El MEG answer set de $\langle P_{\oslash}, B, f\rangle$ es:
$\{x^1_3, s3(1), -l1(1), -l2(1), -l4(1), l3(1)\}$ y como solo estamos interesados en M restamos el abductivo que nos permitió la actualización y por lo tanto, el answer set $\oslash$-actualización es: $\{s3(1), -l1(1), -l2(1), -l4(1), l3(1)\}$

3.3. Discusión de resultados

Observando el único MEG answer set $\{s3(1), -l1(1), -l2(1), -l4(1), l3(1)\}$ nos damos cuenta como el sistema efectivamente actualiza los estados de los sensores y la lámparas y una vez percibida presencia por el sensor s3 presencia el sistema actualiza el estado de l3 de un estado apagado a un estado encendido. con este resultado observamos que el sistema si logró actualizar los estados de las lámparas en este caso de l3 según el sensor s3 que fue el que detecto presencia.

4. Conclusiones

Dentro de las conclusiones de este trabajo podemos decir que cumplimos con el objetivo de modelar y resolver el problema de iluminación inteligente de una oficina utilizando el enfoque de actualización de secuencias de programas lógicos basado en ASP. Cabe mencionar que este escenario es solo un ejemplo que utilizamos para representar el comportamiento del enfoque de actualización. Ya que se puede utilizar cualquier escenario donde sea necesaria la actualización.

Por otra los resultados que obtuvimos fueron resultados satisfactorios por lo tanto concluimos que usando este enfoque es posible modelar problemas con información cambiante o información contradictoria gracias al potencial de ASP. Finalmente podemos decir que con respecto al enfoque de actualización no solo observamos la utilidad que tiene en este sistema sino también podría ser util en cualquier otro sistema o área donde exista la necesidad de actualización.

Agradecimientos. Este trabajo ha sido apoyado por el Proyecto de Ciencia Básica del Fondo Sectorial SEP-CONACyT con número de registro 101581.

Referencias

1. Balduccini, M., Gelfond, M.: Logic programs with consistency-restoring rules. In: on Logical Formalization of Commonsense Reasoning, I.S. (ed.) AAAI 2003 Spring Symposium. In P. Doherty, J. McCarthy, and M.-A. Williams, editors (1988)
2. Baral, C.: Knowledge Representation, reasoning and declarative problem solving with Answer Sets. Cambridge University Press, Cambridge (2003)
3. Carmen L. Garca M., P.R.M.G., Portillo, R.B.J.J.: A. robot platform motion planning using answer set programming. In: Latin American Workshop on Non-Monotonic Reasoning (2011)
4. de Colombia, S.E.: Soluciones en Control de Iluminación

5. Gelfond, M., Lifschitz: The Stable Model Semantics for Logic Programming. In: Kowalski, R., Bowen, K. (eds.) 5th Conference on Logic Programming. pp. 1070–1080. MIT Press (1988)
6. Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. In: Kowalski, R., Bowen, K. (eds.) 5th Conference on Logic Programming. pp. 1070–1080. MIT Press (1988)
7. Lifschitz, V.: What is answer set programming? In: Proceedings of AAAI-08 (2008)
8. Michael Gelfond, R.S., Baral, C.: Answer set programming as the basis for a homeland security qask. In: AAAI Spring Symposium: AI Technologies for Homeland Security (2005)
9. Navarro, J.A.: Tesis profesional de Licenciatura: Lógica Aplicada a Answer Sets (Universidad de las Américas, Puebla, 2003)
10. Niemelä, I.: Logic programming with stable model semantics as a constraint programming paradigm. In: Annals of Mathematics and Artificial Intelligence. pp. 241–273 (1996)
11. Osorio, M., Zepeda, C.: Minimal Extended Generalized Answer Sets and their Applications. In: Proceedings of the Workshop in Logic, Language and Computation (LoLaCOM06), Fifth Mexican Internantional Conference on Artificial Intelligence. vol. 220 (2006)
12. Romero, M.C.: Tesis profesional de Licenciatura: Modelado de problemas de actualización utilizando un enfoque de actualización de secuencias de programas lógicos basados en Answer Set Programming (Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla 2014)
13. You, J.H., Jia, L.Y.Y.X.: Adding Domain Dependent Knowledge into Answer Set Programs for Planning. In: Logic Programming, Lecture Notes in Computer Science. vol. 3132 (2004)
14. Zepeda, C.: Evacuation Planning using Answer Set Programming. Ph.D. thesis, Universidad de las Americas, Puebla and Institut National des Sciences Appliquées de Lyon (2005)