Cómputo Eficiente del Grado de Creencia de un Agente Inteligente sobre 2-FNC Monótonas

por

Lourdes Tecuapetla Quechol

Tesis sometida como requisito parcial para obtener
el grado de
Licenciada en Ciencias de la Computación

BUAP, Facultad de Ciencias de la Computación
Asesorado por Dr. Carlos Guillén Galván

Febrero 2013

A mis padres, Paula y Ricardo,

a mis hermanas, Mercedes y Guadalupe.
A Chamuel, siempre están en mi corazón...

Resumen

Dada una base de conocimiento cuya estructura corresponda a una fórmula monótona en 2 Forma Normal Conjuntiva (2-FNC), se identifican estructuras sobre el grafo asociado a esta fórmula que permitan el diseño de algoritmos eficientes para el cálculo del grado de creencia de un agente inteligente ante la llegada de nueva información. Esta información se recibe por el agente como una literal o cláusula.

Los algoritmos propuestos se basan en la identificación de operadores matriciales sobre estructuras simples que pueden tener los grafos asociados a una base de conocimiento con las características impuestas.

Se utiliza un método basado en modelos de una fórmula booleana monótona para realizar razonamiento y se trata el proceso de razonamiento como el cálculo del grado de creencia. Para llevar acabo el cálculo del grado de creencia en una variable o en una cláusula binaria se analizan cuatro casos. También se muestra una forma de actualizar la representación de la base de conocimiento.

Palabras clave: Forma Normal Conjuntiva, Grafos, Agente Inteligente, Grado de Creencia, Modelos, Cargas, Operadores de Conteo, Operadores de Proyección, Operadores Matriciales.

Índice

Agradecimientos

Agradezco a mi asesor de tesis al Dr. Carlos Guillén Galván y a mis revisores: Dra. Irene Olaya Ayaquica Martínez y al Dr. Rafael Lemuz López, por sus valiosos comentarios y apoyo que trasciende los límites de este trabajo. Mi especial reconocimiento al Dr. Guillén, sin su apoyo no hubiera sido posible la culminación de este trabajo. Mi agradecimiento al profesor Esteban Torres León y Carlos Guevara Hernández por su apoyo incondicional. A todos aquellos compañeros que han contribuido a la realización de este sueño.

Por último, me siento en deuda con la Universidad Autónoma de Puebla (BUAP), y a todo el plantel de docentes de la Facultad de Ciencias de la Computación (FCC), ya que durante mis estancia me forjaron en mi formación profesional. A todos ellos, mi más sincera gratitud.

Índice de figuras

Índice de tablas

Símbolos

$\mathbb{N}$	Números Naturales
$P(A)$	Conjunto Potencia
$[A]^k$	Conjunto de los subconjuntos de A con k-elementos
$G(V, E)$	Grafo no dirigido
$V(G)$	Conjunto de vértices o nodos del grafo G
$E(G)$	Conjunto de aristas del grafo G
$\|A\|$	Cardinalidad del conjunto A
$\|\|G\|\|$	Números de aristas del grafo G
$Var(F)$	Conjunto de variables de F
$Lit(F)$	Conjunto de literales de F
s	Función signo
$grad(x)$	Número de ocurrencias de la variable x en la fórmula F
$M(F)$	Conjunto de asignaciones sobre $Var(F)$ que satisfacen a F
σ	Asignación
$M_{x=i}(F)$	Conjunto de modelos en $M(F)$ que toman el valor de i en la variable x
G_F	Multigrafo
Φ	Teoría proposicional
P_Φ	Distribución uniforme sobre el conjunto $\mathbb{B}^n$
$P_{\alpha\|\Phi}$	Probabilidad condicional de α con respecto a Φ
$\Phi \circ p$	Cambio de Φ al incorporar p a la base de conocimiento
(Φ, F)	Conjunto de subconjuntos maximales de Φ
$\top$	Verdadero
$\bot$	Falso

Capítulo 1

Introducción

Dentro del área de ciencias computacionales, el problema de satisfactiblidad (SAT) es muy importante. En este tipo de problema recae todo lo relacionado al proceso de toma de decisiones (si una fórmula es satisfactible o no). Además, está vinculado con el conteo de satisfactibilidad (#SAT), lo cual ayuda a saber el número de asignaciones que satisfacen una fórmula dada.

Hay gran variedad de aplicaciones donde se hace uso de SAT y #SAT destacando el razonamiento automático, planificación y visión por computadora. Es dentro de Inteligencia Artificial donde ha tenido mayor impacto el problema anterior, debido a que permite calcular el grado de creencia que puede tener un agente inteligente con una base de conocimiento KB ante la llegada de nueva información α. El problema aquí es saber que tan confiable es la nueva información dado que el agente cuenta con información en la base KB y se cumple que,

$$\neg(KB \vdash \neg\alpha) \wedge \neg(KB \vdash \alpha)$$

Donde la base de conocimiento se encuentra representada por una fórmula booleana bajo la condición de que esas fórmulas estén en 2 Forma Normal Conjuntiva (2-FNC). Bajo la anterior premisa, para que un agente inteligente tome una decisión se debe calcular el grado de creencia:

$$P_{\alpha|\Sigma} = Prob\{KB \wedge \alpha \equiv T \mid KB \equiv T\} = \frac{\mid M(\alpha \wedge \Phi) \mid}{M(\Phi)}.$$

El grado de creencia indica la probabilidad de qué tanto cree el agente en la nueva oración α, esto es la proporción de modelos de KB que continúan siendo modelos para $KB \wedge \alpha$.

Como se puede observar el grado de creencia se encuentra estrechamente vinculado con el número de modelos de una fórmula, por lo consiguiente computar el grado de creencia es un problema difícil aún cuando la base de conocimiento y la nueva información se encuentra representada por fórmulas booleanas en 2-FNC.

Debido a la dificultad del cómputo del grado de creencia y su relación con el conteo de modelos de sentencias proposicionales existe la necesidad de identificar estructuras sobre la base de conocimiento que permitan facilitar la tarea a través del diseño de algoritmos eficientes que actúen sobre estas estructuras.

La organización del documento es la siguiente:

El **Capítulo II**, está dedicado a definir de manera formal los conceptos matemáticos fundamentales para el desarrollo de este trabajo. Destacando la definición de Fórmula Booleana, Forma Normal Conjuntiva (FNC) y Grafos.

El **Capítulo III**, presenta una descripción del estado del arte, que incluye el análisis de las principales metodologías para computar de manera eficiente el grado de creencia de un agente inteligente. Además, se analizan los principales trabajos desarrollados para la Revisión-Actualización de Bases de Conocimiento.

En el **Capítulo IV**, se introduce la teoría dedicada a modelos y cargas en grafos y fórmulas. Así, como el diseño de algoritmos basados en operadores matriciales que computa el grado de creencia de una claúsula en el caso de una base de conocimiento en 2-FNC.

En el **Capítulo V**, se utiliza el método basado en los modelos de una fórmula booleana monótona para realizar razonamiento y se trata el proceso de razonamiento como el cálculo del grado de creencia; se analizan para ello 4 casos. También se muestra una forma de actualizar la representación de la base de conocimiento. Así mismo, se describen en detalle las principales aportaciones realizadas con esos algoritmos.

En el **capítulo VI**, se detallan las conclusiones, y trabajo futuro, que se desprenden de la investigación de este trabajo.

Capítulo 2

Preliminares

2.1. Conceptos Básicos

En este capítulo se presentan algunos conceptos de la teoría de grafos que serán de utilidad para el desarrollo del presente documento. La idea es asociarle a una fórmula booleana una estructura de grafo. También, se establece la notación básica que se requiere para el desarrollo de esta tesis.

El conjunto de los números naturales (enteros no negativos) $\{0, 1, 2, 3, ...\}$ se denota por $\mathbb{N}$. Dado un conjunto A el conjunto potencia de A se representa mediante $\mathcal{P}(A)$. El conjunto de todos los k-subconjuntos de A se denota por $[A]^k$, por ejemplo:

$[\{a, b, c\}]^0 = \emptyset$,
$[\{a, b, c\}]^1 = \{\{a\}, \{b\}, \{c\}\}$,
$[\{a, b, c\}]^2 = \{\{a, b\}, \{a, c\}, \{b, c\}\}$,
$[\{a, b, c\}]^3 = \{\{a, b, c\}\}$,
$[\{a, b, c\}]^k = \emptyset$, $k \geq 4$.

Un *grafo no dirigido* G es un par de conjuntos (V, E), donde $V \cap E = \emptyset$, V es llamado conjunto de *vértices* o *nodos* y E es un subconjunto de $[V]^2$ llamado conjunto de *aristas*. Es común escribir $V = V(G)$ y $E = E(G)$. Por ejemplo, si H representa el grafo no dirigido de la figura 2.1, entonces $V(H) = \{a, b, c, d, e, f, g, h, i, j\}$ y $E(H) = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, e\}, \{e, f\}, \{f, a\}, \{i, j\}, \{g, h\}\}$. Por comodidad denotamos $\{v, w\} = vw$. Si v, w son nodos de G, $G + vw$ denota el grafo $(V(G), E(G) \cup \{v, w\})$ [Die05, AK07].

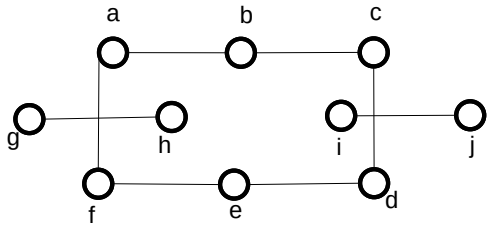


FIGURA 2.1: Grafo H.

Se dice que los nodos v y w son *adyacentes* si vw es una arista en $E(G)$, caso contrario v y w son nodos *independientes*, un subconjunto X de $V(G)$ se dice *independiente* si cualesquiera dos nodos en X son independientes.

El *orden* de un grafo G está definido por el número de nodos en $V(G)$ y se representa por $|G|$. El *tamaño* de G está dado por el número de aristas en $E(G)$ y es representado por $\|G\|$. En el grafo de la figura 2.1 tenemos $|H| = 10$ y $\|H\| = 8$. Una arista e es incidente en el nodo v si $e = vw$ para algún nodo w de G. El grado $d(v)$ de un nodo v es el número de aristas incidentes en este nodo.

Una *trayectoria o camino* es un grafo no vacío $P = (V, E)$ donde $V = \{x_0, x_1, \ldots, x_k\}$ y $E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$. Los vértices x_0 y x_k son *ligados* por P y son sus *extremos*. Para abreviar, frecuentemente escribimos $P = x_0x_1 \ldots x_{k-1}x_k$ y nos referimos a P como un camino *entre* x_0 y x_k.

Si $P = x_0x_1 \ldots x_{k-2}x_{k-1}$ es un camino y $k \geq 3$, entonces el grafo $C = P + x_{k-1}x_0$ es un *ciclo*, por simplicidad denotamos $C = x_0x_1 \ldots x_{k-2}x_{k-1}x_0$. En el grafo de la figura 2.1 tenemos que $P = abcde$ es un camino con extremos a y e, $C = abcdefa$ es un ciclo.

Dado un grafo $G = (V, E)$, el *subgrafo inducido* por el conjunto de nodos $V' \subseteq V$, es el subgrafo $G' \subseteq G$ tal que $V(G') = V'$ y $E(G') = \{e \in E : e \text{ tiene sus extremos en } V'\}$. Por ejemplo, en el grafo de la figura 2.1 tenemos que el subgrafo inducido por el conjunto de nodos $\{a, b, f, g, h, d\}$ es el grafo de la figura 2.2.

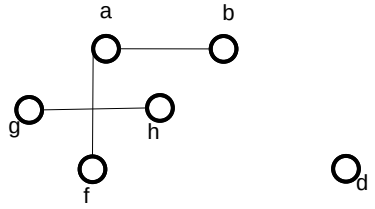


FIGURA 2.2: Grafo inducido por los nodos a, b, f, g, h, d.

Un grafo no vacío G es *conexo o conectado* si cualesquiera dos de sus vértices se encuentran ligados por un camino en G. Una *componente* de G es un subgrafo

conectado maximal de G. En el grafo de la figura 2.1 las componentes conexas están dadas por los subgrafos inducidos por los conjuntos de nodos $\{a, b, c, d, e, f\}$, $\{g, h\}$ y $\{i, j\}$.

Un *árbol* es un grafo conexo acíclico, esto es, un grafo conexo que no contiene cíclos. Si T es un árbol se observa fácilmente que si $t, t' \in V(T)$, existe un único camino que une a t con t', el cual es denotado por tTt'.

Una *rama* de un grafo G es un camino $T = x_0x_1 \cdots x_k$, tal que $T \subseteq G$, $d(x_0) = 1$ o $d(x_k) = 1$ y $d(x_i) = 2$ para $i = 1, ..., k-1$.

Un *grafo etiquetado sobre un conjunto* S es un grafo junto con una función $\psi : E \to S$. Por ejemplo, sea $S = \{+,-\} \times \{+,-\}$ y G el grafo con $E(G) = \{ab, bc, cd, ad, ef\}$, $V(G) = \{a, b, c, d, e, f\}$ y $\psi(ab) = (+,-)$, $\psi(bc) = (+,+)$, $\psi(cd) = (-,+)$, $\psi(ad) = (-,-)$, $\psi(ac) = (-,+)$ y $\psi(ef) = (+,-)$ (ver figura 2.3).

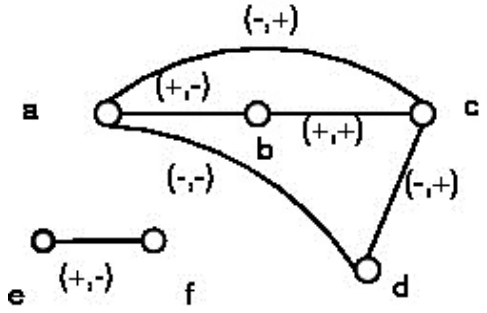


FIGURA 2.3: Grafo Etiquetado.

Si un grafo G es etiquetado sobre $S = \{+,-\} \times \{+,-\}$ las aristas etiquetadas mediante $(+,+)$, $(+,-)$, $(-,+)$ y $(-,-)$ se representan gráficamente por las flechas $\leftrightarrow$, $\leftarrow$, $\rightarrow$ y $\rightarrow\leftarrow$ respectivamente. El grafo del ejemplo anterior con esta notación de las aristas se ve como la figura 2.4.

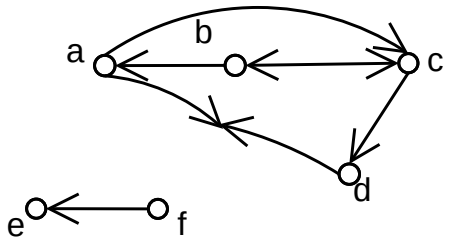


FIGURA 2.4: Grafo Etiquetado con la notación de aristas.

Un *multigrafo* es un terna $G = (V, E, \varphi)$ donde V es un conjunto de vértices o nodos, E es un conjunto de aristas y $\varphi : E \to V \cup [V]^2$ una función. Por ejemplo, sea

$G = (V, E, \varphi)$, donde $V = \{x, y, z\}$, $E = \{e_1, e_2, e_3, e_4\}$, $\varphi(e_1) = x$, $\varphi(e_2) = \{x, z\}$, $\varphi(e_3) = \{y, z\}$ y $\varphi(e_4) = \{y, z\}$ (ver figura 2.5).

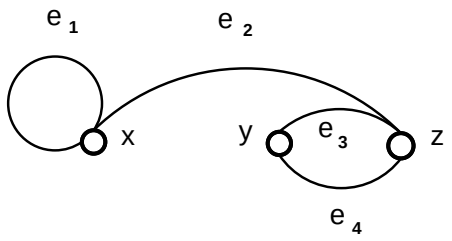


FIGURA 2.5: Multigrafo G.

2.2. Fórmulas Booleanas en FNC

Si V es un conjunto de variables, el conjunto de variables negadas $\{\neg x : x \in V\}$ es denotado por $\neg V$. Cualquier elemento de $V \cup \neg V$ es una *literal*. La variable asociada a la literal l se denota por $\nu(l)$, por ejemplo, $\nu(\neg x_3) = x_3$ y $\nu(\neg x_5) = x_5$.

Una *fórmula booleana*[1] F en forma normal conjuntiva (FNC) de n variables booleanas $X = \{x_1, x_2, ..., x_n\}$ consiste de un conjunto de cláusulas $\{c_1, c_2, ..., c_m\}$, donde cada *cláusula* es un conjunto de literales sobre X. Por ejemplo:

$F = \{\{x_1\}, \{x_1, \neg x_2, \}, \{x_2, \neg x_3, \}, \{x_1, \neg x_2, \neg x_3\}\}$ es una fórmula en FNC.

Una fórmula booleana F es k-FNC si es una FNC y $|c| \leq k$ para toda cláusula c de F. Con $Var(F)$ denotamos el conjunto de variables de F y el conjunto de variables negadas $\neg Var(F)$ es definido como $\{\neg x : x \in Var(F)\}$. $Lit(F)$ denota el conjunto de literales de F, observemos que $Lit(F) \subseteq Var(F) \cup \neg Var(F)$.

La *función signo* s: $Lit(F) \to \{-, +\}$ es definida como $s(l) = +$ si $l \in Var(F)$ y $s(l) = -$, si $l \in \neg Var(F)$. Si l es una literal en la cláusula c y $x = \nu(l)$, entonces el *signo de x en la cláusula c* es $s(l)$.

Dado $x \in Var(F)$ el *grado* de x es el número de ocurrencias de la variable x en la fórmula F, este número es denotado por $grad(x)$.

Una *asignación* σ para la fórmula F es una función $\sigma : Var(F) \to \mathbb{B}$. Decimos que la asignación σ *satisface* a la literal l si y sólo si $\sigma(\nu(l)) = 1$ si $l \in Var(F)$ y $\sigma(\nu(l)) = 0$ si $l \in \neg Var(F)$. Una asignación σ *satisface* a una cláusula c si y sólo

[1]Si no se especifica lo contrario cuando se utilicen los términos "fórmula" o "variable" se sobreentenderá que nos referimos a una fórmula o variable booleana respectivamente.

si existe una literal $l \in c$ tal que σ la satisface. Una asignación σ *satisface* una fórmula Booleana F si y sólo si σ satisface toda cláusula de F. Si $V \subseteq Var(F)$ la función $\sigma : V \to \mathbb{B}$ es una *asignación parcial* para F.

El *conjunto de modelos* de una fórmula booleana F es el conjunto de asignaciones sobre $Var(F)$ que satisfacen a F, este conjunto es denotado por $M(F)$.

Dada una fórmula F en FNC, para $i \in \mathbb{B}$ se denota por $F_{x=i}$, el resultado de hacer $x = i$ en F y simplificar por: (1) remover toda cláusula que contenga "1"; y (2) deshacerse de todo "0". Por ejemplo, si $F = \{\{x, y\}, \{\neg x, w\}, \{\text{z,y}\}\}$, entonces $F_{x=1} = \{\{w\}, \{z, y\}\}$ y $F_{x=0} = \{\{y\}, \{z, y\}\}$. En ocasiones, cuando la fórmula tiene subíndices, se separa con una raya vertical "|" la notación para evitar confusiones y el uso excesivo de paréntesis, por ejemplo $F_j|_{x=i}$ denota la fórmula $(F_j)_{x=i}$.

Para $x, y \in Var(F)$ y $i \in \mathbb{B}$, se denota por $M_{x=i}(F)$ el conjunto de modelos en $M(F)$ que toman el valor de i en la variable x, esto es $M_{x=i}(F) = \{\sigma \in M(F) : \sigma(x) = i\}$. En general, para $i, j, ..., k \in \mathbb{B}$ y $x, y, .., z \in Var(F)$ se tiene la notación

$$M_{x=i,y=j,...,z=k}(F) = \{\sigma \in M(F) : \sigma(x) = i, \sigma(y) = j, ..., \sigma(z) = k\}.$$

2.3. Grafos y Fórmulas en FNC

El estudio de una fórmula en FNC puede realizarse a través de la estructura topológica de su grafo de restricciones. En esta sección se define concepto de grafo de restricciones y se establecen las diferentes representaciones de una fórmula en FNC en términos de su grafo asociado.

El *grafo de restricciones* de una fórmula F en 2-FNC, es un multigrafo G_F tal que $V(G) = Var(F)$ y $E(G) = F$ y el mapeo $\varphi : E \to [V]^1 \cup [V]^2$ es tal que $\varphi(c) = Var(c)$.

Ejemplo 1. *Consideremos la siguiente fórmula,*

$$F = \{\{\neg x, y\}, \{y, z\}, \{\neg y, \neg w\}, \{x\}, \{x, z\}, \{y, \neg z\}\}.$$

El grafo de restricciones G_F es dado en la figura 2.6a.

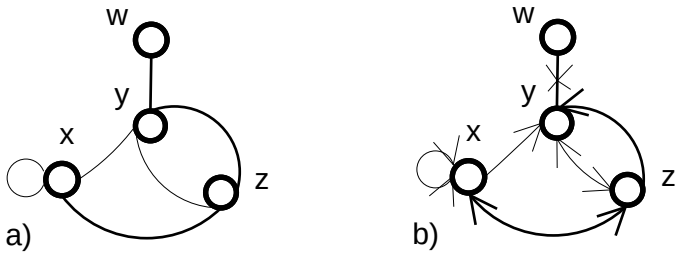


FIGURA 2.6: a) Multigrafo G_F, b) Multigrafo Etiquetado G_F.

El grafo de restricciones de F, *etiquetado sobre* $[S]^2$, donde $S = \{+, -\}$, es su grafo de restricciones G_F junto con la función $\psi : E \to [S]^2$, tal que $\psi(\{l, l'\}) = \{s(l), s(l')\}$. La imagen $\psi(\{l, l'\})$ es llamada *etiqueta de la cláusula* $\{l, l'\}$ y se denota por $S(\{l, l'\})$.

Acordamos representar gráficamente la etiqueta $\{s(l), s(l')\}$ sobreponiéndola a la arista $\{l, l'\}$, colocando de manera adyacente l con $s(l)$ y l' con $s(l')$. Por ejemplo, para la fórmula F dada en el ejemplo 2, la figura 2.6b muestra su grafo de restricciones etiquetado sobre $[S]^2$. Note que si $\nu(l) = \nu(l')$, la arista $\{l, l'\}$ es un bucle.

Una fórmula en 2-FNC es un *camino, árbol, ciclo, etc.* si su grafo de restricciones cumple con ser un camino, árbol, ciclo, etc., respectivamente. Una fórmula booleana en 2-FNC es *conexa o conectada*, si su grafo de restricciones es conexo.

Capítulo 3

Estado del Arte

3.1. Base de Conocimiento

3.1.1. Cómputo del Grado de Creencia

Una base de conocimiento puede ser modelada de diferentes formas y con diferentes niveles de abstracción lógica en dependencia del grado de dificultad del problema que se necesite abordar, por ejemplo, los modelos pueden estar dentro de los distintos niveles: lógica proposicional, lógica de primer orden, lógicas de alto orden, por mencionar algunas. Una primera aproximación es mediante una teoría proposicional; ya en este primer nivel se engloban grandes retos, uno de ellos es el diseño de algoritmos eficientes que puedan computar el grado de creencia que tiene nueva información dada por una fórmula proposicional. Para poder tomar decisiones en un ambiente de incertidumbre se asigna un grado de creencia a la nueva información siguiendo el siguiente principio:

1) Se asigna el mismo grado de creencia a todas las *situaciones básicas* consistentes con la base de conocimiento.
2) Se calcula la parte de las situaciones que son consistentes con la pregunta (query).

Todos los modelos posibles de la teoría tienen igual peso y nos interesa la complejidad computacional del cómputo del grado de creencia de una fórmula proposicional.

Existen trabajos orientados a la aplicación de este principio y a determinar cuáles son las situaciones básicas [DNP04a, DNP04b, DS04, HD05]. Debido al alto costo computacional es de interés computar el grado de creencia en un caso más restrictivo, como el de una base de conocimiento que consiste de una teoría proposicional sin información estadística. Las dificultades computacionales aún en este caso restrictivo son altas, Grove [GHK92] trata la versión de predicados de primer orden de este problema donde demuestra que la gran mayoría de los problemas que pudieran requerir una respuesta son altamente indecidibles.
Dada una teoría proposicional Φ sobre n variables, la probabilidad de que Φ sea satisfactible, P_Φ, es computada con la distribución uniforme sobre el conjunto $\mathbb{B}^n$ de todas las posibles asignaciones de Φ.

$$P_\Phi = Prob\{\Phi \equiv T\} = \frac{\mid M(\Phi) \mid}{2^n},$$

donde $M(\Phi)$ denota el conjunto de todas las asignaciones que satisfacen Φ, se denota su cardinalidad como $\mid M(\Phi) \mid$ y T es la constante "*verdadero*".

Ejemplo 2. *Si tenemos que* $\Phi = \{\{x, y\}, \{\neg x, w\}, \{z, y\}\}$, $\mid M(\Phi) \mid$ *coincide con el número de hojas del árbol de la figura 3.1.*

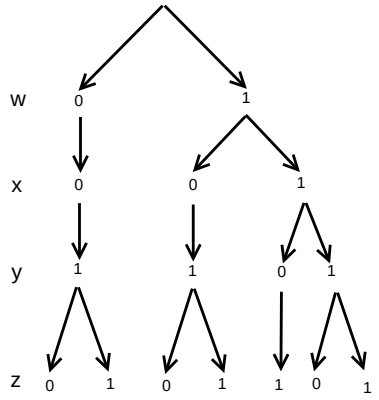


FIGURA 3.1: Árbol de modelos de F .

En este caso tenemos que $P_\Phi = 7/2^4 = 0{,}4375$.

Dada una teoría proposicional Φ y una afirmación proposicional α, **el grado de creencia en** α es la probabilidad condicional de α con respecto a Φ, $P_{\alpha|\Phi}$, esto es, la proporción de modelos de Φ que satisfacen a α:

$$P_{\alpha|\Phi} = Prob\{\alpha \wedge \Phi \equiv T \mid \Phi \equiv T\} = \frac{\mid M(\alpha \wedge \Phi) \mid}{M(\Phi)}.$$

En el ejemplo anterior, si $\alpha = \{\neg x, \neg y\}$, entonces $\alpha \wedge \Phi = \{\{\neg x, \neg z\}, \{x, y\}, \{\neg x, w\}, \{z, y\}\}$ y $\mid M(\alpha \wedge \Phi) \mid$ se puede calcular mediante las hojas del árbol de la figura 3.2:

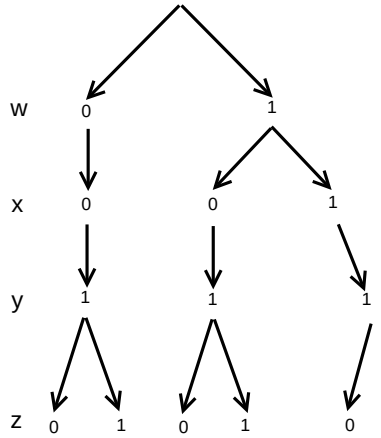


FIGURA 3.2: Árbol de modelos de $\alpha \wedge \Phi$.

Entonces, es claro que $P_{\alpha|\Phi} = 5/16 = 0{,}3125$.

En [RSW01a], Roth demuestra que la complejidad en el cómputo de P_Φ con respecto a una teoría proposicional, está polinomialmente relacionada con la complejidad de cómputo del número de modelos de una afirmación proposicional. También demuestra que el problema de computar el grado de creencia en una afirmación proposicional con respecto a una teoría proposicional es #P-completo.

3.1.2. Revisión-Actualización de Bases de Conocimiento

Dada una base de conocimiento, representada por la teoría α, la importancia de esa teoría se debe a que brinda información que permite describir como es el mundo externo y, va acompañado de los modelos, los cuales permiten conocer el estado de ese mundo. Sobre esa base se pueden realizar dos operaciones básicas: revisión y actualización como se indica en [KM91, ZZC].

Ejemplo 3. *Un robot monitorea una planta nuclear, para ello debe saber el comportamiento de su ambiente, por mínimos cambios que se produzcan en él, le deberan ser notificados. Para ello, debe estar constantemente revisando el ambiente, y ante la llegada de nuevos cambios (o sentencias) se agregarán a su base de conocimiento, a este proceso se le conoce como actualización. Es decir, dada la*

base de conocimiento del agente, ante la llegada de una nueva sentencia se actualiza su conocimiento. Mientras que si al agente se le asignan nuevas tareas a realizar y éstas no contradicen a las anteriores (esto es, a su base de conocimiento), solo es revisión.

A través de la revisión de creencias un agente inteligente puede percibir que su ambiente está cambiando, lo que causa que sus creencias también deban actualizarse para que no sean imprecisas. Para llevar acabo la revisión de creencias se puede seguir un patron básico. Para poder alojar nueva información ésta no deberá ser inconsistente con la información de la base, de lo contrario se tendrá que actualizar la base. Se han aplicado técnicas para llevar acabo la revisión-actualización de creencias como se puede ver en [GP96, NPP02, FP01, ZZC, Bou95]. Observar el siguiente ejemplo.

Ejemplo 4. *Carlos ve a Fernando conduciendo un automóvil. Él cree que si Fernando está conduciendo ese automóvil, él es su dueño. Así, podemos designar como K la creencia que tiene Carlos en ese instante de tiempo t. A continuación Fernando se baja del carro y conversa con Carlos, le dice que ese carro es de su primo, esto sucede en otro tiempo t'. De ahí, que la creencia de Carlos debe de cambiar del tiempo t a t', para actualizar su base de conocimiento con la llegada de la nueva información.*

La mayoría de los resultados para que un agente revise o actualize sus creencias dada una base de conocimiento representada como KB, se basan en métodos como: Mundos posibles de Ginsberg, operador de Nebel, Mundos posibles de Winslett, Satoh, y Dalal.

Mundos Posibles de Ginsberg [Gin86, L$^+$09, EG]. Aquí se consideran situaciones o eventos que no han acontecido en el universo actualmente observable pero que pudiera haber ocurrido en algún otro universo, en lógica modal este tipo de eventos se denominan contrafactuales o contrafácticos. Un contrafactual es una alternativa a un evento que ocurrió en el pasado. Por ejemplo, si consideramos el siguiente contrafactual:

$$p \Rightarrow q$$

considerando que el *mundo posible* es tan parecido como sea posible al mundo real dado que p es verdadero en éste.

Un contrafáctico es una sentencia con la siguiente estructura si p, entonces q. Por lo general se encuentran en forma condicional, y cuando tienen esta estructura se les llama también condicionales contrafácticos, aunque no siempre tiene esta forma y no es una regla. Un ejemplo de un enunciado contrafáctico escrito en forma condicional es el siguiente:

Ejemplo 5. *Si Socrátes no hubiera sido condenado a muerte, hubiera muerto por alguna enfermedad.*

Se espera que el antecedente del contrafáctico sea falso, y desde la semántica de la lógica, al darse la anterior premisa se cumple que un contrafáctico siempre sea cierto.

Una de las principales características de los contrafácticos es la dirección. La dirección hace referencia a si el resultado fue mejor o peor que en el universo actual, éste puede ser ascendente o descendente. Se puede observar su uso en diversas áreas, como en Filosofía, Inteligencia Artificial, Psicología, e Historía. Dentro de ésta última podemos mencionar que es aplicado para dar soluciones alternas a eventos ocurridos, como se puede observar en el siguiente ejemplo [Ver99].

Ejemplo 6. *En el año de 1532, Francisco Pizarro invade Perú, y se dirige a la ciudad Cajamarca, toma como prisionero a su lider inca Atahualpa. Pizarro promete dejarlo en libertad si llena un cuarto de oro, a lo cual accede Atahualpa. Reúne a sus súbditos y cumple lo prometido, posteriormente es ejecutado en la plaza de armas de Cajamarca.*

Si se analiza el ejemplo anterior, si Atahualpa hubiera usado esa influencia y comunicación con su pueblo para revelarse, y no para salvar su vida como lo hizo, entonces la conquista de Cajamarca y de Perú no se hubiera dado. Al final no le hubieran privado de su vida. En este ejemplo planteado, se nota que el contrafáctico es ascendente debido a que los resultados alternos posiblemente hubieran sido mejores.

Estudios de Nebel [Neb91, Neb89]. Un agente inteligente para inferir, requiere de alguna estructura donde pueda acceder a su conocimiento. De ahí, radica la importancia de estudiar diferentes maneras de representarlo. Uno de los principales problemas que se presenta en la representación del conocimiento es la revisión del mismo ante la llegada de nueva información, que posiblemente puede ser contradictoria y causar incosistencia.

Existen diferentes paradigmas para representar el conocimiento, una forma consiste en almacenar un conjunto de creencias, o sólo una base de creencias. Siendo esta última, la más usada, debido a que un pequeño conjunto de creencias permite reproducir a todo el conjunto. Además, se sugiere que se trabaje sobre bases de creencias en lugar de conjuntos de creencias, debido a que esta última puede ser infinita o muy grande.

Un problema que surge al hacer uso de las bases de creencias es la selección de la base, ya que puede haber varias bases de creencias que puedan representar al mismo conjunto de creencias. Nebel propone que a través de la teoría del cambio epistémico [1] se puede seleccionar la base de creencia que contenga sentencias proposicionales que representen observaciones específicas, y reglas. Este enfoque permite que las proposiciones almacenadas en la base sean las creencias fundamentales y el resto se pueden deducir de ellas.

Ejemplo 7. *Juego de Obligatio. A un agente se le proporciona una lista de proposiciones, $\varphi_1, \varphi_2, ..., \varphi_n$ que debe aceptar o rechazar en tiempo real. En cuanto sea proporcionada φ_i al agente, éste debe decidir si la acepta o la rechaza, si la acepta, se esta tomando como verdadera φ_i y si la rechaza será tomada como falsa $\neg\varphi_i$. Además, deberá tomar como verdaderas aquellas que sean consecuencia directa o indirecta de φ_i. El agente debe de ser capaz de procesar todas las proposiciones sin caer en inconsistencia.*

Modelos posibles de Winslett [FKUV86, KM91]. El origen de los *Modelos de Winslett* surgió ante la necesidad de dar solución al problema de distinguir entre *un estado del mundo y la descripción de ese estado del mundo*, que es una deficiencia presente en los Contrafácticos. El que no haya una distinción entre estos conceptos ocasiona dos problemas: *el problema del frame y el problema de ramificación.* El problema del frame, se refiere a qué sentencias se mantienen como verdaderas y no sufren cambios aunque acciones en el mundo externo se produzcan por mínimas que sean. En el problema referente a ramificiación es, qué sentencias se deben modificar ante los cambios que se lleven acabo en el mundo externo.

Básicamente, la ventaja de usar sentencias protejidas presentes en los modelos de Winslett ayuda a dar solución a los dos problemas planteados anteriormente,

[1] La epistemología (del griego, episteme, *conocimiento*; logos, *teoría*) es la rama de la filosofía cuyo objeto de estudio es el conocimiento. Es decir, la epistemología se ocupa de la definición del saber y de los conceptos relacionados, de las fuentes, los criterios, los tipos de conocimiento posible y el grado con el que cada uno resulta cierto.

cuyo objetivo es actualizar información en una base de conocimiento. La etapa de interés es definir que modelo es el adecuado, si se basa en que varios modelos del mundo pueden definir los estados del mundo. Es decir, si se tienen dos modelos $m1$ y $m2$, con una sentencia de diferencia cuál sería la óptima para agregar a la base de conocimiento. Una alternativa sería unir ambos modelos y actualizar la base.

Otra ventaja de este modelo es que se consideran condiciones iniciales de los estados del mundo lo que permitiría definir qué acciones llevarían al éxito en la realización de una tarea asignada a un agente inteligente. Además, este enfoque usa modelos, lo que hace que sea más rápido, a diferencia de los contrafácticos que utiliza fórmulas para incorporarlas a una base de conocimiento dada.

Este modelo, sigue un principio básico de protección de sentencias base que no pueden ser modificadas aún cuando el mundo externo cambie. Basándonos en la premisa de que dos elementos no pueden ocupar el mismo espacio.

Ejemplo 8. *En una habitación hay un robot, dos libros (uno rojo y uno azul), y un librero. Se le ordena al robot que tome el libro rojo que está sobre el suelo y lo coloque en el librero, teniendo en cuenta que solo hay un espacio disponible. Para ello, el robot hace una revisión de los estados del mundo ψ, y se desea incorporar el hecho μ de colocar un objeto en el librero. Si se denota con a, el libro rojo, y con b el libro azul, sea m o n el hecho de colocar a o b en el librero respectivamente. Entonces el robot realiza el siguiente razonamiento, será colocado el libro rojo en el librero, si no esta el libro azul en ese espacio, posteriormente se realiza la actualización de ψ.*

Revisión de Dalal [RSW01b]. El mecanismo de Dalal trabaja con modelos al igual que en *modelos posibles de Winslett* . Dalal propone un mecanismo para la revisión y actualización de creencias, llamado operador de Dalal. Desde el punto de vista sintáctico, hace un conteo de las sentencias verdaderas y que deben cambiar en la base de conocimiento. Siguiendo el principio de que dichas sentencias deban cambiar lo mínimo posible en la base actual. Para ello, se debe buscar el conjunto de sentencias con el menor número de sentencias verdaderas. El revisarlas conlleva a realizar la comprobación de satisfactibilidad múltiple.

Para reducir este problema algunos autores como [RSW01b] proponen que la KB, así como las nuevas sentencias que se deseen incorporar deben tener la misma estructura, es decir se definan sintácticamente siguiendo la Forma Normal Disyuntiva

(FND) o Forma Normal Conjuntiva (FNC), para nuestro estudio consideramos el último [FP]:

$$A = a_1 \wedge a_2 \wedge a_3 \wedge ... \wedge a_n$$

Además de lo anterior, se hará uso del concepto de *quantum conjuntivo*, para ello se requiere de dos elementos; el primero de ellos es a_n que corresponde la literal a buscar. El segundo elemento F_c, representa el subconjunto con las etiquetas o coordenadas de las fórmulas donde aparece a_n en ψ y se representa (a_n, F_c). Donde F_c, indica la situación de la literal a_n en la base. También debe cumplir el siguiente criterio, $|F_c \neq \varnothing|$; pasando la base de conocimiento de FNC a FNC quantum:

$$A = a_1^{F1} \wedge a_2^{F2} \wedge, ..., \wedge a_n^{Fn}$$

Otro término importante es que Dalal, en su teoría considera como cambio mínimo, aquellas sentencias con el menor número de valores verdaderos en las sentencias y, les asigna el mismo peso a cada una de ellas, dada una fórmula simple o compleja. Para nuestro caso de estudio nos interesan solo aquellas que cumplan con el cambio mínimo en FNC, pero de grado dos, como se observa a continuación:

$$A = a_1 \wedge a_2$$

Sólo debe tener la restricción de que A no contenga a la literal a y a su negación ($a_1 \wedge \neg a_1$). Por ejemplo, llegan dos fórmulas B y C a la base actual:

$$B = a_i \wedge a_j, C = a_i \wedge a_k$$

A continuación, se busca la literal a_i en ψ y se realiza la operación de $|a_i \cap a_{i1}^{Fi}|$. Así, se procede con las otras literales (a_j, a_k). Posteriomente, se obtendrá la suma de la cardinalidad de las fórmulas $|B|$ y $|C|$ por separado. Así, $\psi \bigcup C$, donde C, la fórmula con la cardinalidad más pequeña, es la que será anexada a la base de conocimiento, manteniendo la consistencia e integridad en ψ.

El método de Satoh o colección de teorías [RFV86]. La mayoría de los trabajos desarrollados como se puede observar en [L$^+$09, KM91] se basan en una sola teoría. Pero, la limitación es que esa teoría puede no contener toda la información acerca del mundo. Motivo por el cual Satoh sugiere que para la actualización de una base de conocimiento se empleen múltiples teorías (o colección de teorías) para tener una visión lo más completa posible del mundo externo.

A la colección de teorías se le llama "flocks", en el momento de actualizar un "flock", se debe actualizar cada una de las teorías que lo forman por medio de disyunciones de todas las sentencias que se desean actualizar en cada una de las teorías que formen al flock S, que deben seguir el cambio mínimo en la KB. La ventaja de este enfoque es que al momento de ingresar nuevas sentencias a la base, ya no crece de manera exponencial el tamaño de la base de conocimiento.

A continuación enlistamos otros métodos de revisión-actualización que son de interés en el área debido a la naturaleza del problema que se desee tratar:

a) La investigación de Weber [FUV83]
b) El Producto-Cruz [FUV83]
c) El WIDTIO [Gin86, FUV83]
d) La investigación de Forbus [FKUV86]
e) El método de Borgida [FUV83]

Estos métodos obedecen el llamado *principio de mínimalidad de cambio*, este principio establece que cuando se incorpore nueva información a la base de conocimiento ésta deberá cambiar lo mínimo posible . Dependiendo de la aplicación particular que se tenga se tiene un método adecuado correspondiente, hasta la fecha no existe un método general que sea adecuado para cualquier circunstancia. En [KM91] la complejidad computacional de las diferentes propuestas fue presentada como un problema importante. Aunque se sabe que la fórmula de implicación [2]

$$\Phi \circ p \models q$$

es intratable para cada método conocido, la complejidad precisa de este problema es un problema abierto. Además, no se tiene claro cúales son las restricciones del

[2]Dada una base de conocimiento Φ, p una actualización y q una fórmula, $\Phi \circ p$ denota el cambio de Φ al incorporar p a la base de conocimiento aplicando el operador de cambio $\circ$.

problema de revisión-actualización para que sea tratable [EG92].

El siguiente método corresponde a la dirección de este trabajo. Sea Φ una base de conocimiento, es de nuestro interés revisar Φ dada nueva información representada por una fórmula F, esta operación se denota por

$$\Phi^*_M F$$

donde M es el método empleado para realizar la revisión.

Cambio de Fórmula Base. Si una base de conocimiento Φ es inconsistente con una nueva fórmula F, un método simple de ganar consistencia con F es remover las fórmulas de Φ que son inconsistentes con F. Sea $W(\Phi, F)$ el conjunto de subconjuntos maximales de Φ los cuales son consistentes con la revisión de la fórmula F:

$$W(\Phi, F) = \{\Phi' \subseteq \Phi :\sim (\Phi', F \models \bot) \ \& \ \sim \exists \gamma : \Phi' \subset \gamma \subseteq \Phi \ \& \ \sim (\gamma, F \models \bot)\}$$

Donde la notación $F_1 \models F_2$, significa que F_2 es "verdadera" para todo modelo de F_1 y $\bot$ es la fórmula distinguida como siempre "falsa".
$W(\Phi, F)$ contiene a todos los subconjuntos plausibles que podemos retener cuando insertamos F. Una de las operaciones más comunes de revisión de cambio de fórmula base es el método de Ginsberg donde:

$$\Phi^*_G F = \{\Phi' \cup F : \Phi' \in W(\Phi, F)\} \tag{3.1}$$

En el proceso de revisión, cuando se han determinado los conjuntos $\Phi^*_G F$ es usual tener que elegir con base en factores externos o bajo una orientación probabilista qué conjuntos $\Phi^*_G F$ son los más adecuados para ser descartados de F. En este trabajo de investigación se propone un método para determinar el valor lógico relativo de cada elemento de la fórmula F, y con base a éste valor se puede entonces determinar el conjunto $\Phi^*_G F$ que haría perder la mínima información al ser descartado de F.

3.2. Agentes Inteligentes

El nacimiento de la Inteligencia Artificial trajo consigo una gran diversidad de tecnologías, destacando entre ellas la representación del conocimiento, sistemas de razonamiento y sistemas de aprendizaje. A partir de los años 90's y con la implementación de las anteriores tecnologías marcaron el inicio de los *Agentes Inteligentes*. Actualmente, se puede apreciar cada vez más su uso en distintos campos no sólo en el científico, también con fines comerciales. El desafío, sigue siendo la confiabilidad para delegarles tareas y que la toma de decisiones sea la correcta. Para ello se requiere sistemas computacionales con razonamiento monotónico y no-monotónico. Este último, ha despertado gran interés debido a que se debe buscar el mecanismo óptimo para representar el conocimiento dinámico, siendo la revisión de creencias uno de los principales mecanismos para brindar razonamiento a un agente inteligente en un ambiente de incertidumbre [Vaz06].

Ejemplo 9. *Se envía una sonda a Marte con fines de recolectar material, una vez concretada su tarea regrese a la estación base e indique el éxito o fracaso de su misión. Cuando esta sonda está a punto de aterrizar en el planeta Marte, pierde la comunicación con la estación base. Debido a lo anterior debe por si sola tomar la decisión de si aterriza o regresa. De acuerdo a su base de conocimiento decide aterrizar, recolectar la muestra solicitada y regresar.*

En el ejemplo anterior se observa que un agente es un sistema computacional, que habita en un medio ambiente dinámico y complejo, sobre el cual puede actuar.

En el campo de la Inteligencia Artificial (IA), la siguiente definición propuesta por Wooldridge & Jennings en [WJ95], es de las más aceptadas.

Definición 1. *Sistemas computacionales que habitan en algún ambiente dinámico y complejo, pudiendo sentir ese entorno y actuar en consecuencia, tomando en cuenta el conjunto de objetivos y motivaciones que intenta conseguir a través de sus acciones (ver figura 3.3.)*

Para algunos autores como *Cherniak* en [Che86], consideran como punto de partida para describir la característica o propiedad mínima que debe contar un sistema computacional para que sea considerado agente es la *racionalidad* y, se debe cumplir aún cuando su racionalidad sea mínima [Tor06]. Para que un agente sea racional implica el decidir o elegir, lo que conlleva a los siguientes procesos: *deseos y creencias.*

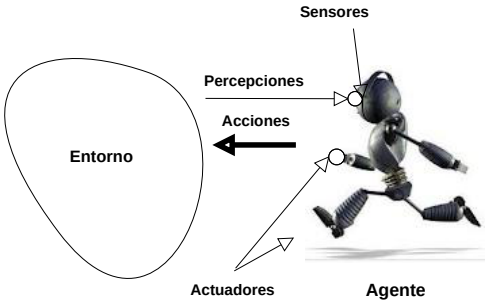


FIGURA 3.3: Visión esquemática de un Agente Inteligente.

3.2.1. Características de un Agente Inteligente

Autores como Franklin & Graesser [FG96] y, Wooldridge & Jennings [WJ95], señalan que las características básicas de un agente serán el ser racional, autónomo, reactivo, pro-activo y social, motivo por el cual sólo se describirán éstas. (ver fig. 3.4). Cabe destacar que el nivel de abstracción serán las características con que cuente ese agente.

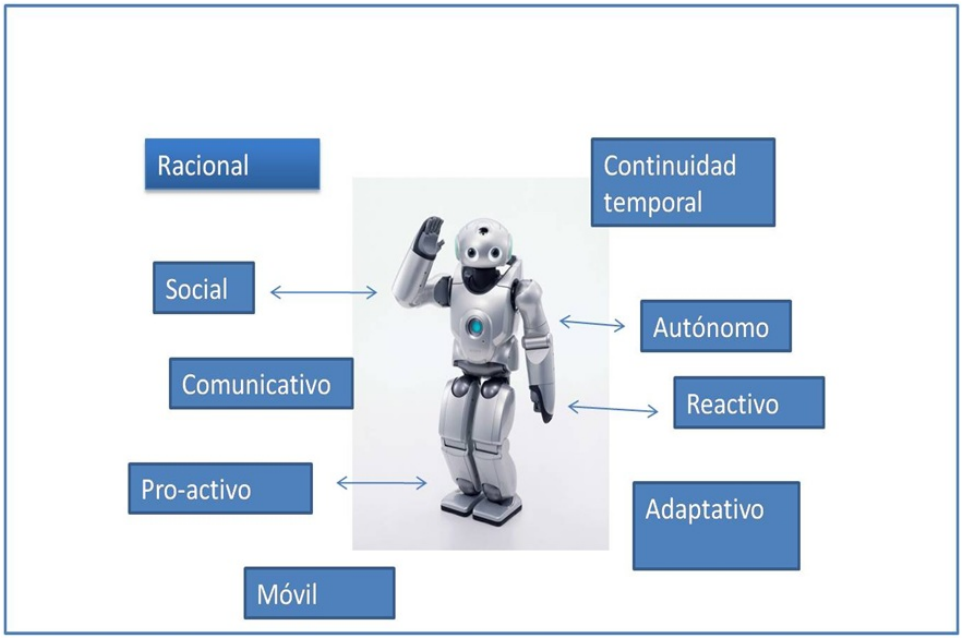


FIGURA 3.4: Características de un Agente.

Racional. Es la propiedad que posee un agente inteligente de tomar una decisión y que además sea correcta, con el objetivo de optimizar sus tareas para llegar a su tarea definida. Es decir, realize la tarea asignada y para ello tome la mejor opción con base en su conocimiento.

Autónomo. Es la capacidad que tiene un agente de tomar decisiones propias, sin necesidad de que alguien lo guie. Aún, cuando llegue a tomar una decisión errónea debe ser capaz de buscar una alternativa que lo lleve a su meta.

Reactivo. Se mencionaba que al ser autónomo un agente puede cometer errores, ello le permite alimentar su base de conocimiento y aprender de su experiencia, que en un futuro sus decisiones sean correctas, consistentes con su base de conocimiento y tenga una mejor visión de su medio ambiente.

Pro-activo. Un agente no sólo debe actuar ante eventos producidos por su ambiente, también debe de ser capaz de tomar la iniciativa para satisfacer sus objetivos.

Social. La capacidad de poder relacionarse y comunicarse con otros agentes.

Como ya se mencionó anteriormente, los agentes inteligentes han sido empleados en diversas ramas, un ejemplo con fines comerciales se puede observar en el *asistente virtual de Bankia*[3]. El cual es un robot con interfaz humana (ver figura 3.5), que está capacitado para comprender preguntas en lenguaje natural y contestar con la información requerida por el usuario, o dar alguna sugerencia. Si alguna pregunta no comprende se lo dirá. Se le realizaron las siguientes preguntas: 3.5 (a) ¿Cómo te llamas? y 3.5 (b) ¿Te gusta observar la luna?.

(a) Pregunta 1 (b) Pregunta 2

FIGURA 3.5: Asistente Virtual de Bankia

[3]http://asistente.bankia.es/ResponseJSP24/Bea.jsp

Capítulo 4

Operadores matriciales para contar modelos

4.1. Modelos y cargas en grafos y fórmulas

Aquí se introduce el concepto de *carga* de una variable relativa a una fórmula dada. De manera informal, la carga de una variable concentra información parcial sobre el número de modelos que satisfacen una subfórmula. Formalmente, la carga de una variable es un vector en donde la primera componente representa el número de asignaciones que satisfacen la fórmula donde la variable toma el valor de *uno* y la segunda componente representa el número de asignaciones que satisfacen la fórmula en donde la variable toma el valor de *cero*.

Ejemplo 10. *Si* $F = \{\{x, y\}, \{y, z\}\}$*, donde las cargas de las variables* x*,* y*,* z *son* $(3, 2)$*,* $(4, 1)$ *y* $(3, 2)$ *respectivamente, ver figura 4.1.*

También se presentan algunos resultados simples sobre modelos que serán de utilidad más adelante.

Para $i \in \mathbb{B}$, $R_F(x = i)$ denota el conjunto de variables diferentes de x eliminadas cuando F es simplificada a $F_{x=i}$, esto es

$$R_F(x = i) = Var(F) \setminus (\{x\} \cup Var(F_{x=i}))$$

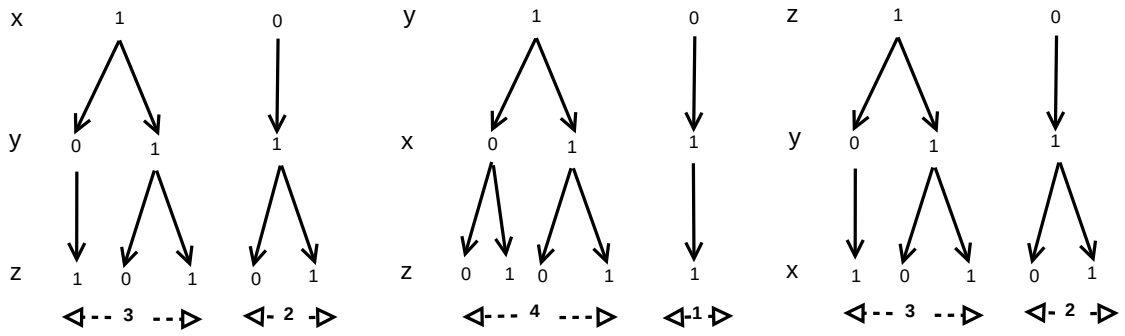


FIGURA 4.1: $F = \{\{x, y\}, \{y, z\}\}$

Por ejemplo, si $F = (x \vee y) \wedge (z \vee w)$ entonces

$$R_F(x = 0) = \{x, y, z, w\} \setminus (\{x\} \cup \{y, w, z\}) = \emptyset,$$

$$R_F(x = 1) = \{x, y, z, w\} \setminus (\{x\} \cup \{w, z\}) = \{y\}$$

Si F y F' son fórmulas tales que los conjuntos $Var(F)$ y $Var(F')$ son disjuntos o tienen como intersección a $\{x\}$, entonces se sigue que

$$R_{F \cup F'}(x = i) = R_F(x = i) \cup R_{F'}(x = i) \text{ y } R_F(x = i) \cap R_{F'}(x = i) = \emptyset \quad (4.1)$$

Dada una fórmula F e $i \in \mathbb{B}$, el número de modelos de F donde la variable x toma el valor de $i \in \mathbb{B}$ es dado por

$$|M_{x=i}(F)| = 2^{|R_F(x=i)|}|M(F_{x=i})| \quad (4.2)$$

ya que cada variable que desaparece puede tomar cualquier valor en $\mathbb{B}$. Por ejemplo, $F = \{(x \vee y \vee z) \wedge (u \vee v)\}$, entonces

$$|M_{x=1}(F)| = 2^2|M(u \vee v)| = 4 \cdot 3$$

Definición 2. *Si $F_1, ..., F_n$ son fórmulas disjuntas a pares, entonces es claro que*

$$|M(F_1 \cup F_2 \cup \cdots \cup F_n)| = |M(F_1)||M(F_2)| \cdots |M(F_n)| \quad (4.3)$$

Ejemplo 11. $|M(x \vee (y \vee z) \vee (u \vee v \vee w))| = |M(x)||M(y \vee z)||M(u \vee v \vee w))| = 2 \cdot 3 \cdot 5$

Definición 3. *Dada F una fórmula, $x \in Var(F)$, la* carga de x relativa a F, *es un par ordenado de enteros no negativos (m, n), en donde m y n son el número*

de modelos de F donde la variable x toma el valor de 0 *y* 1 *respectivamente, esto es*

$$m = |M_{x=1}(F)| \ , \ n = |M_{x=0}(F)|$$

esta pareja es denotada como $\#\boldsymbol{sat}(F, x)$.

Entonces es claro que dada cualquier variable x, la suma $m + n$ es el total de modelos de F, donde (m, n) es la carga de x relativa a F. El problema $\#sat$ se puede ver como el problema de hallar la carga de una variable de la fórmula dada. Note también que si se pueden obtener algunas cargas de las variables de una fórmula F se tiene información más precisa que la que proporciona el número $|M(F)|$, esto es, no sólo se conoce el número de modelos de la fórmula, sino también se sabe el número de modelos donde una determinada variable toma el valor 1 y el número de modelos donde esta misma variable toma el valor 0.
Si F' es una sub-fórmula de F y $x \in Var(F')$, entonces la carga $\#\mathbf{sat}(F', x)$ es llamada *carga parcial* de x en F' o la *carga inducida* por F' en x.

Ejemplo 12. *Si* $F = (x \vee y) \wedge (y \vee z)$ *y* $F' = (x \vee y)$*, entonces* F' *es una sub-fórmula de* F*, la carga de la variable* x *es* $(3, 2)$ *y su carga relativa a la sub-fórmula* F' *es* $(2, 1)$.

Con $\#sat_{y=i}(F, x)$ se representa la carga de x en la sub-fórmula $F_{y=i}$. Por ejemplo si $F = \{\{x, y\}\}$, de la tabla 4.1 se tiene $\#sat(F, x) = (1, 2)$, $\#sat(F, y) = (2, 1)$, $\#sat_{y=0}(F, x) = (0, 1)$ y $\#sat_{y=1}(F, x) = (1, 1)$.

x	y	F
1	1	1
1	0	1
0	1	1
0	0	0

Tabla 4.1: Tabla de Modelos de $F = \{\{x, y\}\}$

4.2. Operadores de Conteo

En esta sección se presenta la descripción de los operadores matriciales que conducen al conteo de modelos sobre fórmulas monótonas simples como cadenas.

Posteriormente este conjunto de operadores es extendido a operadores de conteo sobre ciclos simples [GLI05]. Estos operadores actúan sobre las cargas parciales de las variables y permiten ciertas reducciones en una fórmula dada.

Definición 4. *Sea el operador definido por*

$$(m_1, n_1) \diamond (m_2, n_2) = (m_1 m_2, n_1 n_2) \tag{4.4}$$

El operador "$\diamond$" es llamado producto de Hadamard.

El producto "$\diamond$" es asociativo, distributivo y conmutativo.
Se sabe que, si $F_1, ..., F_k$ son las distintas componentes conexas de F, entonces el número de modelos de F es el producto

$$|M(F_1)||M(F_2)| \cdots |M(F_k)|$$

Entonces para simplificar, se puede suponer que toda fórmula es conectada. La complejidad en tiempo de un procedimiento para una fórmula arbitraria no es afectada, ya que los procedimientos para determinar las componentes conexas pueden llevarse a cabo en tiempo lineal (pueden encontrarse algoritmos eficientes basados en búsquedas en profundidad o amplitud).

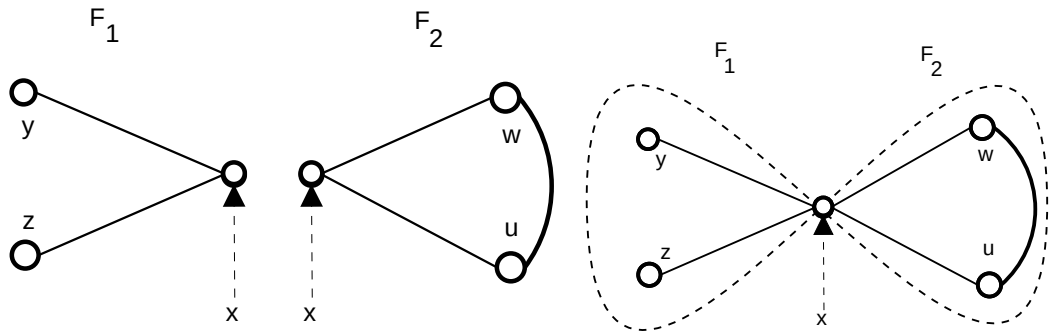
Proposición. Si la fórmula F es la unión de las sub-fórmulas F_1 y F_2, donde x es la única variable en común de estas sub-fórmulas, entonces el número de modelos de F es $(m_1, n_1) \diamond (m_2, n_2)$, donde (m_1, n_1) y (m_2, n_2) son las cargas de x relativas a F_1 y F_2 respectivamente. (Para la demostración ver ref. [GLI05]).

Ejemplo 13. *La fórmula de la figura 4.2 (c) está formada por las sub-fórmulas, F_1 (ver figura 4.2 (a)) y F_2 (ver figura 4.2 (b)). Donde,*

$$F_1 = \{\{x, y\}, \{x, z\}\}$$

$$F_2 = \{\{x, w\}, \{x, u\}\}$$

De la tabla 4.2 se tiene que F_1, es $\#sat(F_1, x) = (4, 1)$, y de F_2 es $\#sat(F_2, x) = (3, 1)$.



(a) F1 (b) F2 (c) $F1 \bigcup F2 = F$

FIGURA 4.2: Sub-fórmulas con una sola variable en común: x

x	y	z	F_1	x	w	u	F_2
1	1	1	1	1	1	1	1
1	1	0	1	1	1	0	1
1	0	1	1	1	0	1	1
1	0	0	1	1	0	0	0
0	1	1	1	0	1	1	1
0	1	0	0	0	1	0	0
0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0

TABLA 4.2: Tabla de Modelos de F_1 y F_2 de fig. 4.2 (a) y (b) respectivamente

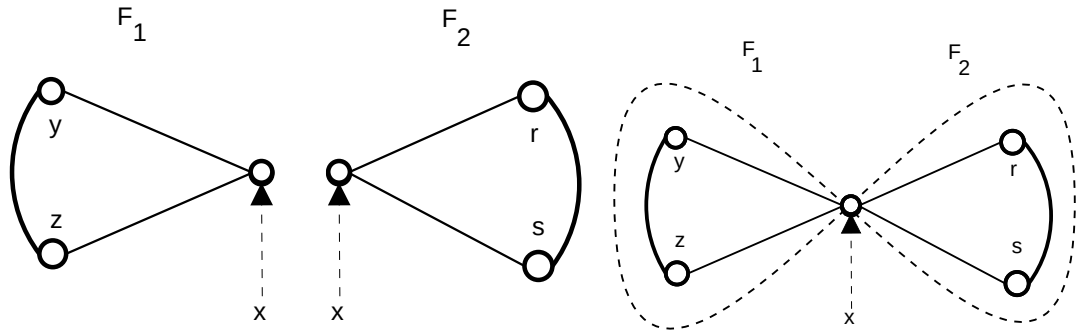
Ejemplo 14. *La fórmula de la figura 4.3 (c)* [1]*, está formada por las sub-fórmulas F_1 y F_2, como se observa en la figura 4.2 (a) y (b) respectivamente.*

$$F_1 = \{\{y, z\}, \{y, x\}, \{z, x\}\}$$

$$F_2 = \{\{x, r\}, \{r, s\}, \{x, s\}\}$$

Para F_1 de la tabla 4.3 se tiene $\#sat(F_1, x) = (3, 1)$, y F_2 de la tabla 4.4 $\#sat(F_2, x) = (3, 1)$. Entonces $\#sat(F_1, x) \diamond \#sat(F_2, x) = (3, 1) \diamond (3, 1) = (9, 1)$. Donde $(9, 1)$ indica el número de modelos de F (ver figura 4.3 (c)), se puede comprobar el mismo resultado llevando acabo todo el proceso, como se observa en la tabla 4.5.

[1]Los valores de la variable **x** que tienen el valor de 0 ó 1 y dan en F el valor 1, están en color verde en la tabla 4.3, 4.4, 4.5, no importando el valor de las otras literales que forman a F



(a) F1 (b) F2 (c) $F1 \bigcup F2 = F$

FIGURA 4.3: Sub-fórmulas F1 y F2 con una sola variable en común x

x	y	z	$\{y,z\}$	$\{y,x\}$	$\{z,x\}$	F_1
1	1	1	1	1	1	1
1	1	0	1	1	1	1
1	0	1	1	1	1	1
1	0	0	0	1	1	0
0	1	1	1	1	1	1
0	1	0	1	1	0	0
0	0	1	1	0	1	0
0	0	0	0	0	0	0

TABLA 4.3: Tabla de Modelos de F_1 de fig. 4.3

x	r	s	$\{x,r\}$	$\{r,s\}$	$\{x,s\}$	F_2
1	1	1	1	1	1	1
1	1	0	1	1	1	1
1	0	1	1	1	1	1
1	0	0	1	0	1	0
0	1	1	1	1	1	1
0	1	0	1	1	0	0
0	0	1	0	1	1	0
0	0	0	0	0	0	0

TABLA 4.4: Tabla de Modelos de F_2 de fig. 4.3

x	y	z	r	s	F	x	y	z	r	s	F
1	1	1	1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	0	1	1	1	0	0
1	1	1	0	1	1	0	1	1	0	1	0
1	1	1	0	0	0	0	1	1	0	0	0
1	1	0	1	1	1	0	1	0	1	1	0
1	1	0	1	0	1	0	1	0	1	0	0
1	1	0	0	1	1	0	1	0	0	1	0
1	1	0	0	0	0	0	1	0	0	0	0
1	0	1	1	1	1	0	0	1	1	1	0
1	0	1	1	0	1	0	0	1	1	0	0
1	0	1	0	1	1	0	0	1	0	1	0
1	0	1	0	0	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0	0	1	1	0
1	0	0	1	0	0	0	0	0	1	0	0
1	0	0	0	1	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0

TABLA 4.5: Tabla de Modelos de F de fig. 4.3

El operador π_+ es llamado *operador de proyección*, y formalmente está dado por la siguiente definición.

Definición 5. *(Operador de Proyección) El operador* $\pi_+ : \mathbb{N}^2 \to \mathbb{N}^2$ *se define como*

$$\pi_+ = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \tag{4.5}$$

el cual es llamado proyección positiva.

Definición 6. *(Operador Arista) El operador* $T_{++} : \mathbb{N}^2 \to \mathbb{N}^2$ *llamado* operador arista*, es definido como*

$$T_{++} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}. \tag{4.6}$$

Observe de la tabla , para $F = \{\{x, y\}\}$ las relaciones:

$$\#sat(F_2, y) = T_{++}(1,1),\ \#sat(F_2, x) = T_{++}(1,1) \tag{4.7}$$

Dada una fórmula F, siguiendo la idea previa la pregunta es si es posible encontrar relaciones entre $|M(F)|$ y los operadores $\pi_+ +$ y T_{++} (ec. 4.6). La respuesta, como se verá más adelante, es encontrada para cierta clase de fórmulas. En esta dirección, se consideran las siguientes fórmulas:

$$F_3 = \{\{x, y\}, \{y, z\}\}, F_4 = \{\{x, y\}, \{y, z\}\},$$

$$F_5 = \{\{x, y\}, \{y, z\}\}, F_6 = \{\{x, y\}, \{y, z\}\}.$$

Para estas fórmulas se obtiene fácilmente que:

$$\#\mathbf{sat}(F_3, z) = T_{++}T_{++}(1, 1) = (3, 2)$$

Si $F = \{(x_1, x_2), (x_2, x_3), ..., (x_n, x_{n+1})\}$, no es díficil comprobar también que

$$\#\mathbf{sat}(F, x_1) = T^n_{++}(1, 1)$$

4.3. Operadores matriciales para conteo sobre ciclos

En esta sección se hace un estudio análogo a la sección previa para obtener operadores que realicen el conteo de modelos sobre ciclos simples. La siguiente definición es de utilidad para dicho fin.

Definición 7. *(Operadores de Ciclo). Sea* $\Psi : \mathbb{N}^4 \to \mathbb{N}^4$ *el operador definido como sigue:*

$$\Psi \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & b \\ c & 0 \end{pmatrix}. \tag{4.8}$$

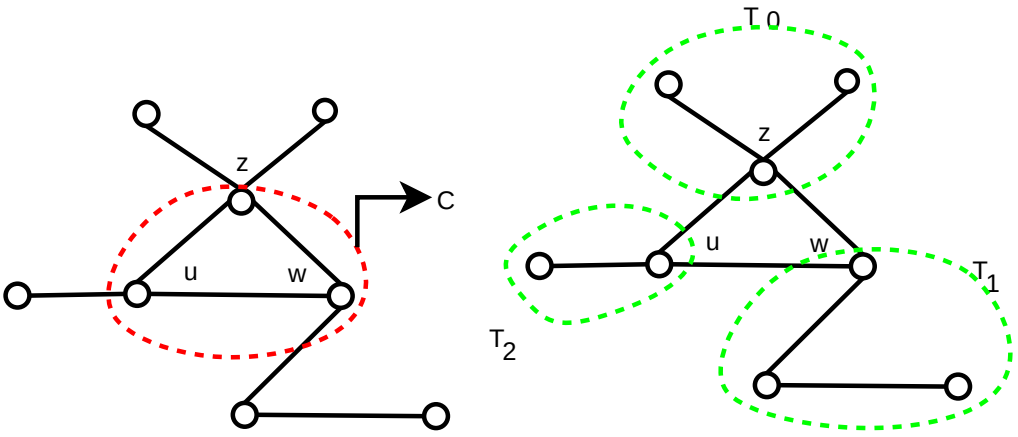
El siguiente teorema muestra como calcular el número de modelos para un ciclo simple de fórmulas con cargas conocidas.

Teorema 1. *Sea* F *un ciclo simple de fórmulas* $\Gamma_0, ..., \Gamma_m$*, conectadas por el ciclo* $\mathcal{C} = \{c_0, ..., c_m\}$ *en las variables* $w_0, ..., w_m$*, con cargas* $\boldsymbol{q}_0, ..., \boldsymbol{q}_m$ *respectivamente, entonces*

$$\#sat(F, w_m) = \Psi T' \boldsymbol{q}_0 \tag{4.9}$$

donde $T' = T'_{m-1}T'_{m-2}\cdots T'_1$ *y* $T'_i = \boldsymbol{q}_{i+1} \circledast T_i$, *para* $i \in [m-1]$.

Ejemplo 15. *La fórmula de la figura 4.4, donde el ciclo* $\mathcal{C} = \{u, w, z\}$ *y formado por tres sub-fórmulas* T_0, T_1 *y* T_2.



(a) Ciclo C (b) Sub-fórmulas

FIGURA 4.4: Sub-fórmulas T_0, T_1 y T_2 con tres variables en común: u, w y z

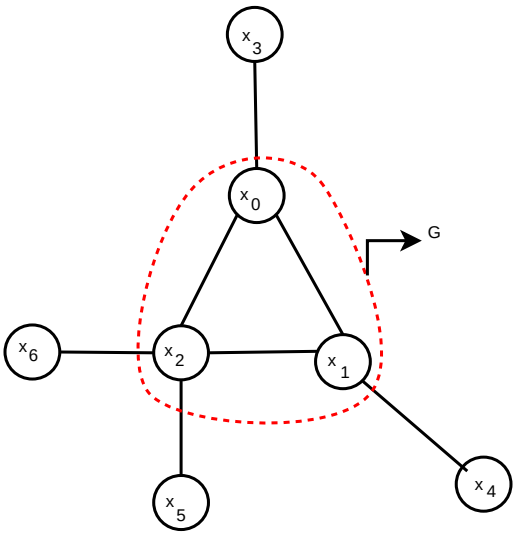
Ejemplo 16. *La fórmula de la figura 4.5 está formada por las sub-fórmulas,* $F = \underbrace{\{\{x_0, x_1\}, \{x_1, x_2\}, \{x_2, x_0\}}_{G}, \underbrace{\{x_0, x_3\}}_{F_0}, \underbrace{\{x_1, x_4\}}_{F_1}, \underbrace{\{x_2, x_5\}, \{x_2, x_6\}}_{F_2}\}$.

$$\#sat(F_0, x_0) = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} = q_0$$

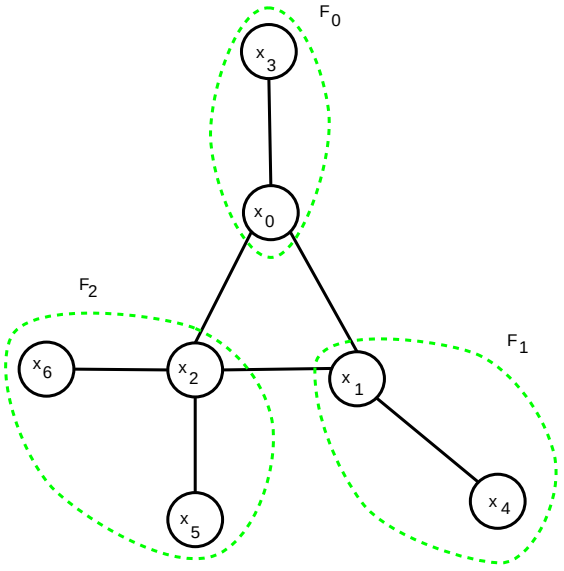
$$\#sat(F_1, x_1) = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} = q_1$$

$$\#sat(F_2, x_2) = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \diamond \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 2 \\ 1 \end{pmatrix} \diamond \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \end{pmatrix} = q_2$$



(a) Ciclo G formado por x_0, x_1 y x_2



(b) Sub-fórmulas F_0, F_1 y F_2

FIGURA 4.5: Sub-fórmulas F_0, F_1 y F_2 con tres variables en común: x_0, x_1 y x_2

Como se puede observar de $\#sat(F_0, x_0) = q_0$, $\#sat(F_1, x_1) = q_1$ y $\#sat(F_2, x_0) = q_2$, lo que arroja la contracción de la figura 4.5 (b) a 4.6.

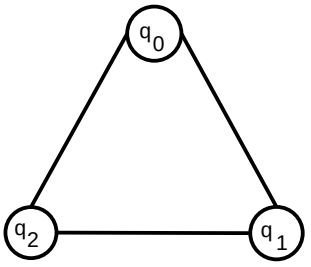


FIGURA 4.6: Sub-fórmula G

$$\#sat(F, x_2) = \Psi_{++} T' \mathbf{q_0} \qquad m = 2$$

$$T = T_1' T_0'$$

$$T_1' = q_2 \circledast T_{++}$$

$$T_1' = q_2 \circledast T_{++} = \begin{pmatrix} 4 \\ 1 \end{pmatrix} * \begin{pmatrix} 1\ 1 \\ 1\ 0 \end{pmatrix} = \begin{pmatrix} 4\ 4 \\ 1\ 0 \end{pmatrix}$$

$$T_0' = q_1 \circledast T_{++} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} * \begin{pmatrix} 1\ 1 \\ 1\ 0 \end{pmatrix} = \begin{pmatrix} 2\ 2 \\ 1\ 0 \end{pmatrix}$$

$$T' = \begin{pmatrix} 4\ 4 \\ 1\ 0 \end{pmatrix} * \begin{pmatrix} 2\ 2 \\ 1\ 0 \end{pmatrix} = \begin{pmatrix} 12\ 8 \\ 2\ 2 \end{pmatrix}$$

$$\Psi_{++} T' = \begin{pmatrix} 12\ 8 \\ 2\ 0 \end{pmatrix}$$

$$\#sat(F, x_2) = \Psi_{++} T' q_0 = \begin{pmatrix} 12\ 8 \\ 2\ 0 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 32 \\ 4 \end{pmatrix}$$

Recuerde del capítulo dos que el ciclo $\mathcal{C}$ puede ser expresado como $\mathcal{C} = P + w_m w_0$, donde P es la trayectoria $w_0 w_1, \cdots, w_{m-1} w_m$, $Var(c_i) = \{w_i, w_{i+1}\}$.

De los resultados previos y el teorema 1, podemos calcular el número de modelos de cadenas simples sobre ciclos de fórmulas con cargas conocidas y por el teorema 2 podemos contar el número de modelos sobre ciclos simples de fórmulas que a su vez son ciclos de fórmulas con cargas conocidas.

Corolario 1. *Sea F un ciclo simple de cláusulas unitarias $\Gamma_1 = \{(w_1)\}, ..., \Gamma_m = \{(w_m)\}$, entonces*

$$\#sat(F, w_m) = \Psi T' \boldsymbol{q}_1$$

donde $T' = T_{m-1}T_{m-2}\cdots T_1$ y $\boldsymbol{q}_1 = (1, 1)$.

Corolario 2. *Si F es un ciclo entonces*

$$\#sat(F, w_m) = \begin{pmatrix} F_{m+1} \\ F_{m-1} \end{pmatrix}$$

donde F_{m+1} y F_{m-1} son números de Fibonacci.

Si se desea consultar la demostración del corolario 1 y 2, ver ref [GLI05].

4.4. Algoritmo #sat_2MON

Los teoremas 1 y 2 nos conducen a un algoritmo eficiente, tal que, dada cualquier fórmula monótona en 2-FNC, ésta es reducida a una fórmula cargada equivalente. Cuando la fórmula es reducida a una variable cargada, entonces este algoritmo cuenta el número de modelos de la fórmula original.

El algoritmo que se presenta a continuación esta basado en el algoritmo presentado en [GLI05], con la diferencia de que ha sido ligeramente adaptado para fórmulas monótonas en 2-FNC y que utiliza los procedimientos llamados:

1. ***poda-ramas***
2. ***poda-ciclos***.

A su vez el procedimiento ***poda-ciclos*** hace uso del procedimiento

- ***Ciclo***.

4.4.1. Procedimiento *poda-ramas*

El procedimiento ***poda-ramas*** suprime las ramas del grafo de restricciones de una fórmula dada hasta que es reducido a un nodo o a un grafo sin nodos de grado uno. El efecto de este procedimiento sobre la fórmula es su reducción a una fórmula equivalente sin variables de grado uno. Cuando la fórmula es reducida a un nodo, su carga lleva el cómputo del número de modelos de la subfórmula de entrada.

A continuación el símbolo $F|_{deg=1}$ denota el conjunto de todas las cláusulas de F que contienen una variable de grado uno, esto es

$$F|_{deg=1} = \{c \in F | \exists x \in c : d(x) = 1\}.$$

El número $\delta(F) = min\{d(x) : x \in Var(F)\}$ es el grado mínimo de la fórmula F.

Procedimiento *poda_ramas*(F)

Entrada: F una fórmula, $\mathbf{q}_0, ..., \mathbf{q}_{|Var(F)|-1}$ las cargas de sus variables.

Salida: F'' donde $\delta(F'') \geq 2$ o $F'' = \{\{v\}\}$ y $\mathbf{q}_v$

B1) **While** $F \neq \emptyset$ or $\delta(F) = 1$ **do**
B2) $F' = F|_{deg=1}$
B3) **While** $F' \neq \emptyset$ **do**
B4) **Let** $\{l, r\} \in F'$
B5) **If** $d(l) \neq 1$ **then**
B6) $l \leftrightarrow r$
B7) **endIf**
B8) $\mathbf{q}_r = T_{++}\mathbf{q}_l \diamond \mathbf{q}_r$
B9) $d(r) = d(r) - 1$
B10) $F' = F' \setminus \{\{l, r\}\}$
B11) $F = F \setminus \{\{l, r\}\}$
B12) **endWhile**
B13) **endWhile**
B14) **Return** $F,\ \mathbf{q}_r$

La complejidad en tiempo de este algoritmo es $O(m^2)$, donde m es el número

de cláusulas de F, ya que la determinación de $\delta(F)$ puede ser realizada en tiempo $O(m)$ y el bucle del while interno toma tiempo de orden $O(m)$.

4.4.2. Procedimiento *Ciclo*

El siguiente procedimiento cuenta la carga de un ciclo simple sobre una variable elegida en el ciclo. En lo que sigue una cláusula unitaria $c = \{l\}$ es considerada como un ciclo simple y es representada por $\{l, l\}$. Si x es una variable en una cláusula unitaria, entonces $d(x) = 2$.

La matriz identidad (en este caso de tamaño 2×2) es denotada por id.

Procedimiento *Ciclo*($\mathcal{C}, w$)

Entrada: $\mathcal{C}$ ciclo, $w \in Var(\mathcal{C})$, $\mathbf{q}_0, ..., \mathbf{q}_{|Var(\mathcal{C})|-1}$ las cargas de sus variables.

Salida: $\#sat(\mathcal{C}, w)$

c1) $T = id$
c2) $\alpha = w$
c3) **While** $| \mathcal{C} | \neq 1$ **do**
c4) **Let** $\{\ell, r\} \in \mathcal{C}$ such that $\alpha \in Var\{\ell, r\}$
c5) **If** $\nu(\ell) \neq \alpha$ **then**
c6) $\ell \leftrightarrow r$
c7) **endIf**
c8) $T = (\mathbf{q}_r \circledast T_{++})T$
c9) $\mathcal{C} = \mathcal{C} \setminus \{\{\ell, r\}\}$
c10) $\alpha = \nu(r)$
c11) **endWhile**
c12) **If** $\nu(r) \neq w$ **then**
c13) $l \leftrightarrow r$
c14) **endIf**
c15) $\mathbf{q}_w = \Psi T \mathbf{q}_w$
c16) **Return** $\mathbf{q}_w$

Para un ciclo de m cláusulas, el procedimiento retorna la carga sobre una sola

variable del ciclo en complejidad en tiempo $O(m^2)$ ya una cláusula que contiene la variable α puede ser encontrada en un ciclo de m cláusulas en tiempo lineal $O(m)$ (línea c4).

4.4.3. Procedimiento *poda_ciclos*

Por un *ciclo-hoja* se entiende una fórmula la cual es un ciclo en el que todo nodo, a excepción posiblemente de uno, es de grado 2. El conjunto de ciclos hoja de una fórmula F se denota por $Cleaf(F)$.

Para una fórmula F y su grafo de restricciones G_F, el procedimiento *poda_ciclos* suprime los ciclos hoja de G_F, hasta reducir G_F a un nodo o a un grafo sin ciclos-hoja. En la fórmula, esto significa una reducción a una fórmula cargada equivalente. Si G_F es reducido a un nodo, la carga de este único nodo nos da el número de modelos de F.

Procedimiento *poda_ciclos*(F)

Entrada: F fórmula, $\mathbf{q}_0, ..., \mathbf{q}_{|Var(F)|-1}$ las cargas de sus variables.

Salida: F'' donde $Cleaf(F'') = \emptyset$

C1) **While** $F \neq \emptyset$ and $Cleaf(F) \neq \emptyset$ **do**
C2) $CF = Cleaf(F)$
C3) **While** $CF \neq \emptyset$ **do**
C4) **Let** $(C, w) \in CF$
C5) $\mathbf{q}_w = Ciclo(C, w)$
C6) $d(w) = d(w) - 2$
C7) $CF = CF \setminus \{C\}$
C8) $F = F \setminus C$
C9) **endWhile**
C10) **endWhile**
C11) **Return** F

Es claro de la líneas C2, C3 y C9 que el procedimiento *poda_ciclos* tiene complejidad en tiempo de orden $O(m^4)$.

4.4.4. Algoritmo #sat2MON

Dada una fórmula monótona en 2FNC, el algoritmo #sat2MON utiliza los procedimientos *poda-ciclos* y *poda-ramas* para reducir su grafo asociado a un nodo o a un grafo simplificado sin ciclos-hoja y sin ramas.

En este algoritmo las cargas de las variables relativas a la fórmula de entrada se toman inicialmente como $(1, 1)$ y se actualizan en cada llamada a los procedimientos.

Algoritmo *#sat_2FNC*

Entrada: Fórmula monótona F en 2-FNC, **Salida:** F'', $\mathbf{q}''_i$, $i \in [|Var(F'')| - 1]$, donde $Cleaf(F'') = \emptyset$ y $F''|_{deg1} = \emptyset$

S0) $\mathbf{q}_i = (1, 1), i \in [|Var(F)| - 1]$
S1) $CF = \{C : C \; es \; ciclo \; de \; F\}$
S2) $d(x) = grado \; de \; x, \forall x \in Var(F)$
S3) **While** $F \neq \emptyset$ and $(Cleaf(F) \neq \emptyset$ or $F|_{deg1} \neq \emptyset)$
S4) $\quad F = poda - ramas(F)$
S5) $\quad F = poda - ciclos(F)$
S6) **endWhile**
S7) **Return** $\quad F, \; \mathbf{q}_w$

Se sabe que dada F una fórmula con m cláusulas y n variables, la complejidad en tiempo de un algoritmo para generar el conjunto de ciclos-fundamentales de G_F es de complejidad en tiempo $O(n + m)$ en consecuencia la complejidad de los procedimientos anteriores implica que $\#sat_2MON$ tiene una complejidad en tiempo polinomial.

Por ejemplo,

- **1- S1.** $CF_1 = \{C_1, C_2, C_3, C_4, C_5\}$, donde
  $C_1 = \{\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_1\}\}, C_2 = \{\{x_2, x_2\}\}, C_3 = \{\{x_3, x_4\}, \{x_4, x_5\}, \{x_5, x_3\}\}$,
  $C_4 = \{\{x_4\}\}$, $C_5 = \{\{x_8, x_9\}, \{x_9, x_{10}\}, \{x_{10}, x_8\}\}$.

- **2- S2.** Los grados de cada uno de los nodos son:
  $d(x_i) = 1$ para $i = 7, 11$, $d(x_i) = 2$ para $i = 1, 9$, $d(x_i) = 3$ para $i = 5, 6, 8, 10$ y $d(x_i) = 4$ para $i = 2, 3, 4$.

- **3- S3-S4.** Entonces, poda-ramas retorna

  $$F_1 = F_1 \setminus \{\{x_{10}, x_{11}\}, \{x_6, x_7\}\}$$

  actualizando los grados y las cargas de las variables x_6 y x_{10} (líneas B8 y B9 del procedimiento poda-ramas) como sigue: $\mathbf{q}_6 = T_{++}\mathbf{q}_7 \diamond \mathbf{q}_6 = (2, 1)$, $d(x_6) = 3 - 1 = 2$, $\mathbf{q}_{10} = T_{++}\mathbf{q}_{11} \diamond \mathbf{q}_{10} = (2, 1)$, $d(x_{10}) = 3 - 1 = 2$.

- **4- S3-S5.** Aquí el procedimiento poda_ciclos es aplicado. Describimos cada paso de este procedimiento, como sigue:

  - **5- C1-C2.** $Cleaf(F_1) = \{C_2, C_4, C_5\}$.
  - **6- C3-C9.** $\mathbf{q}_2 = Ciclo(C_2, x_2) = (1, 0)$, $d(x_2) = 4 - 2 = 2$, $\mathbf{q}_4 = Ciclo(C_4, x_4) = (0, 1)$, $d(x_4) = 4 - 2 = 2$, $\mathbf{q}_8 = Ciclo(C_5, x_8) = (6, 1)$, $d(x_8) = 3 - 2 = 1$ y $F_1 = F_1 \setminus (C_2 \cup C_4 \cup C_5)$.
  - **7- C2.** $Cleaf(F_1) = \{C_1\}$.
  - **8- C3-C9.** $\mathbf{q}_3 = Ciclo(C_1, x_3) = (2, 0)$, $d(x_3) = 4-2 = 2$ y $F = F_1 \setminus C_1$.
  - **9- C2.** $Cleaf(F_1) = \{C_3\}$.
  - **10- C3-C9.** $\mathbf{q}_5 = Ciclo(C_3, x_5) = (2, 0)$, $d(x_5) = 3 - 2 = 1$ y $F_1 = F_1 \setminus C_3$. Entonces obtenemos $poda\_ciclos(F_1) = \{\{x_5, x_6\}, \{x_6, x_8\}\}$.

- **11-Pasos S3-S4.** Finalmente, $poda - ramas(F_1) = (28, 10)$, ya que de las líneas 8-9, tenemos $\mathbf{q}_6 = T_{++}\mathbf{q}_5 \diamond \mathbf{q}_6 = (2, 2) \diamond (2, 1) = (4, 2)$, $\mathbf{q}_6 = T_{++}\mathbf{q}_8 \diamond \mathbf{q}_6 = (7, 5) \diamond (4, 2) = (4, 2) = (28, 10)$. Por lo tanto $\#sat(F_1) = 38$.

Capítulo 5

Cómputo eficiente del grado de creencia

Aquí se utiliza el método basado en los modelos de una fórmula booleana monótona para realizar razonamiento y se trata el proceso de razonamiento como el cálculo del grado de creencia [GLI05].

Sea KB una base de conocimiento definida por una fórmula monótona F en 2-FNC, sabemos del capítulo 2 que esta se representa mediante un grafo, donde cada nodo tiene asociada una carga. Con esta representación de la fórmula F, se puede calcular el grado de creencia en una variable o en una cláusula binaria F eficientemente. También se muestra una forma de actualizar la representación de la base de conocimiento.

5.1. Representación de una base de conocimiento

Como en el capítulo 2, consideremos una base de conocimiento KB representada mediante una fórmula monótona F en 2-FNC y consideramos su grafo asociado $G_F = (V, E)$, con $V = Var(F)$ y $E = \{Var(c) : c \in F\}$.

Supongamos G_F es conexo, $|V| = n$ y $|E| = m$. Sea $v_r \in V$ un nodo de grado mínimo en G_F el cual es elegido como el inicio de una búsqueda a profundidad. Sea T_G el árbol abarcador obtenido con v_r como nodo raíz y sea $\mathcal{C} = \{C_1, ..., C_k\}$ un

conjunto de ciclos fundamentales, donde cada arista de retroceso $c_i \in E$ delimita el comienzo y el final de un ciclo fundamental.

Dado cualquier par de ciclos C_i y C_j de $\mathcal{C}$, $i \neq j$, si C_i y C_j comparten aristas, se dice que los ciclos se *intersectan*, en cualquier otro caso serán llamados ciclos independientes.

Sea A_G el grafo generado por la búsqueda a profundidad sobre G_F. Entonces A_G está formado por el árbol abarcador T_G y el conjunto de ciclos fundamentales $\mathcal{C}$.

El grafo A_G es trasladado en un grafo D_G acíclico dirigido (DAG), asignando una orientación a cada arista $\{u, v\}$ en A_G, como de u a v $(u \rightarrow v)$ si v es un nodo ancestro de u en T_G.

Se aplica el procedimiento de ordenamiento topológico sobre D_G, obteniendo un ordenamiento y un número de orden "o" en D_G tal que $o(u) < o(v)$ si $u \rightarrow v$. Este número de orden indica el orden para el procesamiento de los nodos en D_G cuando se calcula el número de modelos $M(F)$ en la siguiente sección.

Ejemplo. Consideremos el grafo de la figura 5.1,

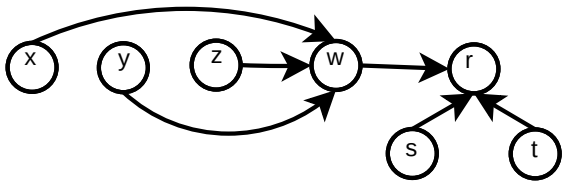


FIGURA 5.1: Grafo D_G.

Su diagrama de Hasse esta dado por la figura 5.2

En cosecuencia tenemos que su orden topológico es:

$$r < s < t < w < x < y < z$$

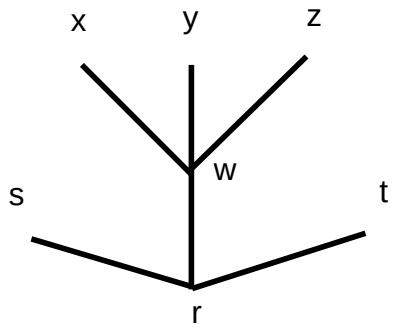


FIGURA 5.2: Diagrama de Hasse de D_G.

5.2. Cómputo de modelos sobre bases de conocimiento

En lo que sigue se muestra de manera general cómo podemos realizar el conteo sobre estructuras más generales del grafo D_G. Cabe mencionar que el costo computacional puede pasar la frontera entre lo tratable y lo intratable dependiendo de la estructura del grafo asociado a la fórmula monótona.

Recordemos que dada F una fórmula monótona en 2-FNC, si G_F es una cadena simple podemos calcular el número de modelos de F mediante el operador

$$T_{++} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}. \tag{5.1}$$

Por ejemplo, si $F = \{\{x, y\}, \{y, z\}, \{z, w\}\}$ (ver figura 5.3), entonces el operador arista T_{++}, se utiliza iniciando con el vector $(1, 1)$ para calcular en dos líneas el número de modelos de F. En la primera línea se calcula de forma incremental el número de modelos de F que hacen a la variable x verdadera ($x = 1$), y en la segunda línea el número de modelos de F donde x es falsa ($x = 0$), de aquí que la suma de las entradas del vector resultante es el número de modelos de F. Para la cadena dada previamente, tenemos

$$(1, 1)\underline{T_{++}} \rightarrow (2, 1)\underline{T_{++}} \rightarrow (3, 2)\underline{T_{++}} \rightarrow (5, 3)$$

esto es $\#SAT(F) = 8$.

Cuando G_F es una ciclo simple, aplicamos el operador ciclo Ψ.

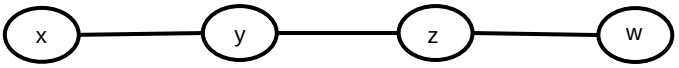


FIGURA 5.3: Cadena G_F.

$$\Psi \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & b \\ c & 0 \end{pmatrix}. \tag{5.2}$$

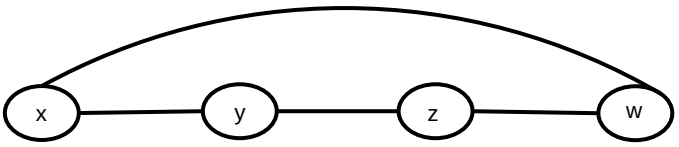


FIGURA 5.4: Ciclo G_F.

Para contar modelos de ciclos simples, se consideran matrices 2×2, el operador T_{++} es aplicado iterativamente sobre cada columna y el operador Ψ es utilizado de acuerdo a los signos de las literales involucradas en el arco que cierra el ciclo. Entonces, la suma de las entradas de la matriz resultante es el número de modelos de F. Por ejemplo, si $F = \{\{x, y\}, \{y, z\}, \{z, w\}, \{x, w\}\}$(ver figura 5.4), entonces:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \underline{T_{++}} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \underline{T_{++}} \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \underline{T_{++}} \begin{pmatrix} 3 & 2 \\ 2 & 1 \end{pmatrix} \underline{\Psi} \begin{pmatrix} 3 & 2 \\ 2 & 0 \end{pmatrix}$$

Por tanto, F tiene 7 modelos.

En el caso de que G_F sea un árbol, calculamos $\#SAT(F)$ usando el operador T_{++} en combinación con otro operador llamado *operador de salto* en [GLI05], definido por

$$J = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{5.3}$$

Ilustramos el uso combinado de estos operadores mediante el ejemplo siguiente. Sea $F = \{\{x, y\}, \{y, z\}, \{z, w\}, \{y, u\}, \{z, v\}\}$, G_F es un árbol (ver figura 5.5).

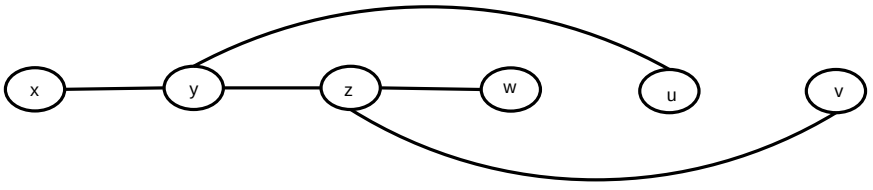


FIGURA 5.5: Árbol G_F.

Cualquier árbol puede ser ordenado de tal manera que sus nodos se encuentren en una disposición lineal y ligados con sus respectivos arcos.

Para realizar el cómputo del número de modelos de una fórmula monótona se sigue de manera similar al procesamiento de ciclos:

- Se recorre el grafo en el orden lineal establecido.
- Cuando nos encontramos un nodo donde uno o más arcos inician se etiquetan con el o los nodos de llegada.
- La matriz correspondiente al cálculo siguiente es transformada en una matriz con una columna adicional por cada valor diferente de cero en la segunda fila de la matriz inicial.

Para ilustrar lo anterior veamos el siguiente:

Ejemplo 17. *Sea antes*

$$F = \{\{x, y\}, \{y, z\}, \{x, z\}, \{x, w\}, \{y, u\}\}$$

siguiendo el procedimiento previo tenemos:

$$\begin{pmatrix} 1 \\ 1_{zw} \end{pmatrix} \underline{_{T_{++}}} \begin{pmatrix} 1 & 0 \\ 0 & 1_{zw} \end{pmatrix} \underline{_{T_{++}}} \begin{pmatrix} 1 & 1_{zw} \\ 1 & 0 \end{pmatrix} \underline{_{A_y}} \begin{pmatrix} 1 & 0 & 1_{zw} \\ 1 & 1 & 0 \end{pmatrix} \underline{_{T_{++}}} \begin{pmatrix} 1 & 1 & 1_{zw} \\ 1 & 0 & 1_{zw} \end{pmatrix}$$

$$\underline{_{\Psi}} \begin{pmatrix} 1 & 1_u & 1_w \\ 1 & 0 & 0 \end{pmatrix} \underline{_{J}} \begin{pmatrix} 2 & 1 & 1_w \\ 2 & 1_u & 1_w \end{pmatrix} \underline{_{\Psi}} \begin{pmatrix} 2 & 1 & 1 \\ 2 & 1_u & 0 \end{pmatrix} \underline{_{J}} \begin{pmatrix} 4 & 2 & 1 \\ 4 & 2_u & 1 \end{pmatrix} \underline{_{\Psi}} \begin{pmatrix} 4 & 2 & 1 \\ 4 & 0 & 1 \end{pmatrix}$$

De donde se sigue que la fórmula F tiene 12 modelos.

5.3. Cómputo del grado de creencia

Supongamos que se asigna el mismo grado de creencia a todas las "situaciones básicas" que aparecen en una base de conocimiento dada a través de F, una

fórmula monótona de n variables para un agente, entonces la probabilidad de que F sea satisfactible es

$$P_F = Prob(F \equiv T) = \frac{|M(F)|}{2^n}$$

donde T es establecido para el valor *verdadero* y $Prob$ es usado para denotar la probabilidad.

La probabilidad condicional de F_1 con respecto a F, denotada por $P_{F_1|F}$ y calculada como:

$$P_{F_1|F} = Prob(F \wedge F_1 \equiv T \mid F \equiv T) = \frac{|M(F_1 \wedge F)|}{|M(F)|}$$

Si se supone que la base de conocimiento F es una fórmula monótona en 2-FNC, entonces es claro que es satisfactible, de donde

$$|M(F)| > 0 \ y \ P_{F_1|F} = \frac{|M(F_1 \wedge F)|}{|M(F)|}$$

De donde la complejidad del cómputo del grado de creencia, $P_{F_1|F}$ se relaciona polinomialmente con la complejidad del cómputo del número de modelos para $F \wedge F_1$, la cual dada las relaciones anteriores, esto se reduce al problema #SAT.

En esta sección se muestra cómo el cálculo del grado de creencia $P_{F_1|F}$ puede realizarse eficientemente para el caso especial en el que nuestra base de conocimiento se encuentre dada por una fórmula monótona en 2-FNC.

Consideremos entonces que F una KB monótona en 2-FNC y F_1 una cláusula unitaria o binaria, en este caso es claro que se mantiene la estructura inicial de F sin pérdida de estructura al realizar actualizaciones. Se puede suponer que la carga (m, n) asociada con cada variable x_i de F, está almacenada en un arreglo: $varpar$.

Tomemos en consideración los siguientes casos:

Caso 1. $F_1 = \{(x)\}$

- $x \in Var(F)$. Como todas las variables de F tienen asociada su carga respectiva (m_i, n_i), se usa x como un apuntador en el arreglo $varpar$ para recuperar

la carga (m_x, n_x) y entonces

$$P_{F_1|F} = m_x/|M(F)|$$

- $x \notin Var(F)$. En tal caso ha llegado información nueva no considerada antes. Por lo tanto el espacio de probabilidad para calcular la probabilidad condicional $P_{F_1|F}$ debe ser ampliado.

Si consideramos un caso más general, donde $F_1 = \{\{x_i\} : i = 1, ..., k\}$ con cláusulas unitarias tal que toda variable de la fórmula F_1 no aparece en $Var(F)$ y $n = |Var(F)|$, entonces hay 2^n asignaciones sobre $Var(F)$ y 2^{n+k} asignaciones definidas sobre $Var(F) \cup Var(F_1)$, entonces se actualiza el dominio del espacio de probabilidad sobre el cual se calcula $P_{F_1|F}$ como:

$$P_{F_1|F} = \frac{Prob(F_1 \wedge F \equiv T)}{Prob(F \equiv T)} = \frac{\frac{|M(F_1 \wedge F)|}{2^{n+k}}}{\frac{|M(F)|}{2^n}} = \frac{|M(F_1 \wedge F)|}{2^k |M(F)|}$$

Puesto que los grafos asociados G_F y G_{F_1} corresponden a dos componentes conexas independientes y

$$|M(F_1)| = \prod_{i=1}^{k} |M(\{x\})| = 1,$$

entonces

$$P_{F_1|F} = \frac{|M(F_1)||M(F)|}{2^k |M(F)|} = \frac{1}{2^k} \tag{5.4}$$

En el caso particular de que la formula F_1 consista de una sola variable, esto es $k = 1$, un agente A cree en la nueva información no relacionada con su base de conocimientos con una fiabilidad de un 50 %.

En el primer caso, debido a que $P_{x|F}$ es calculado vía dos accesos al arreglo *varpar*, una comparación y una división, entonces se tiene una complejidad del orden $O(1)$. En general, el tiempo máximo para computar $P_{x|F}$ es consumido por la determinación de la posición de la variable x en el arreglo *varpar*.

Suponga que un agente A debe tomar una acción de acuerdo con el conjunto de opciones $Q = \{x_1, \ldots, x_k\}$. El método de deducción clásico propone revisar la satisfactibilidad de $(F \cup x)$ para cada $x \in Q$, reconociendo igual valor para toda

literal satisfactible con F, sin embargo la probabilidad condicional $P_{x|F}$ da mucho más información que sólo decir que $(F \cup x)$ es satisfactible. La probabilidad $P_{x|F}$ nos da la proporción de los modelos originales de F los cuales continúan siendo modelos para $(F \cup x)$. Esta clase de información es crucial cuando el agente A tiene que tomar una acción dependiendo del valor estratégico de cada alternativa $x \in Q$. Considere ahora el caso $Q = \{c_1, \ldots, c_k\}$ donde cada c_i es una cláusula binaria . Sea $c = \{x, y\}$ cualquier cláusula de Q, entonces hay cuatro casos para el cálculo de $P_{c|F}$:

1. $x \notin F$ y $y \notin F$: Existen tres modelos sobre las cuatro asignaciones en $Var(c)$ y puesto que los grafos G_F y G_c son independientes, entonces

$$P_{c|F} = \frac{|M(F)| \cdot |M(c)|}{|M(F)| \cdot 2^2} = 3/4.$$

   Como se ha extendido el espacio de probabilidad con dos nuevas variables: x y y este caso es calculado en tiempo $O(1)$.

2. $x \in Var(F)$ y $y \notin Var(F)$: Entonces x es buscado sobre el arreglo *varpar* para recuperar (m_x, n_x) también se tiene que

$$|M(F \wedge c)| = 2 \cdot m_x + n_x$$

   Entonces;

$$P_{c|F} = \frac{2 \cdot m_x + n_x}{|M(F \cup c)|}$$

   En este caso se tiene una complejidad en tiempo logarítmico sobre $| Var(F) |$ porque éste depende principalmente de la recuperación de la carga (m_x, n_x) del arreglo de variables.

3. $x, y \in Var(F)$ y $c \in F$: Como c ha sido ya calculado en $|M(F)|$, $|M(F \wedge c)| = |M(F)|$ entonces $P_{c|F} = 1$. En tal situación se obtiene el máximo valor posible para $P_{c|F}$, así que cualquier alternativa de acción del agente tomaría esta opción.

4. $x, y \in Var(F)$y $c \notin F$: Consideremos en esta opción una situación más general. Sea $F_1 = \{x_1 \vee \cdots \vee x_k\}$ una cláusula con k variables y sea $V = \{x \in F_1 : x \notin Var(F)\}$ y sea $F' = F_1 - V$ las variables en F_1 que no aparecen

en F, sea $t =| V |$. Se calcula $|M(F_1 \wedge F)|$ extendiendo los modelos de F con las nuevas variables V y eliminando de este espacio aquellas asignaciones extendidas las cuales no satisfacen a $(F_1 \wedge F)$. Esto es

$$P_{F_1|F} = \frac{|M(F \wedge F_1)|}{2^t |M(F)|}$$

Ejemplo 18. *Sea* $F = \{\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}, \{x_4, x_5\}, \{x_1, x_4\}, \{x_3, x_5\}, \{x_2, x_6\}\}$ *y la cláusula* $F_1 = c = \{x_2, x_5\}$. *Calculemos* $P_{c|F}$. *Siguiendo el procedimiento 4 de la sección anterior se obtiene:*

$$\begin{pmatrix} 1 \\ 1_3 \end{pmatrix} \underline{_{T_{++}}} \begin{pmatrix} 1 & 0 \\ 0 & 1_4 \end{pmatrix} \underline{_{T_{++}}} \begin{pmatrix} 1 & 1_4 \\ 1 & 0 \end{pmatrix} \underline{_{T_{++}}} \begin{pmatrix} 2 & 1_4 \\ 1_5 & 1_4 \end{pmatrix} \underline{_{abre}} \begin{pmatrix} 2 & 0 & 1_4 & 0 \\ 0 & 1_5 & 0 & 1_{45} \end{pmatrix}$$

$$\underline{_{T_{++}}} \begin{pmatrix} 2 & 1 & 1 & 1_5 \\ 2 & 0 & 1_4 & 0 \end{pmatrix} \underline{_{\Psi}} \begin{pmatrix} 2 & 1_5 & 1 & 1_5 \\ 2 & 0 & 0 & 0 \end{pmatrix} \underline{_{T_{++}}} \begin{pmatrix} 4 & 1 & 1 & 1 \\ 2 & 1_5 & 1 & 1_5 \end{pmatrix} \underline{_{\Psi}} \begin{pmatrix} 4 & 1 & 1 & 1 \\ 2 & 0 & 1 & 0 \end{pmatrix}$$

De donde $|M(F)| = 10$. Es claro que $t = 0$ ya que $V = \emptyset$ y también que $|M(c \wedge F)| = 9$ porque la carga recuperada de x_2 y x_5 es $(7, 3)$ y solo hay una asignación donde ambas son nulas. Entonces de la fórmula del procedimiento 4 se tiene que $P_{c|F} = 9/(2^0)(10) = 0{,}9$.

Ejemplo 19. *Si* F *es como en el ejemplo anterior y* $F_1 = \{\{x_1, x_2\}\}$, *entonces* $P_{c|F} = 1$.

Ejemplo 20. *Tomando* F *como en los ejemplos anteriores y* $F_1 = \{\{x_5, x_7\}\}$, *también* $P_{c|F} = 1$. *Ya que en este caso* $m_5 = 7$ *y* $n_5 = 3$, *de donde*

$$P_{c|F} = \frac{2 \cdot m_x + n_x}{|M(F \cup c)|} = (14 + 3)/17$$

.

Ejemplo 21. *Por último es claro que si* F *es como antes y* $c = \{\{x_7, x_8\}\}$, *entonces*

$$P_{c|F} = \frac{10 * 3}{10 * 4} = 3/4$$

.

Capítulo 6

Conclusiones y trabajo futuro

En la presente tesis se han presentado condiciones sobre fórmulas Booleanas para que el cómputo del número de modelos pueda ser realizado eficientemente. Las condiciones impuestas se establecen sobre su sintáxis y sobre la estructura de su grafo asociado. Las condiciones de sintaxis son la monotonía y la 2-FNC, así como los grafo-estructurales corresponden a árboles, ciclos simples y combinaciones de estas que no formen intersección de ciclos. El cómputo del número de modelos de este tipo de fórmulas ha sido posible mediante la aplicación de los operadores matriciales como el de la matriz de Fibonacci, los operadores proyección y el operador de ciclos. Basado en estos operadores se presenta un algoritmo eficiente para el cómputo de $\#SAT$ para dichas fórmulas.

Dada una base de conocimientos modelada a través de una fórmula Booleana con las restricciones impuestas anteriormente se ha podido presentar un método que nos permite computar eficientemente el grado de creencia $P_{F_1|F}$, cuando F_1 es un *query* compuesto por literales o por una cláusula monótona binaria, la cual incluye tanto variables utilizadas, no utilizadas y alternadas en f. En trabajos futuros se pueden extender los operadores para que se contemplen casos menos restrictivos para las fórmulas. Por ejemplo, se puede tratar el caso no monótono, estructuras más complejas sobre los grafos asociados como ciclos anidados o estructura de hipergrafo. Para el cómputo del grado de creencia se puede evaluar el impacto sobre la eficiencia de los algoritmos generados a partir de las extensiones. Finalmente se puede también analizar la estructura de F_1 para garantizar cómputo eficiente.

Bibliografía

[AK07] Horst Bunke y Mark Last Abraham Kandel. *Applied Graph Theory in Computer Vision and Pattern Recognition.* Studies in Computational Intelligence. Springer, 2007.

[Bou95] Craig Boutilier. Abduction to plausible causes: An event-based model of belief update. Technical report, January 1995.

[Che86] Christopher Cherniak. Minimal rationality, 1986.

[Die05] Reinhard Diestel. *Graph Theory.* Springer-Verlag Heidelberg, 2005.

[DNP04a] J. P. Delgrande, A. C. Nayak, and M. Pagnucco. Gricean belief change. *Studia Logica*, (1):1–18, August 2004.

[DNP04b] James P. Delgrande, Abhaya C. Nayak, and Maurice Pagnucco. Conservative belief revision. *AAAI*, pages 251–256, 2004.

[DS04] James P. Delgrande and Torsten Schaub. Two approaches to merging knowledge bases. *JELIA*, pages 426–438, 2004.

[EG] Thomas Eiter and Georg Gottlob. The complexity of nested counterfactuals and iterated knowledge base revisions. pages 526–531.

[EG92] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *PODS*, pages 261–273, 1992.

[FG96] Stan Franklin and Art Graesser. Is it an agent, or just a program? a taxonomy for autonomus agents. *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, pages–, 1996.

[FKUV86] Ronald Fagin, Gabriel M. Kuper, Jeffrey D. Ullman, and Moshe Y. Vardi. Updating logical databases. *Advances in Computing Research*, pages 1–18, 1986.

[FP] George Flouris and Dimitris Plexousakis. Belief revision in propositional knowledge bases. pages 1–10.

[FP01] George Flouris and Dimitris Plexousakis. Belief revision using table transformation. pages 1–38, July 2001.

[FUV83] Ronald Fagin, Jeffrey D. Ullman, and Moshe Y. Vardi. On the semantics of updates in databases. *Advances in Computing Research*, pages 352–365, 1983.

[GHK92] Adam J. Grove, Joseph Y. Halpern, and Daphne Koller. Asymptotic conditional probabilities for first-order logic. *STOC*, pages 294–305, 1992.

[Gin86] Matthew L. Ginsberg. Counterfactuals. *Artificial Intelligence*, (1):80–86, 1986.

[GLI05] Carlos Guillén, Aurelio Lopez, and Guillermo De Ita. Diseño de algoritmos combinatorios para #sat y su aplicación al razonamiento proposicional. Technical report, Noviembre 2005.

[GP96] Moises Goldszmidt and Judea Pearl. Qualitative probabilities for default reasoning, belief revision, and causal modeling. *Artificial Intelligence*, 84:57–112, June 1996.

[HD05] Aaron Hunter and James P. Delgrande. Iterated belief change: A transition system approach. *IJCAI*, pages 460–465, 2005.

[KM91] Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. pages 387–394, 1991.

[L+09] Germán Cárdenas Leroy et al. Análisis de los efectos de la elaboración de enunciados contrafácticos en una tarea de elección. 8(2):385–398, 2009.

[Neb89] Bernhard Nebel. A knowledge level analysis of belief revision. *IWBS*, 1989.

[Neb91] Bernhard Nebel. Belief revision and default reasoning: Syntax-based approaches. *Proceedings of the Second Internatinal Conference*, pages 417–428, April 1991.

[NPP02] Abhaya C. Nayak, Maurice Pagnucco, and Pavlos Peppas. Dynamic belief revision operators. *Artificial Intelligence*, 146:193–228, April 2002.

[RFV86] Jeffrey D. Ullman Ronald Fagin, Gabriel M. Kuper and Moshe Y. Vardi. Updating logical databases. *Advances in Computing Research*, pages 1–18, 1986.

[RSW01a] Ron M. Roth, Paul H. Siegel, and Jack K. Wolf. Efficient coding schemes for the hard-square model. *IEEE Transactions on Information Theory*, (3):1166–1176, 2001.

[RSW01b] Ron M. Roth, Paul H. Siegel, and Jack K. Wolf. A syntactical approach to revision. *IEEE Transactions on Information Theory*, pages 1166–1176, 2001.

[Tor06] Fernando G. Torres. Las distintas miradas acerca de la racionalidad instrumental mínima y la ética, 2006.

[Vaz06] Francisco Javier Tobon Vazquez. *Pizarron electrónico usando Agentes Inteligentes.* 2006.

[Ver99] Susana Segura Vera. *Razonamiento contrafáctico: la posición serial y el número de antecedentes en los pensamientos sobre lo que podría haber sido.* PhD thesis, Junio 1999.

[WJ95] M. Wooldridge and N. Jennings. *Intelligent agents: Theory and practice.* 1995.

[ZZC] Dongmo Zhang, Zhaohui Zhu, and Shifu Chen. Deault reasoning and belief revision: A sintax-independent approach. pages 1–13.